# ZT 8832
# I/O Control Processor

**OPERATING MANUAL**

For ZT 8832 Revision 0.5
and ZT 88CT32 Revision 0.4

Reorder Part Number ZT M8832
May 10, 1994

ZIATECH CORPORATION
1050 Southwood Drive
San Luis Obispo, CA 93401 USA
FAX (805) 541-5088
Telephone (805) 541-0488

# ZIATECH 5+5 WARRANTY

## For Ziatech Board- and System-Level Computer Products

### FIVE-YEAR LIMITED WARRANTY

Products manufactured by Ziatech Corporation are covered from the date of purchase by a five-year warranty against defects in materials, workmanship, and published specifications applicable to the date of manufacture. During the warranty period, Ziatech will repair or replace, solely at its option, defective units provided they are returned at customer expense to an authorized Ziatech repair facility. Products which have been subjected to misuse, abuse, neglect, alteration, or unauthorized repair, determined at the sole discretion of Ziatech, whether by accident or otherwise, are excluded from warranty. The warranty on fans and disk drives is limited to two years and the warranty on flat panel displays is limited to nine months from date of purchase. Other products and accessories not manufactured by Ziatech are limited to the warranty provided by the original manufacturer. Consumable items (fuses, batteries, etc.) and software are not covered by this warranty. Within 90 days of shipping date, Ziatech will replace software disk media should it prove defective.

Ziatech may offer, where applicable and available, replacement products; otherwise, repairs requiring components, assemblies, and other purchased materials may be limited by market availability.

Ziatech assumes no liability resulting from changes to government regulations affecting use of materials, equipment, safety, and methods of repair. Ziatech may, at its discretion, offer replacement products.

THE ABOVE WARRANTY IS IN LIEU OF ANY OTHER WARRANTY, WHETHER EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY FOR FITNESS OF PURPOSE, MERCHANTABILITY, OR FREEDOM FROM INFRINGEMENT OR THE LIKE, AND ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATIONS, OR SAMPLE.

Ziatech neither assumes nor authorizes any person to assume for it any other liability. The liability of Ziatech under this warranty agreement is limited to a refund of the purchase price. In no event shall Ziatech be liable for loss of profits, use, incidental, consequential, or other damage, under this agreement.

## SPECIAL EXTENDED WARRANTY OPTION

In addition to the standard five-year warranty, Ziatech offers, for a nominal fee, an extended period of warranty up to five extra years. This extended warranty period provides similar coverage and conditions as stated above in the five-year limited warranty agreement.

## LIFE SUPPORT POLICY

Ziatech products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Ziatech Corporation. As used herein:

1. Life support devices or systems are devices or systems which support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be expected to cause the failure of the life support device or system, affect its safety, or limit its effectiveness.

## CUSTOMER SUPPORT

If you have a technical question, please call Ziatech's Customer Support Service at the following number:

Corporate Headquarters: (805) 541-0488
(805) 541-5088 (FAX)

You can also use a modem to leave a message on the 24-hour Ziatech Bulletin Board Service (BBS) by calling (805) 541-8218. The BBS will also provide you with current Ziatech product revision and upgrade information.

If you have a sales question, please contact your local Ziatech Sales Representative.

# PREFACE

This manual describes the operation and use of the ZT 8832 and the ZT 88CT32. The boards are functionally identical. However, the ZT 88CT32 consumes less power and operates over a wider temperature range. Refer to Appendix B for actual specifications. The term ZT 8832 is used throughout the manual to reference both products, except where otherwise noted.

The following organizational outline describes the focus of each chapter in this manual. Section headings enclosed in boxes indicate the locations of labeled tabs for quick access to the appropriate information.

## I. INTRODUCTION

**Chapter 1, "Introduction,"** offers an overview of the ZT 8832. It includes a product definition, a list of product features, a functional block diagram, and a brief description of each block. This chapter is most useful to those comparing the features of the ZT 8832 against the needs of a specific application.

## II. GETTING STARTED

**Chapter 2, "Getting Started,"** summarizes the information essential to getting your ZT 8832 operational. This includes system requirements, memory and I/O mapping, and a brief description of the jumper options. In many cases this information is all that is needed to begin using the ZT 8832. The section entitled "USER'S REFERENCE" includes more detailed discussions of the topics presented in this chapter.

## III. USER'S REFERENCE

**Chapter 3, "Theory Of Operation,"** presents a detailed description of ZT 8832 system level operation. Topics discussed include commonly asked questions, memory and I/O organization, and interrupt structure.

**Chapter 4, "Application Examples,"** gives specific examples of the ZT 8832 in operation, including code to implement these applications.

**Chapter 5, "Processor Description (V40),"** divides the V40 into functional blocks and presents a theory of operation for each. This chapter is most useful to those not familiar with the architecture of the V40/8088 series microprocessor.

**Chapter 6, "Processor Configuration (V40),"** describes the architecture of a V40 software programmable register set used to configure peripherals resident on the V40 for specific applications. These peripherals are described in detail in the following chapters.

**Chapter 7, "Counter/Timers (V40),"** describes the function, configuration, and operation of the V40 Counter/Timer Control Unit, which includes three 16-bit counter/timers and is functionally equivalent to the 8254 Programmable Interval Timer. The chapter also includes register descriptions.

**Chapter 8, "Interrupt Controller (V40),"** describes the function, configuration, and operation of the V40 Interrupt Control Unit, a programmable interface between interrupt-generating peripherals and the CPU. The chapter also includes register descriptions.

**Chapter 9, "DMA Controller (V40),"** describes the function, configuration, and operation of the V40 Direct Memory Access Control Unit, a programmable peripheral device used to direct high speed data transfers between the ZT 8832 and SBX expansion module I/O. The chapter also includes register descriptions.

**Chapter 10, "Serial Communications (V40),"** describes the function, configuration, and operation of the V40 Serial Control Unit, a single serial channel that performs asynchronous serial communication between the V40 and a serial device external to the ZT 8832. The chapter also includes register descriptions.

**Chapter 11, "Serial Communications (82050),"** describes the function, configuration, and operation of the serial controller external to the V40. This is the Intel 82050 Asynchronous Communications Controller, or equivalent. The chapter also includes register descriptions.

**Chapter 12, "Parallel I/O,"** describes the function, configuration, and operation of the ZT 8832 parallel I/O. Parallel I/O is used to control industry standard I/O modules, such as those manufactured by Opto 22, Gordos, Crydom, and Adtek. The on-board programmable LEDs are also controlled through parallel I/O.

**Chapter 13, "Watchdog Timer,"** describes the function, configuration, and operation of the ZT 8832 two-stage watchdog timer, which is used to monitor ZT 8832 operation and take corrective action if the ZT 8832 fails to function as programmed.

**Chapter 14, "SBX Expansion Module,"** contains a description and installation information for the optional SBX expansion module, which can be used to expand the I/O capabilities of the ZT 8832.

**Chapter 15, "Numeric Data Processor (8087),"** explains installation and operation of the optional 8087 Numeric Data Processor, used to enhance the math capabilities of the ZT 8832 for numerically intensive applications.

## IV. APPENDICES

**Appendix A, "Jumper Selections,"** provides a detailed explanation of the ZT 8832 options that are selectable through jumper configuration or the use of cuttable traces.

**Appendix B, "Specifications,"** contains the electrical, environmental, and mechanical specifications for the ZT 8832. Frontplane connector pinouts, cable drawings, and timing diagrams are also included.

**Appendix C, "Customer Support,"** offers a product revision history, technical support information, and instructions for returning the ZT 8832 for service.

**Appendix D, "Glossary,"** defines important terms and acronyms used throughout the manual.

# CONTENTS

# TABLES

## Tables

# ILLUSTRATIONS

Chapter 1

# INTRODUCTION

**Contents** **Page**

# OVERVIEW

This chapter provides a brief introduction to the ZT 8832. It includes a product definition, a list of product features, a functional block diagram, and a description of each block. Unpacking information and installation instructions can be found in Chapter 2, "Getting Started."

## Product Definition

The ZT 8832 is an 8MHz V40-based single board computer designed to operate by itself or as an I/O control processor in an STD bus system. The ZT 8832 increases system throughput by managing time-critical operations normally assigned to the STD bus master processor. These operations include numerical computation, serial communication, and digital I/O using resources local to the ZT 8832, as well as a large selection of digital and analog I/O via the SBX expansion module. Complex systems are simplified by partitioning the design into smaller tasks and assigning these tasks to one or more ZT 8832s.

The ZT 8832 and the STD bus master processor communicate through 32 Kbytes of shared (dual port) RAM, and I/O mapped control and status registers, located on the ZT 8832. These registers include features such as maskable and non-maskable interrupt generation between the ZT 8832 and the master processor.

Arbitration for the 32 Kbytes of dual ported RAM is implemented entirely in hardware. The ZT 8832 also supports a board select option that reduces the amount of STD bus addressing space needed in a multiple ZT 8832 configuration. With board select, up to seven ZT 8832s can be mapped into a single 32 Kbyte range.

## Features of the ZT 8832

- STD 32 bus compatible
- 8088/8086 code compatible
- One 32-pin EPROM socket (480 Kbyte capacity)
- Two 32-pin RAM sockets (512 Kbyte capacity)
- 32 Kbyte dual port RAM (populated)
- Optional battery backup on all RAM
- Interrupt controller (V40)
- Three 16-bit counter/timers (V40)
- Three 8-bit parallel ports, Opto 22 compatible
- Two serial ports (V40, 82050), RS-232/422/485
- Two-stage watchdog timer
- zSBX expansion module with DMA support
- 8087 Numeric Data Processor socket
- 20- or 24-bit STD bus memory address decoding
- 16-bit STD bus I/O address decoding
- Programmable board select for multiple ZT 8832s
- Software programmable LED
- Pushbutton reset
- Optional cables for parallel and serial interface
- Optional STD ROM development/debug monitor
- Optional DOS Multiprocessor Extension (DOS MPX) device drivers
- Five-year warranty

## Development Considerations

Ziatech offers two software development systems for ZT 8832 applications: STD ROM and DOS MPX.

STD ROM (Borland's Turbo Debugger® environment using Paradigm Systems' DEBUG/RT™) connects the ZT 8832 to an IBM-compatible personal computer through a high speed serial link. The computer is used as a development station to create, download, and debug applications written in assembly, C, and other high level languages. Paradigm's DEBUG/RT used during the debug phase is similar to Microsoft Codeview. It includes features such as source level debugging, single stepping, breakpoints, watchpoints, and pull down menus.

The DOS MPX package includes a boot ROM for the ZT 8832, an installable device driver for the STD bus CPU, and a utility for downloading programs to the ZT 8832. Also included is a utility to permit several 8832s to share the same console as the STD bus CPU.

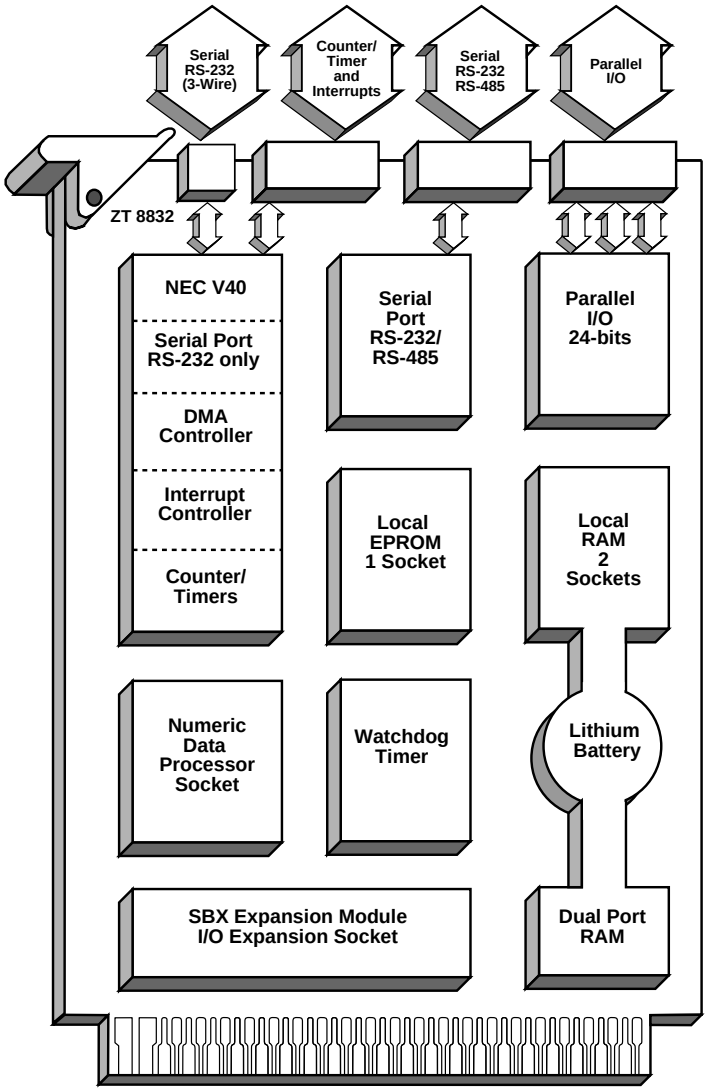



*Figure 1–1. Functional Block Diagram.*

## FUNCTIONAL BLOCKS

Figure 1-1 illustrates the functional blocks of the ZT 8832. A description of each block follows.

### V40 (μPD70208) Processor

The NEC V40 is a highly integrated microprocessor that includes an 8088 compatible CPU and many standard I/O devices. The 8088 compatible CPU executes all code written for the 8088/8086 family of microprocessors. In addition, the V40 instruction set includes expanded rotate and shift, bit and nibble manipulation, and string I/O. The standard I/O devices found in the V40 include an interrupt controller (8259 architecture), counter/timers (8284 architecture), a serial controller (8251 architecture), and a DMA controller. The V40 is fabricated with CMOS technology to provide an increase in both temperature range and noise immunity with a reduction in power consumption.

### Local Memory

The ZT 8832 is populated with three 32-pin memory sockets dedicated for use by the V40 microprocessor. One socket supports 8K, 16K, 32K, 64K, 128K, and 512 Kbyte EPROMs. The remaining two sockets each support 32 Kbyte and 128 Kbyte RAMs, with optional battery backup. Additionally, one of the RAM sockets can be configured to support a 512 Kbyte RAM. This memory is not accessible by the STD bus CPU.

## Dual Port Memory

The ZT 8832 is populated with 32 Kbytes of RAM that is accessible from both the local CPU and the STD bus CPU. Arbitration for simultaneous access is done entirely in hardware. To increase system performance, the dual port memory is physically separated from the local memory. This permits the local CPU to continue executing local operations while the STD bus CPU is accessing the dual port memory. The STD bus CPU can also continue operations while the local CPU accesses dual port memory. Only when both CPUs attempt a dual port memory access is the operation of one suspended until the other is completed. Repetitive access from both processors is interleaved on a machine cycle basis. The dual port memory also supports software programmable interrupts, a locking mechanism, and optional battery backup. The locking mechanism allows either the STD bus CPU or the local CPU to keep the other from accessing the dual port memory until the lock is removed.

## Board Select Option

One ZT 8832 occupies 32 Kbytes of STD memory addressing space. For applications using multiple ZT 8832s, each can be mapped into a unique 32 Kbyte space, or up to seven can be mapped into a common 32 Kbyte space by using the board select option. This option is enabled by mapping each ZT 8832 to the same address range and a unique board select address. In operation, all commonly mapped ZT 8832s monitor the same I/O port for a board select address. To communicate with a ZT 8832, the STD bus CPU writes the appropriate board select address to the common I/O port. The STD bus CPU is then free to communicate with the selected ZT 8832. It is important to note that all functions local to the ZT 8832 continue to operate when it is not selected by the STD bus CPU.

## Serial Communications

The ZT 8832 includes two asynchronous serial communication channels, each with a programmable baud rate generator. The V40 provides one serial channel with an architecture similar to the asynchronous portion of the 8251. Configured as RS-232 DTE, this serial channel supports transmit data (TxD), receive data (RxD), and ground. These signals are available through a 3-pin frontplane connector. The optional 1 meter ZT 90069 cable is available to connect the 3-pin connector to a male 25-pin D type connector.

The second serial channel (Intel 82050, or equivalent) is functionally equivalent to the serial controller used in the IBM line of personal computers. This channel supports all handshaking and modem control signals and can be jumper programmed for RS-232 or RS-422/485. Other features include loopback diagnostics, maskable interrupt generation, and jumper selectable DCE or DTE configuration. The 82050 serial channel is accessed through a 14-pin frontplane connector. The optional 1 meter ZT 90014 and ZT 90027 cables connect the 14-pin connector to a male or female 25-pin D type connector, respectively.

## Parallel Input/Output

There are three 8-bit parallel ports on the ZT 8832, for a total of 24 I/O lines. Each I/O line can be programmed as input or as output with readback. The I/O signals, fused power, and ground are accessible through a 26-pin frontplane connector. The optional 1 meter ZT 90068 cable is available to connect the 26-pin connector directly to an 8-, 16-, or 24-position I/O module mounting rack, such as Ziatech's ZT 2226 or the rack offered by Opto 22.

## Interrupt Controller

The ZT 8832 includes an eight-input programmable interrupt controller with an 8259 architecture. Features of the interrupt controller include level- and edge-triggered sensing, fixed and rotating priorities, and the ability to mask individual inputs. Two of the interrupt request inputs are available through a frontplane connector, to be used as needed by the application. The remaining inputs are dedicated to interrupt sources such as counter/timers for timed or periodic interrupt generation, serial controllers for interrupt driven data transfers, zSBX expansion module for interrupt driven I/O, and the STD bus to support interrupt-driven data transfers from the STD bus CPU through dual port memory.

## Counter/Timers

The ZT 8832 has three independent 16-bit counter/timers with an 8254 architecture that can be used as timers or as event counters. There are six programmable counter/timer modes: interrupt on end of count, frequency divider, square wave generator, software-triggered, hardware-triggered, and retriggerable one-shot. One of the counter/timers is available through a frontplane connector, to be used as required by the application. The other two counter/timers are dedicated to uses such as baud rate generation for the V40 serial channel and interrupt generation for timed and periodic interrupts.

## Watchdog Timer

The ZT 8832 includes a two-stage watchdog timer. The main function of a watchdog timer is to monitor system operation and take corrective action if the system fails to operate as designed. While in operation, the watchdog timer must be strobed at a predetermined rate by toggling a parallel port bit. Failure to strobe the watchdog results in a non-maskable interrupt to the V40. The ZT 8832 is reset unless the non-maskable interrupt service routine takes corrective action that includes strobing the watchdog timer within a certain time period. The watchdog timer is enabled through jumper selection.

## zSBX Expansion Module Socket

The zSBX expansion module socket is provided to customize the I/O capabilities of the ZT 8832 to the needs of the application. The expansion module interface is electrically, mechanically, and functionally equivalent to the Intel iSBX MULTIMODULE™ standard, including DMA support for high speed transfers to ZT 8832 memory. This makes hundreds of off-the-shelf modules available to the STD bus designer. Functions available include servo and step motor control, analog-to-digital and digital-to-analog conversion, serial and parallel I/O, graphics, and modem control. There are also prototyping boards for those who need to design custom I/O.

## Numeric Data Processor Socket

The math capabilities of the V40 include addition, subtraction, multiplication, and division of 8-bit and 16-bit numbers. The STD bus is often used in numerically intensive applications needing more powerful arithmetic operations than those provided by the V40. These requirements are realized with the addition of an 8087 Numeric Data Processor (NDP) to the ZT 8832. Simply plugging the NDP into the empty socket on the ZT 8832 provides 68 added instructions for extended arithmetic, trigonometric, exponential, and logarithmic functions. These added instructions are supported by seven additional data types including integers (16-, 32-, and 64-bit), floating point (32-, 64-, and 80-bit), and BCD (18-digit). Using the NDP increases performance up to 100 times that of equivalent routines implemented in software.

Chapter 2

# GETTING STARTED

**Contents** **Page**



## OVERVIEW

This chapter provides a summary of the information essential to getting the ZT 8832 operational. Read this chapter before attempting to use the board.

## UNPACKING

Please check the shipping carton for damage. If the shipping carton and contents are damaged, notify the carrier and Ziatech for an insurance settlement. Retain the shipping carton and packing material for inspection by the carrier. Do not return any product to Ziatech without a Return Material Authorization (RMA) number.

Appendix C explains the procedure for obtaining an RMA number from Ziatech.

## WHAT'S IN THE BOX?

The items listed below are included in a standard ZT 8832 order. System level products, such as the STD ROM and DOS MPX, include additional items not shown. If you have ordered a system level product, refer to the system manual for the packing list.

- ZT 8832 I/O Control Processor in anti-static bag
- Optional ZT 8832 Operating Manual in binder
- Anti-static packing material

Attach the sticker packaged with the manual to the spine of the binder for easy identification. Save the anti-static packing material for storing or returning the ZT 8832.

*WARNING!*

*Like all equipment utilizing MOS devices, the ZT 8832 must be protected from static discharge. Never remove or install any of the socketed parts except at a static-free work station.*

## SYSTEM REQUIREMENTS

The ZT 8832 is designed for use with or without an STD bus backplane. For STD bus applications, the ZT 8832 is mechanically and electrically compatible with Version 1.1 of the *STD 32 Bus Specification and Designer's Guide*.

Vertical mounting is recommended in applications not using a fan. Horizontal mounting requires a minimum air flow of 30 cubic feet/min. passing over the surface of the board.

Refer to Appendix B for additional specifications.

### ZT 8832 Requirements

The ZT 8832 requires +5 VDC ±5% at 1.5 A maximum, +12 VDC ±10% at 0.10 A maximum, and -12 VDC ±10% at 0.10 A maximum. The ambient temperature range must be maintained between 0 and +65˚ Celsius to guarantee proper operation and to avoid damage. Relative humidity must be less than 95% at 40˚ Celsius, non-condensing.

### ZT 88CT32 Requirements

The ZT 88CT32 requires +5 VDC ±5% at 0.5 A maximum, +12 VDC ±10% at 0.10 A maximum, and -12 VDC ±10% at 0.10 A maximum. The ambient temperature range must be maintained between -40˚ and +85˚ Celsius to guarantee proper operation and to avoid damage. Relative humidity must be less than 95% at 40˚ Celsius, non-condensing.

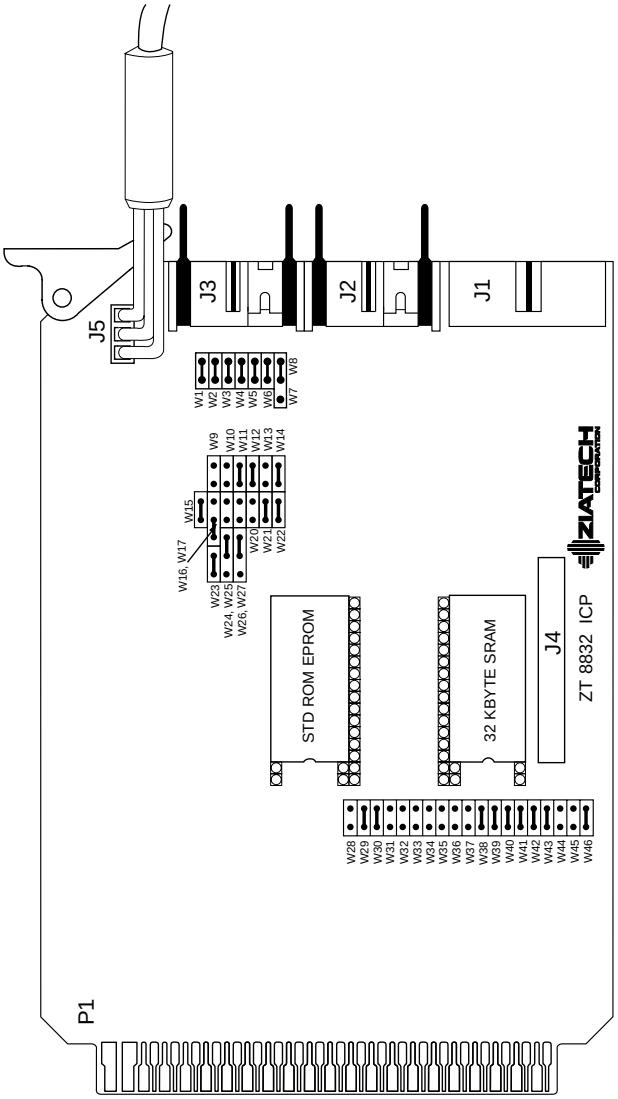



*Figure 2–1. STD ROM Jumper Configuration.*

## INSTALLING THE ZT 8832 WITH STD ROM

The fastest way to begin using the ZT 8832 is with the addition of development software from Ziatech. The STD ROM development system connects the ZT 8832 to an IBM-compatible personal computer through a serial link. The personal computer is used as a development station to create, download, and debug applications written in assembly, C, and other high level languages.

If the ZT 8832 and STD ROM are ordered together, Ziatech configures the ZT 8832 and tests the system level operation. If the ZT 8832 and STD ROM are ordered separately, or the system has been altered and no longer functions properly, refer to Figure 2-1 and check the following configuration requirements.

### Memory Requirements

The STD ROM firmware is shipped in a 64 Kbyte EPROM. Install the EPROM into the ZT 8832 socket labeled *ROM*. The 28-pin device must be right-justified in the 32-pin socket, as shown in Figure 2-1.

STD ROM requires 2 Kbytes of RAM memory, from address 0 through 7FFh. Installing a 32 Kbyte RAM in the socket labeled *RAM LOW* meets this requirement. The 28-pin device must be right-justified in the 32-pin socket, as shown in Figure 2-1. If more than 32 Kbytes is needed, a second 32 Kbyte device can be added to the *RAM HIGH* socket, or larger RAM devices can be used.

### Cable Requirements

STD ROM requires a serial link between the ZT 8832 and the terminal or PC. The ZT 90069 cable shipped with the STD ROM system is used for this purpose. Plug the cable into the J5 connector of the ZT 8832 as shown in Figure 2-1.

## Jumper Requirements

Table 2-1 and Figure 2-1 show the correct jumper positioning for the ZT 8832 configured to support STD ROM.

Table 2-1
STD ROM Jumper Configuration.

| | |
|---|---|
| Installed | W1-6, 8, 11, 12, 14-16, 21-23, 25, 27, 29, 30, 37, 39, 41-43, 46 |
| Removed | W7, 9, 10, 13, 17-20, 24, 26, 28, 31-36, 38, 40, 44, 45 |

## Operation

Refer to the STD ROM documentation for software installation and operation procedures.

# INSTALLING THE ZT 8832 WITH DOS MPX

The DOS Multiprocessing Extension (MPX) software provides a development and operating environment for one or more ZT 8832s in a DOS application. DOS MPX allows programmers to develop applications using a high-level language such as C without the problems associated with placing the program in ROM.

DOS MPX includes a boot EPROM for the ZT 8832, an installable device driver for the system master processor, and a utility for downloading programs from the master processor to the ZT 8832.

## Memory Requirements

The DOS MPX firmware is shipped in a 64 Kbyte EPROM. Install the EPROM into the ZT 8832 socket labeled *ROM*. The 28-pin device must be right justified in the 32-pin socket, as shown in Figure 2-2.

DOS MPX requires at least 32 Kbytes of local RAM. Installing a 32 Kbyte RAM in the socket labeled *RAM LOW* meets this requirement. The 28-pin device must be right-justified in the 32-pin socket, as shown in Figure 2-2. If more than 32 Kbytes is needed, a second 32 Kbyte device can be added to the *RAM HIGH* socket, or larger RAM devices can be used.

## Cable Requirements

DOS MPX shares video and keyboard with the system master processor. The only cabling required is that dictated by the application requirements.

## Jumper Requirements

Table 2-2 and Figure 2-2 show the correct jumper positioning for the ZT 8832 configured to support DOS MPX. This is the same jumper configuration required for STD ROM.

Table 2-2
DOS MPX Jumper Configuration.

| | |
|---|---|
| Installed | W1-6, 8, 11, 12, 14-16, 21-23, 25, 27, 29, 30, 37, 39, 41-43, 46 |
| Removed | W7, 9, 10, 13, 17-20, 24, 26, 28, 31-36, 38, 40, 44, 45 |

## Operation

Refer to the DOS MPX documentation for software installation and operation procedures.

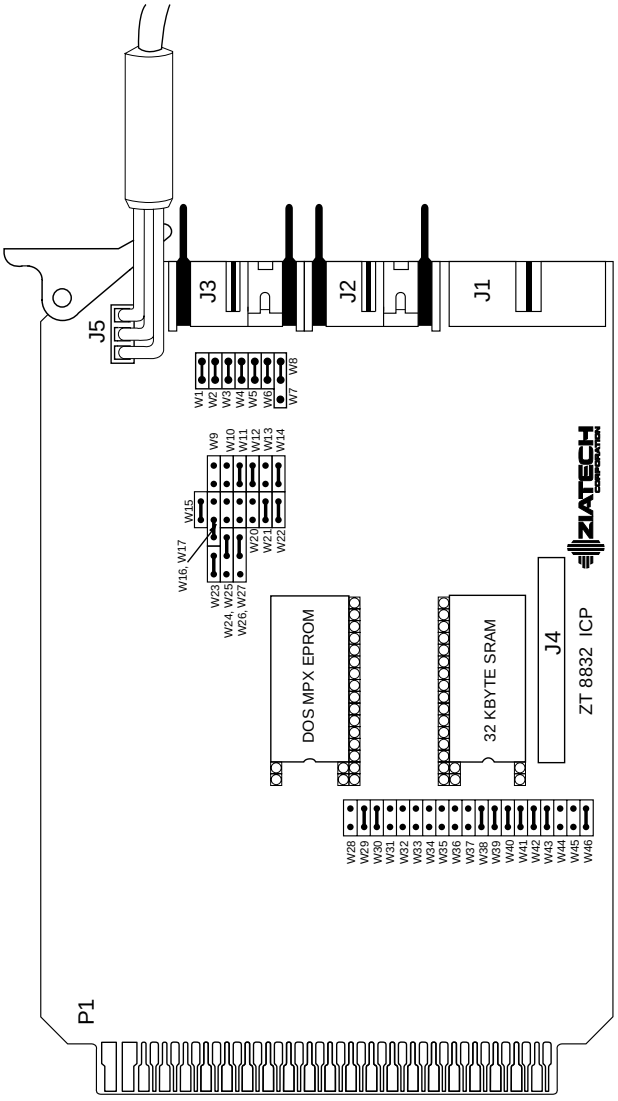



*Figure 2–2. DOS MPX Jumper Configuration.*

## MEMORY

The ZT 8832 includes memory that is addressable by the local CPU and memory that is dual ported between the local and STD bus CPU. This is shown in two memory maps: Figure 2-3 illustrates the address ranges of the memory devices available to the local CPU, and Figure 2-4 illustrates the address ranges of the memory devices available to the STD bus CPU.

The local ROM is a single memory device located in a 32-pin JEDEC socket. While this socket is jumper configurable to support 8K, 16K, 32K, 64K, 128K, 256K, and 512 Kbyte devices, the ROM address range is always 88000h through FFFFFh. This means that 8K, 16K, 32K, 64K, 128K, and 256 Kbyte devices are redundantly mapped, and only 480 Kbytes of the 512 Kbyte device is used. Please note that while the 256K and 512 Kbyte devices are supported, they may not be currently available. The access time of the local ROM device must not be greater than 250 ns. This ROM is not addressable by the STD bus CPU.

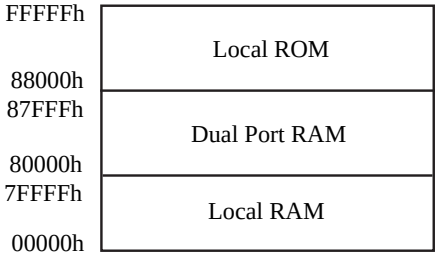



*Figure 2–3. Local CPU Memory.*

The dual port RAM is a 32 Kbyte RAM populated on the ZT 8832. It is shown on both local and STD bus memory maps because, even through it is the same physical memory, it is accessible from both the local CPU and the STD CPU. The address range of the dual port RAM as seen by the local CPU is fixed from 80000 through 87FFFh. The address range of the dual port RAM as seen by the STD bus CPU is jumper selectable to any 32 Kbyte memory block within a 1 Mbyte address range of an STD bus CPU. Some STD bus microprocessors support a 16 Mbyte address range by generating a 24-bit memory address. The ZT 8832 can be configured to jumper select the dual port RAM into any 32 Kbyte memory block within this 16 Mbyte address range. As shipped, the address range is XC8000 through XCFFFFh.

The local RAM consists of two contiguously mapped 32-pin JEDEC sockets. The sockets are shipped to support 32 Kbyte RAM devices. Jumper options also permit both sockets to support 128 Kbyte devices, and one of the sockets to support a 512 Kbyte device. Please note that while the 512 Kbyte RAM is supported by the ZT 8832, it may not be currently available. The access time of the local RAM devices must not be greater than 250 ns. This RAM is not addressable by the STD bus CPU.

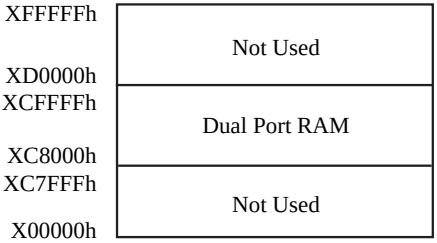



*Figure 2–4. STD Bus CPU Memory.*

## I/O

The ZT 8832 includes some I/O devices addressable by the local CPU and other I/O devices addressable by the STD bus CPU. This is shown in two I/O maps. Figure 2-6 illustrates the devices accessible by the local CPU. Figure 2-5 illustrates the devices available to the STD bus CPU. Note that none of the I/O devices are accessible from both the local CPU and the STD bus CPU.

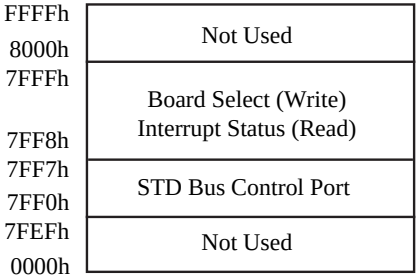



*Figure 2–5. STD Bus CPU I/O.*

Several I/O devices available to the local CPU must be enabled and mapped by programming the V40 configuration registers. These devices are the interrupt controller, counter/timers, DMA controller, and V40 serial controller. The address mapping of all other local I/O devices is fixed to the locations shown in Figure 2-6. Note that these I/O devices are not accessible by the STD bus CPU.

The ZT 8832 requires 16 consecutive STD bus I/O port addresses. The STD bus control port is a single byte value redundantly mapped over the lower eight addresses. The STD bus control port can only be written; no readback is available. The upper eight addresses are shared between the Board Select and Interrupt Status Port. Both the Board Select and Interrupt Status Ports are redundantly mapped single byte values. The Board Select Port can only be written; no readback is available. The Interrupt Status Port can only be read.

| | |
|---|---|
| FFF0-FFFFh | V40 Configuration |
| 0400-FFEFh | Not Used |
| 03F0-03FFh | 82050 Serial Port |
| 0380-03EFh | Not Used |
| 0300-037Fh | SBX Module - Select 1 |
| 0280-02FFh | SBX Module - Select 0 |
| 0240-027Fh | Not Used |
| 0230-023Fh | Local Control Port |
| 0220-022Fh | Parallel Port 2 |
| 0210-021Fh | Parallel Port 1 |
| 0200-020Fh | Parallel Port 0 |
| 00E0-01FFh | Not Used |
| 00D0-00DFh | DMA Controller [1] |
| 00B8-00CFh | Not Used |
| 00B0-00B7h | V40 Serial Port [1] |
| 0048-00AFh | Not Used |
| 0040-0047h | Counter/Timers [1] |
| 0028-003Fh | Not Used |
| 0020-0027h | Interrupt Controller [1] |
| 0000-0001Fh | Not Used |

[1] These devices are internal to the V40 and must be enabled and mapped through software configuration.

*Figure 2–6. Local CPU I/O.*

## JUMPER OPTIONS

The ZT 8832 includes several jumper options that tailor the operation of the board to the requirements of specific applications. Refer to Appendix A for a description of configurable jumpers.

# III. USER'S REFERENCE

Chapter 3

# THEORY OF OPERATION

**Contents** **Page**

## OVERVIEW

This chapter presents a detailed description of ZT 8832 system level operation. Topics discussed include:

- Commonly asked questions regarding the ZT 8832
- Issues surrounding STD bus compatibility
- Operation of dual port memory and control port I/O used to interface the ZT 8832 to the STD bus
- Using the board select option to reduce STD bus memory and I/O requirements in a system with more than one ZT 8832
- Using STD bus interrupts
- The reset state of the ZT 8832 and the relationship between reset on the ZT 8832 and reset on the STD bus

## COMMONLY ASKED QUESTIONS

1. **What software development tools are available?**

   Ziatech supports two software development tools for the ZT 8832: STD ROM and DOS Multiprocessing Extension (DOS MPX).

   The STD ROM software includes a PROM based debugger and a PC based floppy disk. The ZT 8832 is connected to a PC through a serial link. Programs are developed on the PC and transferred across the serial link to the ZT 8832 for debugging.

   DOS MPX provides software support for one or more ZT 8832s in an STD bus application running under Ziatech DOS. The DOS MPX package includes a boot ROM for the ZT 8832, an installable device driver for the STD bus CPU, and a utility for downloading programs to the ZT 8832. Also included is a Virtual Processor Console (VPC) utility to multiplex several ZT 8832s and the STD bus CPU on a single console.

2. **Are emulators available for the NEC V40?**

   Yes. Several manufacturers of V40 emulators are listed below.

| | |
|---|---|
| Tektronix, Inc. | P.O. Box 4828<br>Portland, OR 97208<br>(818) 999-1711 |
| Sophia Computer Systems | 3337 Kifer Road<br>Santa Clara, CA 95051<br>(408) 733-1571 |
| Zax Corporation | 2572 White Road<br>Irvine, CA 92714<br>(714) 474-1170 |
| American Automation | 14731 Franklin Avenue<br>Tustin, CA 92680<br>(714) 731-1661 |
| Hewlett-Packard | 1501 Page Mill Road<br>Palo Alto, CA 94304<br>Eastern (301) 258-2000<br>Midwestern (312) 255-9800<br>Southern (404) 955-1550<br>Western (818) 508-2319 |

3. **Can the ZT 8832 access STD bus memory and I/O?**

   The ZT 8832 includes 32 Kbytes of dual port RAM mapped into the STD bus memory addressing space. This is the only STD bus memory accessible by the local CPU. The local CPU does not have access to any STD bus I/O.

4. **Can the STD bus CPU access memory and I/O local to the ZT 8832?**

   The STD bus CPU has access to 32 Kbytes of dual port RAM local to the ZT 8832. The three 32-pin JEDEC sockets are not available to the STD bus CPU. The ZT 8832 includes 16 I/O ports that are accessible by the STD bus CPU. This I/O is divided into the STD bus control port, board select port, and interrupt status port. This is the only I/O on the ZT 8832 available to the STD bus CPU.

5. **How does the ZT 8832 compare with Ziatech's earlier I/O control processor, the ZT 8830?**

   The ZT 8832 and ZT 8830 are both designed to increase the modularity and performance of STD bus applications through parallel processing. Both boards include 32 Kbytes of dual port RAM accessible by both the STD bus CPU and the CPU local to the board. Both boards also include an SBX expansion module for I/O expansion with off-the-shelf or custom I/O. It should be noted that the SBX expansion module connector is in the same place on both boards; this permits custom SBX designs for the ZT 8830 to be used on the ZT 8832.

The ZT 8832 is a newer generation board that removes many of the restrictions found on the ZT 8830. The major differences are outlined below.

- **Increased performance.** Both the ZT 8830 and ZT 8832 are based on an 8 MHz microprocessor with an 8088 architecture. The major performance increase of the ZT 8832 is realized in the dual port memory design. On the ZT 8830, all local memory is dual ported. This means that the local CPU is suspended any time the STD bus CPU accesses the ZT 8830. The ZT 8832 separates the dual port memory from local memory so the local CPU continues to operate while the STD bus CPU accesses the dual port memory. It is not until both CPUs attempt a dual port access that one is suspended until the other has completed its dual port access. The ZT 8832 also supports DMA on the SBX expansion module for high speed I/O transfers and an 8087 Numeric Data Processor for math operations up to 100 times faster than possible with a ZT 8830.
- **Reduced STD bus memory requirements.** The ZT 8830 supports 32 Kbytes of EPROM and 32 Kbytes of RAM, all of which are dual ported. This means that each ZT 8830 requires 64 Kbytes of STD bus memory. This requirement is compounded when multiple ZT 8830s are used in a single system. The ZT 8832 requires only 32 Kbytes of STD bus memory for the dual port RAM. The ZT 8832 also supports a board select option that permits up to seven ZT 8832s to be mapped into a single 32 Kbyte STD bus address range.

- **More local memory.** The ZT 8830 supports a maximum of 32 Kbytes of RAM and 32 Kbytes of EPROM. With memory devices currently available on the market, the ZT 8832 supports 256 Kbytes of RAM and 128 Kbytes of EPROM. When larger devices become available, the ZT 8832 can be expanded to 512 Kbytes of RAM and 480 Kbytes of EPROM.

6. **How many ZT 8832s can be used in a system?**

   The number of ZT 8832s is limited by the number of slots in the STD bus card cage and power supply capabilities.

7. **Can the ZT 8832 operate without an STD bus backplane?**

   The ZT 8832 is designed to operate with or without an STD bus backplane. It can be useful to operate small dedicated control applications without a backplane when the ZT 8832 contains all of the I/O needed.

## DUAL PORT MEMORY

The ZT 8832 is shipped with 32 Kbytes of dual port RAM. Dual port means that is it accessible by both the ZT 8832 CPU and the STD bus CPU. The dual port memory is physically separate from the ZT 8832 local memory to permit the local CPU to continue operating even during a dual port access by the STD bus CPU. It is not until both CPUs attempt a simultaneous dual port access that arbitration logic suspends the operation of one CPU until the other is completed. For continuous accesses by both CPUs, the arbitration logic alternates the access grant on a machine cycle boundary to ensure both processors equal time. The arbitration is done entirely in hardware.

While using the dual port is as simple as writing to a standard memory device, it may be beneficial to implement interrupt driven or locked transfers. These are supported through the STD bus control port and the local control port discussed on the following pages. Note that the dual port RAM is battery backed if the optional battery is installed.

## STD BUS AND LOCAL CONTROL PORTS

The STD bus and local control ports increase the control and flexibility of the communication link between the STD bus and the ZT 8832.

### STD Bus Control Port Overview

The STD bus control port is an I/O mapped device programmed by the STD bus CPU to perform the following operations.

- Generate maskable and non-maskable interrupts
- Lock dual port access
- Reset the maskable interrupt generated by the local CPU
- Reset the ZT 8832

The STD bus CPU control port does not have readback capabilities and is not accessible by the local CPU.

## STD Bus Control Port Architecture

Figure 3-1 shows the architecture of the STD bus control port. Bit definitions are given on the following pages.

**Note:** Writing a 0Fh followed by a logical 0 to the STD bus control port resets the ZT 8832.

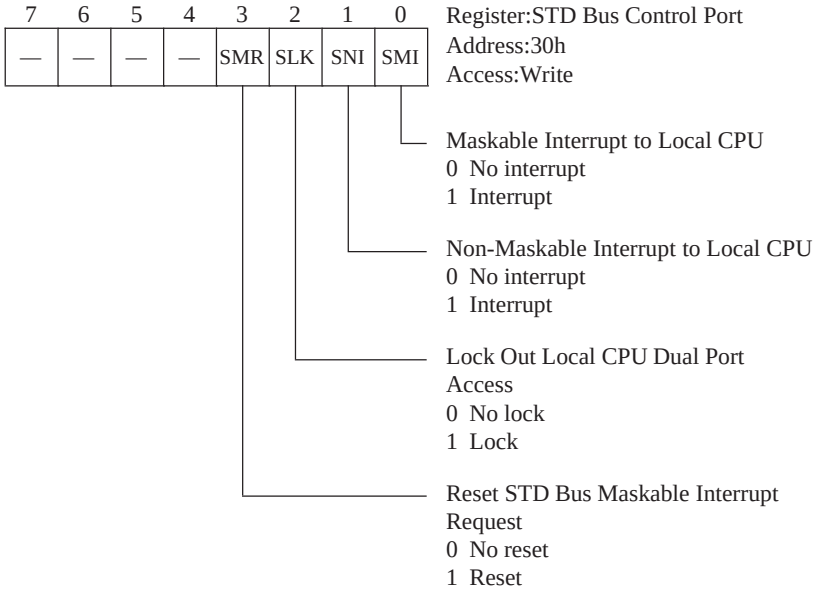



*Figure 3–1. STD Bus Control Port Architecture.*

The *STD Bus Control Port Maskable Interrupt (SMI)* bit (bit 0) is programmed by the STD bus CPU to generate a maskable interrupt to the local CPU. The SMI bit is connected to interrupt request 5 (IR5) of the interrupt controller. The local CPU must complete the following operations before the SMI bit is active:

- Enable the interrupt controller with the OPCN V40 configuration register, as discussed in Chapter 6;

- Program the interrupt controller to enable IR5, as discussed in Chapter 8;
- Remove the hardware mask by programming the LMR bit of the local control port with a logical 0.

After the local CPU has performed these operations, the STD bus CPU generates a maskable interrupt to the local CPU by writing a logical 1 to the SMI bit of the STD bus control port. After a logical 1 is written by the STD bus CPU, all other writes to the SMI bit are ignored until the interrupt request is cleared by the local CPU writing a logical 1 followed by a logical 0 to the LMR bit of the local control port.

The *STD Bus Control Port Non-maskable Interrupt (SNI)* bit (bit 1) is programmed by the STD bus CPU to generate a non-maskable interrupt to the local CPU. The STD bus CPU generates a non-maskable interrupt to the local CPU by writing a logical 1 followed by a logical 0 to the SNI bit. The SNI bit shares the non-maskable interrupt with the jumper selectable watchdog timer and the optional numeric data processor. While there is no status available to the local CPU that indicates SNI is active, there is status for both the watchdog timer and the numeric data processor. This means that if more than one of the non-maskable interrupt sources is used, the local CPU can determine that the SNI bit generated a request by eliminating the other possible sources through polling.

The *STD Bus Control Port Lock Dual Port Access (SLK)* bit (bit 2) is programmed by the STD bus CPU to prevent the local CPU from accessing the dual port RAM. The STD bus CPU arms Lock by programming the SLK bit with a logical 1. At this point, the local CPU can still access the dual port RAM. It is not until the STD bus CPU performs the first read, or a write of the dual port RAM, that the local CPU is no longer granted access. Any attempt by the local CPU to access dual port RAM is denied, suspending local CPU operation until the STD bus CPU programs SLK with a logical 0.

The *STD Bus Control Port Maskable Interrupt Reset (SMR)* bit (bit 3) is programmed by the STD bus CPU to reset an active STD bus maskable interrupt request generated by the local CPU through the LMI bit of the local control port. The Interrupt Status Port (ISP) is available to the STD bus CPU to determine if the local control port LMI bit is active. The STD bus CPU must first program the SMR bit with a logical 0 to enable the local CPU to generate a maskable interrupt. To reset an active maskable interrupt, the STD bus CPU must program the SMR bit with a logical 1 followed by a logical 0.

### STD Bus Control Port Programmable Reset

The STD bus CPU resets the local CPU by writing a 0Fh to the STD bus control port followed by a logical 0. This software programmable reset does not affect the STD bus. The ZT 8832 enters reset a maximum of 250 ms after the 0Fh is written and remains reset until the logical 0 is written. The STD bus CPU should not access the ZT 8832 before the logical 0 is written.

## Local Control Port Overview

The local control port is an I/O mapped device programmed by the local CPU to perform the following operations.

- Generate maskable and non-maskable interrupts
- Lock dual port access
- Reset the maskable interrupt generated by the STD bus CPU

The local control port does not have readback and is not accessible by the STD bus CPU.

## Local Control Port Architecture

Figure 3-2 shows the architecture of the local control port. Bit definitions are given on the following pages.

**Note:** The STD bus cannot be reset through the local control port. All other functions are symmetrical between the STD bus control port and the local control port.

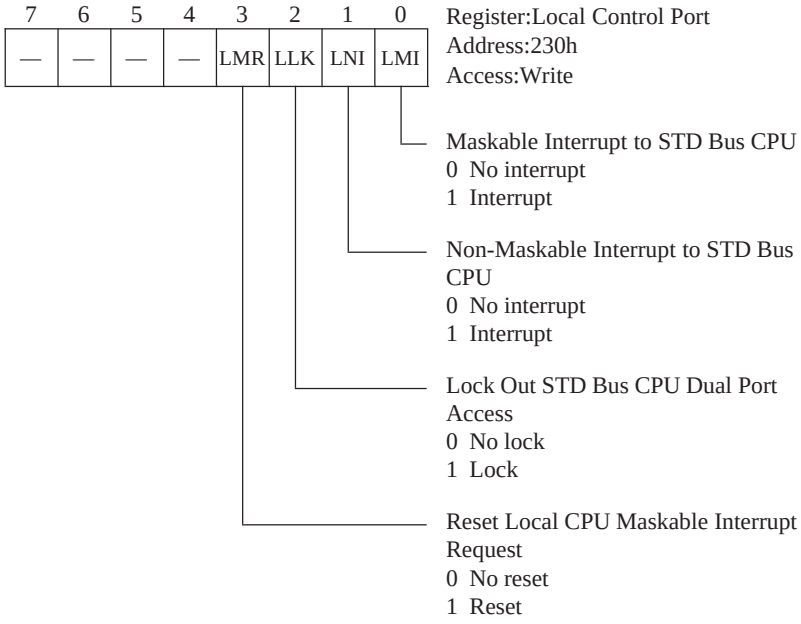



*Figure 3–2. Local Control Port Architecture.*

The *Local Control Port Maskable Interrupt (LMI)* bit (bit 0) is programmed by the local CPU to generate a maskable interrupt to the STD bus CPU. The LMI bit is connected to the STD bus INTRQ* (pin 44), INTRQ1* (pin 37), or INTRQ2* (pin 50) through jumper selection. Before the LMI bit is programmed, the STD bus CPU must program the SMR bit of the STD bus control port with a logical 0. Once this is done, the local CPU generates a maskable interrupt to the STD bus CPU by writing a logical 1 to the LMI bit of the local control port. After a logical 1 is written by the local CPU, all other

writes to the LMI bit are ignored until the interrupt request is cleared by the STD bus CPU writing a logical 1 followed by a logical 0 to the SMR bit of the STD bus control port.

The *Local Bus Control Port Non-Maskable Interrupt (LNI)* bit (bit 1) is programmed by the local CPU to generate a non-maskable interrupt to the STD bus CPU. The local CPU generates a non-maskable interrupt to the STD bus CPU by writing a logical 1 followed by a logical 0 to the LNI bit. The ZT 8832 does not provide any status to the STD bus CPU that flags an active non-maskable interrupt request. If this is necessary for the application, a status word can be implemented in the dual port RAM.

The *Local Control Port Lock (LLK)* bit (bit 2) is programmed by the local CPU to prevent the STD bus CPU from accessing the dual port RAM. The local CPU arms Lock by programming the LLK bit with a logical 1. At this point, the STD bus CPU can still access the dual port RAM. It is not until the local CPU performs the first read, or a write of the dual port RAM, that the STD bus CPU is no longer granted access. Any attempt by the STD bus CPU to access dual port RAM is denied, suspending STD bus CPU operation until the local CPU programs LLK with a logical 0.

The *Local Control Port Maskable Interrupt Reset (LMR)* bit (bit 3) is programmed by the local CPU to reset an active maskable interrupt request generated by the STD bus CPU through the SMI bit of the STD bus control port. The local CPU must first program the LMR bit with a logical 0 to enable the STD bus CPU to generate a maskable interrupt. To reset an active maskable interrupt, the local CPU must program the LMR bit with a logical 1 followed by a logical 0.

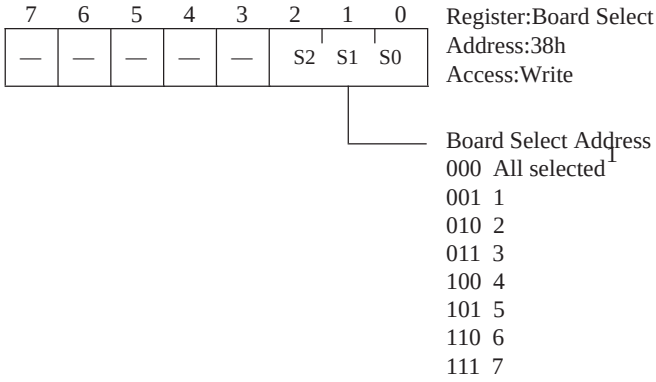
## BOARD SELECT OPTION

A ZT 8832 occupies 32 Kbytes of STD bus memory address space and 16 bytes of STD bus I/O address space. The memory is jumper selectable to any 32 Kbyte block and the I/O is jumper selectable to any of 16 possible locations. Systems using multiple ZT 8832s can map each one to a unique memory and I/O address range. However, in some systems this may not be possible because of limited STD bus resources. The board select option solves this problem by reducing the memory and I/O requirements of up to seven ZT 8832s to those of a single ZT 8832; up to 14 ZT 8832s to those of two ZT 8832s; and so on.

Follow the steps listed below to begin using the board select option. The mapping options discussed below are found in the jumper table beginning on page A-2.

1. Map the dual port RAM of each ZT 8832 to the same STD bus memory addressing range.
2. Map the STD bus control port, board select port, and interrupt status port of each ZT 8832 to the same STD bus I/O address range.
3. Jumper program a unique board select address for each ZT 8832. Please note that with three jumpers there are eight combinations. One combination is used to disable the board select feature, leaving seven unique combinations for up to seven boards.

After the above steps are completed, install the ZT 8832s into the STD bus card cage. The ZT 8832s power up *not selected.* This means that the dual port RAM, STD bus control port, and interrupt status port are not accessible by the STD bus CPU. The Board Select Port is the only port accessible. To begin communicating with a ZT 8832, the STD bus CPU must write the board select address of that ZT 8832 to the Board Select Port (see Figure 3-3). After the board select address is written, the STD bus CPU is free to communicate with the ZT 8832 as though there were no bank selection. To begin communicating with a different ZT 8832, the STD bus CPU simply writes the new board select address. This automatically switches out the currently enabled ZT 8832 and switches in the new one.




[1]Programming the All Selected address selects the ZT 8832 regardless of the jumper programmed board select address. This operation is useful for writing the same data to the dual port RAM or control port of all ZT 8832s at the same time.

**Warning:** Do not perform an STD bus read from the ZT 8832 dual port RAM or I/O if the All Selected option is programmed; this may damage the STD bus system.

*Figure 3–3. Board Select Port Architecture.*

## STD BUS INTERRUPTS

The ZT 8832 is capable of generating maskable and non-maskable interrupts to the STD bus CPU. The maskable interrupt is jumper selectable to the STD bus INTRQ* (pin 44), INTR1* (pin 37), or INTRQ2* (pin 50). The non-maskable interrupt is dedicated to the STD bus NMIRQ* (pin 46). This section discusses the issues surrounding the use of the ZT 8832 and STD bus interrupts.

The local CPU generates a non-maskable interrupt to the STD bus CPU by writing to the local control port. The ZT 8832 does not provide any status to the STD bus master to indicate that it generated the non-maskable interrupt. If it is necessary to distinguish between multiple sources of non-maskable interrupt, the application software can implement a status byte in the dual port RAM. In general, non-maskable interrupts are reserved for orderly shutdown of a system in response to a catastrophic event.

There are several issues surrounding the use of maskable interrupts. The first thing to determine is which STD bus interrupts are supported by the STD bus CPU. Most STD bus CPU boards support INTRQ*; many of the newer models also support INTRQ1* and INTRQ2*. For example, the ZT 8806/8807 supports INTRQ*, while the ZT 8808/8809 and ZT 8816/8817 (revision B and later) support INTRQ*, INTRQ1*, and INTRQ2*. If the STD bus CPU supports more than one interrupt request, the next thing to determine is which to use. The best choice is to use an interrupt request that is not shared with other STD bus boards. This greatly simplifies the application software. Once the STD bus interrupt request is selected, the ZT 8832 must be jumper configured to support it.

The simplest interrupt architecture is one in which the ZT 8832 does not share the interrupt with any other STD bus boards, including other ZT 8832s. For this architecture, an interrupt cycle is outlined below.

- The local CPU activates the STD bus interrupt request by writing to the local control port. Please note that the STD bus CPU must first remove the interrupt mask by writing to the STD bus control port.
- The STD bus CPU responds to the interrupt request by vectoring to the dedicated interrupt service routine.
- The interrupt service routine resets the interrupt request by writing to the STD bus control port. To prevent missing a subsequent interrupt, it is best to reset the request as early as possible.

The ZT 8832 provides an Interrupt Status Port if it is necessary to share the ZT 8832 interrupt with other STD bus boards, including other ZT 8832s. The Interrupt Status Port, shown in Figure 3-4, provides a means by which the STD bus CPU can determine, in general, if any of the shared resources generated an interrupt request, and specifically, if the ZT 8832 generated an interrupt request.

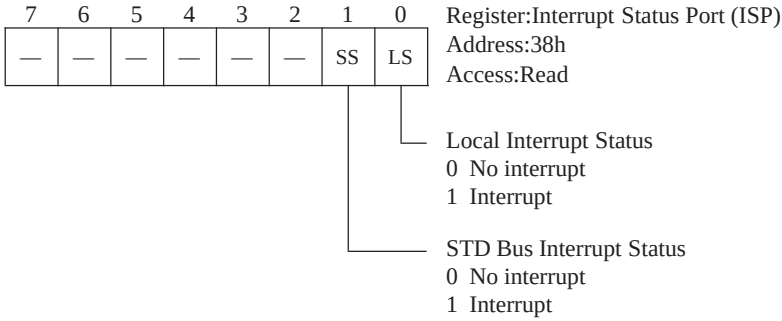



*Figure 3–4. Interrupt Status Port Architecture.*

The local interrupt status bit (bit 0) indicates the status of the ZT 8832 interrupt request to the STD bus CPU. The STD bus interrupt status bit (bit 1) indicates whether any of the interrupt sources sharing the STD bus interrupt request with the ZT 8832 has an interrupt pending, including the ZT 8832. The following steps outline an interrupt cycle for a shared STD bus interrupt configuration:

1. The local CPU activates the interrupt request by writing to the local control port. Note that the STD bus CPU must first remove the interrupt mask by writing to the STD bus control port.
2. The STD bus CPU responds to the interrupt request by vectoring to the interrupt service routine for the shared resources.
3. The interrupt service routine reads the Interrupt Status Port to determine if the ZT 8832 generated the request.
4. If the local status bit of the Interrupt Status Port is active, then the ZT 8832 has a request pending.
5. If the local status bit is not active and the STD bus status bit is active, then one of the other shared resources has an interrupt pending. If other ZT 8832s share the interrupt, the Interrupt Status Port of each one can be read to determine if a request is pending. If the board select option is in use, each ZT 8832 must be selected by writing the appropriate board address to the Board Select Port before the Interrupt Status Port becomes accessible.
6. The interrupt service routine resets a ZT 8832 request by writing to the STD bus control port. To prevent missing a subsequent interrupt, it is best to reset the request as early as possible. If the board select option is in use, the ZT 8832 must be selected before the STD bus control port is accessible.

7. This step applies only to systems with the STD bus CPU configured for edge triggered as opposed to level triggered interrupts. To avoid missing an interrupt request, the interrupt service routine must continue servicing the shared interrupt sources until the STD bus status bit of the ZT 8832 Interrupt Status Port is inactive. This prevents the possibility of missing a request because the edge occurred while servicing another request. If there is more than one ZT 8832 sharing an interrupt request, the STD bus status bit is redundant.

## RESET

The ZT 8832 is reset by any of the following events:

- Programming the STD bus control port with a 0Fh, followed by a logical 0, from the STD bus CPU. This resets the ZT 8832 only. The STD bus is not affected.
- Activating the pushbutton switch located on the ZT 8832. This resets the ZT 8832 only. The STD bus is not affected.
- Dropping Vcc applied to the ZT 8832 through pins 3 and 4 of the STD bus connector to below a typical value of 4.37 V (worst case specifications are 4.49 V maximum and 4.25 V minimum). This resets the ZT 8832 only. The STD bus is not affected.
- Failure to strobe the watchdog timer during stage 2 time delay. This is an optional source of reset. If the two-stage watchdog timer is enabled, it generates a non-maskable interrupt if it is not strobed within the first-stage time out period. If the non-maskable interrupt service routine does not strobe the watchdog timer within the second-stage time out period, a local reset is generated. This resets the ZT 8832 only. The STD bus is not affected.
- An active low on pin 47 (SYSRESET*) of the STD bus. The STD bus CPU typically drives SYSRESET* to a logical 0 when it is being reset. In response to a SYSRESET*, the ZT 8832 drives STD bus pin 48 (PBRESET*) to a logical 0 to prevent the STD bus CPU from coming out of reset and attempting an access to a ZT 8832 still in reset. An STD bus CPU is typically held in reset by an active PBRESET*. In a system with multiple ZT 8832s, PBRESET* is held active until the last ZT 8832 is out of reset and operational.

In response to the STD bus control port reset, the ZT 8832 enters a reset state for a variable time period. The ZT 8832 enters reset a maximum of 250 ns after a 0Fh is written to the control port and remains reset until a logical 0 is written to the control port. While it is

possible for the STD bus CPU to perform these writes as two sequential operations, the SBX expansion module interface requires a 15 µs reset pulse width.

In response to all other reset sources, the ZT 8832 enters a reset state that typically lasts 600 ms (worst-case specifications are 250 ms minimum and 1000 ms maximum). The following events occur during the reset period:

- The STD bus control port and local control port are reset to a logical 0 during power up, STD bus SYSRESET*, local pushbutton reset, and watchdog timer stage 2 timeout.
- If the Board Select Port is jumper enabled, the board is de-selected during power-up. The Board Select Port is unaffected by any other source of reset.
- The dual port RAM controller is reset by all of the above sources of reset. Data being transferred to or from the dual port RAM at the time of reset will be corrupted. If the STD bus CPU attempts a dual port RAM access while the controller is in reset, it is held off until the reset period is complete. This will cause problems in systems that have critical interrupt latencies, DMA latencies, asynchronous data transfers, or dynamic RAM that must be refreshed by the STD bus CPU.

Table 3-1 includes a list of devices affected by all sources of reset and a page number for a detailed discussion on the reset state of each.

Table 3-1
Devices Affected by Reset.

| Device | Page # |
|---|---|
| V40 CPU | 5-23 |
| V40 Configuration Registers | 6-12 |
| Counter/Timers | 7-14 |
| Interrupt Controller | 8-19 |
| DMA Controller | 9-17 |
| V40 Serial Port | 10-11 |
| 82050 Serial Port | 11-20 |
| Parallel I/O | 12-6 |
| Watchdog Timer | 13-5 |

Chapter 4

# APPLICATION EXAMPLES

**Contents** **Page**



## OVERVIEW

This chapter includes programming examples for initializing and managing several devices found on the ZT 8832. In most cases, the procedures will require some modification to meet the needs of specific applications. Each example includes a discussion of program objectives followed by an assembly language listing.

# EXAMPLE 1: V40 INITIALIZATION

## Objectives

The ZT 8832 is designed around the NEC 70208 (V40) microprocessor. This high integration microprocessor includes counter/timers, interrupt controller, DMA controller, serial controller, and wait state generator, in addition to a CPU with an 8088 architecture. The V40 also includes programmable configuration registers to provide flexibility when using these devices. Chapter 5 discusses the V40 configuration registers and the restrictions placed on these registers by the ZT 8832. The following program code, taken from the STD ROM software, initializes the V40 configuration registers.

## Program Code

```
;
```

```
                                EXAMPLE #1
                          PROGRAMMING ABSTRACT
```

```
;
;                       Ziatech Corporation
;                       San Luis Obispo, CA
;                            06/01/89

; THIS PROGRAMMING EXAMPLE ILLUSTRATES THE CODE USED BY
;  STD ROM TO INITIALIZE THE
;  V40 CONFIGURATION REGISTERS.


;
```

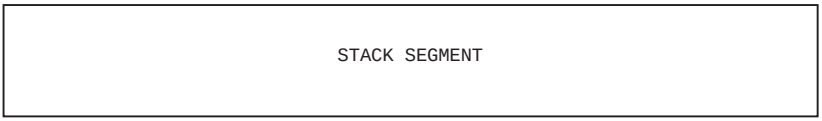
```
                                SYSTEM EQUATES
```

```
RESV1           EQU     0FFFFH          ; NEC RESERVED

; ON-CHIP PERIPHERAL CONNECTION REGISTER (OPCN)
OPCN            EQU     0FFFEH          ; OPCN I/O ADDRESS
OPCN_INIT       EQU     00000110B       ; SERIAL, IRQ2

; ON-CHIP PERIPHERAL SELECTION REGISTER (OPSEL)
OPSEL           EQU     0FFFDH          ; OPSEL I/O ADDRESS
OPSEL_INIT      EQU     00001110B       ; ENABLE SCU AND TCU

; ON-CHIP PERIPHERAL HIGH ORDER ADDRESS (OPHA)
OPHA            EQU     0FFFCH          ; OPHA I/O ADDRESS
OPHA_INIT       EQU     0               ; BOTTOM 256 BYTES

; DMA CONTROL UNIT
DULA            EQU     0FFFBH          ; DULA I/O ADDRESS
DULA_INIT       EQU     0D0H            ; DCU OFFSET ADDRESS
DCU             EQU     256*OPHA_INIT+DULA_INIT; DCU I/O ADDRESS

; INTERRUPT CONTROL UNIT
IULA            EQU     0FFFAH          ; IULA I/O ADDRESS
IULA_INIT       EQU     20H             ; ICU OFFSET ADDRESS
ICU             EQU     256*OPHA_INIT+IULA_INIT ; ICU I/O ADDRESS

; TIMER CONTROL UNIT
```

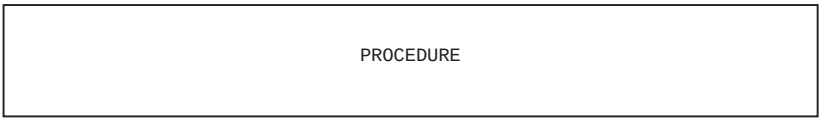
```
TULA            EQU     0FFF9H          ; TULA I/O ADDRESS
TULA_INIT       EQU     40H             ; TCU OFFSET ADDRESS
TCU             EQU     256*OPHA_INIT+TULA_INIT; TCU I/O ADDRESS

; SERIAL CONTROL UNIT
SULA            EQU     0FFF8H          ; SULA I/O ADDRESS
SULA_INIT       EQU     0B0H            ; SCU OFFSET ADDRESS
SCU             EQU     256*OPHA_INIT+SULA_INIT; SCU I/O ADDRESS

RESV2           EQU     0FFF7H          ; NEC RESERVED

; WAIT REQUEST CONTROL UNIT
WCY2            EQU     0FFF6H          ; WCY2 I/O ADDRESS
WCY2_INIT       EQU     00001000B       ; 2 DMA WAIT STATES
WCY1            EQU     0FFF5H          ; WCY1 I/O ADDRESS
WCY1_INIT       EQU     10000000B       ; 2 I/O WAIT STATES
WMB             EQU     0FFF4H          ; WMB I/O ADDRESS
WMB_INIT        EQU     0               ; 0 MEMORY WAIT STATES

RESV3           EQU     0FFF3H          ; NEC RESERVED

; REFRESH CONTROL UNIT
RFC             EQU     0FFF2H          ; RFC I/O ADDRESS
RFC_INIT        EQU     0               ; DISABLE REFRESH

RESV4           EQU     0FFF1H          ; NEC RESERVED

; TIMER AND CLOCK SELECTION
TCKS            EQU     0FFF0H          ; TCKS I/O ADDRESS
TCKS_INIT       EQU     0               ; INT CLOCK, DIV BY 2

;
```

MACRO DEFINITIONS

```
PUT     MACRO   SRCE,DATA               ; I/O WRITE MACRO
        MOV     DX,SRCE
        MOV     AL,DATA
        OUT     DX,AL
        ENDM

;
```




```
STACK                   SEGMENT         STACK
                DW      20 DUP (?)      ; UNINITIALIZED STACK
STACK_TOP       LABEL   WORD            ; TOP OF STACK
STACK                   ENDS

;
```




```
CODE                    SEGMENT         PARA

ASSUME  CS:CODE, SS:STACK, DS:NOTHING, ES:NOTHING

MAIN:
                MOV     BX,SEG STACK
                MOV     SS,BX
                MOV     SP,OFFSET STACK_TOP
                                        ; INITIALIZE CONFIG REGS
                PUT     OPCN,OPCN_INIT
                PUT     OPSEL,OPSEL_INIT
                PUT     OPHA,OPHA_INIT
                PUT     DULA,DULA_INIT
                PUT     IULA,IULA_INIT
                PUT     TULA,TULA_INIT
                PUT     SULA,SULA_INIT
                PUT     WCY2,WCY2_INIT
                PUT     WCY1,WCY1_INIT
                PUT     WMB,WMB_INIT
                PUT     RFC,RFC_INIT
                PUT     TCKS,TCKS_INIT
CODE            ENDS
                END     MAIN
```

# EXAMPLE 2: PERIPHERAL INITIALIZATION

## Objectives

The ZT 8832 contains many of the most commonly used peripherals found in STD bus applications. The following procedures show example initialization sequences for the devices listed below. While these procedures are general purpose in nature, they can be used as a starting point for most application software.

- Parallel Ports
- Light Emitting Diode (LED)
- 82050 Serial Controller
- V40 Serial Controller
- V40 Interrupt Controller
- V40 DMA Controller

## Program Code

```
;
```

```
                              EXAMPLE #2
                         PROGRAMMING ABSTRACT
```

```
;                       Ziatech Corporation
;                       San Luis Obispo, CA
;                            06/01/89

; THE FOLLOWING ARE EXAMPLE INITIALIZATION AND CONTROL PROCEDURES
;  FOR SEVERAL OF THE PERIPHERAL DEVICES FOUND ON THE ZT 8832.
;  WHILE THESE PROCEDURES ARE GENERAL PURPOSE IN NATURE, THEY CAN
;  BE USED AS A STARTING POINT FOR APPLICATION SOFTWARE.
;  PROCEDURES ARE INCLUDED FOR THE FOLLOWING DEVICES:
;
;       PARALLEL PORTS
;       LIGHT EMITTING DIODE (LED)
```

```
;       82050 SERIAL CONTROLLER
;       V40 SERIAL CONTROLLER AND BAUD RATE TIMER
;       V40 INTERRUPT CONTROLLER
;       V40 DMA CONTROLLER

;
```

SYSTEM EQUATES

```
; V40 CONFIGURATION REGISTERS
OPCN            EQU     0FFFEH          ; OPCN I/O ADDRESS
OPCN_INIT       EQU     00000010B       ; EXTRN IRQS
OPSEL           EQU     0FFFDH          ; OPSEL I/O ADDRESS
OPSEL_INIT_S    EQU     00001000B       ; ENABLE SCU
OPSEL_INIT_T    EQU     00000100B       ; ENABLE TCU
OPSEL_INIT_I    EQU     00000010B       ; ENABLE ICU
OPSEL_INIT_D    EQU     00000001B       ; ENABLE DCU
OPHA            EQU     0FFFCH          ; OPHA I/O ADDRESS
OPHA_INIT       EQU     0               ; BOTTOM 256 BYTES
DULA            EQU     0FFFBH          ; DULA I/O ADDRESS
DULA_INIT       EQU     0D0H            ; DCU OFFSET ADDRESS
IULA            EQU     0FFFAH          ; IULA I/O ADDRESS
IULA_INIT       EQU     20H             ; ICU OFFSET ADDRESS
TULA            EQU     0FFF9H          ; TULA I/O ADDRESS
TULA_INIT       EQU     40H             ; TCU OFFSET ADDRESS
SULA            EQU     0FFF8H          ; SULA I/O ADDRESS
SULA_INIT       EQU     0B0H            ; SCU OFFSET ADDRESS
TCKS            EQU     0FFF0H          ; TCKS I/O ADDRESS
TCKS_INIT       EQU     0               ; INT CLOCK, DIV BY 2

; PARALLEL PORT
PAR_PORT_0      EQU     200H            ; PARALLEL PORT 0 ADDRESS
PAR_PORT_1      EQU     210H            ; PARALLEL PORT 1 ADDRESS
PAR_PORT_2      EQU     220H            ; PARALLEL PORT 2 ADDRESS
PAR_PORT_INIT   EQU     0               ; INITIALIZE VALUE

; LED
PAR_PORT_LED    EQU     01000000B       ; PARALLEL PORT LED CTRL

; 82050 ACC SERIAL CONTROLLER
ACC_PORT        EQU     03F8H           ; 82050 ACC PORT ADDRESS
ACC_DIV1        EQU     ACC_PORT        ; DIVISOR LSB
ACC_DIV2        EQU     ACC_PORT+1      ; DIVISOR MSB
ACC_INTC        EQU     ACC_PORT+1      ; INTERRUPT CONTROL
ACC_LINC        EQU     ACC_PORT+3      ; LINE CONTROL
ACC_MODC        EQU     ACC_PORT+4      ; MODEM CONTROL
ACC_OUT2        EQU     00001000B       ; OUT2 BIT
ACC_DLAB        EQU     10000000B       ; DIVISOR LATCH SELECT
ACC_BAUD_LO     EQU     0CH             ; 9600 BAUD
ACC_BAUD_HI     EQU     0               ;
ACC_INTC_INIT   EQU     0               ; MASK INTERRUPTS
ACC_LINC_INIT   EQU     00000011B       ; 8 DATA, 0 PARITY, 1 STOP
ACC_MODC_INIT   EQU     00000011B       ; ENABLE RTS AND DTR
```

```
; V40 SCU SERIAL CONTROLLER AND BAUD RATE TIMER
TCU_PORT        EQU     256*OPHA_INIT+TULA_INIT  ; TCU I/O ADDRESS
TCU_TMR1        EQU     TCU_PORT+1      ; TIMER 1 CONTROL
TCU_TMR1_BD1    EQU     26              ; SCU BAUD OF 9600
TCU_TMR1_BD2    EQU     0               ;
TCU_MODE        EQU     TCU_PORT+3      ; TCU MODE ADDRESS
TCU_MODE_INIT   EQU     01110110B       ; BIN, SQ, 2 BYTE, T1
SCU_PORT        EQU     256*OPHA_INIT+SULA_INIT ; SCU I/O ADDRESS
SCU_CMND        EQU     SCU_PORT+1      ; COMMAND ADDRESS
SCU_CMND_INIT   EQU     00000101B       ; ENABLE TXD AND RXD
SCU_MODE        EQU     SCU_PORT+2      ; MODE ADDRESS
SCU_MODE_INIT   EQU     01001110B       ; DIV BY 16, 8 BT, 0 PY, 1 SP
SCU_MASK        EQU     SCU_PORT+3      ; MASK ADDRESS
SCU_MASK_INIT   EQU     00000011B       ; MASK INTERRUPTS

; V40 INTERRUPT CONTROLLER
ICU_PORT        EQU     256*OPHA_INIT+IULA_INIT ; ICU I/O ADDRESS
ICU_IIW1        EQU     ICU_PORT+0      ; INIT WORD 1 ADDRESS
ICU_IIW1_INIT   EQU     00010011B       ; EDGE TRIG, IIW4 ENABLE
ICU_IIW2        EQU     ICU_PORT+1      ; INIT WORD 2 ADDRESS
ICU_IIW2_INIT   EQU     00001000B       ; BASE VECTOR TYPE OF 8
ICU_IIW4        EQU     ICU_PORT+1      ; INIT WORD 4 ADDRESS
ICU_IIW4_INIT   EQU     00000001B       ; NORMAL NEST, FI COMMAND
ICU_MASK        EQU     ICU_PORT+1      ; MASK ADDRESS
ICU_MASK_INIT   EQU     11111111B       ; MASK COUNTER/TIMER 0
                                        ; MASK V40 SERIAL PORT
                                        ; MASK WATCHDOG TIMER
                                        ; MASK COUNTER/TIMER 1
                                        ; MASK SBX REQUEST 0
                                        ; MASK SBX REQUEST 1
                                        ; MASK 82050 SERIAL PORT
                                        ; MASK STD BUS CTRL PORT
                                        ; MASK FRONTPLANE J3-8
                                        ; MASK FRONTPLANE J3-10

; V40 DMA CONTROLLER
DCU_PORT        EQU     256*OPHA_INIT+DULA_INIT ; DCU I/O ADDRESS
DCU_DICM        EQU     DCU_PORT+0      ; DICM ADDRESS
DCU_DICM_INIT   EQU     00000001B       ; RESET DCU
DCU_DCH         EQU     DCU_PORT+1      ; DCH ADDRESS
DCU_DCH_INIT    EQU     0               ; CHANNEL 0
DCU_DBC         EQU     DCU_PORT+2      ; DCU BASE COUNT
DCU_DBA_OFF     EQU     DCU_PORT+4      ; DCU BASE ADDRESS
DCU_DBA_SEG     EQU     DCU_PORT+6      ;  OFFSET AND SEGMENT
DCU_DDC         EQU     DCU_PORT+8      ; DCU DEVICE CONTROL
DCU_DDC_INIT    EQU     0               ; ENABLE DMA
DCU_DMD         EQU     DCU_PORT+0AH    ; DCU MODE
DCU_DMD_WRTE    EQU     01000100B       ; SNGL, INC, WRITE
DCU_DMD_READ    EQU     01001000B       ; SNGL, INC, READ
DCU_DMK         EQU     DCU_PORT+0FH    ; DCU MASK
DCU_DMK_INIT    EQU     0EH             ; NOT MASKED

;
```

```
                        MACRO DEFINITIONS
```

```
PUT     MACRO   SRCE,DATA               ; I/O WRITE MACRO
        MOV     DX,SRCE
        MOV     AL,DATA
        OUT     DX,AL
        ENDM

;
```

```
                          STACK SEGMENT
```

```
STACK                   SEGMENT         STACK
                DW      20 DUP (?)      ; UNINITIALIZED STACK
STACK_TOP       LABEL   WORD            ; TOP OF STACK
STACK                   ENDS

;
```

```
                          DATA SEGMENT
```

```
; THIS DATA SEGMENT PROVIDES A DATA BUFFER FOR DMA OPERATIONS
;  BETWEEN THE SBX EXPANSION MODULE I/O AND LOCAL RAM.

DATA                    SEGMENT
DMA_BUF         DB      256 DUP (?)        ; DMA DATA BUFFER
DATA                    ENDS

CODE                    SEGMENT         PARA
ASSUME  CS:CODE, SS:STACK, DS:DATA, ES:NOTHING

MAIN:
                MOV     BX,SEG STACK
                MOV     SS,BX
                MOV     SP,OFFSET STACK_TOP
                MOV     BX,SEG DATA
                MOV     DS,BX

;
```

PARALLEL PORT PROCEDURE

```
;
; THE ZT 8832 INCLUDES THREE PARALLEL PORTS.  THE PARALLEL PORT
;  OUTPUTS ARE ENABLED AND DISABLED WITH THE OUT2 BIT OF THE
;  82050 SERIAL PORT TO PREVENT GLITCHES DURING POWER-UP.  THE
;  POWER-UP STATE OF OUT2 IS A LOGICAL 0 THAT DISABLES THE
;  PARALLEL PORT OUTPUTS.  PASSIVE TERMINATION MAINTAINS A TTL
;  HIGH ON THE PARALLEL I/O SIGNALS AT CONNECTOR J1 WHEN THE
;  PARALLEL PORTS ARE DISABLED.  BEFORE THE PARALLEL PORTS CAN
;  BE USED, THE OUT2 BIT MUST BE PROGRAMMED.  SINCE THE PARALLEL
;  PORT BITS DO NOT POWER UP INITIALIZED, IT IS RECOMMENDED THAT
;  THEY BE PROGRAMMED WITH A LOGICAL 0 BEFORE BEING ENABLED.
;  THE PARALLEL PORTS ARE INVERTING, SO PROGRAMMING THEM WITH
;  A LOGICAL 0 MAINTAINS THE TTL HIGH ON THE I/O SIGNALS AT
;  CONNECTOR J1.  THIS INITIALIZATION ALSO ENSURES THAT THERE
;  IS NO CONTENTION BETWEEN THE PARALLEL PORT I/O SIGNALS THAT
;  ARE DRIVEN AS INPUTS FROM EXTERNAL DEVICES.  AFTER THIS
;  INITIALIZATION SEQUENCE, THE APPLICATION SOFTWARE IS FREE TO
;  COMMUNICATE  WITH THE PARALLEL PORTS USING STANDARD INPUT
;  AND OUTPUT INSTRUCTIONS, INCLUDING STRING INPUT AND OUTPUT.
;
; INPUTS:       NONE
; OUTPUTS:      PARALLEL PORTS ARE INITIALIZED AND ENABLED
; CALLS:        NONE
; DESTROYS:     FLAGS
PAR_INIT        PROC
                                ; PRESERVE REGISTER STATUS
                PUSH    AX
                PUSH    DX
                                ; INITIALIZE PARALLEL PORTS
                PUT     PAR_PORT_0,PAR_PORT_INIT
                PUT     PAR_PORT_1,PAR_PORT_INIT
                PUT     PAR_PORT_2,PAR_PORT_INIT
                                ; ENABLE PARALLEL PORTS
                MOV     DX,ACC_MODC
                IN      AL,DX
                OR      AL,ACC_OUT2
                OUT     DX,AL
                                ; RESTORE REGISTER STATUS
                POP     AX
                POP     DX
PAR_INIT        ENDP

;
```

LED PROCEDURE

```
; THE LED CAN BE TURNED ON AND OFF UNDER SOFTWARE CONTROL.  THIS
;  IS A VALUABLE STATUS INDICATOR, ESPECIALLY DURING THE DEBUG
;  PHASE OF APPLICATION DEVELOPMENT.  THE FOLLOWING CODE SHOWS
;  THE PROCEDURE FOR ARMING THE LED AND TURNING IT ON AND OFF.
;
; INPUTS:       NONE
; OUTPUTS:      LED IS ARMED
; CALLS:        NONE
; DESTROYS:     FLAGS

LED_INIT        PROC
                                ; PRESERVE REGISTER STATUS
                PUSH    AX
                PUSH    DX
                                ; ARM THE LED
                PUT     PAR_PORT_0,PAR_PORT_INIT
                PUT     PAR_PORT_1,PAR_PORT_INIT
                PUT     PAR_PORT_2,PAR_PORT_INIT
                MOV     DX,ACC_MODC
                IN      AL,DX
                OR      AL,ACC_OUT2
                OUT     DX,AL
                                ; RESTORE REGISTER STATUS
                POP     DX
                POP     AX
LED_INIT        ENDP

; ONCE THE LED IS ARMED, IT IS TURNED ON WITH THE FOLLOWING
;  PROCEDURE.
;
; INPUTS:       NONE
; OUTPUTS:      LED IS TURNED ON
; CALLS:        NONE
; DESTROYS:     FLAGS

LED_ON          PROC
                                ; PRESERVE REGISTER STATUS
                PUSH    AX
                PUSH    DX
                                ; TURN ON THE LED
                MOV     DX,PAR_PORT_2
                IN      AL,DX
                OR      AL,PAR_PORT_LED
                OUT     DX,AL
                                ; RESTORE REGISTER STATUS
                POP     DX
                POP     AX
LED_ON         ENDP

; ONCE THE LED IS ARMED AND TURNED ON, IT IS TURNED OFF WITH THE
;  FOLLOWING PROCEDURE.
```

```
;
; INPUTS:       NONE
; OUTPUTS:      LED IS TURNED OFF
; CALLS:        NONE
; DESTROYS:     FLAGS

LED_OFF         PROC
                                ; PRESERVE REGISTER STATUS
                PUSH    AX
                PUSH    DX
                                ; TURN OFF THE LED
                MOV     DX,PAR_PORT_2
                IN      AL,DX
                AND     AL,NOT PAR_PORT_LED
                OUT     DX,AL
                                ; RESTORE THE REGISTER STATUS
                POP     DX
                POP     AX
LED_OFF         ENDP

;
```

82050 SERIAL CONTROLLER PROCEDURE

```
; THE 82050 SERIAL CONTROLLER IS A PC COMPATIBLE ASYNCHRONOUS
;  COMMUNICATION CHANNEL SUPPORTING JUMPER SELECTABLE DCE/DTE
;  AND RS-232/RS-485 CONFIGURATION.  THIS PROCEDURE INITIALIZES
;  THE 82050 WITH THE FOLLOWING PARAMETERS:
;
;       BAUD RATE       9600
;       DATA BITS       8
;       STOP BITS       1
;       PARITY          NONE
;       INTERRUPTS      DISABLED
;
;  ONCE THE SERIAL PORT IS INITIALIZED AS SHOWN BELOW, IT IS
;  READY TO TRANSMIT AND RECEIVE CHARACTERS.
;
; INPUTS:       NONE
; OUTPUTS:      SERIAL PORT INITIALIZED
; CALLS:        NONE
; DESTROYS:     FLAGS

ACC_INIT        PROC
                                ; PRESERVE REGISTER STATUS
                PUSH    AX
                PUSH    DX
                                ; SET BAUD RATE
                PUT     ACC_LINC,ACC_DLAB
                PUT     ACC_DIV1,ACC_BAUD_LO
                PUT     ACC_DIV2,ACC_BAUD_HI
                                ; SET SERIAL CHARACTER ATTRIBUTES
                PUT     ACC_LINC,ACC_LINC_INIT
                                ; SET MODEM HANDSHAKE
```

```
                PUT     ACC_MODC,ACC_MODC_INIT
                                ; MASK INTERRUPTS
                PUT     ACC_INTC,ACC_INTC_INIT
                                ; RESTORE REGISTER CONTENTS
                POP     DX
                POP     AX
ACC_INIT        ENDP

;
```

V40 SERIAL CONTROLLER PROCEDURE

```
; THE V40 SERIAL CONTROLLER IS AN 8251 COMPATIBLE ASYNCHRONOUS
;  COMMUNICATION CHANNEL CONFIGURED AS THREE-WIRE (TRANSMIT
;  DATA, RECEIVE DATA, AND GROUND) RS-232 DTE.  THE PROCEDURE
;  SHOWN BELOW INITIALIZES THE V40 SCU WITH THE FOLLOWING
;  PARAMETERS:
;
;       SCU AND TCU ENABLED
;       SCU BASE ADDRESS OF 00B0H
;       TCU BASE ADDRESS OF 0040H
;       BAUD RATE       9600
;       DATA BITS       8
;       STOP BITS       1
;       PARITY          NONE
;       INTERRUPTS      DISABLED
;
;  ONCE THE SERIAL PORT IS INITIALIZED AS SHOWN BELOW, IT IS
;  READY TO TRANSMIT AND RECEIVE CHARACTERS.
;
; INPUTS:       NONE
; OUTPUTS:      SERIAL PORT INITIALIZED
; CALLS:        NONE
; DESTROYS:     FLAGS

SCU_INIT        PROC
                                ; PRESERVE REGISTER STATUS
                PUSH    AX
                PUSH    DX
                                ; INITIALIZE V40 CONFIG REGS
                PUT     OPCN,OPCN_INIT
                PUT     OPSEL,OPSEL_INIT_S+OPSEL_INIT_T
                PUT     OPHA,OPHA_INIT
                PUT     TULA,TULA_INIT
                PUT     SULA,SULA_INIT
                PUT     TCKS,TCKS_INIT
                                ; SET BAUD RATE
                PUT     TCU_MODE,TCU_MODE_INIT
                PUT     TCU_TMR1,TCU_TMR1_BD1
                PUT     TCU_TMR1,TCU_TMR1_BD2
                                ; SET SERIAL CHARACTER ATTRIBUTES
                PUT     SCU_MODE,SCU_MODE_INIT
                                ; ENABLE TXD AND RXD
```

```
                PUT     SCU_CMND,SCU_CMND_INIT
                                ; MASK INTERRUPTS
                PUT     SCU_MASK,SCU_MASK_INIT
                                ; RESTORE REGISTER STATUS
                POP     DX
                POP     AX
SCU_INIT        ENDP

;
```

V40 INTERRUPT CONTROLLER PROCEDURE

```
; THE V40 INTERRUPT CONTROLLER IS USED TO SERVICE ASYNCHRONOUS
;  EVENTS SUCH AS A COUNTER/TIMER TIMEOUT, DATA TRANSFERS
;  THROUGH A SERIAL LINK, AND DATA TRANSFERS THROUGH DUAL PORT
;  RAM.  THE EXAMPLE CODE IS A GENERAL PURPOSE INITIALIZATION
;  PROCEDURE THAT PROGRAMS THE FOLLOWING:
;
;       ICU ENABLED
;       BASE ADDRESS OF 0020H
;       EDGE TRIGGERED
;       BASE VECTOR OF TYPE 8
;       NORMAL NESTING
;       FI COMMAND
;       MASK ALL INPUTS
;
;  IN ADDITION TO THE INITIALIZATION PROCEDURE PROVIDED BELOW,
;  THE APPLICATION SOFTWARE MUST DEVELOP THE NECESSARY INTERRUPT
;  SERVICE ROUTINES, STORE THE ADDRESS OF THE INTERRUPT SERVICE
;  ROUTINES AT THE APPROPRIATE TYPE ADDRESS, UNMASK THE SUPPORTED
;  REQUESTS THROUGH THE ICU MASK REGISTER, AND ENABLE V40
;  INTERRUPTS WITH THE ENABLE INTERRUPT INSTRUCTION.  AFTER THESE
;  STEPS ARE COMPLETE, THE INTERRUPTING PERIPHERALS WILL BE
;  SERVICED ON DEMAND.
;
; INPUTS:       NONE
; OUTPUTS:      INTERRUPT CONTROLLER INITIALIZED
; CALLS:        NONE
; DESTROYS:     FLAGS

ICU_INIT        PROC
                                ; PRESERVE REGISTER CONTENTS
                PUSH    AX
                PUSH    DX
                                ; INITIALIZE V40 CONFIG REGS
                PUT     OPCN,OPCN_INIT
                PUT     OPSEL,OPSEL_INIT_I
                PUT     OPHA,OPHA_INIT
                PUT     IULA,IULA_INIT
                                ; INITIALIZE PIC
                PUT     ICU_IIW1,ICU_IIW1_INIT
                PUT     ICU_IIW2,ICU_IIW2_INIT
                PUT     ICU_IIW4,ICU_IIW4_INIT
```

```
                PUT     ICU_MASK,ICU_MASK_INIT
                                ; RESTORE THE REGISTER STATUS
                POP     DX
                POP     AX
ICU_INIT        ENDP

;
```

V40 DMA CONTROLLER PROCEDURE

```
; THE V40 DMA CONTROLLER IS USED FOR HIGH SPEED DATA TRANSFERS
;  BETWEEN AN SBX EXPANSION MODULE THAT SUPPORTS DMA AND EITHER
;  DUAL PORT OR LOCAL RAM.  THE APPROXIMATE DATA RATE FOR TRANS-
;  FERS BETWEEN THE EXPANSION MODULE AND RAM USING STANDARD INPUT
;  AND OUTPUT INSTRUCTIONS IS 235 KBYTES PER SECOND.  USING DMA,
;  THE TRANSFER RATE IS INCREASED TO APPROXIMATELY 1.3 MBYTES PER
;  SECOND, FOR A PERFORMANCE INCREASE OF OVER 80 PERCENT.  THE
;  FIRST PROCEDURE IS A GENERAL PURPOSE INITIALIZATION ROUTINE
;  THAT PROGRAMS THE FOLLOWING:
;
;       DCU ENABLED
;       BASE ADDRESS OF 00D0H
;       DMA ENABLED
;
; INPUTS:       NONE
; OUTPUTS:      DMA CONTROLLER INITIALIZED
; CALLS:        NONE
; DESTROYS:     FLAGS

DCU_INIT        PROC
                                ; PRESERVE REGISTER CONTENTS
                PUSH    AX
                PUSH    DX
                                ; INITIALIZE V40 CONFIG REGS
                PUT     OPSEL,OPSEL_INIT_D
                PUT     OPHA,OPHA_INIT
                PUT     DULA,DULA_INIT
                                ; INITIALIZE DMA CONTROLLER
                PUT     DCU_DICM,DCU_DICM_INIT
                PUT     DCU_DCH,DCU_DCH_INIT
                PUT     DCU_DDC,DCU_DDC_INIT

                                ; RESTORE THE REGISTER STATUS
                POP     DX
                POP     AX
DCU_INIT        ENDP

; ONCE THE DMA CONTROLLER IS INITIALIZED, IT IS PROGRAMMED FOR
;  DMA WRITE TRANSFERS WITH THE FOLLOWING PROCEDURE.  A DMA
;  WRITE DEFINES THE SBX I/O AS THE DATA SOURCE AND THE RAM
;  MEMORY AS THE DESTINATION.  NOTE THAT EITHER DUAL PORT OR
;  LOCAL RAM MAY BE USED AS RAM MEMORY.  THE FOLLOWING
;  PARAMETERS ARE SELECTED:
```

```
;
;       SINGLE BYTE TRANSFERS
;       INCREMENT MEMORY ADDRESS
;       SBX I/O TO RAM OPERATION
;
; INPUTS:       NONE
; OUTPUTS:      SBX I/O TO RAM MEMORY TRANSFERS ARMED
; CALLS:        NONE
; DESTROYS:     FLAGS

DCU_WRITE       PROC
                                ; PRESERVE REGISTER STATUS
                PUSH    AX
                PUSH    BX
                PUSH    CX
                PUSH    DX
                                ; SET UP ADDRESS REGISTERS
                MOV     AX,SEG DATA
                MOV     DX,OFFSET DMA_BUF
                MOV     BL,AH
                MOV     CL,4
                SHL     AX,CL
                SHR     BL,CL
                ADD     DX,AX
                ADC     BL,0
                MOV     AX,DX
                MOV     DX,DCU_DBA_OFF
                OUT     DX,AX
                MOV     AX,BX
                MOV     DX,DCU_DBA_SEG
                OUT     DX,AL
                                ; SET UP COUNT REGISTERS
                MOV     AX,SIZE DMA_BUF-1
                MOV     DX,DCU_DBC
                OUT     DX,AX
                                ; SET UP MODE AND ENABLE
                PUT     DCU_DMD,DCU_DMD_WRTE
                PUT     DCU_DMK,DCU_DMK_INIT
                                ; RESTORE REGISTER STATUS
                POP     DX
                POP     CX
                POP     BX
                POP     AX
DCU_WRITE       ENDP

; ONCE THE DMA CONTROLLER IS INITIALIZED, IT IS PROGRAMMED FOR
;  DMA READ TRANSFERS WITH THE FOLLOWING PROCEDURE.  A DMA
;  READ DEFINES THE RAM MEMORY AS THE DATA SOURCE AND THE SBX
;  I/O AS THE DESTINATION.  NOTE THAT EITHER DUAL PORT OR LOCAL
;  RAM MAY BE USED AS RAM MEMORY.  THE FOLLOWING PARAMETERS ARE
;  SELECTED:
;
;       SINGLE BYTE TRANSFERS
;       INCREMENT MEMORY ADDRESS
;       RAM TO SBX I/O OPERATION
;
; INPUTS:       NONE
; OUTPUTS:      RAM MEMORY TO SBX I/O TRANSFERS ARMED
```

```
; CALLS:        NONE
; DESTROYS:     FLAGS

DCU_READ        PROC
                                ; PRESERVE REGISTER STATUS
                PUSH    AX
                PUSH    BX
                PUSH    CX
                PUSH    DX
                                ; SET UP ADDRESS REGISTERS
                MOV     AX,SEG DATA
                MOV     DX,OFFSET DMA_BUF
                MOV     BL,AH
                MOV     CL,4
                SHL     AX,CL
                SHR     BL,CL
                ADD     DX,AX
                ADC     BL,0
                MOV     DX,DCU_DBA_OFF
                MOV     AX,DX
                OUT     DX,AX
                MOV     DX,DCU_DBA_SEG
                MOV     AX,BX
                OUT     DX,AL
                                ; SET UP COUNT REGISTERS
                MOV     AX,SIZE DMA_BUF
                MOV     DX,DCU_DBC
                OUT     DX,AX
                                ; SET UP MODE AND ENABLE
                PUT     DCU_DMD,DCU_DMD_READ
                PUT     DCU_DMK,DCU_DMK_INIT
                                ; RESTORE REGISTER STATUS
                POP     DX
                POP     CX
                POP     BX
                POP     AX
                                ; RESTORE REGISTER STATUS
                POP     DX
                POP     CX
                POP     AX
DCU_READ        ENDP

CODE            ENDS
                END     MAIN
```

# EXAMPLE 3: WATCHDOG TIMER

## Objectives

A watchdog timer is useful in applications in which a microprocessor controls a physical process that may be damaged if the microprocessor fails to function as programmed. The ZT 8832 includes a jumper selectable watchdog timer which monitors operation of the V40 and initiates corrective action if necessary.

As discussed in Chapter 13, the watchdog timer has two stages. In normal operation, the application software strobes the watchdog timer at a periodic rate less than the stage 1 timeout. If this strobe does not take place, it is because the V40 is not operating as programmed. If this happens, stage 1 times out and generates a non-maskable interrupt.

The service routine designed to handle the non-maskable interrupt is application specific. This example simply sets a flag in battery-backed RAM to indicate the system failure, and loops until the stage 2 timeout resets the ZT 8832. While not implemented in this example, the reset can be prevented by the non-maskable interrupt service routine strobing the watchdog timer before the stage 2 timeout.

## Program Code

```
;
```

```
                                EXAMPLE #3
                            PROGRAMMING ABSTRACT
```

```
;                       Ziatech Corporation
;                       San Luis Obispo, CA
;                           06/01/89

; THIS PROGRAMMING EXAMPLE ILLUSTRATES THE CODE USED TO ARM AND
;  STROBE THE WATCHDOG TIMER.  ALSO INCLUDED IS A NON-MASKABLE
;  INTERRUPT SERVICE ROUTINE THAT FLAGS A WATCHDOG TIMEOUT IN
;  DUAL PORT RAM.  NOT SHOWN IS THE ZT 8832 STARTUP CODE THAT
;  TESTS THE FLAG LOCATION TO SEE WHETHER THE RESET WAS CAUSED
;  BY A WATCHDOG TIMEOUT OR BY ANOTHER SOURCE OF RESET.

;
```

```
                                SYSTEM EQUATES
```

```
; 82050 SERIAL PORT
SER_PORT        EQU     03F8H           ; 82050 BASE PORT
SER_PORT_CTRL   EQU     SER_PORT+4      ; MODEM CONTROL PORT
SER_PORT_OUT2   EQU     00001000B       ; OUT2 BIT USED FOR
                                        ;  PARALLEL PORT OUTPUT
                                        ;  ENABLE
; PARALLEL PORT

PAR_PORT_0      EQU     200H            ; PARALLEL PORT 0 ADDRESS
PAR_PORT_1      EQU     210H            ; PARALLEL PORT 1 ADDRESS
PAR_PORT_2      EQU     220H            ; PARALLEL PORT 2 ADDRESS
PAR_PORT_INIT   EQU     0               ; INITIALIZE VALUE
PAR_PORT_WD     EQU     10000000B       ; WATCHDOG STROBE BIT

;
```

```
                        STACK SEGMENT
```

```
STACK                   SEGMENT         STACK
                DW      20 DUP (?)      ; UNINITIALIZED STACK
STACK_TOP       LABEL   WORD            ; TOP OF STACK
STACK                   ENDS

;
```

```
                        DATA SEGMENT
```

```
; A MULTIPLE CHARACTER FLAG IS USED TO REDUCE THE CHANCE THAT
;  THE ZT 8832 STARTUP CODE (NOT SHOWN HERE) WILL FALSELY DETECT
;  A WATCHDOG TIMER RESET AFTER A POWER CYCLE.

DATA                    SEGMENT
WATCHDOG_FLAG   DB      'WATCHDOG TIMEOUT' ; WATCHDOG FLAG MESSAGE
DATA                    ENDS

;
```

```
                        DUALPORT SEGMENT
```

```
; THE DUAL PORT RAM IS MAPPED AT 8000H IN THE LOCAL CPU MEMORY
;  ADDRESS SPACE.

DUALPORT                SEGMENT AT 8000H
WATCHDOG_STAT   DB      16 DUP (?)      ; DUAL PORT WATCHDOG FLAG
DUALPORT                ENDS

;
```

```
                    INTERRUPT POINTERS SEGMENT
```

```
INT_POINT               SEGMENT AT 0
                ORG     0
TYPE_0          DD      ?               ; DIV BY ZERO (NOT USED)
```

```
TYPE_1          DD      ?               ; SINGLE STEP (NOT USED)
TYPE_2          DD      ?               ; NON-MASKABLE INTERRUPT
INT_POINT       ENDS
```

PROCEDURES

```
CODE                    SEGMENT         PARA
ASSUME  CS:CODE, SS:STACK, DS:DATA, ES:DUALPORT

; THE WATCHDOG TIMER GENERATES A NON-MASKABLE INTERRUPT IF ARMED
;  AND ALLOWED TO TIME OUT.  THE TASKS PERFORMED BY THE SERVICE
;  ROUTINE ARE VERY APPLICATION SPECIFIC.  THIS EXAMPLE SIMPLY
;  SETS A FLAG IN DUAL PORT RAM AND LOOPS UNTIL RESET.  SINCE THE
;  SERVICE ROUTINE DOES NOT STROBE THE WATCHDOG TIMER, THE
;  SECOND STAGE WILL TIME OUT AND GENERATE A LOCAL RESET.  THE
;  ZT 8832 STARTUP CODE (NOT SHOWN HERE) CAN TEST THE WATCHDOG
;  FLAG TO DETERMINE WHETHER OR NOT A WATCHDOG TIMEOUT HAS
;  OCCURED, AND CAN THEN TAKE THE NECESSARY ACTION.
;
; INPUTS:       NON-MASKABLE INTERRUPT REQUEST
; OUTPUTS:      WATCHDOG_FLAG SET
; CALLS:        NONE
; DESTROYS:     ALL

WATCHDOG_NMI            PROC
                MOV     AX,SEG DATA
                MOV     DS,AX
                MOV     SI,OFFSET WATCHDOG_FLAG
                MOV     AX,SEG DUALPORT
                MOV     ES,AX
                MOV     DI,OFFSET WATCHDOG_STAT
                CLD
                MOV     CX,16
            REP MOVSB
;LP1:             JMP     LP1

WATCHDOG_NMI    ENDP

; THE FOLLOWING PROCEDURE STROBES THE WATCHDOG TIMER.  THIS
;  PROCEDURE MUST BE CALLED BY THE MAIN PROCEDURE AT A PERIODIC
;  RATE LESS THAN THE STAGE 1 DELAY OF THE WATCHDOG TIMER.  THE
;  DEFAULT STAGE 1 DELAY IS 60 MILLISECONDS MINIMUM.
;
; INPUTS:       ASSUMES THE PARALLEL PORTS ARE ENABLED
; OUTPUTS:      STROBES THE WATCHDOG TIMER
; CALLS:        NONE
; DESTROYS:     FLAGS

WATCHDOG_STB    PROC
                PUSH    AX              ; PRESERVE REGISTER STATUS
                PUSH    DX
                MOV     DX,PAR_PORT_2   ; WRITE WATCHDOG BIT LOW
                IN      AL,DX
```

```
                AND     AL, NOT PAR_PORT_WD
                OUT     DX,AL
                OR      AL,PAR_PORT_WD  ; WRITE WATCHDOG BIT HIGH
                OUT     DX,AL
                POP     DX              ; RESTORE REGISTER STATUS
                POP     AX
                RET                     ; EXIT
WATCHDOG_STB    ENDP

; THE MAIN PROCEDURE INCLUDES A LOOP AND A CALL TO THE WATCHDOG
;  STROBE PROCEDURE.  THE LOOP REPRESENTS THE APPLICATION SOFT-
;  WARE.  IN NORMAL OPERATION, THE WATCHDOG STROBE PROCEDURE IS
;  CALLED AT A PERIODIC RATE LESS THAN THE WATCHDOG TIMER STAGE
;  1 DELAY OF 60 MILLISECONDS.  A LOOP COUNT OF 6000H IS SHORT
;  ENOUGH TO PREVENT THE TIMEOUT.  IF ANYTHING PREVENTS THE
;  APPLICATION SOFTWARE FROM CALLING THE WATCHDOG STROBE PROCEDURE
;  IN TIME, STAGE 1 TIMES OUT AND THE NMI SERVICE ROUTINE IS INVOKED.
;  THIS CAN BE SEEN BY CHANGING THE LOOP COUNT TO 8000H.
;
; INPUTS:       NONE
; OUTPUTS:      NONE
; CALLS:        WATCHDOG_STB
; DESTROYS:     ALL

MAIN:
                MOV     BX, SEG DATA    ; INITIALIZE SEGMENTATION
                MOV     DX,BX
                MOV     BX, SEG STACK
                MOV     SS,BX
                MOV     SP,OFFSET STACK_TOP

ASSUME  CS:CODE, SS:STACK, DS:INT_POINT, ES:DUALPORT

                PUSH    DS              ; INITIALIZE INT VECTOR
                MOV     BX, SEG INT_POINT
                MOV     DS,BX
                MOV     WORD PTR TYPE_2,OFFSET WATCHDOG_NMI
                MOV     WORD PTR TYPE_2+2,SEG CODE
                POP     DS

ASSUME  CS:CODE, SS:STACK, DS:DATA, ES:DUALPORT

                MOV     AL,PAR_PORT_INIT; INITIALIZE PARALLEL I/O
                MOV     DX,PAR_PORT_0   ;  SO THE J1 CONNECTOR
                OUT     DX,AL           ;  OUTPUTS DO NOT CHANGE
                MOV     DX,PAR_PORT_1   ;  WHEN PARALLEL PORTS ARE
                OUT     DX,AL           ;  ENABLED
                MOV     DX,PAR_PORT_2
                OUT     DX,AL
                MOV     DX,SER_PORT_CTRL; ENABLE PARALLEL I/O
                IN      AL,DX
                OR      AL,SER_PORT_OUT2
                OUT     DX,AL
                MOV     DX,PAR_PORT_2   ; ARM WATCHDOG TIMER
                IN      AL,DX
                OR      AL,PAR_PORT_WD
                OUT     DX,AL
LP2:            MOV     CX,6000H        ; SIMULATED PROGRAM LOOP
```

```
LP3:            LOOP    LP3
                CALL    WATCHDOG_STB    ; STROBE WATCHDOG TIMER
                JMP     LP2             ; REPEAT
CODE            ENDS
                END     MAIN
```

Chapter 5

# PROCESSOR DESCRIPTION (V40)

**Contents** **Page**

## OVERVIEW

The NEC 70208, commonly known as the V40, is a CMOS microprocessor with a 16-bit internal and 8-bit external data bus structure. The V40 instruction set includes all of the instructions of the 8088 and 80188 microprocessors, plus a few more. The added instructions include string I/O, expanded rotate and shift, bit and nibble manipulation, BCD arithmetic, and 8080 emulation mode.

The V40 contains several peripherals frequently used in STD bus applications. These peripherals include a serial controller, interrupt controller, direct memory access (DMA) controller, counter/timers, and a programmable wait-state generator.

This chapter divides the V40 microprocessor into functional blocks and presents an overview of each. More detailed descriptions of the programmable functional blocks are found in subsequent chapters.

## ZT 8832 SPECIFICS

The V40 includes a dynamic RAM (DRAM) refresh controller for applications supporting DRAM devices. Since the ZT 8832 contains only static RAM, the refresh controller is not needed.

The V40 includes four DMA channels. The ZT 8832 makes use of one of these channels to provide high speed data transfers between I/O on the SBX expansion module and dual port or local RAM. The remaining three channels are not supported by the ZT 8832.

## COMMONLY ASKED QUESTIONS

1. **Is the V40 pin-compatible with the 80188?**

   The V40 and 80188 are not pin-compatible. This means an 80188 cannot be plugged into the V40 socket. This is unlike the V20 and V30, which are interchangeable with the 8088 and 8086, respectively.

2. **What are the hardware differences between the V40 and the 80188?**

   One of the most notable differences is that the V40 is fabricated with a CMOS process. CMOS technology provides an increase in both temperature range and noise immunity, with a reduction in power consumption. The CMOS V40 has a maximum power dissipation of less than 1/2 W and a standby dissipation of less than 1/10 W. The 80188 has a maximum power dissipation of 3 W; that is, too hot to touch.

   Other differences are as follows: the V40 includes an asynchronous serial port; the V40 interrupt controller and counter/timers have the same architecture as the interrupt controller and counter/timers used in the Personal Computer; and the data transfer rate for the V40 DMA controller is twice as fast.

3. **Is the V40 software compatible with the 8088 and 80188 microprocessors?**

   Yes. The V40 instruction set is 100% object code compatible with the 80188 instruction set. This means that a program written for the 8088 or 80188 will execute on the V40. Application software using the peripherals internal to the 80188 requires some modification.

4. **What are the V40 instructions not in the 8088 or 80188 and how are they used?**

   The V40 instruction set is a superset of the 8088 and 80188. This means the V40 includes all of the instructions found in

these microprocessors, plus a few more. The added instructions are outlined below.

The following instructions are useful in testing and setting status bits for I/O operations and in bit manipulation for graphics applications.

| | |
|---|---|
| INS | Insert bit field |
| EXT | Extract bit field |
| TEST1 | Test bit |
| SET1 | Set bit |
| CLR1 | Clear bit |
| NOT1 | Complement bit |

The instructions shown below are useful for manipulating binary numbers in a decimal format.

| | |
|---|---|
| ADD4S | BCD string addition |
| SUB4S | BCD string subtraction |
| CMP4S | BCD string comparison |
| ROL4 | Rotate BCD digit left |
| ROR4 | Rotate BCD digit right |

The string I/O instructions shown below can be combined with the repeat prefixes for high speed data transfers between I/O and memory.

| | |
|---|---|
| INM | String input |
| OUTM | String output |

Other instructions not found in the 80188 instruction set are listed below.

| | |
|---|---|
| REPC | Repeat while carry set |
| REPNC | Repeat while carry cleared |
| FPO2 | Floating point operation 2 |
| BRKEM | Break for emulation mode |
| RETEM | Return from emulation mode |

## FUNCTIONAL BLOCKS

The V40 can be divided into the major functional blocks listed below and shown in Figure 5-1 on page 5-6.

**CPU** Central Processing Unit

**BIU** Bus Interface Unit

**BAU** Bus Arbitration Unit

**CGU** Clock Generator Unit

**VCR** V40 Configuration Registers

**WCU** Wait Control Unit

**SCU** Serial Control Unit

**TCU** Counter/Timer Control Unit

**ICU** Interrupt Control Unit

**DCU** DMA Control Unit

## CPU - Central Processing Unit

The architecture of the CPU functional block is compatible with the 8088. The CPU recognizes all of the instructions found in the 8088 and 80188 microprocessors. Figure 5-2 shows a block diagram of the CPU divided into two elements: the Bus Control Unit (BCU) and the Execution Unit (EXU). The BCU prefetches instructions and data into a 4-byte instruction queue. The EXU executes the instructions. This pipelined architecture increases the throughput over the typical microprocessor that must wait for an instruction or operand to be fetched before operation is continued.

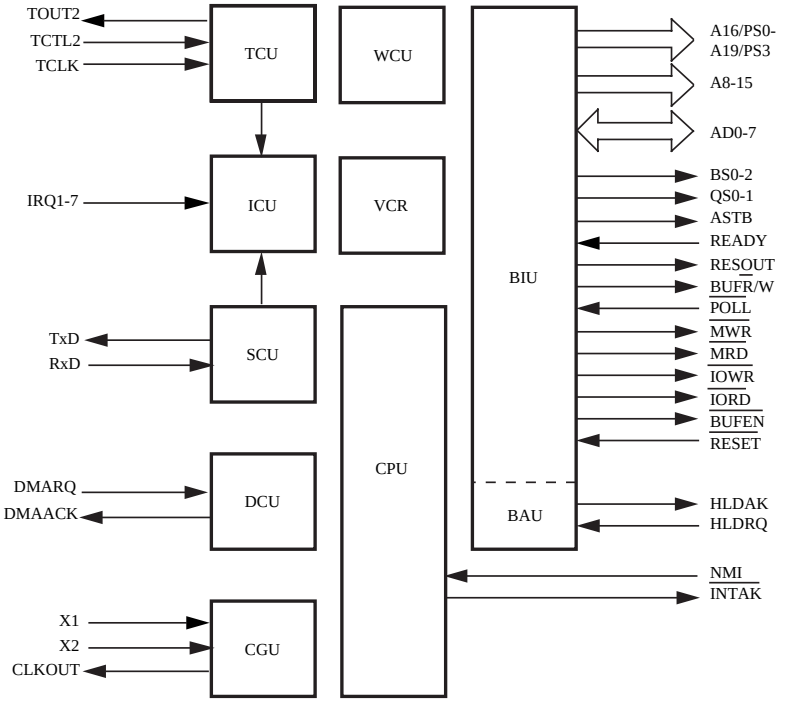



*Figure 5–1. V40 Block Diagram.*

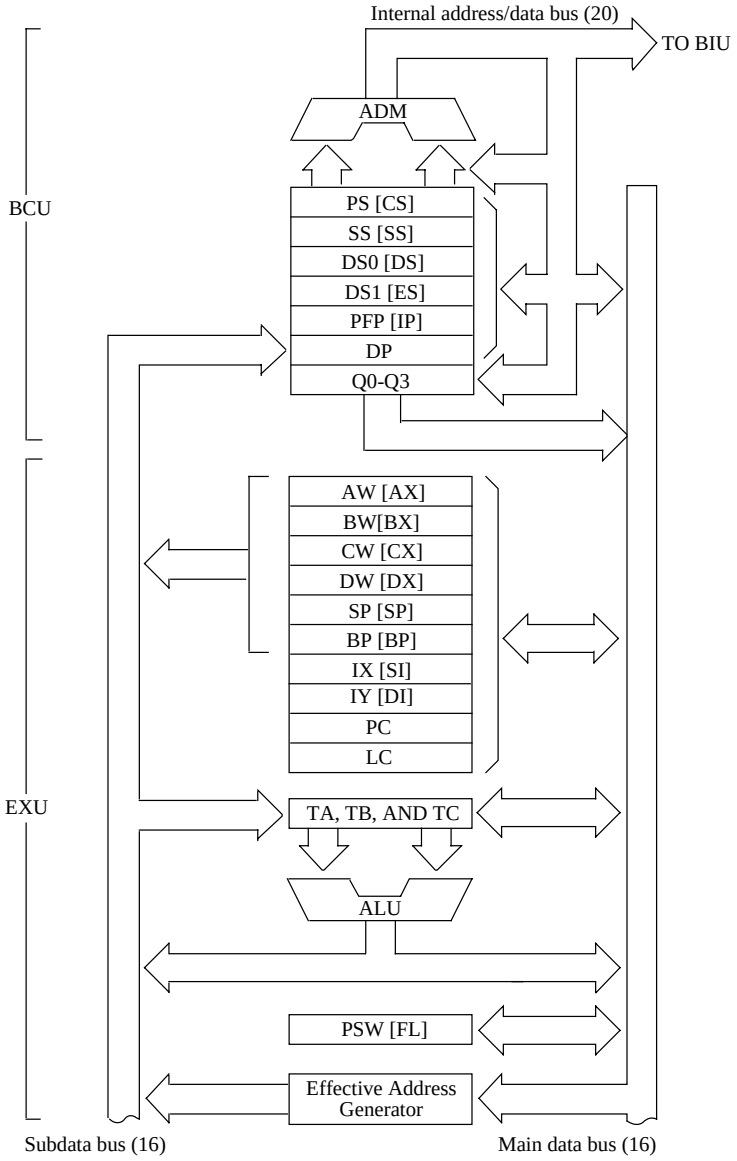



*Figure 5–2. CPU Block Diagram.*

## CPU Functional Blocks

The functional blocks in Figure 5-2 are described below. The NEC mnemonic is shown for each block, followed by the Intel mnemonic in brackets. For example, the CPU flag register is represented by PSW [FL] because NEC labels it Processor Status Word and Intel labels it Flags.

Segment Registers

PS [CS], SS [SS], DS0 [DS], and DS1 [ES]

The CPU can address up to 1 Mbyte of memory in segments of 64 Kbytes or less. The starting address of a segment is specified in a segment register. The four segment registers are as follows:

PS [CS] - Program Segment register
SS [SS] - Stack Segment register
DS0 [DS] - Data Segment register 0
DS1 [ES] - Data Segment register 1

All memory addresses are specified with a segment and offset as shown in Table 5-1. The segment and offset used depend on the type of instruction being executed.

The program always resides in a program segment pointed to by the PS [CS] register. The PFP [IP] always contains the offset within the program segment. The program stack always resides in the stack segment pointed to by the SS [SS] register. The SP [SP] contains the offset of the top of the stack. Stack variables can be addressed using the BP [BP] register because the default segment register is SS [SS].

Program variables generally reside in the data segment with the segment address in the DS0 [DS] register. The offset of the variable within DS0 [DS] is called the effective address. The EXU calculates the effective address by summing any combination of displacement, base register, and index register. The possible combinations of offset, base, and index provide the programmer with a large variety of addressing modes.

Strings are addressed differently from other variables. The segment register used to point to the source string is DS0 [DS], unless an override is used. The offset for the string source is the IX [SI] register. The segment register for the string destination is always DS1 [ES] and the offset is specified in IY [DI].

Table 5-1
Segment Registers.

| Memory Reference | Default Segment | Alternate Segment | Offset |
|---|---|---|---|
| Instruction Fetch | PS[CS] | NONE | PFP[IP] |
| Stack Operation | SS[SS] | NONE | SP[SP] |
| Variable (except following) | DS0[DS] | PS[CS], DS1[ES], SS[SS] | Effective Address |
| String Source | DS0[DS] | PS[CS], DS1[ES], SS[SS] | IX[SI] |
| String Destination | DS1[ES] | NONE | IY[DI] |
| BP[BP] Used As Base Register | SS[SS] | PS[CS], DS0[DS], DS1[ES] | Effective Address |
| BW[BX] Used As Base Register | DS0[DS] | PS[CS], DS1[ES], SS[SS] | Effective Address |

Prefetch Pointer

PFP [IP]

The prefetch pointer PFP [IP] is a 16-bit binary counter that maintains the offset of the next instruction to be fetched into the instruction queue. The BCU fetches a program instruction based on the segment value in the PS [CS] register and the offset in the PFP [IP].

For sequentially addressed instructions, the PFP [IP] is incremented by the number of bytes of the current instruction to point to the next. For program branching, such as intrasegment and intersegment jumps, the PFP [IP] is programmed with a value contained within the jump instruction. The PFP [IP] is not accessible to the programmer.

Data Pointer

DP

This 16-bit register is the destination for the offset address calculated by the effective address generator. The offset address calculation is done by hardware instead of the traditional microcode, saving three to ten clock cycles for every calculation. The DP register is not accessible to the programmer.

Instruction Queue

Q0 - Q3

The instruction queue is a temporary storage location for program instructions and variables that have been fetched by the BCU to be executed by the EXU. The instruction queue consists of four 8-bit registers, Q0 through Q3. These registers allow instruction fetching by the BCU and instruction execution by the EXU to be independent operations. This overlap essentially eliminates the time required to fetch program instructions and data. Q0 through Q3 are not accessible to the programmer.

Address Modifier

ADM

The V40 uses a 20-bit memory address to access any location in the 1 Mbyte addressing range. The 20-bit memory address is the sum of a segment (shifted left four bits) and an offset. The offset is taken from the PFP [IP] if a program instruction is being addressed or from the DP for all other data. The ADM does this addition. If the PFP [IP] was used, the ADM increments it for the next instruction. The ADM is not accessible to the programmer.

General Purpose Registers

AW [AX], BW [BX], CW [CX], and DW [DX]

The CPU has four 16-bit general purpose registers. Each of these registers can be addressed as one 16-bit register or two 8-bit registers. The 16-bit registers are referred to as AW [AX], BW [BX], CW [CX], and DW [DX]. The high order bytes of the 16-bit registers are AH, BH, CH, and DH, while the low order bytes are AL, BL, CL, and DL. The most common use of these registers is to provide a temporary storage location for data. Some instructions do assign specific meanings to the general purpose registers, as shown in Table 5-2.

Table 5-2
Implied Use of General Registers.

| Register | Implied Use | Register | Implied Use |
|---|---|---|---|
| AW [AX]: | Word Multiplication/ Division<br>Word Input/Output | BW [BX]: | Translation |
| AL: | Byte Multiplication/ Division<br>Byte Input/Output<br>Translation<br>BCD and Decimal Arithmetic | CW [CX]: | String Operations |
| AH: | Byte Multiplication/ Division | CL: | Variable Shift and Rotate |
| | | DW [DX]: | Word Multiplication/ Division<br>Indirect Input/Output |

Pointers and Index Registers

SP [SP], BP [BP], and IX [SI], IY [DI]

The two 16-bit pointer registers are used primarily for stack operations. The Stack Pointer (SP [SP]) is the offset to the top of the stack within the stack segment. This pointer is adjusted automatically each time a stack operation is performed. The Base Pointer (BP [BP]) is an offset to any location within the stack segment. The BP [BP] is useful as a pointer to variables being passed on the stack. Both pointer registers are accessible to the programmer.

The 16-bit index registers are used primarily for string operations. Strings are linear arrays of data that can be organized as words, bytes, nibbles, or even as bit values. The index registers specify the offset of the string source (IX [SI]) and destination (IY [DI]) within Data Segment 0 (DS0 [DS]) and Data Segment 1 (DS1 [ES]), respectively. The index registers are adjusted automatically during string transfers. Both IX [SI] and IY [DI] are accessible to the programmer.

Program Counter

PC

The program counter is a 16-bit binary counter that contains the offset address of the next instruction to be executed by the EXU. The PC is automatically incremented each time the EXU reads an instruction from the queue. If the instruction causes a branch in program execution, the PC is programmed with the branch address. At this point the contents of the PC are the same as the PFP [IP]. The difference between the PFP [IP] and the PC is that the PFP [IP] contains the offset of the next instruction to be fetched by the BCU and the PC contains the address of the next instruction to be executed by the EXU. The PC is not accessible to the programmer.

Loop Counter

LC

LC is a binary counter used to regulate iterative operations such as string transfers controlled by the repeat prefix and multiple-bit shifts and rotations. The CPU uses hardware for a loop counter, as opposed to microcode used by the traditional microprocessor.

Temporary Registers A, B, and C

TA, TB, and TC

These 16-bit registers are used by the ALU during arithmetic and logical instructions such as multiplication, division, and shift and rotate. TA and TB combine for 32-bit temporary storage during multiplication and division. Access to the temporary registers is not available to the programmer.

Arithmetic and Logic Unit

ALU

The ALU performs arithmetic and logic operations as well as bit manipulation. Arithmetic and logic operations include add, subtract, multiply, divide, increment, decrement, compare, complement, AND, OR, and exclusive OR. Bit manipulation includes shifting, rotating, comparing, setting, clearing, and inverting of individual bits.

Effective Address Generator

EAG

The 16-bit offset address calculated by the EXU for memory operations is called the effective address. The effective address can include a displacement, base, index, or combination of the three, depending on the addressing mode specified in the instruction being executed. Traditional microprocessors calculate this effective address using microcode. The EAG does this calculation in hardware. As an example, the 8088 requires up to

12 clocks to calculate the effective address using microcode. However, the V40 does all effective address calculations in two clocks with the hardware EAG.

The effective address, once calculated by the EAG, is transferred to the DP register, where it can be used by the BCU to transfer data between the CPU and memory.

Program Status Word

PSW [FL]

There are six status flags and four control flags in the 16-bit PSW [FL], as seen in Figure 5-3. Notice that not all 16 bits are defined. Those not defined are reserved; that is, they may be used in later versions of the processor. Because of this, a program should never rely on a value in any of these reserved bits.

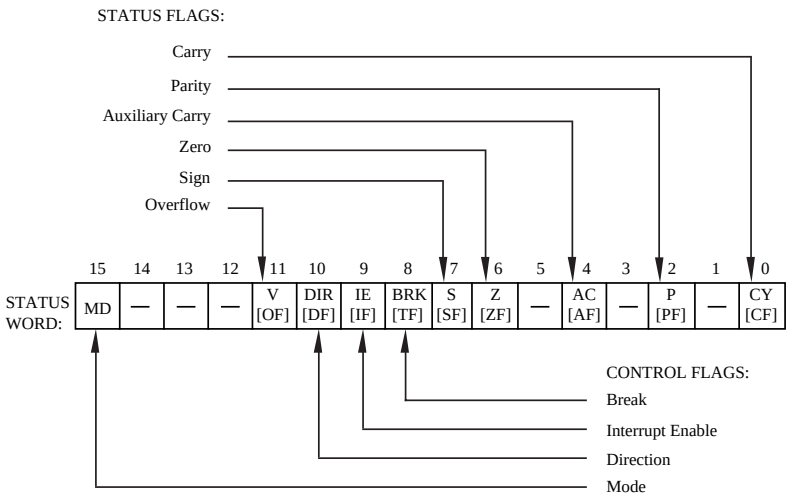



*Figure 5–3. Processor Status Word.*

The status flags provide information about the result of arithmetic and logic operations. The status flags are set (logical 1) and reset (logical 0) by the EXU based on the result of an arithmetic or logic operation. These flags can be tested by conditional jump instructions to change the order of program execution.

The control flags are used by the programmer to direct CPU operation. The control flags are set (logical 1) and reset (logical 0) with dedicated instructions. The IE [IF] and BRK [TF] flags are automatically reset when the program enters an interrupt service routine.

The PSW [FL] is automatically preserved on the stack at the start of an interrupt service routine for both hardware and software initiated interrupts, and at the start of a procedure initiated with the CALL instruction. A return from interrupt instruction, a return from procedure instruction, or a return from emulation instruction will restore the contents of the PSW [FL] from the stack.

Different instructions affect the status flags differently. A detailed description of each status flag is given on the following pages. Reference to bit position is based on the least significant bit being bit 0. The state of a flag is referred to as *set* when a a logical 1 is present, and as *reset* when a logical 0 is present.

*Status Flags*

**CY [CF] (Carry Flag)** - The carry flag is set if an addition results in a carry out of bit 7 for byte operations or bit 15 for word operations. The CY [CF] flag is also set if a subtraction results in a borrow into bit 7 for byte operations or bit 15 for word operations.

For unsigned byte multiplication, CY [CF] is reset if the most significant byte of the result (register AH) is 0. The same is true of the most significant word (register DW [DX]) for unsigned word multiplication.

For signed multiplication, CY [CF] is reset if the sign bit of the least significant byte (register AL) is extended to the most significant byte (register AH). The same is true for signed word multiplication with the least significant word in register AX and the most significant in register DX.

**P [PF] (Parity Flag)** - The P [PF] flag is set if the least significant byte of an arithmetic or logical result has an even

number of bits set. This flag is useful for checking the parity of ASCII characters.

**AC [AF] (Auxiliary Flag)** - AC [AF] is set if an addition results in a carry from the four least significant bits of the result. This is true for both byte and word addition. This flag is used by the CPU for BCD arithmetic operations.

**Z [ZF] (Zero Flag)** - Z [ZF] is set if the result of an arithmetic or logical operation is zero. A common use of this flag is to determine if two numbers are equal. The program subtracts the two values and if Z [ZF] is set, the values are equal.

**S [SF] (Sign Flag)** - Arithmetic and logic operations set S [SF] equal to the high order bit of the result. This is bit 7 for byte operations and bit 15 for word operations. For signed binary operations, S [SF] is reset for positive results and set for negative results. Programs using unsigned operations usually ignore S [SF] because the high order bit does not reflect the sign of the result.

**V [OF] (Overflow Flag)** - The V [OF] flag is set if the result of an operation is too large a positive number or too small a negative number to fit into the destination. The application program can use the overflow flag to determine if the result of two's complement arithmetic operation is out of range.

*Control Flags*

**MD (Mode Flag)** - The CPU operates in either native or emulation mode. In native mode, the CPU executes the standard 8086/186 instructions in addition to instructions unique to the V40. In emulation mode, the CPU executes an 8080 based instruction set. The MD flag is used to distinguish between the two modes. MD is programmed using specific instructions to put the CPU in the native mode (MD is set) or emulation mode (MD is reset). Refer to page 5-33 for more information.

**DIR [DF] (Direction Flag)** - The CPU supports string operations to manipulate linear arrays of data organized as words, bytes, nibbles, or bits. Index registers are used to point to elements of

the array during a string operation. After a string operation is completed, the index registers are incremented or decremented, depending on the state of the DIR [DF] flag. If the DIR [DF] flag is set, the index is incremented to point to the next array element. If the DIR [DF] flag is reset, the index is decremented.

**IE [IF] (Interrupt Enable Flag)** - The IE [IF] flag determines how the CPU responds to maskable external interrupts. If IE [IF] is set, the CPU recognizes maskable external interrupts. The CPU ignores all maskable external interrupts if IE is reset. IE [IF] has no effect on external non-maskable interrupts or on internally generated interrupts. IE [IF] is set or reset with dedicated instructions, but will also be reset automatically with a return from interrupt instruction.

**BRK [TF] (Break Flag)** - Setting the break (or trap) flag puts the CPU into a single-step operation useful for testing program execution. With BRK [TF] set, the CPU automatically generates an internal interrupt after each instruction. The programmer need only develop an interrupt service routine to examine contents of registers, dump memory, or do whatever is necessary for testing.

The BRK [TF] flag is set or reset by transferring the PSW [FL] to the program stack and using memory manipulation instructions to modify it. Once BRK [TF] is modified, it must be transferred back to the PSW [FL] to generate a type 1 interrupt after the execution of each instruction. As part of the interrupt acknowledge, the PSW [FL] is saved on the stack and the BRK [TF] flag is reset. This is done so that the processor will not single-step through the interrupt service routine. Once the service routine is completed, the PSW [FL] is restored from the stack automatically, setting the BRK [TF] flag to trap the next instruction.

### Enhanced Architecture

The V40 CPU includes several enhancements that provide an increase in performance over the 8088 microprocessor found on many STD bus designs. The most noticeable performance improvements come from additional hardware for the Effective Address Generator, Loop Counters and Shifters, and the use of dual internal data buses.

Using a hardware-effective address generator rather than microcode reduces the time needed to fetch memory operands by as much as 10 clocks per fetch. Using hardware counters and shifters instead of the conventional microcode increases the speed of multiply and divide instructions by as much as four times. Dual internal data buses reduce traffic for instructions with two operands for effective address calculation. These enhancements add up to a speed increase of as much as 30 percent over that of the 8088 microprocessor.

### Standby Mode

The CPU's standby mode reduces power consumption by more than one tenth during idle periods. Standby mode is automatically entered when the HALT instruction is executed from the native or 8080 emulation mode. This does not affect any of the internal peripherals, such as the counter/timers, interrupt controller, refresh controller, or DMA controller. The CPU automatically exits the standby mode after a reset or an interrupt.

## BIU - Bus Interface Unit

The BIU controls the external address, data, and control buses. The BIU also synchronizes the RESET and READY inputs with the clock, as shown in Figure 5-4. The synchronized RESET signal is used internally. It is provided externally as a signal called RESOUT. The synchronized READY signal is combined with the output of the Wait Control Unit to control the number of wait states inserted during bus operations.

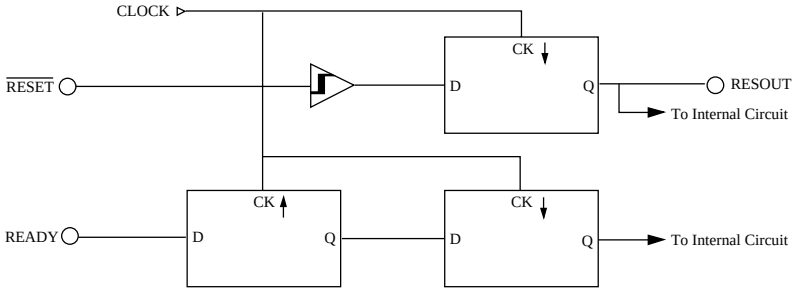



*Figure 5–4. RESET and READY Synchronization.*

## BAU - Bus Arbitration Unit

The V40 includes two internal bus masters and two signals supporting one external bus master. The two internal bus masters are the Central Processing Unit and DMA Control Unit. An external master, such as the 8087 numeric data processor, is supported with the HLDRQ and HLDAK signals. Each of the three bus masters mentioned above needs access to the address, data, and control buses to perform its function. The BAU controls which of these bus masters has access to

the buses at any given time. The bus masters are prioritized in the following order:

(1st) DCU - DMA Control Unit
(2nd) HLDRQ - External Bus Master
(3rd) CPU - Central Processing Unit

If the bus is being used by one bus master and another with higher priority makes a request, the BAU inactivates the current bus master's acknowledge. The BAU grants access to the higher priority bus master after the current bus master removes the request.

The BUSLOCK prefix prevents all bus masters, other than the CPU, from gaining access to the bus. Care must be taken to prevent loss of data in dynamic RAM when using the BUSLOCK prefix with instructions that have a long execution time.

## CGU - Clock Generator Unit

The CGU halves the frequency of the external oscillator to provide a clock reference with a 50% duty cycle to the CPU. This same signal is available on an external pin called CLKOUT to which all of the V40 timing parameters are referenced.

## VCR - V40 Configuration Registers

Twelve programmable registers are used to configure the V40 to meet the needs of varying applications. The V40 configuration registers are located in the top 16 bytes of the 64 Kbytes of I/O address space. The configuration registers define the functions of the programmable pins; the enabling and disabling of the SCU, TCU, ICU, and DCU; the location of the SCU, TCU, ICU, and DCU programmable registers in I/O address space; the wait-state configuration; DRAM refresh; and the counter/timer clock source.

## WCU - Wait Control Unit

The WCU provides added flexibility for interfacing to memory and I/O that have varying speed requirements. The V40 includes three internal bus masters that access memory and I/O devices. The number of wait states inserted can be programmed separately for the CPU, RCU, and DCU. The memory space can be divided into three separate areas and the number of wait states defined differently for each. The wait states are programmed through the WCY2, WCY1, and WMB V40 configuration registers.

## SCU - Serial Control Unit

The serial control unit is a single asynchronous serial channel used for serial communication between the V40 and a serial device external to the V40. Programming the SCU is similar to programming the 8251A Serial Control Unit for asynchronous modes of operation.

## TCU - Counter/Timer Control Unit

The Counter/Timer Control Unit includes three 16-bit programmable counter/timers. These counter/timers can be used for SCU baud rate generation, timing loops, timed and periodic interrupts, and external asynchronous event counters. Programming the TCU is similar to programming the 8254 Programmable Interval Timer, with a few restrictions placed on operating modes because of the way the TCU is connected internally to the V40.

## ICU - Interrupt Control Unit

Interrupts provide an efficient interface between the V40 CPU and supporting peripheral devices. The ICU supports eight interrupts directly and can be cascaded with other interrupt controllers, such as the 8259A Programmable Interrupt Controller, to support additional interrupting sources. Programming the ICU is similar to programming the 8259A.

## DCU - DMA Control Unit

The DCU controls high speed data transfer between I/O and memory devices. The DCU is similar to the 8257 Programmable DMA Controller except that the DCU supports the full 1 Mbyte of V40 addressing space.

## RESET

Resetting the V40 initializes registers internal to the CPU, VCR, SCU, TCU, ICU, and DCU. The reset states for the CPU registers are given in Table 5-3. Reset states for registers internal to the VCR, TCU, ICU, DCU, and SCU are given in their respective chapters.

The reset states of the program segment and instruction pointer combine to produce a physical address of FFFF0h. This is the address from which the V40 fetches the first instruction after reset.

Table 5-3
CPU Reset State.

| Register | Reset Value |
|---|---|
| PFP [IP] | 0000h |
| PC | 0000h |
| PS [CS] | FFFFh |
| SS [SS] | 0000h |
| DS0 [DS] | 0000h |
| DS1 [ES] | 0000h |
| PSW [FL] | F002h |
| AW [AX], BW [BX] | Undefined |
| CW [CX], DW [DX] | Undefined |
| IX [SI], IY [DI] | Undefined |
| BP [BP], SP [SP] | Undefined |
| Instruction Queue | Cleared |

## MEMORY AND I/O ADDRESSING

This section discusses how the V40 communicates with memory and I/O devices. The V40 has a 20-bit address bus and an 8-bit data bus. With 20 bits of address, the V40 can directly access up to 1 Mbyte of memory. The address range is from 0 to FFFFFh, as shown in Figure 5-5. Address locations 0 to 7Fh are reserved for dedicated interrupts and future enhancements. The address range from 80 to 3FFh completes the interrupt vector table and may be used as needed by the application. The 12 bytes (six words) from FFFF0 to FFFFBh are the area vectored to by the V40 after a reset. The most common practice is to program this area with an intersegment jump to the start of the application program. The upper four bytes are reserved and must not be programmed.

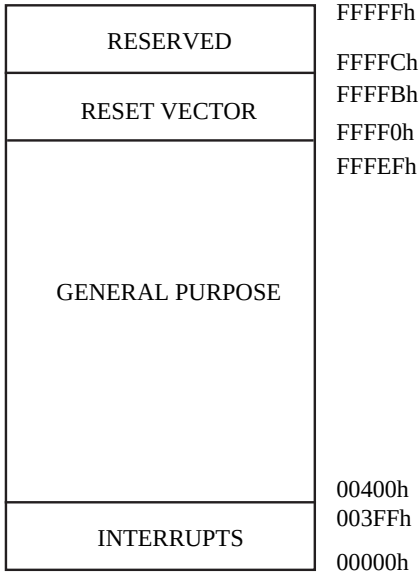



*Figure 5–5. Memory Map.*

To the programmer, the V40 address space is organized as a contiguous sequence of up to 1 Mbyte. Data can be addressed in units of bytes, words, and double words. Word and double word values are stored in memory with the most significant byte at the higher address and the least significant at the lower. Figure 5-6 illustrates these data formats.

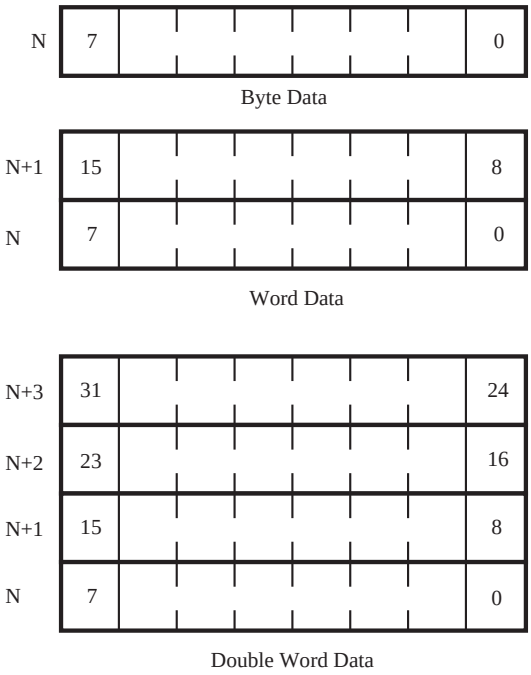



*Figure 5–6. Data Formats.*

The lower 16 of the 20 address lines are also used to address I/O devices. With 16 bits of address, the V40 can directly access up to 64 Kbytes of I/O. The address range is from 0 to FFFFh, as shown in Figure 5-7. Address locations FF00 through FFEFh are reserved for future use. The address range from FFF0 through FFFFh is currently used for the V40 configuration registers. These registers define programmable options in the V40, such as wait-state insertion, location of internal peripheral device registers, enabling DRAM refresh, and selecting the period of refresh.

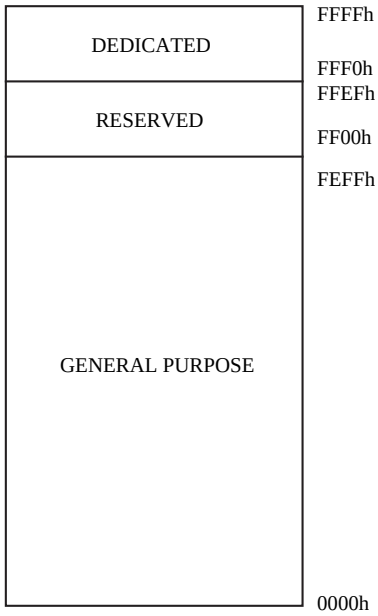



*Figure 5–7. I/O Map.*

## INTERRUPTS

The V40 includes a versatile interrupt structure that supports both hardware and software initiated interrupts. Hardware interrupts are external inputs to the V40 and can be classified as maskable or non-maskable. Maskable interrupts are routed to the CPU through the ICU. The ICU provides the maskable interrupt inputs with the ability to be level- or edge-triggered, have fixed or rotating priorities, and be individually masked. Non-maskable interrupts are routed directly to the CPU and are not maskable through programming.

Software interrupts are internally generated during program execution. They include instructions that can be executed to generate interrupts, and error handling instructions for such things as a divide overflow. Table 5-4 on page 5-28 lists the sources of hardware and software interrupts. The remainder of this section explains these interrupts and how the V40 handles them.

Table 5-4
Interrupt Sources.

| | Interrupt Source | Clocks | Priority |
|---|---|---|---|
| **Software** | DIVU divide error | 45 | 1 |
| | DIV divide error | 45-55 | 1 |
| | CHKIND breakout error | 53-56 | 1 |
| | BRKV | 40 | 1 |
| | BRK3 | 38 | 1 |
| | BRK imm8 | 38 | 1 |
| | BRKEM imm8 | 38 | 1 |
| | CALLN imm8 | 38 | 1 |
| | BRK single step | 38 | 4 |
| **Hardware** | NMI | 38 | 2 |
| | ICU inputs | 49 | 3 |

The purpose of an interrupt is to redirect the CPU from its current activity to an interrupt service routine designed to handle the needs of the interrupting source. Every interrupting source is associated with a number that points the CPU to a location in memory that contains the address of the interrupt service routine. This number is called an interrupt vector. The area in memory where the addresses of the interrupt service routines are stored is called the interrupt vector table. During an interrupt cycle, the CPU multiplies the vector by four to obtain the location of the service routine address in the vector table. The CPU then transfers control to the service routine at the address read from the vector table. (See Figure 5-8.) It is the programmer's responsibility to load the address of the service routine into the vector table. The address includes a segment and offset value in the format shown.

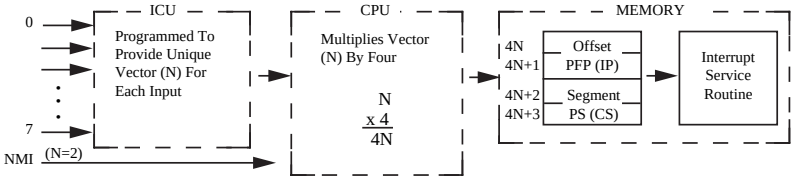



*Figure 5–8. Interrupt Processing.*

The vector of an interrupt source must be known before the location of the service routine address can be determined. The interrupt vector table, shown in Table 5-5, lists the vectors for the sources of interrupts. As an example, assume you need to write a service routine to handle a non-maskable interrupt (NMI) request. The vector for NMI is two, as seen in the Table 5-5. The address of the NMI service routine is determined by the application, but that address must be programmed in the vector table at 8h.

Table 5-5
Interrupt Vector Table.

| | **Interrupt Source** | **Vector Number** |
|---|---|---|
| **Software** | DIVU divide error | 0 |
| | DIV divide error | 0 |
| | CHKIND breakout error | 5 |
| | BRKV | 4 |
| | BRK3 | 3 |
| | BRK imm8 | 32-255 |
| | BRKEM imm8 | 32-255 |
| | CALLN imm8 | 32-255 |
| | BRK single step | 1 |
| **Hardware** | NMI | 2 |
| | ICU inputs | 32-255 |

Before describing each source of interrupt shown in the vector table, it is useful to summarize the operation of the CPU in response to an interrupt. Interrupts come to the CPU from three sources: the NMI signal external to the V40, the output of the ICU, and from inside to the CPU itself. A vector is supplied in all cases to distinguish between the interrupting sources. The CPU determines the address of the service routine by multiplying the vector times four. Before transferring execution to the service routine, the CPU saves the machine status by pushing the current contents of the PSW [FL] and the return address onto the stack. The CPU then clears the BRK [TF] and IE [IF] flags to prevent subsequent single-step and maskable interrupts, and transfers program execution to the service routine. The service routine is terminated with a "return from interrupt" instruction. This instruction causes the CPU to restore the PSW [FL] from the stack and return execution to the interrupted program. Restoring the PSW [FL] automatically enables single-step and maskable interrupts.

**Divide Error**

The divide error interrupt is generated by the CPU following execution of a DIV [DIV] or DIVU [IDIV] instruction if the calculated quotient is larger than the specified destination. The interrupt is not maskable and the vector is fixed at zero.

**Single-Step**

The single-step interrupt is a powerful software debugging tool. The purpose of the single-step interrupt is for software single stepping through a sequence of code. This interrupt is controlled by the BRK [TF] flag in the PSW [FL]. There is no instruction to set the BRK [TF] flag. To set the BRK [TF] flag, the PSW [FL] register must be pushed on the stack, the flag set, and the PSW [FL] popped back off the stack. With the BRK [TF] flag set, a single-step interrupt is generated after each instruction. The CPU responds to the interrupt by pushing the PSW [FL], PS [CS], and PFP [IP] on the stack. The BRK [TF] and IE [IF] are reset to a logical 0 to prevent another single-step or maskable interrupt. Upon completion of the single-step routine, the

CPU restores the PSW [FL], PS [CS], and PSP [IP]. The single-step interrupt is not masked by the IE [IF] bit in the PSW [FL].

### Non-Maskable

The V40 has a non-maskable interrupt input called NMI. NMI is rising edge triggered but must remain active for two CPU clocks to guarantee recognition. This interrupt is not maskable and the vector is fixed at 2.

### Fixed Vector Instruction

The fixed vector instruction (BRK3 [INT3]) is a special form of the more general variable vector instruction. The difference is that the fixed vector is a single byte while the variable vector requires 2 bytes. The primary use of this instruction is for breakpoint execution during program development. Because it is a single byte, the instruction can be mapped over the smallest possible instruction. This interrupt is not maskable and has a fixed vector of 3.

### Overflow

The overflow interrupt is generated if the V [OF] flag is set to a logical 1 and the BRKV [INTO] instruction is executed. This interrupt is useful for trapping overflow errors for mathematical operations. The overflow interrupt is not maskable and the vector is fixed at 4.

## Check Index

The purpose of the check index instruction is to test the index of an array against an upper and lower limit. The CHKIND [BOUND] instruction generates a check index interrupt if the index value is less than the lower limit or greater than the upper limit. The vector for the check index instruction is fixed at 5 and is not maskable.

## Variable Vector Instruction

Interrupts can be generated using a variable vector interrupt instruction with the format BRK [INT] *xx*, where *xx* is the vector number. Accessing subroutines in this manner means only the subroutine vector is fixed. The location and length of the subroutine can vary without affecting the main program. These interrupts cannot be masked.

## Emulation Mode

Two interrupt instructions deal with 8080 emulation mode. The BRKEM instruction is used to transfer V40 operating modes from native to emulation for execution of 8080 based programs. The CALLN instruction is used in emulation mode to call an 8088 procedure. Both instructions have the format and operation of the variable vector instruction. The following pages discuss 8080 emulation in detail.

## 8080 EMULATION

Designs based on 8080 and 8085 microprocessors have two major limitations: inadequate performance and lack of development tools. Upgrading an 8-bit design to a higher performance microprocessor requires time to convert the software. The V40 solves these problems by supporting two modes of operation: emulation mode and native mode. When the CPU is in emulation mode, it executes the 8080 instruction set. Emulation mode is used for the existing software base. In native mode, the CPU executes the 8088 instruction set. All future software development is done in native mode to take advantage of the 8088 instruction set and the large number of development tools designed around it.

The CPU powers up in native mode, the normal mode of operation. Two instructions are provided to switch the CPU from native mode to emulation mode and back. Break for Emulation (BRKEM) is the instruction used to switch from native to emulation mode; Return from Emulation (RETEM) is used to switch back. The effect of these instructions and emulation mode operation is discussed below.

The BRKEM instruction is similar to the BRK [INT] software interrupt. BRKEM includes an 8-bit vector that, when multiplied by four, points to the location in the interrupt vector table that contains the address of the 8080-based routine. During execution of this instruction, the CPU saves the machine status by pushing the current contents of the PSW [FL] and the return address onto the stack. The CPU then clears the mode (MD) flag to a logical 0 and loads the address of the emulation mode routine into the PS [CS] and PFP [IP].

The RETEM instruction is one of the four methods of terminating emulation mode. The execution of RETEM is identical in operation to the RETI [IRET] instruction. Upon executing this instruction, the CPU restores the contents of the PSW [FL], PS [CS], and PFP [IP] from the stack, returning program execution to the instruction following BRKEM. The other three methods of exiting emulation mode are a system reset, a hardware interrupt, or the CALLN instruction.

A hardware interrupt suspends the 8080 emulation mode. The CPU pushes the PSW [FL] and return address onto the native mode stack, sets the MD flag to a logical 1, and transfers program execution to the native mode interrupt service routine. When the CPU executes the RETI [IRET] instruction, the PSW [FL] is restored with the MD flag set to a logical 0, and program execution continues in the emulation mode. The CALLN instruction permits the execution of native mode subroutines from emulation mode. The CPU responds to CALLN in the same manner as a hardware interrupt.

The emulation mode cannot be nested. For example, assume that the CPU is operating in native mode and executes the BRKEM instruction. The CPU switches to native mode and begins executing emulation code. Next, assume that a hardware interrupt (such as the 16450 serial controller) suspends emulation mode and the CPU begins executing the interrupt service routine in native mode. That interrupt service routine cannot include a BRKEM instruction.

Table 5-6 on page 5-36 shows the relationship between the native and emulation mode registers and flags. The native mode registers not shown are inaccessible to 8080 programs; they are AH [AH], PS [CS], SS [SS], DS0 [DS], DS1 [ES], IX [SI], IY [DS], and the upper eight bits of the PSW [FL].

Memory addressing and stack referencing must also be considered. The 8080 addresses a maximum of 64 Kbytes. This block of memory can be located anywhere in the 1 Mbyte address space by programming the PS [CS] word in the interrupt vector table before the BRKEM instruction is executed. All data and stack operations are referenced from the DS0 [DS] register. This register must be initialized before the BRKEM instruction is executed. The values in the PS [CS] and DS0 [DS] registers must be equal for complete compatibility with the 8080 structure.

Emulation mode uses the BP [BP] register for the stack pointer, rather than the native mode SP [SP] register, to reduce the possibility of programming errors in one mode corrupting the stack of the other. This feature is helpful during program development.

Table 5-6
Emulation Mode Registers and Flags.

| | **8080** | **8088** |
|---|---|---|
| **Registers** | A | AL |
| | B | CH |
| | C | CL |
| | D | DH |
| | E | DL |
| | H | BH |
| | L | BL |
| | SP | BP |
| | PC | PC |
| **Flags** | C | CY |
| | Z | Z |
| | S | S |
| | P | P |
| | AC | AC |

Chapter 6

# PROCESSOR CONFIGURATION (V40)

**Contents** **Page**



## OVERVIEW

The V40 is a high integration microprocessor containing a CPU and several peripherals most commonly found in STD bus systems. The V40 also includes a software-programmable register set that configures these peripherals to specific applications. This chapter explains the architecture and use of these configuration registers.

## VCR - V40 CONFIGURATION REGISTERS

The 12 V40 configuration registers are mapped from I/O address FFF0h through FFFFh. The registers are listed in Table 6-1 and are discussed in detail on the following pages. All of the registers can be written to with the output instruction and read from with the input instruction. *The value input may be different from the value output, but only in the bits not defined.*

Table 6-1
V40 Configuration Registers.

| I/O Address | Register | Function |
|---|---|---|
| FFFFh | Reserved | -- |
| FFFEh | OPCN | V40 multiplexed pin assignment |
| FFFDh | OPSEL | V40 peripheral enable |
| FFFCh | OPHA | V40 peripheral I/O address (MSB) |
| FFFBh | DULA | DCU I/O address (LSB) |
| FFFAh | IULA | ICU I/O address (LSB) |
| FFF9h | TULA | TCU I/O address (LSB) |
| FFF8h | SULA | SCU I/O address (LSB) |
| FFF7h | Reserved | -- |
| FFF6h | WCY2 | DCU wait states |
| FFF5h | WCY1 | CPU memory and I/O wait states |
| FFF4h | WMB | Memory wait state boundaries |
| FFF3h | Reserved | -- |
| FFF2h | RFC | Refresh enable, frequency select |
| FFF1h | Reserved | -- |
| FFF0h | TCKS | Timer/counter clock selection |

## OPCN - On Chip Peripheral Connection Register

Figure 6-1 shows the OPCN register. Bit 0 must be programmed with a logical 0 and bit 1 must be programmed with a logical 1.

The two bits of the IRSW field select the interrupt source to be assigned to IRQ1 and IRQ2 of the interrupt controller. The values programmed into bits 2 and 3 depend on the use of the interrupt controller in the application. The pins external to the V40 used for IRQ1 and IRQ2 are connected to Stage 1 of the watchdog timer and Interrupt 0 of the SBX expansion module, respectively. The STD ROM software initializes OPCN to a 06h to use the V40 serial port.

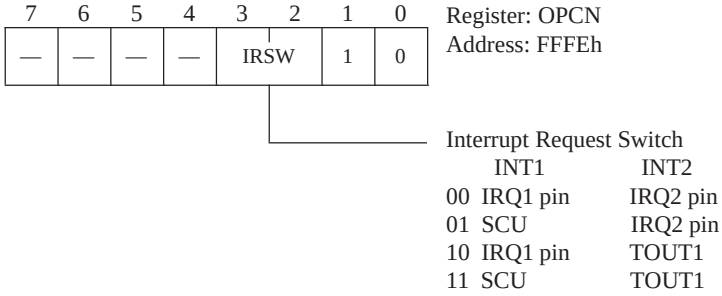



*Figure 6–1. On Chip Peripheral Connection Register.*

## OPSEL - On Chip Peripheral Selection Register

The V40 integrates several of the most common peripheral devices with a CPU in one package. The peripheral devices include a serial port, an interrupt controller, a DMA controller, and three counter/timers. The OPSEL register enables or disables these peripheral devices. The format of the OPSEL register is shown in Figure 6-2. No restrictions are placed on the use of the OPSEL register. The STD ROM software initializes OPSEL to a 0Eh to enable the V40 serial port and the counter/timers for baud rate generation.

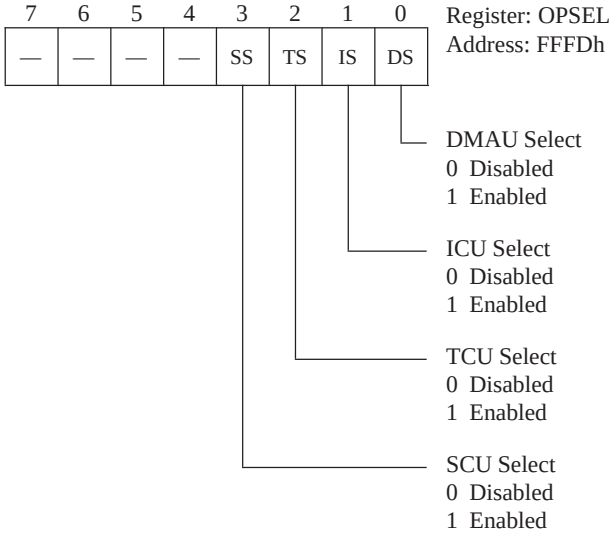



*Figure 6–2. On Chip Peripheral Selection Register.*

## OPHA, DULA, IULA, TULA, and SULA

Five registers determine the I/O base address of the programmable registers used to communicate with the DMA controller, interrupt controller, timer/counters, and serial controller.

OPHA - On Chip Peripheral High Address register (FFFCh)
DULA - DMA Unit Low Address register (FFFBh)
IULA - Interrupt Unit Low Address register (FFFAh)
TULA - Counter/timer Unit Low Address register (FFF9h)
SULA - Serial Unit Low Address register (FFF8h)

The I/O base address is a 16-bit value made up of two 8-bit values. The upper 8 bits of the address for the DMA controller, interrupt controller, counter/timers, and serial controller are defined by the OPHA register. The lower 8 bits for the DMA channel are programmed in the DULA register. The same holds true for the interrupt controller and IULA register, for the counter/timers and TULA register, and for the serial controller and SULA register.

In operation, OPHA permits the four internal peripheral devices to be mapped to any 256 byte block in the 64K I/O address space. The individual registers DULA, IULA, TULA, and SULA are programmed to define the base address of each of these devices anywhere within this block. For example, if the interrupt controller is to be mapped starting at I/O address FF20h, OPHA must be programmed with a FFh and IULA with a 20h.

The only restriction placed on programming these registers is to be sure the peripherals internal to the V40 are not mapped in the same address range as other I/O devices local to the ZT 8832. The STD ROM software programs these registers with the values shown in Table 6-2.

Table 6-2
STD ROM Programmable Address Selection.

| Register | Value | I/O Port Address |
|---|---|---|
| OPHA | 00 | ---- |
| DULA | D0 | 00D0h |
| IULA | 20 | 0020h |
| TULA | 40 | 0040h |
| SULA | B0 | 00B0h |

## WCY2 - Wait Cycle 2 Register

The V40 includes a programmable wait-state generator that interfaces to memory and I/O devices that are not fast enough to operate without wait states. The wait-state generator is programmed through the WCY2, WCY1, and WMB configuration registers. The format of the WCY2 register is shown in Figure 6-3. Bit 2 must be programmed with a logical 0 and bit 3 must be programmed with a logical 1 to select two wait states into DMA cycles. Two wait states are required to meet the SBX expansion module timing requirements. The STD ROM software initializes the WCY2 register with a 08h.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | 1 | 0 | — | — |

Register:WCY2
Address:FFF6h

*Figure 6–3. Wait-Cycle 2 Register.*

## WCY1 - Wait Cycle 1 Register

The WCY1 register is divided into four fields as shown in Figure 6-4 on page 6-8. The first three fields define the number of wait states inserted into three different regions of memory defined by the WMB register. The Lower Memory Wait (LMW) field defines the number of wait states inserted for memory accesses into the low memory region. The Middle Memory Wait (MMW) field defines the number of wait states inserted for memory accesses into the middle memory region. The Upper Memory Wait (UMW) field defines the number of wait states inserted for memory accesses into the upper memory region.

The ZT 8832 does not require any memory wait states if memory devices with access times less than 250 ns are used. To select zero memory wait states, program bits 0 through 5 with logical 0s. Programming bit 6 with a logical 0 and bit 7 with a logical 1 selects the two I/O wait states required by peripherals on the ZT 8832. The STD ROM software programs the WCY1 register with an 80h.

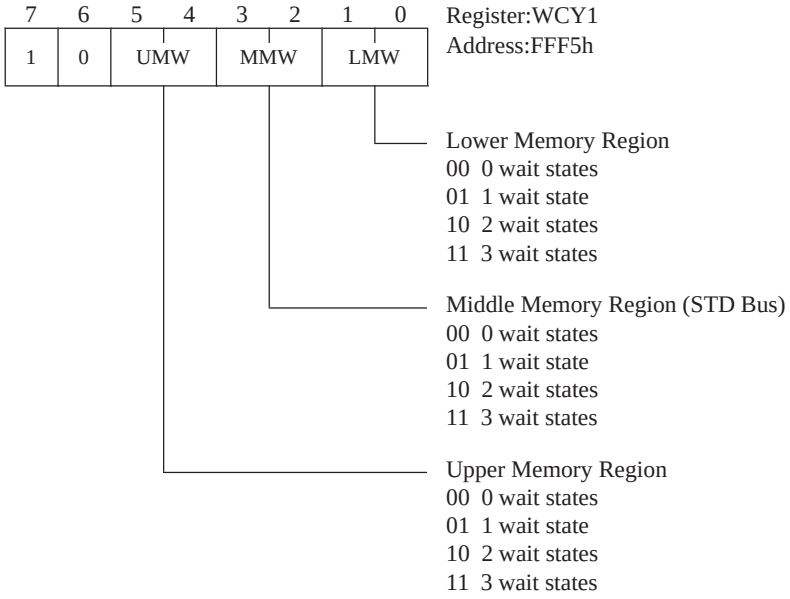



*Figure 6–4. Wait-Cycle 1 Register.*

## WMB - Wait Memory Boundary Register

The ZT 8832 does not require any wait memory wait states if memory devices with access times less than 250 ns are used. If slower memory devices are used, the WMB register divides the ZT 8832 memory into three regions and the WCY1 register defines the number of wait states inserted into each. As shown in Figure 6-5, the WMB register is divided into Upper Memory Boundary (UMB) and Lower Memory Boundary (LMB) fields.

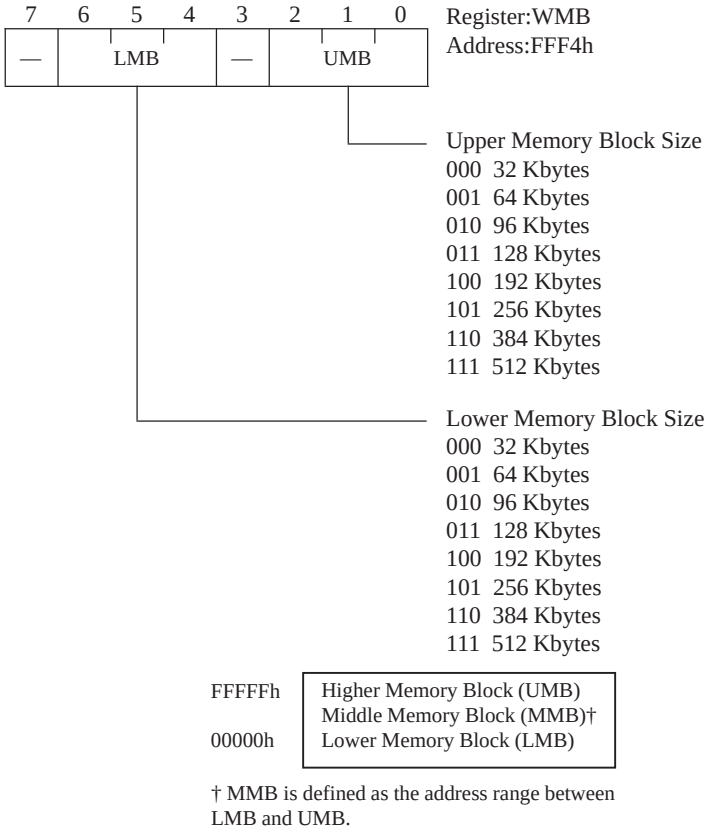



*Figure 6–5. Wait-Cycle Memory Boundary Register.*

The Middle Memory Block is defined between the top of the LMB and the bottom of the UMB. Offboard memory (STD bus) requires one wait state in all cases and is defined by the MMB.

The LMB field defines the lower memory address range starting from zero. This field can be programmed to include the memory devices inserted into the RAM LOW and RAM HIGH sockets. The UMB field defines the upper memory address range ending at FFFFFh. This field can be programmed to include the memory device inserted into the ROM socket. The STD ROM software initializes all bits of the WMB register to logical 0s.

## RFC - Refresh Control Register

The refresh controller is not used by the ZT 8832. To prevent the refresh controller from affecting system performance, program bit 7 with a logical 0 (see Figure 6-6).

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Register:RFC<br>Address:FFF2h |
|---|---|---|---|---|---|---|---|---|
| 0 | — | — | — | — | — | — | — | |

*Figure 6–6. Refresh Control Register.*

## TCKS - Counter/Timer Clock Selection Register

The V40 includes three programmable counter/timers. The TCKS register, shown in Figure 6-7, selects the clock source for each counter/timer, and also selects a frequency divisor that is used by all three counter/timers.

The CS0, CS1, and CS2 (Clock Select 0, 1, and 2) bits select the counter/timer clock source to be either the reference clock internal to the V40 or the TCLK pin available on an external V40 pin. The V40 clock operates at 8 MHz with a 50% duty cycle. The TCLK signal is available through connector J3. The Prescale (PS) field selects a prescale value that divides the clock frequency of all counter/timers using the V40 internal clock, then applies that value to the counter/timers. The STD ROM software programs all bits of TCKS with logical 0s to configure counter/timer 1 for baud rate generation.

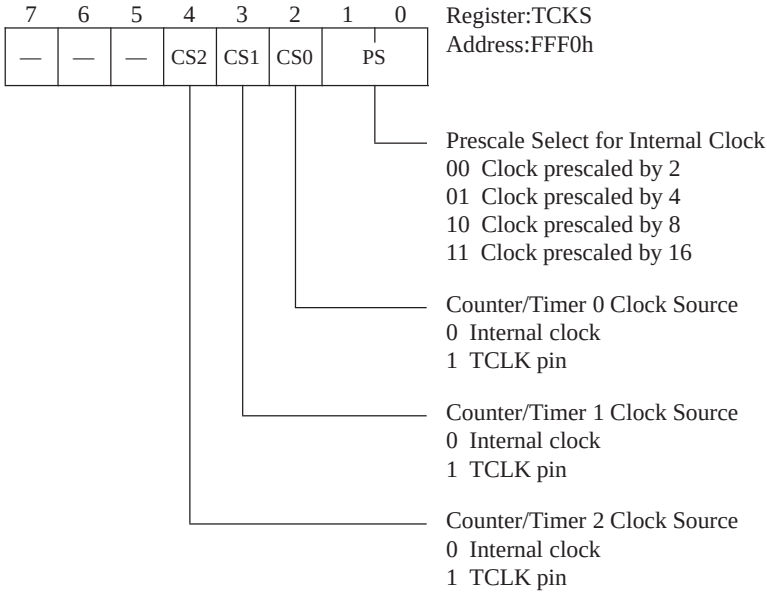



*Figure 6–7. Timer/Counter Clock Selection Register.*

## RESET

The V40 configuration registers are automatically initialized to a default state when power is applied to the V40 and also during reset from an external source. Table 6-3 shows the default state of the configuration registers.

Table 6-3
V40 Configuration Register Defaults.

| Registers | Default Bit Values[1] 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| OPCN | - | - | - | - | 0 | 0 | 0 | 0 |
| OPSEL | - | - | - | - | 0 | 0 | 0 | 0 |
| OPHA | - | - | - | - | - | - | - | - |
| DULA | - | - | - | - | - | - | - | - |
| IULA | - | - | - | - | - | - | - | - |
| TULA | - | - | - | - | - | - | - | - |
| SULA | - | - | - | - | - | - | - | - |
| WCY2 | - | - | - | - | 1 | 1 | 1 | 1 |
| WCY1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| WMB | - | - | - | - | - | - | - | - |
| RFC [2] | - | - | - | 0 | 1 | 0 | 0 | 0 |
| TCKS | - | - | - | 0 | 0 | 0 | 0 | 0 |

[1] Bit Positions marked with a dash (-) can default to 1 or 0.

[2] The refresh enable bit of the RFC register is not affected by resets other than power on.

Chapter 7

# COUNTER/TIMERS (V40)

**Contents** **Page**

## OVERVIEW

This chapter describes the Counter/Timer Control Unit (TCU) and provides register descriptions.

The TCU includes three 16-bit programmable counter/timers. Applications for the counter/timers include baud rate generation for the serial controller, timing loops, timed and periodic interrupts, and asynchronous event counters.

The main features of the TCU are:

- Three 16-bit counter/timers
- Six programmable operating modes
- Binary and BCD counting
- Interrupt and polled operation
- Functionally equivalent to 8254

## ZT 8832 SPECIFICS

The clock source for each counter/timer is defined in the TCKS V40 configuration register. Choices for the clock source are either the V40 internal clock or the TCLK signal. The clock internal to the V40 has a frequency of 8 MHz and a duty cycle of 50%. The TCLK signal is available through connector J3. The TCLK signal must meet the following requirements:

- Operating frequency between DC and 8 MHz
- Rise and fall times less than 25 ns
- Clock low and clock high times greater than 50 ns

Counter/timers 0 and 1 have implied uses because of their dedicated output connections to other devices internal to the V40. The output of counter/timer 0 is connected to IRQ0 of the interrupt controller. This dedicates counter/timer 0 to generating timed or periodic interrupts.

The output of counter/timer 1 is connected to the V40 serial port for baud rate generation. The output of counter/timer 1 can also be connected to IRQ2 of the interrupt controller if the V40 serial port is not needed. This connection is made with the OPCN V40 configuration register.

Counter/timer 2 does not have an implied use. The output (TOUT2) and control (TCTL2) signals for counter/timer 2 are available through connector J3, to be used as required by the application.

## FUNCTIONAL DESCRIPTION

The TCU is similar to the 8254 Programmable Interval Timer in that the programmable registers are the same. Some restrictions apply to the operating modes of individual counter/timers because they are connected to the V40 internally and external control inputs are not available. These restrictions are discussed below in an explanation of individual operating modes. Figure 7-1 illustrates the major functional blocks of the TCU. These functional blocks are discussed in the following paragraphs.

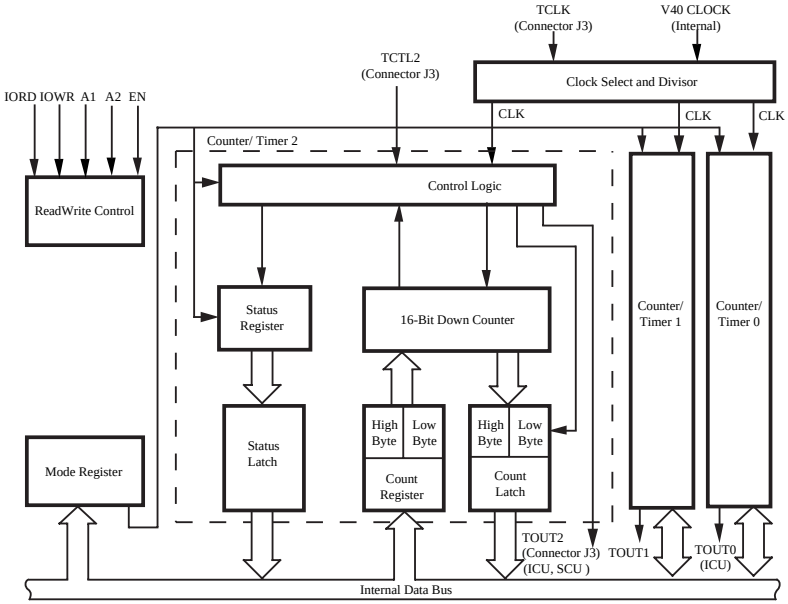



*Figure 7–1. Counter/Timer Block Diagram.*

### Read/Write Control

This functional block monitors I/O control and address signals to select programmable registers and control data flow.

## Mode Register

The 8-bit Mode register is programmed to control the operation of the counter/timers.

## Clock Select and Divisor

The difference between a timer and a counter is the clock source. The clock source for a timer is periodic, while the clock source for a counter is usually not periodic. A timer is used to reduce the frequency of a periodic signal; a counter is used to monitor the occurrence of asynchronous events. The Clock Select module defines the clock source for each counter/timer based on the value programmed into the TCKS V40 configuration register.

Clock source choices are either the V40 internal clock or the TCLK signal available on an external V40 pin. The internal clock is useful for counter/timers used in applications needing baud rate generation for the V40 serial port, accurate timing loops, timed interrupts, or periodic interrupts. The TCLK selection is useful for applications needing to count external events.

The Divisor, which operates on the counter/timers using the V40 internal clock as the clock source, is programmed through the TCKS V40 configuration register to divide the V40 clock by two, four, eight, or 16 before input to the counter/timer occurs.

## Counter/Timers 0, 1, and 2

The functional block diagram shown in Figure 7-1 illustrates the internal blocks of counter/timer 2. These internal blocks are the same as for counter/timer 0 and counter/timer 1. Each counter/timer is based on a 16-bit synchronous Down Counter, which can be preset to an initial number of counts and programmed for either binary or BCD (Binary-Coded-Decimal) operation.

The Count register and Count Latch are the interface through which the count data is transferred between the TCU and the CPU. The Count register receives the count value programmed to the TCU. The count value can be either 8 bits or 16 bits. Eight-bit values can be specified as either upper or lower bytes.

A useful feature of counter/timers is that they can be read at any time. It may be necessary to read a counter while an application is operating to determine how much time remains in a timing loop, or how many external events have occurred. The contents of the Down Counter, however, can change during the read operation, producing undefined results.

While one solution is to stop the Down Counter, this causes inaccuracies in the overall timing loop, or may cause an external event to be missed. The TCU solves this problem with the Count Latch and with commands that freeze the contents of the Count Latch without affecting the Down Counter. The Count Latch normally holds the current value of the Down Counter. When the TCU receives a count latch command, the Count Latch stops changing long enough for the read operation to occur, then returns to tracking the contents of the Down Counter.

The Status register contains information about the counter/timer's operation, such as the programmed operating mode. This eliminates the need for application software to store this information. The Status Latch allows the status to be read during times when status bits may be changing.

The Control Logic manipulates other functional blocks of the counter/timer, depending on which of the six programmable operating modes is selected, the state of the clock (TCLK), and, for counter/timer 2 only, the control input (TCTL2).

## PROGRAMMABLE REGISTERS

Four separately addressable registers are used for communication with the TCU. The TMD (Timer Mode) register specifies the operation of the three counter/timers. The TMD is a write-only register. The other three bidirectional registers, called the Count and Status registers, are used to write the count to the counter/timers and read the count and status back.

The base I/O address of the TCU registers is defined by the OPHA and TULA registers. OPHA is programmed with the high byte and TULA with the low byte of the 16-bit address. Refer to Chapter 6 for a complete discussion on the V40 configuration registers, including OPHA and TULA. The address of the TCU registers, relative to the base address, is shown in Table 7-1.

Table 7-1
TCU Register Addressing.

| Address | Register | Operation | Page Number |
|---|---|---|---|
| Base + 0 | TCT0 Count | Read/Write | 7-12 |
| Base + 0 | TCT0 Status | Read | 7-12 |
| Base + 1 | TCT1 Count | Read/Write | 7-12 |
| Base + 1 | TCT1 Status | Read | 7-12 |
| Base + 2 | TCT2 Count | Read/Write | 7-12 |
| Base + 2 | TCT2 Status | Read | 7-12 |
| Base + 3 | TMD | Write | 7-8 |

## Timer Mode Register (TMD)

The counter/timers must be initialized with the 8-bit TMD register. The three formats for the TMD register are shown in Figures 7-2, 7-3, and 7-4 (pages 7-9 to 7-11). The General Mode format is programmed initially to define the operation of the counter/timers. The Count Latch Mode and Multiple Latch Mode are programmed at any time to read the count and status data while the counter/timers are operating.

General Mode

The General Mode format, shown in Figure 7-2, specifies the operating mode of the individual counter/timers. The Select Counter bits specify the Multiple Latch command, or which counter/timer will receive the mode. Selecting the Multiple Latch command changes the definition of the TMD register bits to that of the Multiple Latch Mode format. The following bit definitions apply only if the Multiple Latch and Count Latch options are not programmed.

The Read/Write Mode bits specify the Count Latch command, or the format of the count transferred between the CPU and the TCU. Selecting the Count Latch command redefines the bits of the TMD register to that of the Count Latch Mode format.

The count transferred to or from the 16-bit counter/timers is one or two 8-bit values, depending on the Read/Write Mode bits. If the low byte option is chosen, the 8-bit count transferred to the counter/timer is placed into the low byte of the Down Counter and the high byte is automatically set to zero. The high byte option means that the 8-bit count is transferred to the upper byte of the Down Counter with the low byte set to zero. Selecting the two-byte option prepares the counter/timer to receive two bytes, placing the first into the lower byte of the Down Counter and the second into the upper.

A new count can be written into the counter/timers at any time without reprogramming the TMD register. Care must be taken to be consistent with the Read/Write Mode each time the new count is programmed. As an example, assume that counter/timer 0 is programmed with a Read/Write Mode of two bytes. Two bytes must be written to counter/timer 0 each time a new count is specified. The same applies for reading counter/timer 0; that is, two count bytes must be read at a time.

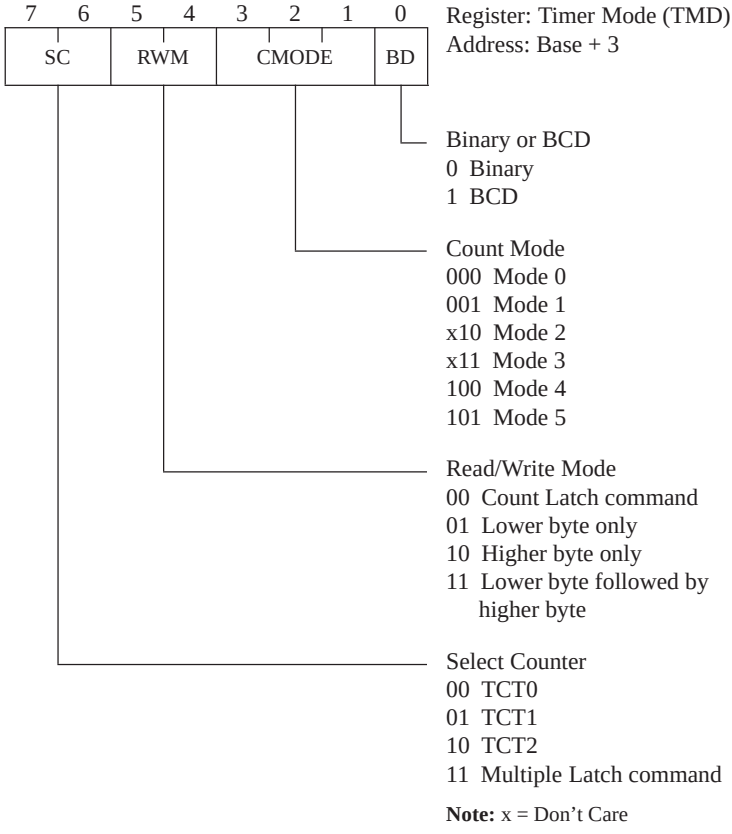



*Figure 7–2. Counter/Timer General Mode Register.*

Each counter/timer must be programmed to operate in one of six possible count modes. Selection of the count mode is based on the needs of the application. The counting operation of each counter/timer is programmed as either binary or Binary-Coded-Decimal (BCD). The range of a counter/timer programmed for binary operation is 0 to FFFFh, while the BCD operation range is 0 to decimal 9999.

## Count Latch Mode

The Count Latch Mode, shown in Figure 7-3, requires that the first six bits be set to a logical 0. The Select Counter bits specify which counter/timer's data is to be latched.

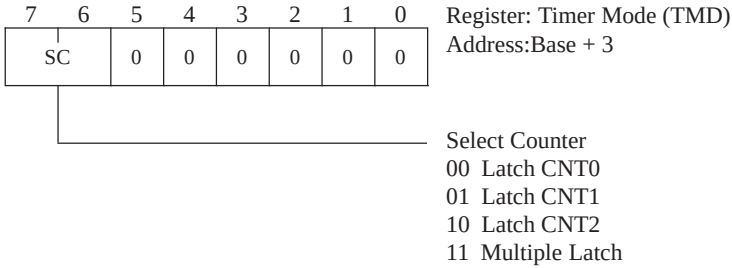



*Figure 7–3. Counter/Timer Count Mode Register.*

Multiple Latch Mode

Programming the Select Counter bits of the TMD to logical 1s defines the Multiple Latch command (see Figure 7-4). The CNT0, CNT1, and CNT2 bits select which of the counter/timers will be latched. The Status Latch and Count Latch bits determine whether the status, the count, or both are to be latched. The status must be latched to be read. The count can be read without being latched, but will be invalid if it is changing at the time of the read.

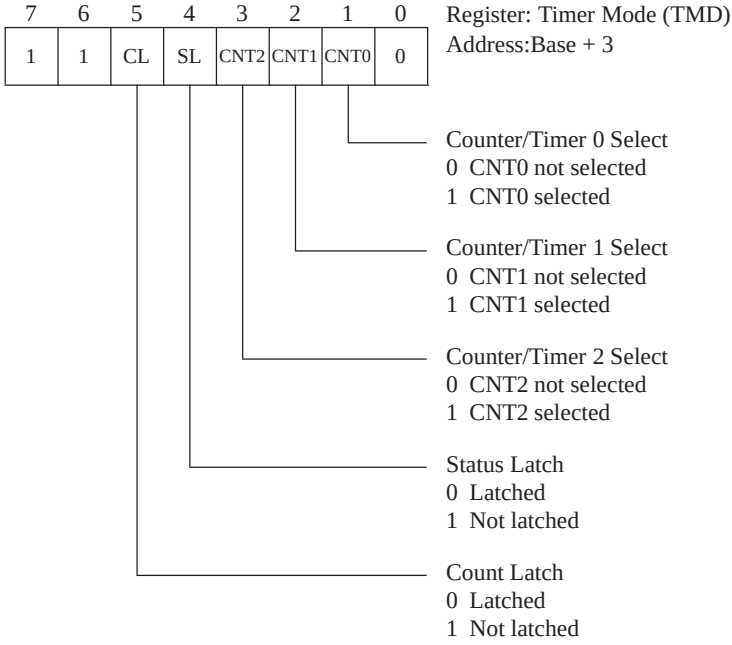



*Figure 7–4. Counter/Timer Multiple Mode Register.*

## Count Registers

The Count register is illustrated in Figure 7-5. Unlike the Mode register, there is one Count register for each of the three counter/timers. The Count register transfers count values to and from the Down Counter. The 16-bit register is programmed with a high byte, low byte, or both high and low byte, as specified with the Read/Write Mode bits in the Mode register. If the high byte or low byte mode is selected, only one read or write operation is needed for data transfers. Two read or write operations are required for the two-byte mode, with the low byte transferred first, followed by the high byte.

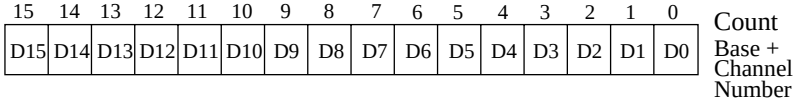



*Figure 7–5. Counter/Timer Count Register.*

## Status Registers

Each counter/timer includes a Status register. The status can be read from the Status register at any time. The format for the status is shown in Figure 7-6.

The first six bits of the Status register provide information about the programmed state of the selected counter/timer. This information is in the Status register to prevent the application software from having to save it. The Null Count bit flags when the last count written to the Count register is transferred to the Down Counter. This is designed to prevent the application software from reading the Down Counter before it is updated to the last count written. The Output Level bit contains the current state of the counter/timer output (TOUT).

The Multiple Latch command must be used to read the status. The number of required read operations depends on the Read/Write Mode and the Multiple Latch command. If the Read/Write Mode is high byte or low byte and only the status is latched with the Multiple Latch command, one read operation is all that is needed. Two reads are required if both the status and the count are latched by the Multiple Latch command. The first read is for the status and the second is for the data. If the Read/Write Mode is programmed for two-byte transfers and the Multiple Latch command is programmed to latch only the status, one read is all that is required. If both the status and data are latched, three reads are required. The first read is for the status, and the next two are for the low byte and high byte of the count, respectively.

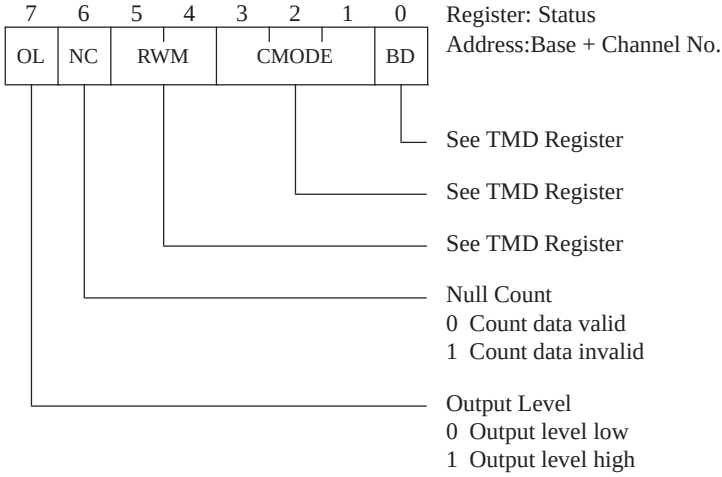



*Figure 7–6. Counter/Timer Status Register.*

# OPERATION

## Reset

The TCU registers are not initialized to a default state after power on or reset.

## Count Latch Command

The count of any counter/timer can be read at any time. The Count Latch command must be used to guarantee accurate results. Without the Count Latch command, the count may change during the read operation and the data will be undefined. After the latch command is specified in the Mode register, the present value of the Down Counter is latched into the Count Latch. This value remains in the latch until it is read by the CPU or the counter/timer is reinitialized. More than one latch command can be used to latch more than one counter/timer at a time.

The count latched at the time of a latch command will not be changed if a second latch command is specified for the same counter/timer before the count is read by the CPU.

The count must be read according to the programmed Read/Write Mode. If the Read/Write Mode specifies two-byte operation, two read operations from the Count register are required, the first for the low byte and the second for the high byte. If the Read/Write Mode is not programmed for two-byte operation, only one read operation is needed.

## Multiple Latch Command

The Multiple Latch command extends the capabilities of the Count Latch command. The Multiple Latch command is used to selectively latch the count and status of any or all counter/timers simultaneously. This command is the only method of reading the status of a counter/timer.

Once the status or count is latched with this command, it is unaffected by further latch commands. The latched data must be read or the counter/timer reinitialized before the Count Latch and Status Latch will resume tracking the operation of the Down Counter.

The count and status must be read according to the Read/Write Mode programmed for the selected counter/timer. Two read operations are required if the Read/Write Mode specifies two-byte operation and the status is not latched. The first read is for the low byte of the count and the second for the high byte. If the status is latched, three read operations are required with the status being the first byte read, followed by the low and then the high bytes of the count. One read operation is all that is needed for single-byte mode, if the status is not latched. For single-byte operations with latched status, two read operations are needed, the first for the status and the second for the data.

## Modes of Operation

There are six possible count modes for the counter/timers. There are restrictions for counter/timers 0 and 1 because the control inputs are not available externally.

### Mode 0 - Interrupt on Count Termination

In Mode 0 operation, the counter/timers count down the programmed number of counts and transition the output signal. This mode is commonly used to count external events. Counter/timer 2 fully supports this mode with the TCTL input available through connector J3 of the ZT 8832. Timer/counters 0 and 1 support Mode 0 with the exception of the TCTL input signal.

Examples of Mode 0 operation are shown in Figure 7-7. TOUT remains low until the Down Counter reaches zero, at which point TOUT returns high. TOUT stays high until the Count or Mode registers are reprogrammed.

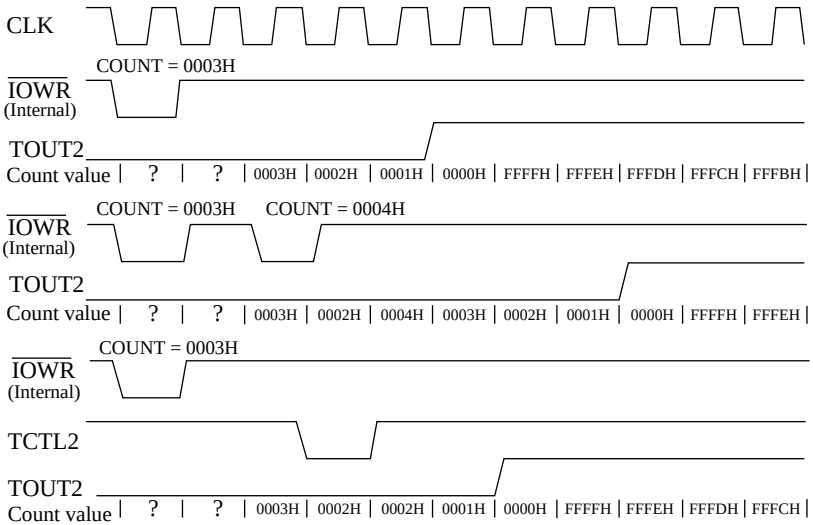



*Figure 7–7. Mode 0 Operation.*

The TCTL signal is used to enable and disable the counting operation. The counting operation is enabled if TCTL is high and disabled if TCTL is low. TCTL has no effect on the TOUT signal.

The Down Counter does not begin decrementing until the clock pulse after the count is programmed. This is because it takes one clock pulse to transfer the count from the Count register to the Down Counter. This means that for an initial count of N, TOUT does not transition high until N + 1 clock pulses later.

If a new count is written to the counter/timer before the previous count is decremented to zero, the Down Counter begins counting from the new count after the next clock pulse. If the count is two bytes long, the first byte disables counting and lowers TOUT. Not until one clock pulse after the second byte is programmed will the entire count be transferred to the Down Counter.

It is possible to start the Down Counter with the TCTL signal by programming the initial count with TCTL low. The programmed count will be transferred to from the Count register to the Down Counter on the next clock pulse. TOUT will not go low until N clock pulses after TCTL goes high.

### Mode 1 - Retriggerable One-Shot

In Mode 1, counter/timer 2 is triggered to generate a pulse of programmed length. Applications can use this mode to signal the occurrence of a single external event without having to monitor for a change in count. This mode is supported only by counter/timer 2.

Examples of Mode 1 operation are shown in Figure 7-8. TOUT is high until one clock pulse after TCTL triggers the count operation to start. This is the start of the one-shot pulse. The pulse terminates by lowering TOUT after the count reaches zero. TOUT remains low until one clock pulse following the next TCTL trigger.

The Down Counter is armed after the Mode and Count are programmed. A TCTL trigger transfers the count from the Count register to the Down Counter and lowers TOUT on the next clock pulse. This means an initial count of N results in a low level pulse duration of N clock pulses. Since the one-shot is retriggerable, TOUT remains low for N clock pulses after any trigger.

Programming the counter/timer with a new count does not affect the current pulse width unless a TCTL trigger occurs. If this happens, the new count is transferred to the Down Counter and the TOUT signal remains low for the new number of programmed clock pulses.

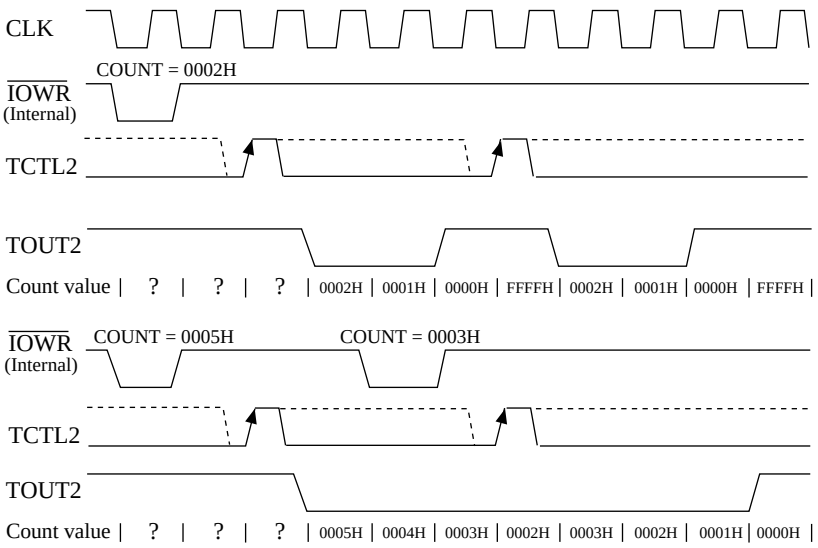



*Figure 7–8. Mode 1 Operation.*

### Mode 2 - Rate Generator

The output of a counter/timer in Mode 2 is high until the programmed count reaches one, then pulses low for a single count before returning to a high value and repeating the operation.

Counter/timer 2 fully supports Mode 2 operation with TCTL available through connector J3 of the ZT 8832. Counter/timers 0 and 1 support Mode 2 with the exception of the TCTL input. The output of counter/timer 0 is connected to INT1 of the interrupt controller, which makes it the first choice for generating periodic interrupts. The output of counter/timer 1 can be connected to INT2 of the interrupt controller by programming the OPCN V40 configuration register. Counter/timer 1 should be used to generate periodic interrupts only when the V40 serial controller is not needed.

Figure 7-9 includes examples of Mode 2 operation. TOUT remains high until the programmed number of counts is decremented to one. At this point, TOUT transitions low for only one clock pulse. As TOUT becomes high, the Down Counter is reloaded automatically and the process is repeated. The entire cycle is periodic, with a period of N clock cycles, for an initial count of N.

The TCTL signal provides a means of stopping the Down Counter for counter/timer 2 only. If TCTL is high, the operation continues as defined above. If TCTL is lowered during the count operation, the Down Counter stops and the TOUT signal is immediately set high. A TCTL trigger causes the count to be transferred from the Count Latch to the Down Counter, and operation to be resumed.

Programming the Count register with a new count while the Down Counter is counting does not affect the current counting sequence. The new count will be loaded automatically after the current cycle is completed, or on the occurrence of a TCTL trigger.

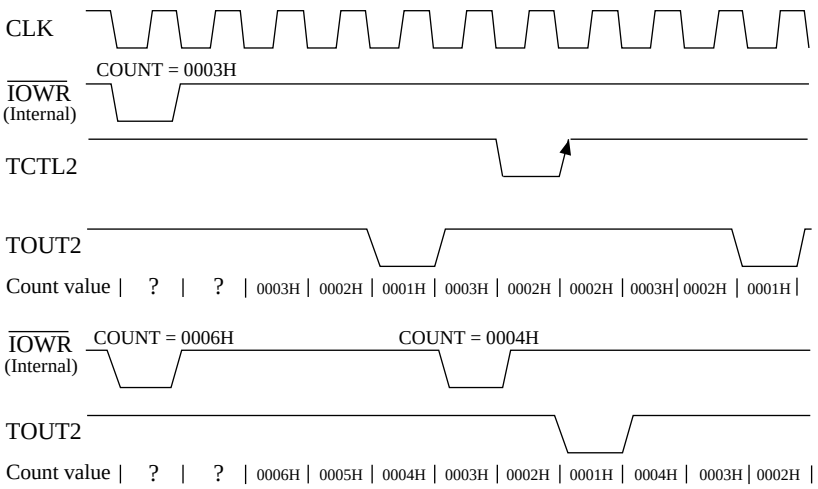



*Figure 7–9. Mode 2 Operation.*

### Mode 3 - Square Wave Generator

The most common use for Mode 3 is baud rate generation. The V40 Serial Control Unit requires that counter/timer 1 be configured in Mode 3 to provide the serial transmit and receive clock. The only difference between Mode 2 and Mode 3 is the duty cycle of the counter/timer output. In Mode 2, the output pulses low for the last count, while in Mode 3 the output is low for half of the programmed count. All counter/timers support Mode 3 with the exception that counter/timers 0 and 1 do not have a TCTL input. The TCTL input for counter/timer 2 is available through connector J3 of the ZT 8832.

Refer to Figure 7-10 for examples of Mode 3 operation. The TOUT signal remains high until the programmed number of counts is decremented by half. TOUT then goes low for the remainder of the count. The entire operation is periodic because it repeats automatically. An initial count of N produces a square wave with a period of N clock cycles.

The TCTL signal provides a means of suspending the Down Counter for counter/timer 2 only. If TCTL is high, the operation continues as defined above. If TCTL is lowered during the count operation, the Down Counter stops and the TOUT signal is immediately set high. A TCTL trigger causes the count to be transferred from the Count Latch to the Down Counter, and counting to resume.

The operation of the Down Counter is not affected if a new count is written while it is counting. The new count will be loaded into the Down Counter at the end of the current half-cycle or after a TCTL trigger.

Actual counter operation is different for even and odd counts. For even counts, the initial count is loaded in one clock pulse and decremented by two on succeeding clock pulses. TOUT changes state when the count expires and the operation is repeated. For odd counts, the initial count minus one is loaded in one clock pulse and decremented by two on succeeding clock pulses. TOUT goes low one clock pulse after the count expires and the operation is repeated. For an odd count of N, TOUT is high for (N + 1)/2 and low for (N -1)/2.

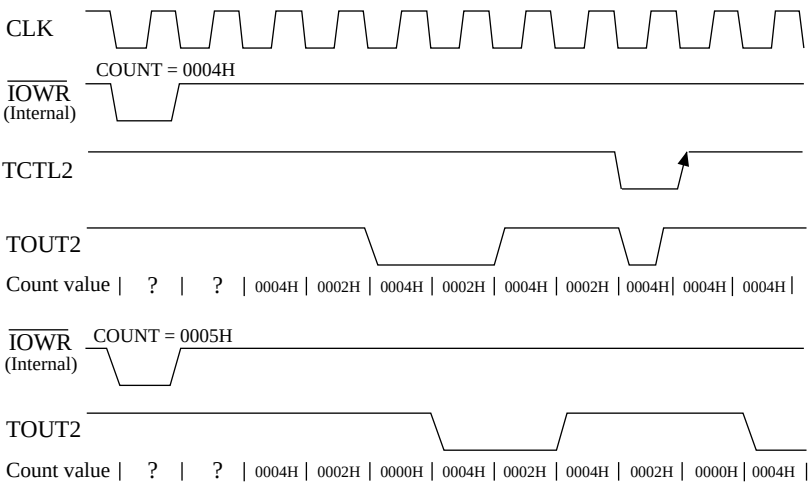



*Figure 7–10. Mode 3 Operation.*

### Mode 4 - Software Triggered Strobe

Mode 4 operation provides a means of generating a hardware delay triggered by software. Counter/timer 2 fully supports Mode 4 operation with the TCTL input available through connector J3 of the ZT 8832. Counter/timers 0 and 1 support Mode 4 with the exception of the TCTL input signal.

Examples of Mode 4 operation are included in Figure 7-11. Counting begins automatically one clock pulse after the count is programmed. TOUT remains high until the count reaches zero. At a count of zero, TOUT goes low for one clock pulse and returns high. The count is not decremented until after it is transferred to the Down Counter by the first clock pulse. This means TOUT does not go low until N + 1 clock pulses later, for an initial count of N.

The TCTL signal can be used to stop the Down Counter for counter/timer 2. A high TCTL signal enables counting and a low on TCTL stops it. TOUT is not affected by TCTL.

A new count can be written to the counter/timer while it is counting. The new count will be transferred to the Down Counter on the next clock pulse and counting will continue from the new value. A two-byte count will not be transferred to the Down Counter until the second byte is written.

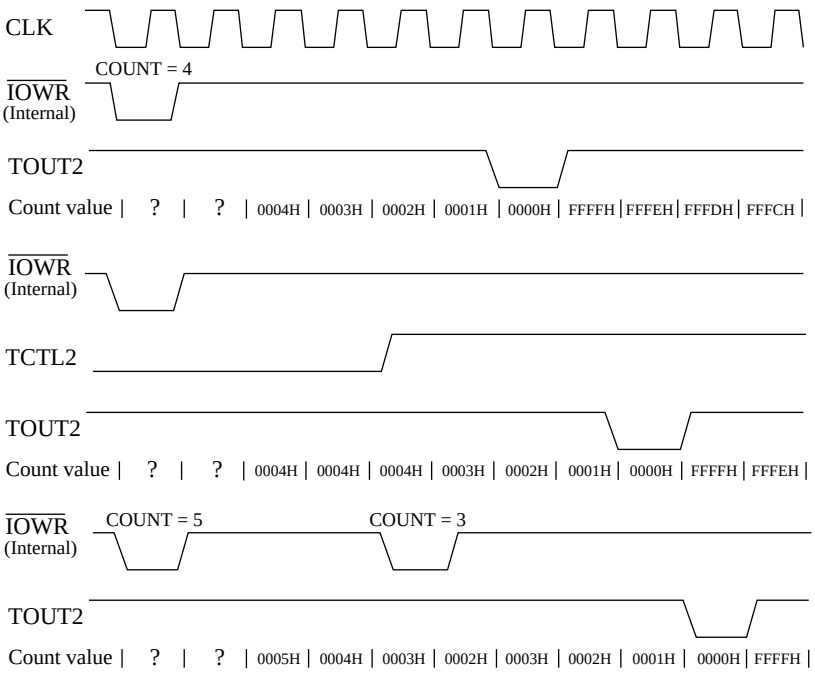



*Figure 7–11. Mode 4 Operation.*

### Mode 5 - Hardware Triggered Strobe

Mode 5 operation provides a means of generating a hardware delay triggered by a hardware signal. This mode is similar to Mode 4 except for the counting process, which is started by a signal external to the ZT 8832. Counter/timer 2 is the only one that supports Mode 5.

Examples of Mode 5 operation are illustrated in Figure 7-12. A TCTL trigger transfers the programmed count from the Count Latch to the Down Counter and starts the counting process. The TOUT signal remains until the count reaches zero. At a count of zero, TOUT goes low for one clock pulse before returning high. It takes one clock pulse to transfer the initial count from the Count Latch to the Down Counter. This means for an initial count of N, TOUT will not go low until N + 1 clock pulses after the TCTL trigger.

The counting sequence can be retriggered at any time with TCTL. If a TCTL trigger occurs while the Down Counter is operating, the programmed count is transferred to the Down Counter on the next clock pulse and counting starts over. Writing a new count will not affect counter/timer operation. If a TCTL trigger occurs after a new count is programmed but before the current count expires, the new count will be loaded and counting will begin at this new value.

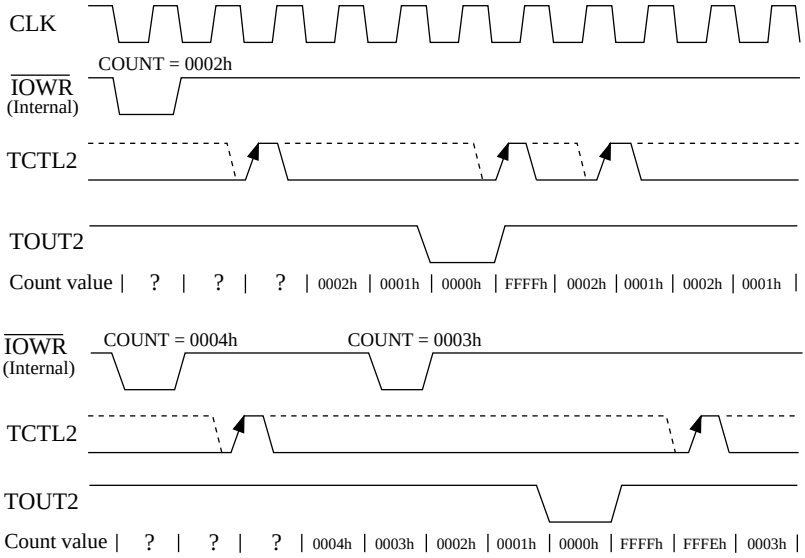



*Figure 7–12. Mode 5 Operation.*

## Programming

The TCU is enabled and mapped into an I/O address range using the V40 configuration register. The TCU includes three counter/timers and four programmable registers. The first register to be programmed is the Mode register. This register selects one of the three counter/timers and information needed for initialization. The next register to be programmed is the Count register for the counter/timer being used. When programming the Count register, it is important to always write the number of bytes specified in the Read/Write Mode bits of the Mode register.

The Count Latch and Multiple Count Latch commands are recommended for reading count data from the Status register. When reading the count, the number of bytes read must be consistent with the Read/Write Mode bits programmed to the counter/timer. The low byte is transferred on the first read operation of a two-byte read. The Multiple Latch command must be used to read the status. If the count and status are latched with the Multiple Latch command, add one to the number of reads defined by the Read/Write Mode for the status. As an example, if the Read/Write Mode specifies two bytes, three read operations are required: one for the status, one for the low byte of the count, and one for the high byte of the count, respectively.

Chapter 8

# INTERRUPT CONTROLLER (V40)

**Contents** **Page**

## OVERVIEW

This chapter describes the Interrupt Control Unit (ICU) and provides register descriptions.

The ICU is a programmable interface between interrupt-generating peripherals and the CPU. The ICU monitors eight interrupt inputs with programmable priority. When peripherals request service, the ICU interrupts the CPU with a pointer to a service routine for the highest priority device. This type of interrupt management is needed for an efficient interface between the CPU and supporting peripheral devices, such as serial controllers and counter/timers. The major features of the ICU are as follows:

- Eight individually maskable interrupts
- Level- or edge-triggered interrupts
- Fixed and rotating prioritization
- Status available for polled operation
- Functionally equivalent to the 8259

## ZT 8832 SPECIFICS

The inputs to the interrupt controller are connected as shown in Table 8-1.

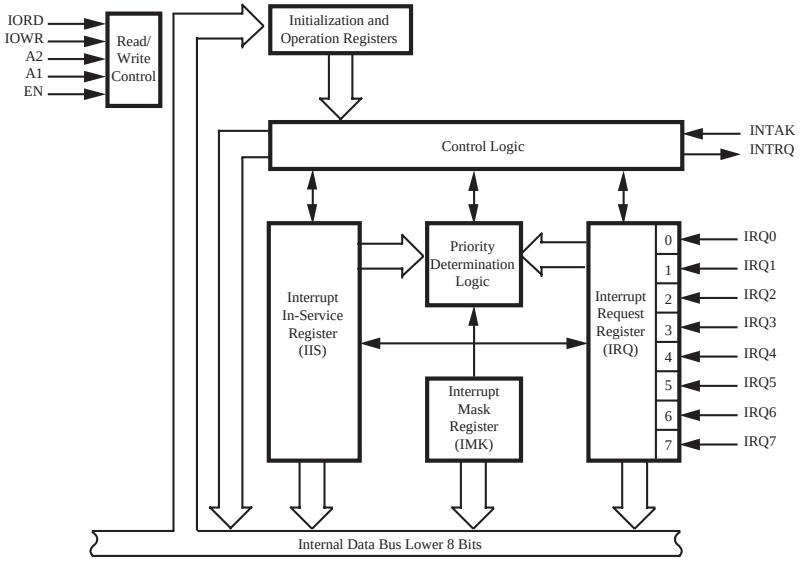
Table 8-1
Interrupt Controller Inputs.

| Input | Connection |
|---|---|
| IRQ0 | Counter/timer 0 output |
| IRQ1 | V40 serial port or watchdog timer stage 1 |
| IRQ2 | Counter/timer 1 output<br>or SBX expansion module interrupt 0 |
| IRQ3 | SBX expansion module interrupt 1 |
| IRQ4 | 82050 serial port |
| IRQ5 | STD bus control port |
| IRQ6 | Connector J3 pin 8 |
| IRQ7 | Connector J3 pin 10 |

The interrupt controller includes a cascade mode not supported by the ZT 8832. Cascade mode is a scheme to expand the number of interrupt request inputs by connecting seven slave interrupt controllers to the inputs of a master interrupt controller. In this scheme, the master interrupt controller must provide a cascade address to the slave controllers during interrupt acknowledge. The cascade address is not provided by the ZT 8832.

## FUNCTIONAL DESCRIPTION

The format of the ICU programmable registers is the same as the industry standard 8259 Programmable Interrupt Controller, with one exception: 8085 operation is not supported. The ICU can be divided into seven functional blocks as shown in Figure 8-1.




[1] See Jumper Configuration tables for signal selection information.

*Figure 8–1. ICU Block Diagram.*

### Interrupt Request Register

The eight interrupt requests, IRQ0 through IRQ7, are input to the Interrupt Request register. The 8-bit IRQ register maintains a bit position for each interrupt input. A requesting interrupt sets the bit position to a logical 1. The bit is automatically reset during the interrupt acknowledge cycle. The IRQ register can be read by the application program to determine the status of the requesting interrupts.

### Interrupt Mask Register

All interrupt requests are latched by the Interrupt Request register. The Interrupt Mask register acts as a programmable filter to selectively disable requesting interrupts from being serviced. The IMK is an 8-bit register with one bit position for each interrupt input. Setting a bit to a logical 1 prevents the respective interrupt request from being transferred from the Interrupt Request register to the Interrupt In-Service register.

### Priority Resolver

All interrupt requests are latched into the Interrupt Request register. Those not masked by the Interrupt Mask register are input to the Priority Resolver to determine which is to be serviced. The interrupt request with the highest priority is transferred from the Interrupt Request register to the Interrupt In-Service register during the interrupt acknowledge cycle. The ICU includes several programmable operating modes that define the rules by which the Priority Resolver determines the highest priority interrupt request. These modes range from all inputs having equal priority to rotating the priorities each time an interrupt is serviced.

### Interrupt In-Service Register

The 8-bit Interrupt In-Service register maintains a bit position for each interrupt request that is currently being serviced. More than one bit can be set if an interrupt is currently under service and a second interrupt request is acknowledged. The IIS register is read to determine the status of the interrupts currently being serviced. A logical 1 in a bit position means that the interrupt is currently being serviced.

### Control Logic

This functional block directs the operation of the other ICU blocks based on the programmed mode of operation. The Control Logic also interfaces to the CPU for interrupt request and acknowledge signals. An interrupt request is generated to the CPU if an ICU input has the correct priority and is not masked. If the CPU interrupts have been enabled with the "set interrupt" command, it will respond with an interrupt acknowledge.

### Read/Write Control Logic

The Read/Write Control Logic controls command and data transfer between the ICU and the CPU. This functional block selects an ICU register and determines the direction of data travel based on the address and I/O control inputs.

### Initialization and Operation Registers

The ICU is programmable to provide flexibility in the way interrupts are handled. The ICU is initialized by writing up to four 8-bit values called Interrupt Initialization Word 1 through Interrupt Initialization Word 4. Once initialized, the operation of the ICU is controlled with three 8-bit values called Interrupt Mask Word, Interrupt Priority and Finish Word, and Interrupt Mode Word.

## PROGRAMMABLE REGISTERS

The ICU is initialized with Interrupt Initialization Word 1 (IIW1) through Interrupt Initialization Word 4 (IIW4). Once initialized, the operation of the ICU is controlled with the Interrupt Mask Word (IMKW), Interrupt Priority and Finish Word (IPFW), and Interrupt Mode Word (IMDW). Three status words can also be read to interrogate the operation of the ICU: the Interrupt Request (IRQ), Interrupt In-Service (IISI), and Interrupt Poll (IPOL). Note that the "word" reference does not mean the values are 16 bits; all communication to the ICU is done through 8-bit data.

All initialization, operation, and status words are accessed through two I/O addresses, as shown in Table 8-2. As might be expected, a specific sequence of read and write operations is needed to pass multiple bytes through a single I/O address. A complete description of all the programmable words and how they are accessed is given on the following pages.

Table 8-2
ICU Register Addressing.

| Address | Value | Operation | Page Number |
|---|---|---|---|
| Base + 0 | IRQ, IIS, IPOL | Read | 8-16 |
| Base + 0 | IIW1 | Write | 8-8 |
| Base + 0 | IMDW, IPFW | Write | 8-13,8-15 |
| Base + 1 | IMKW | Read/Write | 8-12 |
| Base + 1 | IIW2, IIW3, IIW4 | Write | 8-8 |

## Initialization Words (IIW1, IIW2, IIW3, and IIW4)

The ICU must be initialized before it can be used. Initialization consists of writing from two to four bytes called interrupt initialization words.

The sequence in which these words are programmed is outlined in the flow chart shown in Figure 8-2. IIW1 and IIW2 must be programmed during any initialization sequence. IIW3 is not supported by the ZT 8832 and should be initialized to zero. IIW4 may or may not be programmed as required by the application. The ICU decodes the first two bits of IIW1 to know whether or not to expect IIW4 and IIW3, respectively.

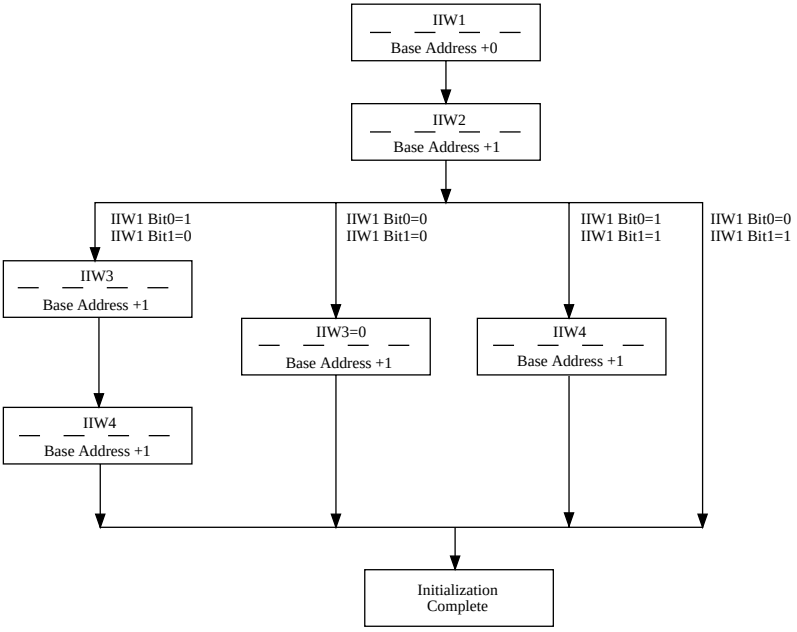



*Figure 8–2. Interrupt Initialization Programming.*

IIW1 and IIW2

Interrupt Initialization Words 1 (IIW1) and 2 (IIW2) are required for ICU initialization. The IIW1 register, shown in Figure 8-3, is divided into two fields labeled II4 (Interrupt Initialization 4) and LEV (Level). The II4 bit selects whether or not the IIW4 register is to be programmed. If II4 is set to a logical 1, the ICU expects IIW4 to be written as part of the initialization. The LEV bit of IIW1 selects between level- or edge-triggered interrupt request inputs. Setting LEV to a logical 1 selects level-triggered inputs, while a logical 0 selects edge-triggered.

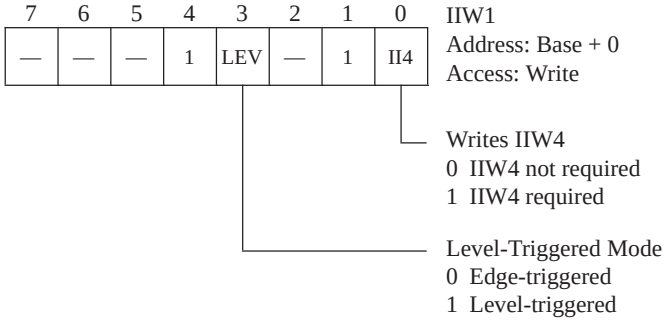



*Figure 8–3. Interrupt Initialization Word 1.*

The ICU responds to an interrupt acknowledge by supplying the CPU with an interrupt vector based on which interrupt generated the request and the value programmed into IIW2. The format for IIW2 is shown in Figure 8-4. Bits V3 through V7 define the upper 5 bits of the vector address.

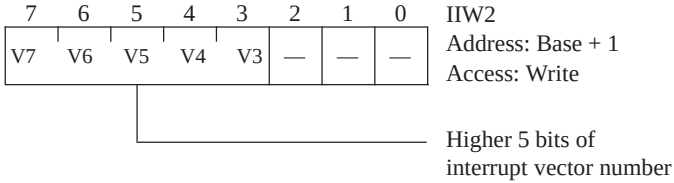



*Figure 8–4. Interrupt Initialization Word 2.*

IIW3

The ZT 8832 does not support cascading the interrupt controller inputs to other interrupt controllers. Bits 0 through 7 must be programmed with logical 0s, as shown in Figure 8-5.

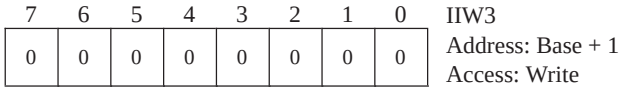



*Figure 8–5. Interrupt Initialization Word 3.*

IIW4

Figure 8-6 shows the architecture for IIW4. IIW1 must be programmed with a logical 1 in the II4 bit if IIW4 is used. A logical 1 in the SFI bit enables the Self Finish Interrupt and a logical 0 disables it. The interrupt service routine must include an EOI command when the Self Finish Interrupt is disabled.

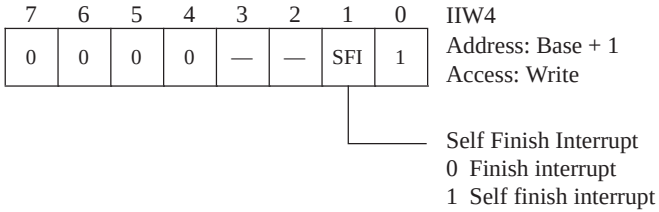



*Figure 8–6. Interrupt Initialization Word 4.*

## Operation Words (IMKW, IPFW, and IMDW)

Once initialized, the operation of the ICU is controlled with three 8-bit values called the Interrupt Mask Word (IMKW), Interrupt Priority and Finish Word (IPFW), and the Interrupt Mode Word (IMDW). The Operation Words can be transferred in any sequence to perform such functions as enabling and disabling individual interrupt requests and changing interrupt priorities.

IMKW

The IMKW masks interrupt request inputs. Interrupts are masked by writing IMKW to the IMK register. IMKW can be read directly from the IMK register to determine the current status of the mask. This eliminates the need for application software to maintain a copy of the mask in program memory.

As shown in Figure 8-7, each of the eight bits in IMKW represents an interrupt input. Bit M0 is used to mask IRQ0, M1 is used to mask IRQ1, and so on. Setting a bit in IMKW to a logical 1 prevents the interrupt request for the respective input from being acknowledged by the ICU. The interrupt request is latched in the IRQ register but it never reaches the IIS register.

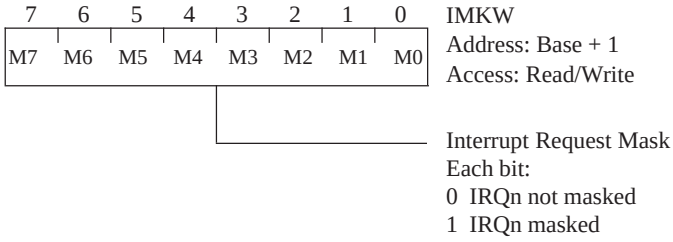



*Figure 8–7. Interrupt Mask Word.*

IPFW

IPFW selects fixed or rotating priorities and the method of informing the ICU that an interrupt has been serviced. Operation of the ICU can be changed at any time by writing a new IPFW. Refer to Figure 8-8 when programming the IPFW.

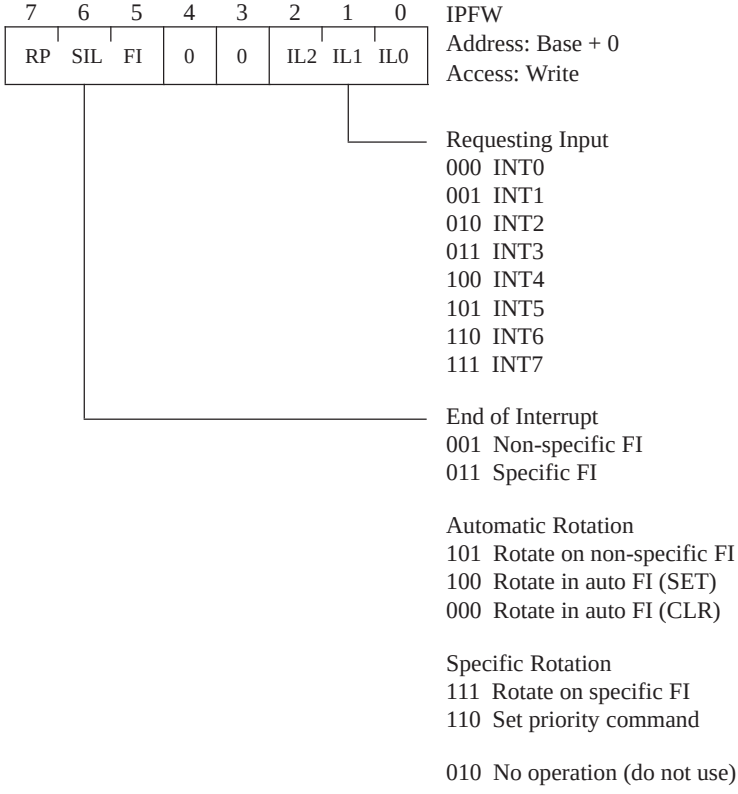



*Figure 8–8. Interrupt Priority and Finish Word.*

The IL0 through IL2 bits designate an interrupt level. This level is used by certain combinations of the FI, SIL, and RP bits either to reset an interrupt request that has been recognized or to set a specific priority.

The ICU uses the IIS register to keep track of which interrupts are being serviced and their relative priorities. The ICU updates the IIS register based on a Finish Interrupt command. There are three methods of generating the finish interrupt command: specific FI, nonspecific FI, and automatic FI. The automatic FI is programmed with IIW4. The specific and nonspecific FI are selected with the FI bit. A logical 1 in FI enables the specific or nonspecific FI, based on the SIL and RP bits.

The SIL bit enables bits IL0 through IL2 for selected operations. IL0 through IL2 indicate an interrupt level to be reset during the finish interrupt commands or a new priority for priority rotation commands.

The ICU provides several methods of establishing priorities for the interrupt request inputs. The RP bit selects the priority rotation options. A logical 1 in the RP bit indicates that rotation in priorities is to take place based on the values of the FI and SIL bits. A logical 0 in the RP bit means that no priority rotation will take place.

IMDW

IMDW controls the method of reading status from the ICU and enables a special type of interrupt masking.

The format of the IMDW is shown in Figure 8-9. The first two bits are used to select the IRQ and IIS registers so they can be read by the application software. A logical 1 in bit 0 selects the IIS register and a logical 0 selects the IRQ register. A logical 1 in the SR bit (bit 1) enables the reading of the IRQ and IIS registers.

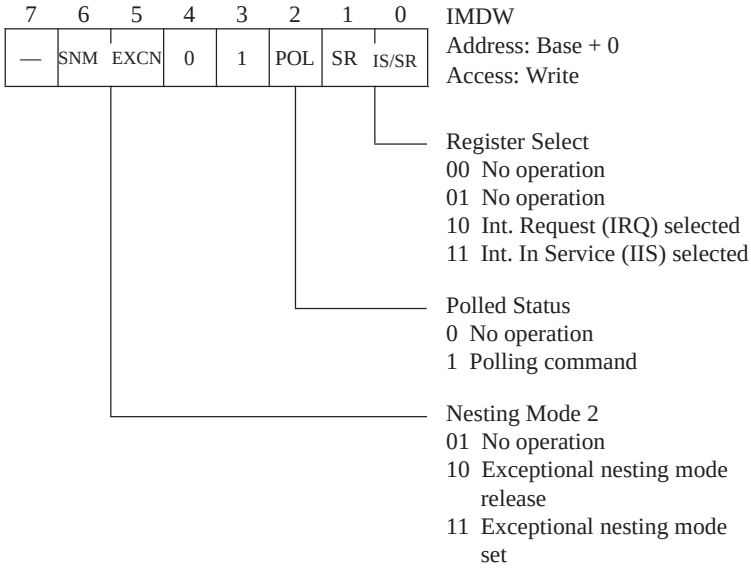



*Figure 8–9. Interrupt Mode Word.*

The POL bit selects the poll command. The two most commonly used methods of servicing peripherals in a microprocessor system are polling and interrupts. Although interrupts are the fastest method of servicing peripherals, using the ICU in a polled operation is still faster than polling each peripheral one at a time. Setting the POL bit to a logical 1 enables the reading of the poll status. The POL bit overrides the SR bit if both are set.

The EXCN and SNM bits can be programmed to enable or disable the exceptional nesting mode of operation. The exceptional nesting mode is armed by setting SNM (bit 6) to a logical 1. The EXCN bit (bit 5) can then be used to set or release the exceptional nesting mode. This operating mode is used to permit interrupts of lower priority than the one currently under service to be recognized.

## Status Words (IRQ, IIS, and IPOL)

Three 8-bit status words can be read from the ICU. These are the Interrupt Request (IRQ), Interrupt In-Service (IIS), and Interrupt Poll (IPOL). These words can be read at any time by programming the first three bits of the IMDW. Once the IMDW is programmed to select one of the status words, that word can be read as many times as needed. The IMDW must be programmed with a new value to read another of the status words. The formats of the IRQ, IIS, and IPOL are illustrated on the following pages.

IRQ and IIS

The IRQ and IIS status words, shown in Figure 8-10 below, are taken directly from the Interrupt Request register and Interrupt In-Service register, respectively. The IRQ status word contains all the interrupt levels requesting service. The IIS status word contains all the interrupt levels currently being serviced. Bit 0 of both status words corresponds to IRQ0, bit 1 corresponds to IRQ1, and so on.

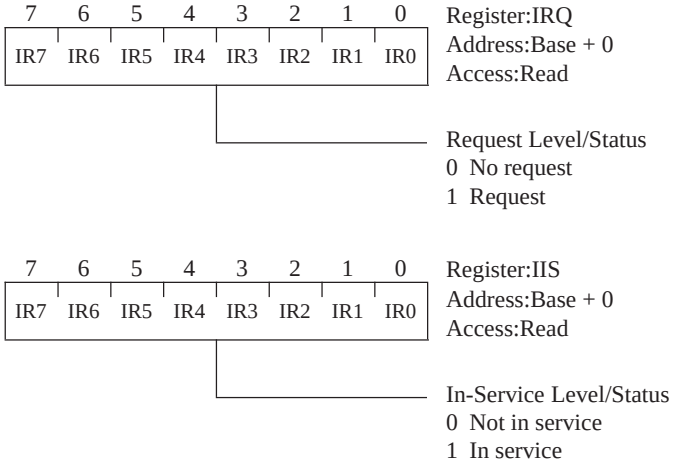



*Figure 8–10. Interrupt Status Registers IRQ and IIS.*

IPOL

The two most common methods of servicing peripherals in most applications are polling and interrupts. Although the ICU is designed primarily for interrupt control, it can be used in a polled system to increase the efficiency of servicing peripherals. The efficiency is increased because the CPU can poll the ICU to determine the status of several peripheral devices at one time.

Figure 8-11 shows the IPOL status word. Bits PL0 through PL2 define the highest priority interrupt input requesting service. For example, if all three bits are set to a logical 1, then IRQ7 is the highest priority request.

The INT bit (bit 7) indicates whether there are any interrupt requests. A logical 1 signals an interrupt request and a logical 0 signals no interrupt request. If INT is a logical 0, PL0 through PL2 are all set to a logical 1. The typical polling sequence is to set the POL bit in IMDW, read the IPOL register, and test the INT bit. If INT is a logical 1, decode PL0 through PL2 to determine which peripheral to service.

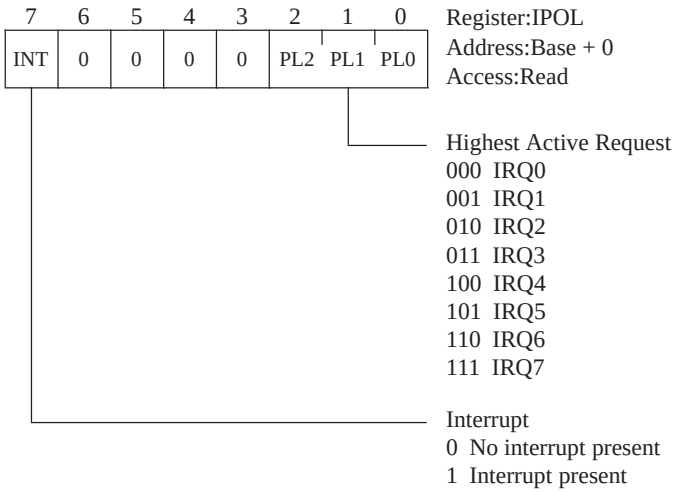



*Figure 8–11. Interrupt Status Register IPOL.*

# OPERATION

## Reset

The ICU registers are not initialized to a default state when power is applied to the ZT 8832 or after a reset. The OPCN V40 configuration register is initialized to disable the ICU, which disables interrupts.

## Interrupts

Most microprocessor systems include peripheral devices designed to perform specific tasks. Examples include counter/timers, serial controllers, and real-time clocks. The CPU has the job of managing the peripherals to meet the needs of specific applications. One method of peripheral management is for the CPU to start them on a job and periodically poll for completion. The problem with this method is that the overhead associated with polling reduces system throughput. As the number of peripheral devices increases, so does the amount of time spent polling. Interrupts provide the peripheral device with a method of informing the CPU when it is ready to be serviced. The following discussion explains how interrupting peripherals are serviced in a V40-based system.

A peripheral device needing service generates an interrupt request on one of the inputs to the ICU. The ICU responds by interrupting the CPU if the incoming interrupt is not masked and has the highest priority. Interrupt masking and priorities are explained later in this section. Interrupts must be enabled in the CPU before the CPU can recognize the interrupt request from the ICU. The CPU interrupts are disabled after reset and must be enabled using the "set interrupt" instruction. The CPU recognizes interrupts by storing the current program location (Instruction Pointer and Program Segment) and status (Processor Status Word) on the stack and entering an interrupt acknowledge cycle. The current program location and status are preserved on the stack as a return pointer that allows program execution to continue after the interrupt is serviced.

The V40 interrupt acknowledge cycle is two machine cycles long and looks much the same as two I/O read cycles. The difference is that the CPU interrupt acknowledge signal is pulsed low for two clock periods each machine cycle instead of the read signal. The first interrupt acknowledge pulse prepares the ICU to provide an interrupt vector on the second. Preparation includes freezing the state of the interrupts internal to the ICU so the highest priority request can be determined. The ICU supplies an interrupt vector onto the CPU data bus in response to the second interrupt acknowledge pulse.

The interrupt vector is an 8-bit value used to point the CPU to an interrupt service routine for the peripheral device being acknowledged. A unique vector is generated by the ICU for each interrupt request input. The CPU reads the vector from the data bus and multiplies it by four to point to the address of the service routine. The CPU then begins executing the interrupt service routine at this address. See the following section, entitled "Interrupt Vectors," for more information on the interrupt vector and how it relates to the address of the service routine. The CPU automatically disables interrupts at the start of the interrupt service routine and does not enable them again until the interrupt service routine is completed. This means the CPU will not acknowledge interrupt requests of a higher priority from the ICU until it is finished servicing the current request. The "clear interrupt" instruction can be used to enable interrupts before completion of the current service routine.

The "return from interrupt" is the last instruction of the interrupt service routine. This instruction enables interrupts and transfers program execution back to the interrupted program by restoring the Instruction Pointer, Program Segment, and Processor Status Word from the stack.

## Interrupt Vectors

The CPU responds to all external interrupt requests by reading an 8-bit value from the interrupt device that indirectly points to the subroutine used to service the device. The 8-bit value is called the interrupt vector and the subroutine used to service the device is the interrupt service routine.

The relationship between the interrupt vector provided by the ICU and the address of the interrupt service routine is illustrated in Figure 8-12. The ICU provides a unique 8-bit vector for each interrupt input, based on the IMKW value programmed. The CPU multiplies the vector by four to point to the memory location that contains the address of the service routine. The address of the interrupt service routine includes the Instruction Pointer and Program Segment. The application software must program the Instruction Pointer and Program Segment into the appropriate memory locations before the interrupt request is generated.

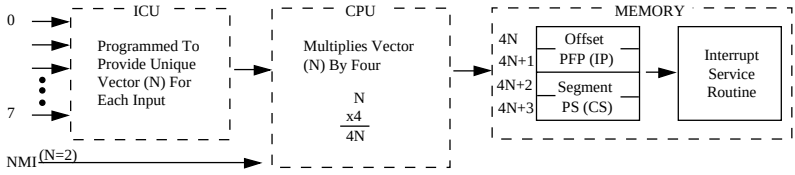



*Figure 8–12. CPU Interrupt Vector Processing.*

## Interrupt Nesting

Interrupt nesting is a powerful structure that allows an interrupt currently under service to be suspended while a second interrupt is serviced. The ICU supports nested interrupts. In most cases, the second interrupt must be a higher priority than the one currently being serviced. The exceptions to the rule are listed below.

- Any ICU using the Self Finish mode
- Any ICU using the Exceptional Nesting mode

The best way to understand nested interrupts is to examine the operation of the Interrupt Request (IRQ) and Interrupt In-Service (IIS) registers when multiple interrupt requests are generated. A bit in the IRQ register is set when an interrupt request is generated. More than one bit of the IRQ register will be set if more than one request occurs between interrupt acknowledge cycles. During the interrupt acknowledge cycle, the highest priority request is selected from the IRQ register and the vector for that request sent to the CPU. The corresponding bit of the IIS register is also set to flag that the request is currently being serviced. This bit remains set until a finish interrupt command is sent to the ICU.

In nested operation, further requests of the same or lower priority are inhibited while the IIS bit is set. A higher priority request can still interrupt the CPU and vector program operation to its own service routine if the "set interrupt" command is executed in the service routine of the device currently being serviced.

An example of nested interrupt execution is illustrated in Figure 8-13. This example assumes that the interrupt request inputs are prioritized with IRQ0 having the highest priority and IRQ7 having the lowest. The example begins in the main program. The "set interrupt" command must be executed before an interrupt is recognized after power on or reset. The CPU vectors program execution to the IRQ3 service routine in response to the IRQ3 request.

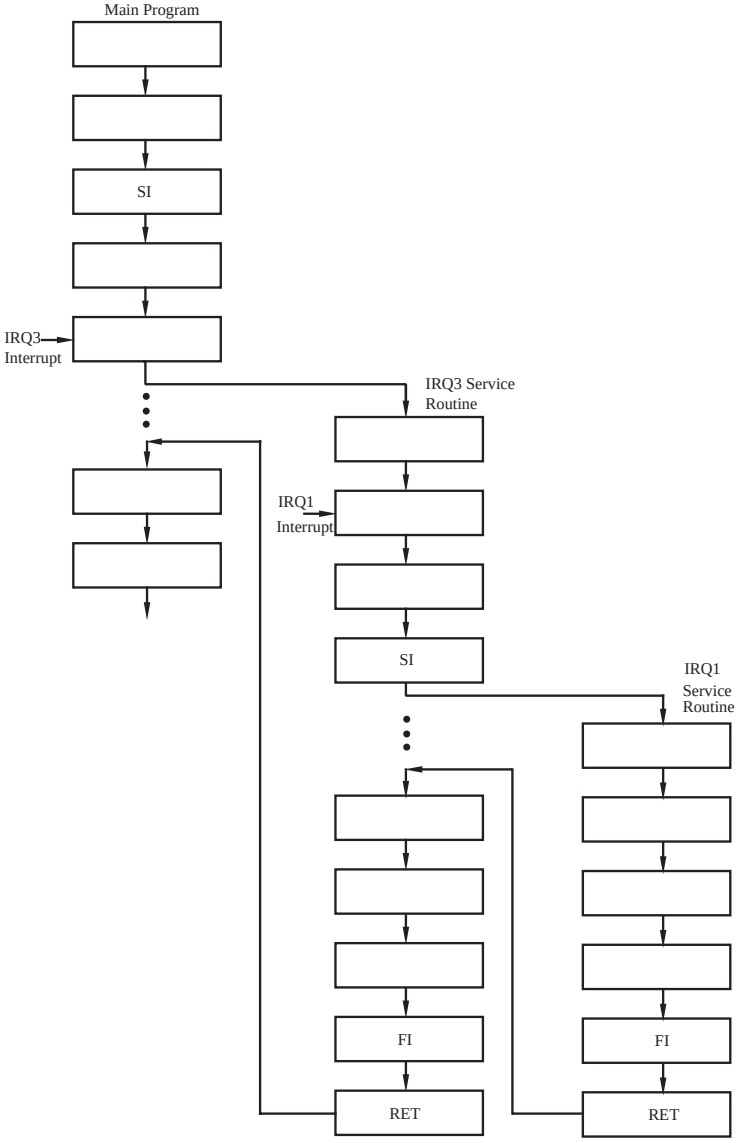



*Figure 8–13. Nested Interrupt Structure.*

Next, an IRQ1 request occurs. Since interrupts are automatically disabled upon entering a service routine, the IRQ1 request is not acknowledged until the "set interrupt" command is executed. Note that if a lower priority input such as IRQ4 generates a request, it will not be serviced until all higher priority requests are serviced. The CPU vectors program execution to the IRQ1 service routine.

At this point in the sequence, the IIS register has bits IRQ3 and IRQ1 set. This means that IRQ0 is the only input with a high enough priority to generate a request to the CPU. The IRQ1 service routine is terminated with the "finish interrupt" command. This command clears the IRQ1 bit in the IIS register. The "interrupt return" instruction vectors program execution back to the IRQ3 service routine. At this point IRQ0 through IRQ2 have high enough priority to generate an interrupt to the CPU. This routine is again completed with a "finish interrupt" command and "interrupt return" instruction.

## Level- or Edge-Triggered

The two primary methods of sensing interrupt requests are to sense the logical state (level) or the transition between logical states (edge) of the interrupt request inputs. The ICU is programmed for level- or edge-triggered interrupt sensing with the IIW1 register.

### Level-Triggered Sensing

If programmed for level-triggered mode, the ICU recognizes a logical 1 as an interrupt request. There is a level of inversion between the ICU and the interrupt request block. This means that all STD bus and frontplane (J3) interrupt requests are active low.

There is a minimum and maximum pulse width for the interrupt request input to guarantee proper operation. The minimum pulse width requirement states that the request must remain active until the falling edge of the second interrupt acknowledge pulse. The maximum amount of time the request remains active is determined by when the "finish interrupt" command is written. If the request remains active after the "finish interrupt" command is issued, a second interrupt is generated on the same request.

### Edge-Triggered Sensing

Edge-triggering is the second mode of interrupt sensing. The ICU recognizes a transition from a logical 0 to a logical 1 as an interrupt request. There is a minimum pulse width for the interrupt request input to guarantee proper operation. As in the level-triggered mode, the request must remain active until the falling edge of the second interrupt acknowledge pulse. Unlike the level-triggered mode, there is no maximum pulse width. This means that once the request transitions to the active state and is serviced, there are no further interrupts generated until the request returns inactive and then transitions back into the active state.

## Finish Interrupts

The ICU must be told when a service routine is completed so that the in-service bit of the IIS register can be cleared. The three finish interrupt formats that the ICU recognizes are the nonspecific finish interrupt, specific finish interrupt, and automatic finish interrupt.

### Nonspecific FI

The ICU accepts a nonspecific FI as an indication of interrupt service completion of the highest priority request. The advantage of the finish interrupt format is that you need not specify the level to be cleared. The catch is that the nonspecific FI command must be used only in operating modes in which the service routine of the highest priority request is the first one completed. Two instances in which the nonspecific FI command should not be used are if priorities are changed during an interrupt service routine and if the exceptional nesting mode is used.

### Specific FI

The second finish interrupt format is the specific FI command. This command includes the level of the request to be cleared. The specific FI command must be used if the highest priority interrupt is not always the first one completely serviced. Such is the case if the interrupt priorities are changed during a service routine (called specific rotation). If the nonspecific FI were used in this case, it is possible that the wrong in-service bit might be reset by the ICU. The specific FI can be used in all modes of ICU operation.

### Automatic Finish Interrupt

When programmed for automatic FI, the ICU automatically executes a nonspecific FI during the interrupt acknowledge cycle. The advantage of the finish interrupt format is that you need no longer issue a command to the ICU that a service routine is completed. Like the nonspecific FI, the automatic FI must be used only in operating modes in which the service routine of the highest priority request is the first one completed. Two instances in which the automatic FI command should not be used are if priorities are changed during an interrupt service routine and if the exceptional nesting mode is used.

When using the automatic FI format, you must disable interrupt requests during the execution of an interrupt service routine. This is because the IIS bit is cleared immediately after the interrupt acknowledge cycle or, in other words, before the service routine is started. If interrupts are enabled during the service routine, there is no indication that an interrupt is currently being serviced, and a second request can be generated by the same interrupt if level-triggered interrupts are used. This can continue until the level is removed, resulting in an undefined operation.

## Automatic Priority Rotation

Automatic priority rotation is used in applications with interrupt devices that are of equal priority. In this mode of operation, after an interrupt request is serviced the ICU rotates it into the lowest priority slot. This guarantees that all interrupt requests will be serviced. The nonspecific FI or specific FI commands can be used for automatic priority rotation.

### Rotate on Nonspecific FI

The ICU response to a rotate on nonspecific FI is the same as previously discussed, except that the interrupt request is rotated into the lowest priority position after the IIS bit of the Interrupt In-Service is cleared. The other ICU inputs are also rotated into the next higher priority position.

### Rotate on Self Finish FI

The ICU handles a rotate on self finish FI in much the same manner as the self finish FI previously discussed. The difference is that priority rotation is done automatically after the IIS bit is reset. The same precautions discussed under the self finish FI explanation still apply.

## Specific Priority Rotation

Specific rotation, like automatic rotation, can be used to change the priorities of the ICU inputs. With automatic rotation, the last request serviced is rotated into the lowest priority, and other ICU inputs are rotated up one level. With specific rotation, you specify which interrupt receives the lowest or highest priority. Either the set priority command or the rotate on specific FI command can be used to select this operating mode.

### Set Priority

The set priority command is used to define the priority of the ICU inputs. You specify the lowest priority interrupt as part of the command. If, for example, you define IRQ4 as part of the set priority command, then IRQ3 is automatically the second lowest, IRQ2 the third lowest, and so on up to IRQ5, which is the highest. The nonspecific FI command must not be used with the set priority command. This is because the nonspecific FI clears the highest priority IIS bit, which is not the one most recently serviced if the set priority command changed priorities. The self finish FI can be used because it clears the IIS bit during the interrupt acknowledge cycle or, in other words, before the set priority command can change the priorities. The specific FI command is the simplest to use because the level is not selected by the ICU, but specified by you.

### Rotate on Specific FI

The rotate on specific FI command is a combination of the set priority and the specific FI command. The command clears the ISR bit and changes the priority of interrupt inputs, based on the specified level. The primary advantage of this command is that it performs the functions of two commands in one operation.

## Interrupt Masking

The ICU inputs are all maskable. The "clear interrupt" instruction can be executed to disable all ICU inputs from generating interrupts. In certain applications it may be necessary to disable, or mask, selected ICU inputs. The ICU Interrupt Mask register (IMKW) includes one bit for each ICU input to permit selective masking. An interrupt, masked or not, flags a request by setting a bit in the IRQ register. If the request is masked it is not passed to the IIS register until the mask bit is cleared. If the interrupt input to the ICU is removed before the mask bit is cleared, the IRQ bit is reset and the request is missed.

In certain instances it may be necessary to enable interrupts of a lower priority than the one currently being serviced. The special mask mode can be used to enable all levels of interrupts except the one being serviced. The special mask mode is set and cleared using the IMDW register. A non-specific FI must not be used if the ICU is programmed for the special mask mode. This is because the nonspecific FI will not clear an IIS bit if it is masked. It is best to use the specific FI command when operating in the special mask mode.

## Interrupt Status

The Interrupt Request (IRQ), Interrupt In-Service (IIS), and Interrupt Mask (IMKW) registers are available to the programmer. The IRQ and IIS registers are read by first writing the appropriate read register command to the ICU IMDW register. After the read register command is written, the selected register can be read any number of times. It is possible for an interrupt to occur after the read register command is written to the IMDW register and before the selected register is read. The interrupt service routine could alter the IMDW register, causing the wrong register to be read. This is prevented by disabling interrupts with the "disable interrupt" instruction before programming the IMDW register, and not enabling them until after reading the register. The IMKW register can be read directly from the ICU without first writing to the IMDW register.

The ICU supports a polled operation for systems not wishing to use interrupts. Although not as efficient as an interrupt-driven system, it is more efficient than having to poll eight different I/O ports to see if a peripheral is in need of servicing. To use the ICU in a polled mode, the application program must not enable CPU interrupts. The ICU is polled by first writing the poll command and then reading the IPOL register.

## Programming

The ICU is enabled and mapped into an I/O address range using the V40 configuration registers. The ICU must be initialized with 2, 3, or 4 initialization control words before operation can begin. Once the initialization control words have been programmed, ICU operation is controlled by command words. The command words define the operating mode and can be programmed at any time after initialization.

Chapter 9

# DMA CONTROLLER (V40)

**Contents** **Page**

## OVERVIEW

This chapter describes the Direct Memory Access Control Unit (DCU) and provides register descriptions.

The DCU is a programmable peripheral device used to direct high speed data transfers between the ZT 8832 and SBX expansion module I/O. The approximate data rate between the expansion module and local RAM using standard input and output instructions is 235 Kbytes per second. Using DMA, the data transfer rate is increased to approximately 1.3 Mbytes per second, for a performance increase of over 80 percent. DMA transfers to dual port RAM are slightly slower because of arbitration time. If no dual port RAM accesses are made by the STD bus CPU during the DMA operation, the data transfer rate is approximately 1.1 Kbytes per second.

There are two primary reasons for this performance increase. The first is that the DCU requires no instruction fetches to perform the data transfer. All address manipulations are done with hardware internal to the DCU, as opposed to the CPU having to fetch and execute the address adjustment instructions. The second reason is that the DCU transfers the data in one CPU machine cycle, rather than two. The DCU allows data to travel from the source device through to the destination device without temporarily storing it in a register, as is done with the CPU.

The DCU has the following features:

- 1 Mbyte of direct memory addressability
- Single, demand, and block data transfers
- Autoinitialization

## ZT 8832 SPECIFICS

The ZT 8832 uses one of the four DMA controllers contained in the V40. DMA channel 0 is used to coordinate high speed data transfers between the SBX expansion module I/O and local or dual port memory. The SBX expansion module must support DMA by driving Expansion Module DMA Request (MDRQT) on J4 pin 34 and receiving Expansion Module DMA Acknowledge (MDAK) on J4 pin 32. The DMA interface does not support Terminate DMA (TDMA) on J4 pin 26.

To meet SBX expansion module timings, the WCY2 V40 configuration register must be must be programmed to insert two wait states into DMA cycles. See page 6-7 for details.

## FUNCTIONAL DESCRIPTION

Figure 9-1 illustrates a block diagram of the DCU. The DCU is divided into six major blocks for explanation purposes. Each of these functional blocks is discussed below.

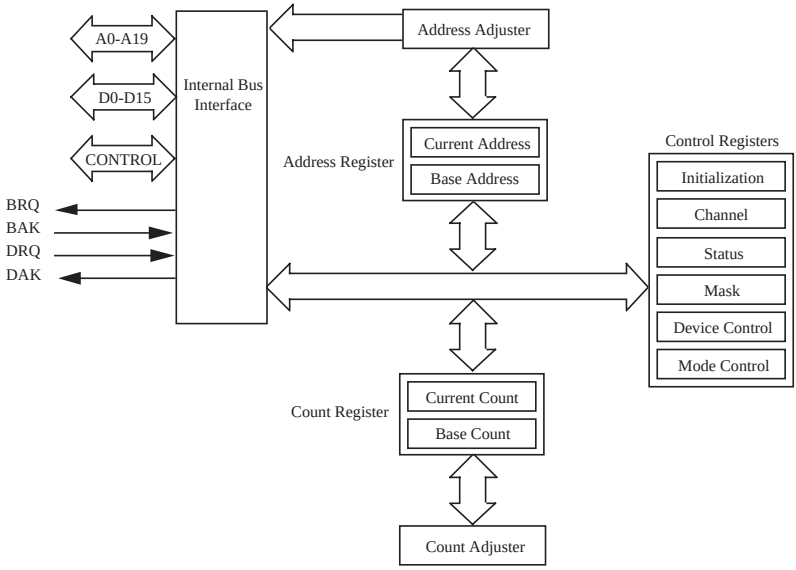



*Figure 9–1. DCU Block Diagram.*

## Internal Bus Interface

The Internal Bus Interface monitors address and data buses for programming information. The Internal Bus Interface also generates Bus Request (BRQ) to request access to the address, data, and control buses in response to a DMA Request (DRQ) from the SBX expansion module. The CPU acknowledges BRQ with Bus Acknowledge (BAK), signaling the Internal Bus Interface to generate DMA Acknowledge (DAK) to the SBX expansion module and perform the data transfer.

## Address Register

The Address register consists of a 20-bit base address and a 20-bit current address. The base and current addresses are programmed with the starting address of the memory block to be used in the transfer. The memory is considered the source if the transfer is from ZT 8832 memory to the SBX expansion module. The memory is considered the destination if the transfer is from the SBX expansion module to the ZT 8832 memory. The current address is automatically adjusted with hardware internal to the DCU as the transfers are made.

## Address Adjuster

The Address Adjuster automatically updates the current address register each time a data transfer is completed. The address register is incremented or decremented by one for each byte transfer.

## Count Register

The Count register includes a 16-bit base count and a 16-bit current count. The base count and current count are programmed with the number of bytes to be transferred by the DMA operation. The base count remains the same until a new count is specified. If the auto-initialize feature is used, the base count is transferred to the current count when the count register is decremented to zero and the last DMA operation is complete. The current count register is decremented by one after each transfer to indicate the number of bytes not yet transferred.

## Count Adjuster

The Count Adjuster automatically updates the current count register each time a data transfer is completed. The Count register is incremented or decremented by one for each byte transfer.

## Control Registers

The Control register functional block consists of six registers that are used to control the DCU operation and provide DCU status. The registers are explained in the following section.

## PROGRAMMABLE REGISTERS

The DCU occupies 16 consecutive I/O port addresses. Of those 16 addresses, 12 are used by the programmer to access DCU functions and 4 are reserved. Table 9-1 lists the address of each of the registers relative to a programmable base address. The base address is selected with the OPHA and DULA V40 configuration registers; see page 6-5 for details.

Table 9-1
DCU Register Addressing.

| Address | Register | Operation | Page Number |
|---|---|---|---|
| Base + 0 | DICM | Write | 9-8 |
| Base + 1 | DCH | Read/Write | 9-8 |
| Base + 2 | DBC/DCC-low | Read/Write | 9-10 |
| Base + 3 | DBC/DCC-high | Read/Write | 9-10 |
| Base + 4 | DBA/DCA-low | Read/Write | 9-11 |
| Base + 5 | DBA/DCA-middle | Read/Write | 9-11 |
| Base + 6 | DBA/DCA-high | Read/Write | 9-11 |
| Base + 7 | Reserved | -- | -- |
| Base + 8 | DDC-low | Read/Write | 9-12 |
| Base + 9 | DDC-high | Read/Write | 9-12 |
| Base + A | DMD | Read/Write | 9-13 |
| Base + B | DST | Read | 9-15 |
| Base + C | Reserved | -- | -- |
| Base + D | Reserved | -- | -- |
| Base + E | Reserved | -- | -- |
| Base + F | DMK | Read/Write | 9-16 |

## DMA Initialize Command (DICM)

The initialization command, shown in Figure 9-2, includes one bit that can be set to a logical 1 to reset the DCU. This register must be written to with the byte output instruction.

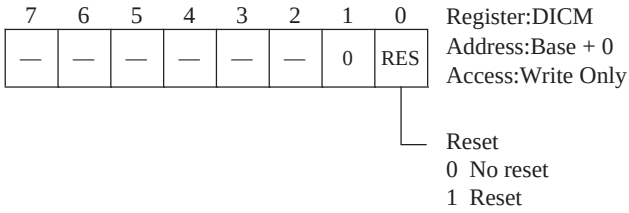



*Figure 9–2. DMA Initialization Command Register.*

## DMA Channel (DCH)

The DMA Channel register, shown in Figure 9-3, must be accessed with byte output and input instructions. The DCH register has a different format for read and write operations. For the write operation, the BASE bit is used to select between the base and current register groups for both the address and count. Both base and current registers are written or only the current register is read if BASE is first programmed with a logical 0. Programming BASE with a logical 1 selects the base to be read or written to.

For read operations, the BASE bit set to a logical 0 defines whether the current register is made available for a read operation or whether the base and current registers are written to during a write operation. A logical 1 in the BASE bit means the base register is selected.

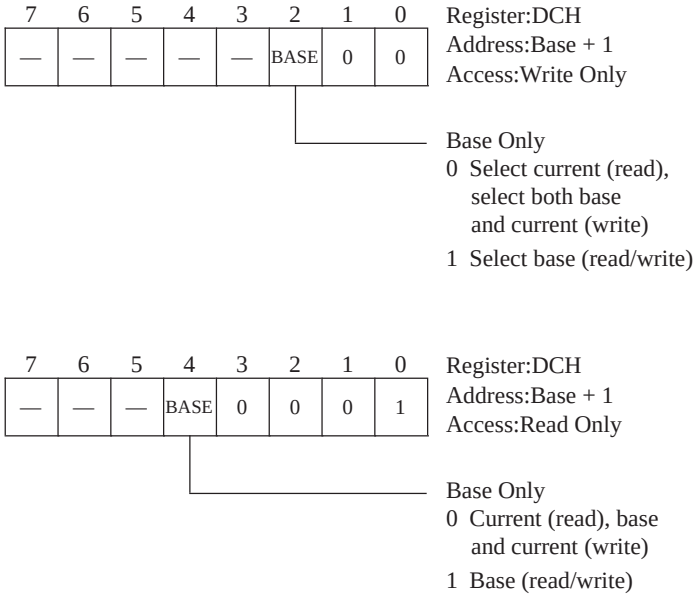



*Figure 9–3. DMA Channel Register.*

## DMA Base Count/Current Count (DBC/DCC)

Two DBC/DCC registers make up the 16-bit DMA count, as shown in Figure 9-4. The two DBC/DCC registers can be accessed with byte or word instructions. The function of these registers depends on the BASE bit of the DCH register. If the BASE bit is set to a logical 0, the values written to the Count registers are programmed into both base and current count values. The current count is read from the Count registers if the BASE bit is set to a logical 0. If the BASE bit is set to a logical 1, the values written to the Count registers are programmed into the base value only. The base value is read from the Count registers if the BASE bit is set to a logical 1.

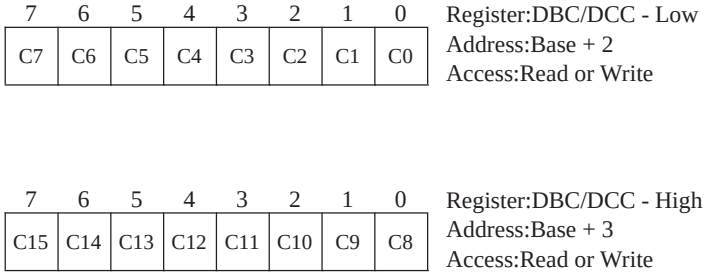



*Figure 9–4. DMA Base and Current Count Registers.*

## DMA Base Address/Current Address (DBA/DCA)

Three DBA/DCA registers specify the 20-bit address. The format of these registers is shown in Figure 9-5. The lower 16 bits of the address can be accessed with byte or word instructions. The upper four bits must be accessed with byte instructions. As is the case with the DBC/DCC registers, the BASE bit of the DCH register defines the operation of the DBA/DCA registers. A logical 0 in the BASE bit specifies that values written to the Address registers are programmed to both base and current values and data read will be current values. If the BASE bit is a logical 1, the values written to the Address registers are programmed into the base value only. The base value is read from the Count registers if the BASE bit is set to a logical 1.

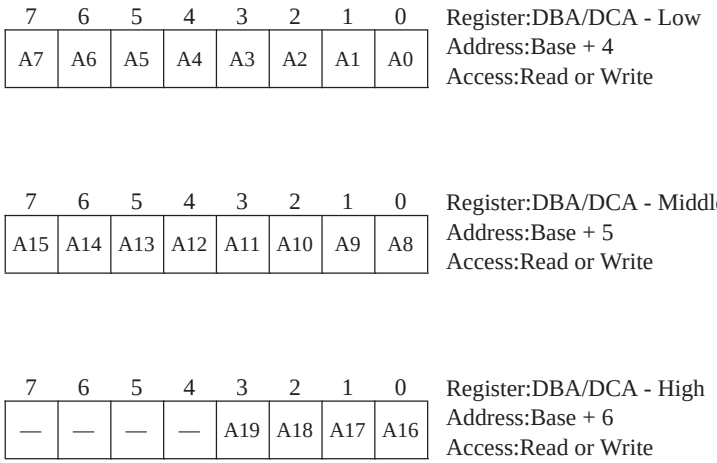



*Figure 9–5. DMA Base and Current Address Registers.*

## DMA Device Control (DDC)

Two DDC registers select various DCU operating modes. The format for these registers is shown in Figure 9-6. These registers can be accessed with byte or word operations. The DDMA bit can be set to a logical 1 to prevent the DCU from requesting bus access. This should be done when programming any of the DCU registers to prevent incorrect DMA operation,

The WEV bit enables or disables wait states to be inserted by the V40 WCU during the verify operation. Programming WEV with a logical 0 disables wait state insertion and programming a logical 1 enables it.

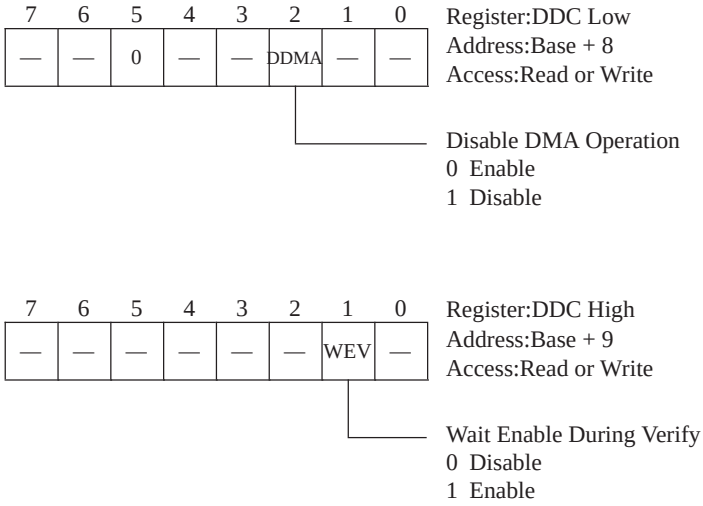



*Figure 9–6. DMA Device Control Registers.*

## DMA Mode (DMD)

Figure 9-7 shows the format of the DMD register. The DMD register can be accessed with byte or word instructions. The TDIR field defines the mode of data transfer. A logical 0 in both bits selects the verify operation. A logical 1 in bit 2 and a logical 0 in bit 3 selects I/O-to-memory transfers. For memory-to-I/O transfers, bit 2 must be programmed with a logical 0 and bit 3 with a logical 1.

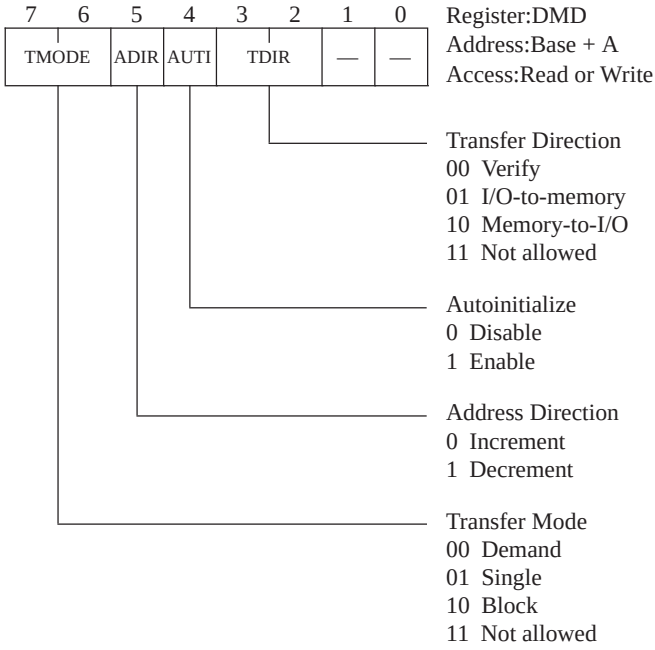



*Figure 9–7. DMA Mode Register.*

Autoinitialize is a feature that automatically reloads the DCU Current Address and Current Count registers from the Base Address and Base Count registers, respectively. The reload is done when the Count register reaches zero. The autoinitialize feature is disabled by programming AUTI with a logical 0 and enabled with a logical 1.

The ADIR bit defines the operation of the DCU address adjuster. If ADIR is programmed with a logical 0, the address adjuster increments the memory address after each data transfer. If ADIR is programmed with a logical 1, the address is decremented. TMODE defines the transfer mode to be demand, single, or block. These operating modes are explained later in this chapter.

## DMA Status (DST)

The Status register includes information about the currently programmed state of the DMA channel. The format for DST is shown in Figure 9-8. DST is accessed with the byte read instruction. The TC0 bit indicates when the count register has reached zero and the DMA transfer is completed. A logical 0 in TC0 means that the operation has not been terminated and a logical 1 means that it has. The RQ0 bit defines the state of the DMA request input. A logical 0 indicates no request active and a logical 1 indicates a request is pending.

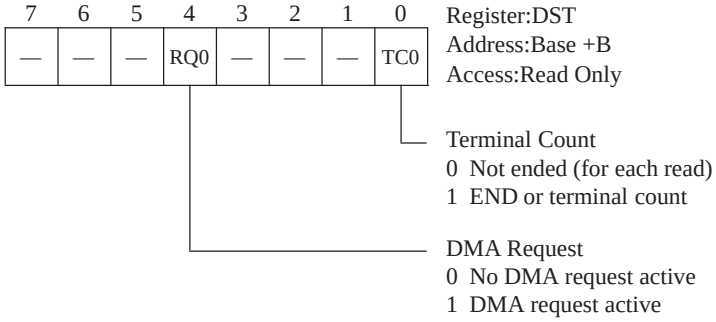



*Figure 9–8. DMA Status Register.*

## DMA Mask (DMK)

The DMK register, shown in Figure 9-9, is used to mask DMA requests made by the DMA channel. The register is accessed with either byte write or read instructions. To mask a DMA channel, the respective bit must be programmed with a logical 1. A logical 0 enables the DMA channel to make requests.

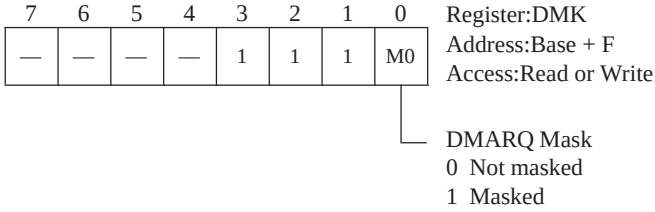



*Figure 9–9. DMA Mask Register.*

# OPERATION

## Reset

The DCU registers are initialized after power-on or after a pushbutton reset. Table 9-2 shows the initialized state.

Table 9-2
DCU Register Default State.

| Register | Default Bit Value [1] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| DCH | - | - | - | 0 | 0 | 0 | 0 | 1 |
| DMD | 0 | 0 | 0 | 0 | 0 | 0 | - | 0 |
| DDC (low) | - | - | 0 | 0 | - | 0 | - | - |
| DDC (high) | - | - | - | - | - | - | 0 | 0 |
| DST | - | - | - | - | 0 | 0 | 0 | 0 |
| DMK | - | - | - | - | 1 | 1 | 1 | 1 |

[1]Bit positions marked with a dash (-) can default to 1 or 0.

### Single Mode Transfers

In single mode transfers, the DCU releases control of the bus after each data transfer. The DCU then enters a slave mode, permitting lower priority bus masters to gain access to the bus resources.

### Demand Mode Transfers

With the DCU programmed for demand mode, the DMA channel is serviced until the DMA request for that channel is removed. The DCU then releases control of the bus and enters a slave mode, permitting lower priority bus masters to gain access to the bus resources.

### Block Mode Transfers

In block transfer mode, the DCU continues servicing the DMA channel until the count register is decremented to zero. At this time the DCU releases control of the buses and enters a slave mode of operation, permitting lower priority bus masters to gain access to the bus resources.

### Autoinitialization

Autoinitialization is useful when doing repetitive DMA transfers using the same amount of data and the same memory locations. The DMA channel can be programmed for autoinitialization with the DMD register. If autoinitialize is selected, the DCU automatically transfers the previously programmed base address and base count to the current address and current count, respectively.

### Programming

The DCU is enabled and mapped into an I/O address range using the V40 configuration register. DMA operation must be disabled with the DDMA bit of the DDC register during DCU programming. Erroneous operation may otherwise result. For example, if the two-byte address register is being programmed by the CPU and a DMA request occurs, it is possible that only one of the bytes may be updated before the DMA channeled is serviced. The address would then be incorrect, and the DMA operation invalid.

Chapter 10

# SERIAL COMMUNICATIONS (V40)

**Contents** **Page**

## OVERVIEW

This chapter describes the Serial Control Unit (SCU) and provides register descriptions and baud rate information.

The SCU is a single serial channel that performs asynchronous serial communication between the V40 and a serial device external to the ZT 8832.

The major features of the SCU are listed below.

- Full-duplex asynchronous operation
- Clock divisor of 16 or 64
- Baud rates to 38.4 Kbaud
- Programmable character format
- Automatic break detect and handling
- Parity, overrun, and framing detection
- Interrupt and polled operation

## ZT 8832 SPECIFICS

The SCU is implemented as a 3-wire RS-232-C serial port, including Transmit Data (TxD), Receive Data (RxD), and Ground (GND). These signals are available at connector J5; pin assignments are shown on page B-16.

## FUNCTIONAL DESCRIPTION

The SCU is similar to the 8251 Serial Control Unit for asynchronous operation. The SCU does not support synchronous communication protocols. Figure 10-1 shows a functional block diagram of the SCU. A description of each functional block follows.

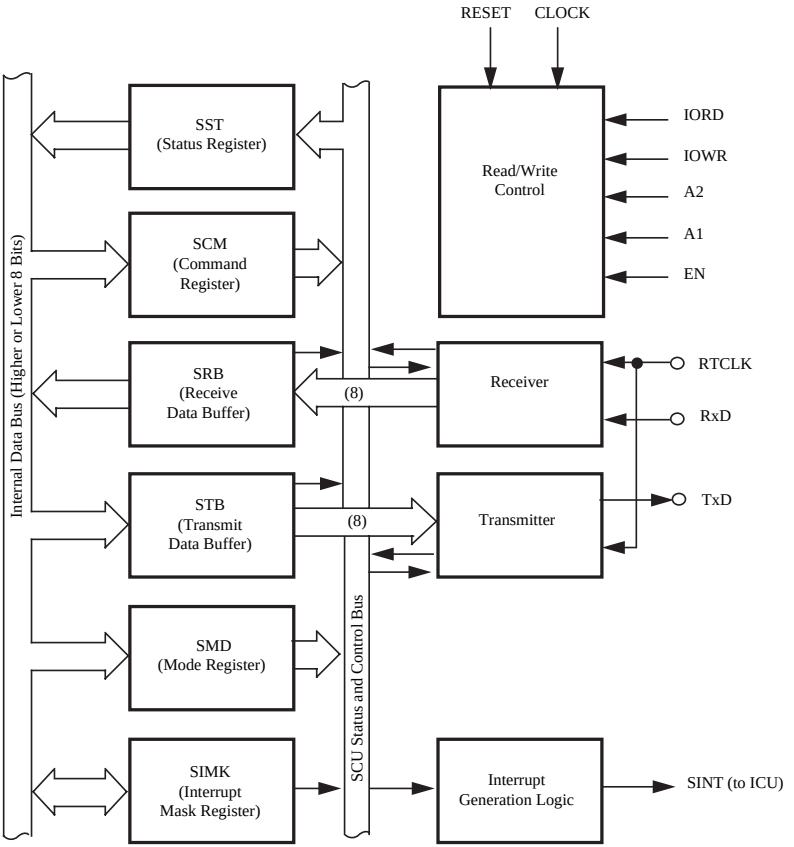



*Figure 10–1. SCU Block Diagram.*

### Read/Write Control

The Read/Write Control block acts as an interface between the internal registers of the SCU and the CPU. The control signals input to the Read/Write Control logic, select internal registers, and control the transfer of information between the CPU and the SCU.

### Receiver

The Receiver block converts serial data input on the RxD signal to a parallel format with the start, stop, and parity bits removed. The parallel data is placed in the Serial Receive Buffer, and the Receive Buffer Ready bit in the Serial Status register is set to a logical 1. Activating the Receive Buffer Ready bit generates an interrupt if the interrupts are not masked. The Receiver is also responsible for testing data parity and monitoring for a Receive Break.

### Transmitter

The Transmitter block converts the contents of the Serial Transmit Buffer from a parallel format into a serial string. Start, stop, and parity bits are added, as specified by the programmer in the Serial Mode register. Next, the modified serial string is transmitted out of the TxD signal at a rate equal to 1/16 or 1/64 that of the Receiver/Transmitter Clock, also defined in the Serial Mode register. The Transmit Buffer Ready bit in the Serial Status register is set to alert the application program that it is clear to send the next character. Activating the Transmit Buffer Ready bit generates an interrupt if the interrupts are not masked by the Serial Interrupt Mask register.

### Interrupt Generation Logic

The Interrupt Generation Logic block can interrupt the CPU when data is received into the Serial Receive Buffer or transmitted from the Serial Transmit Buffer. These interrupts are used to vector the application program to service routines that read a character from the Serial Receive Buffer and write a character to the Serial Transmit Buffer, respectively.

## PROGRAMMABLE REGISTERS

Six registers are used for communication with the SCU. The Serial Transmit Buffer (STB) and Serial Receive Buffer (SRB) store data to be transferred to the serial link and from the serial link, respectively. The Serial Command (SCM) and Serial Mode (SMD) registers define the operating mode. The Serial Interrupt Mask (SIMK) register controls the receive and transmit interrupts. The Serial Status (SST) register provides information on the current state of the transmitter and receiver.

The base I/O address of the SCU registers is defined by the OPHA and SULA registers. OPHA is programmed with the high byte and SULA with the low byte of the 16-bit address; see page 6-5 for details. The address of each register, relative to the base address, is shown in Table 10-1.

Table 10-1
SCU Register Addressing.

| Address | Register | Operation | Page Number |
|---|---|---|---|
| Base + 0 | Receive Buffer | Read | 10-5 |
| Base + 0 | Transmit Buffer | Write | 10-5 |
| Base + 1 | Status Register | Read | 10-6 |
| Base + 1 | Command Register | Write | 10-8 |
| Base + 2 | Mode Register | Write | 10-9 |
| Base + 3 | Interrupt Mask Register | Read/Write | 10-10 |

## Serial Status Register (SST)

Figure 10-2 shows the architecture of the SST register, which can be read at any time.

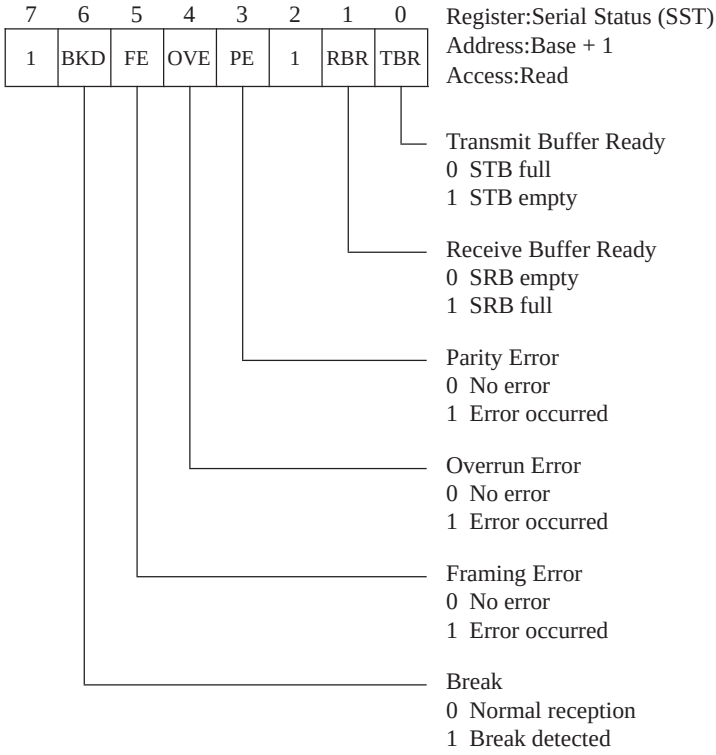



*Figure 10–2. Serial Status Register.*

The first two bits define the state of the STB (Serial Transmit Buffer) and SRB (Serial Receive Buffer). TBRDY (Transmit Buffer Ready) is set to a logical 1 when a character is transferred out of the STB. Application software uses TBRDY to determine when it is clear to send a new character to the STB. TBRDY is automatically reset when a new character is transferred into the STB. The RBRDY (Receive

Buffer Ready) bit is set to a logical 1 when a character is transferred into the SRB. Application software uses RBRDY to determine if a character is available. RBRDY is automatically reset when the character is read from the SRB.

The PE (Parity Error), OVE (Overrun Error) and FE (Framing Error) bits flag data communication errors. The PE bit is set to a logical 1 if a character is received into the SRB with a mismatch between the parity bit and the character itself. The test for parity can be enabled and disabled using the Serial Mode register.

An overrun error occurs when a character is written to the STB before the character currently in the buffer is transmitted. The overrun error is flagged with a logical 1 in the OVE bit. Avoid overrun errors by having the application software test the TBRDY bit before writing the character. A second type of overrun occurs if a character is received into the SRB before the character currently in the buffer is read by the application program. This type of overrun is avoided if the application software reads the character before another is received. This can be done on a polled basis by testing the RBRDY flag at a frequency greater than the rate in which the characters are transferred or by using the SCU interrupt capabilities to signal the presence of a character to be read.

The FE bit is set to a logical 1 to flag a third type of error called a framing error. A framing error signals a missing stop bit. This happens when a break sequence is detected or when the programmed baud rate is not the same as device on the other end of the serial link.

Once set to flag an error, the PE, OVE and FE bits remain set until a logical 1 is written to the Error Clear bit of the Serial Command register.

The BKD bit is set to a logical 1 when a break sequence is detected. A break sequence is detected when a complete character, from the start bit through the stop bit, is received with RxD in a low state. The BRK bit is reset when RxD returns to a high state or the SCU is reset.

## Serial Command Register (SCM)

Figure 10-3 illustrates the SCM register bit map. The SCU is configured with the SCM and the SMD registers. The SCM register includes the functions that are most likely to be modified during operation. The SMD register will more than likely be programmed just once for initialization.

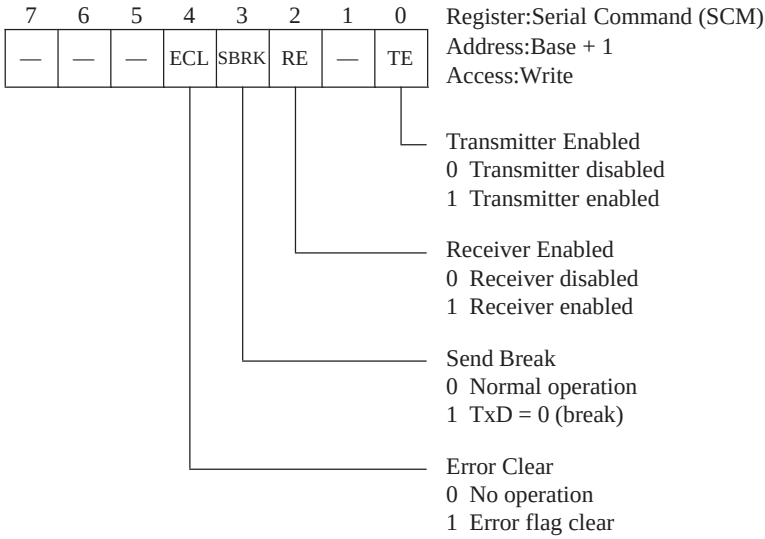



*Figure 10–3. Serial Command Register.*

Bits 0 and 2 are the Transmitter and Receiver control bits. Setting TE to a logical 1 enables the transmitter and a logical 0 disables it. If TE is reset during transmission, the data in the Serial Transmit Buffer will be sent before transmission stops and TxD goes to a high level. Setting the RE bit to a logical 1 enables the receiver and resetting the RE bit disables it.

A break sequence is transmitted if the SBRK bit is set to a logical 1. This mode of operation is independent of the Receiver Enable bit. The break sequence is transmitted until SBRK is reset.

The ECL bit is used to reset the parity, overflow, and framing error bits of the Serial Status register. These bits are set to a logical 1 if

error conditions occur and remain set until a logical 1 is written to the ECL bit.

## Serial Mode Register (SMD)

Figure 10-4 shows the format for the SMD register. This register includes all the functions that are not likely to change after they have been initialized. Bits 1 and 2 combine to define the clock divisor for the baud rate. Programming the baud rate is defined in more detail on page 10-13.

The character length is programmed with the CL field. If a character length of seven is selected, the SCU transmits the lower seven bits of the 8-bit character written by the CPU. Characters read by the CPU will have a logical 0 in the most significant bit position.

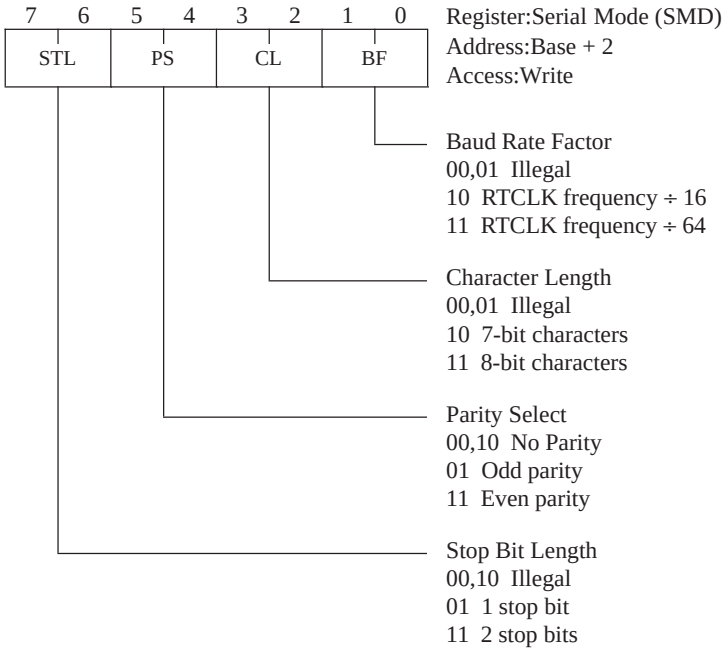



*Figure 10–4. Serial Mode Register.*

Enabling even or odd parity is the function of the PS field. Parity is disabled if a logical 0 is written to bit 4. If parity is disabled, the parity bit will not be appended to the characters transmitted and the characters received will not be tested. If bit 4 is a logical 1, parity is enabled. Bit 5 selects between even or odd format. A character has even parity if it includes an even number of bits set to a logical 1. A character has odd parity if it includes an odd number of bits set to a logical 1.

The number of stop bits appended to the character during transmission or stripped off during reception is defined by the STL bit field. The choice is between one or two stop bits.

## Serial Interrupt Mask Register (SIMK)

The SCU is capable of interrupting the CPU when a character is received into the Serial Receive Buffer or transmitted out of the Serial Transmit Buffer. The SIMK register includes two programmable bits, as illustrated in Figure 10-5, to enable the interrupts. Setting the RM bit to a logical 1 prevents the SCU from generating an interrupt when a character is received. A logical 0 in the RM bit enables the interrupt. The TM bit provides the same control for transmitted characters.

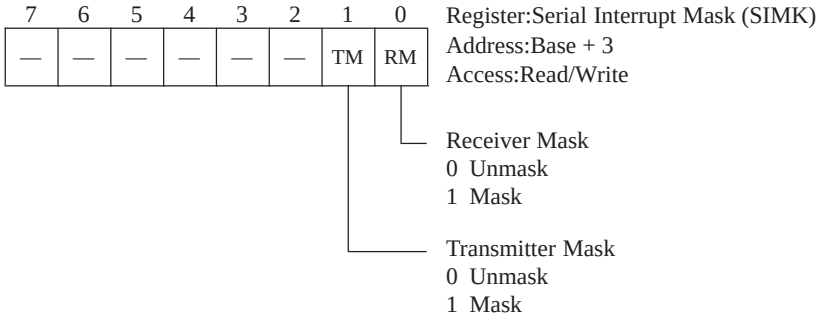



*Figure 10–5. Serial Interrupt Mask Register.*

# OPERATION

## Reset

The SCU registers are automatically initialized to a default state when power is applied to the ZT 8832, or during a reset. Table 10-2 shows the default state for these registers.

Table 10-2
SCU Register Defaults.

| Register | Default Bit Value [1] 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SST | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| SCM | - | - | 0 | 0 | 0 | 0 | - | 0 |
| SMD | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| SIMK | - | - | - | - | - | - | 1 | 1 |

[1] Bit positions marked with a dash (-) can default to 1 or 0.

## Serial Data Format

The SCU supports asynchronous communication. The asynchronous data format is shown in Figure 10-6 to include start, stop, and optional parity bits, as well as seven or eight data bits.

The state of the TxD line, when data is not being transmitted, is the "mark" state. The start bit indicates the beginning of the serial data or character bits. The parity bit is inserted for transmission and tested for reception if parity is enabled in the SMD register. The stop bit flags the end of the serial data. The number of stop bits is programmed into the SMD register.

Also shown in Figure 10-6 is the format for a break sequence. The detection of a break sequence is flagged in the Serial Status register. The Serial Command register can be programmed to transmit the break sequence.

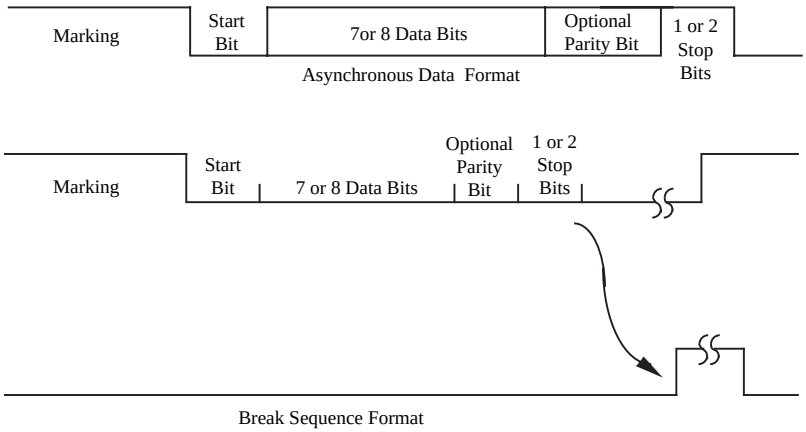



*Figure 10–6. SCU Serial Data Format.*

## Baud Rate

The SCU baud rate is determined by the output of counter/timer 1. Counter/timer 1 must be initialized for a specific mode of operation and programmed with a count that defines the required baud rate. The discussion below explains initialization and how to calculate the count.

To use counter/timer 1 as a baud rate generator, the TCKS register ust specify that counter/timer 1 has an internal clock with a divisor determined by the formula shown below. The TCU Counter/Timer Mode register must also be programmed to configure counter/timer 1 for binary operation as a square wave generator (Mode 3). Different baud rates are obtained by programming counter/timer 1 with different counts. The relationship between the baud rate and the count is given below:

Count = V40 Clock / (Prescale x Baud Rate x Baud Factor)

This formula includes all the factors that determine the baud rate as a function of the V40 clock. The best way to understand the formula is to trace the clock signal from the V40 clock to the SCU. The V40 clock starts with a value of $8.0 \times 10^6$ Hz. This frequency is Prescaled as defined by the PS bits in the TCKS V40 configuration register. At this point the clock is input to the counter/timers. Counter/timer 1 further divides the signal by the programmed count before it is input to the SCU. The last division is done internal to the SCU by the baud factor programmed in the BF bits of the Serial Mode register.

As an example, assume the following parameters are specified:

V40 Clock = $8.0 \times 10^6$ Hz
Prescale = divide by two (TCKS PS bits = 00)
Baud Rate = 9600
Baud Factor = divide by 16 (SMD BF bits = 10)

The calculation to determine the count value to be programmed into counter/timer 1 is shown below to be 26. Note that the calculated count and the programmed count differ by 0.16 percent. To guarantee proper operation, the percent difference must never be greater than four. Table 10-3 lists the count values to be programmed into counter/timer 1 to generate the more common baud rates.

$$\text{Count} = 8 \times 10^6 / (2 \times 9600 \times 16) = 26.04$$

Table 10-3
ZT 8832 Baud Rate Counts.

| Baud Rate | Count Dec/Hex [1] Baud Factor Divided By: 16 | 64 |
|---|---|---|
| 110 | 2273/08E1 | 568/0238 |
| 150 | 1667/0683 | 417/01A1 |
| 300 | 833/0341 | 208/00D0 |
| 600 | 417/01A1 | 104/0068 |
| 1200 | 208/00D0 | 52/0034 |
| 2400 | 104/0068 | 26/001A |
| 4800 | 52/0034 | 13/000D |
| 9600 | 26/001A | [2] |
| 19200 | 13/000D | [2] |
| 38400 | [2] | [2] |

[1] The count values listed assume the PS bits of the TCKS V40 configuration register are programmed with logical 0s.

[2] This combination exceeds the recommended 4% error allowance.

If the counter/timers are driven with the external TCLK pin through connector J3, instead of with the V40 clock, use the formula below for calculating the value to be loaded into counter/timer 1. Note that the frequency of the external clock signal is not dependent on the PS bit of the TCKS register.

Count = External Clock / (Baud Rate x Baud Factor)

## Interrupt and Polled Communication

Serial data can be transferred and received either by polling status bits or with interrupts. In a polled mode, the application program monitors the SST register Transmit Buffer Ready and Receive Buffer Ready signals to determine when to transfer a character to the SCU or when a character is available.

The SCU is capable of generating interrupts for applications that cannot afford time spent polling for status. A transmit interrupt is generated when the transmitter is enabled, the transmit interrupt is not masked, and the Serial Transmit Buffer is empty. A receive interrupt occurs when the receiver is enabled, the receive interrupt is not masked, and a character is received by the Serial Receive Buffer.

## Programming

The SCU is enabled and mapped into an I/O address range using the V40 Configuration register. The SCU includes four programmable registers. The steps listed below outline the procedure for programming the SCU for serial operation. The steps include initializing counter/timer 1 to generate the baud rate.

1. Program the TCKS V40 configuration register for counter/timer 1 to use the internal clock divided by two. Note that an external clock or another divisor can be used, but the baud rate counts shown in Table 10-3 must be adjusted.
2. Program the TMD register of the TCU for binary count, square wave generation (mode 3), low byte followed by high byte, and selection of TCT1.
3. Program the TCT1 Count register of the TCU with the selected baud rate. Use a two-byte output operation — low byte followed by high byte.
4. Program the SMD register of the SCU with the selected baud factor, character and stop bit length, and parity selection.
5. Program the SCM and SIMK registers for the desired operation.

Chapter 11

# SERIAL COMMUNICATIONS (82050)

**Contents** **Page**

## OVERVIEW

This chapter describes the Asynchronous Communication Controller (ACC) and provides register descriptions and baud rate information.

The ACC is the Intel 82050 or equivalent. It is a single asynchronous serial channel that performs serial communication between the ZT 8832 and an external serial device. The major features of the ACC are listed below.

- Full-duplex asynchronous operation
- Programmable baud rates up to 56 Kbaud
- Programmable character format
- Automatic break detect and handling
- Parity, overrun, and framing detection
- Interrupt and polled operation
- Loopback diagnostics
- Double buffering

## ZT 8832 SPECIFICS

The ACC serial port, available at connector J2, includes Transmit Data (TxD), Receive Data (RxD), and support for a complete selection of the following modem control functions:

- Clear To Send (CTS)
- Request To Send (RTS)
- Data Terminal Ready (DTR)
- Data Set Ready (DSR)
- Ring Indicator (RI)
- Data Carrier Detect (DCD)

The serial port is jumper configured to operate as either Data Communication Equipment (DCE) or Data Terminal Equipment (DTE) with either RS-232 or RS-485 protocols. Pin assignments for connector J2 are shown in Appendix B for both RS-232 and RS-485 operation.

The ACC supports interrupt driven communications. The interrupt output of the ACC is connected to the IR4 interrupt input of the V40 interrupt controller.

When the ACC serial port is jumper configured to operate in RS-485 mode, the DTR bit of the ACC Modem Control register enables and disables the transmit and request to send buffers. (See page 11-13.) Regardless of whether the ACC is configured to operate in RS-232 or RS-485 mode, the OUT2 bit of the Modem Control register enables and disables the 24 bits of parallel I/O local to the ZT 8832.

## FUNCTIONAL DESCRIPTION

The ACC is functionally compatible with the industry standard 16450/8250A found in most Personal Computers. Figure 11-1 illustrates the major functional blocks of the ACC. A description of each functional block follows.

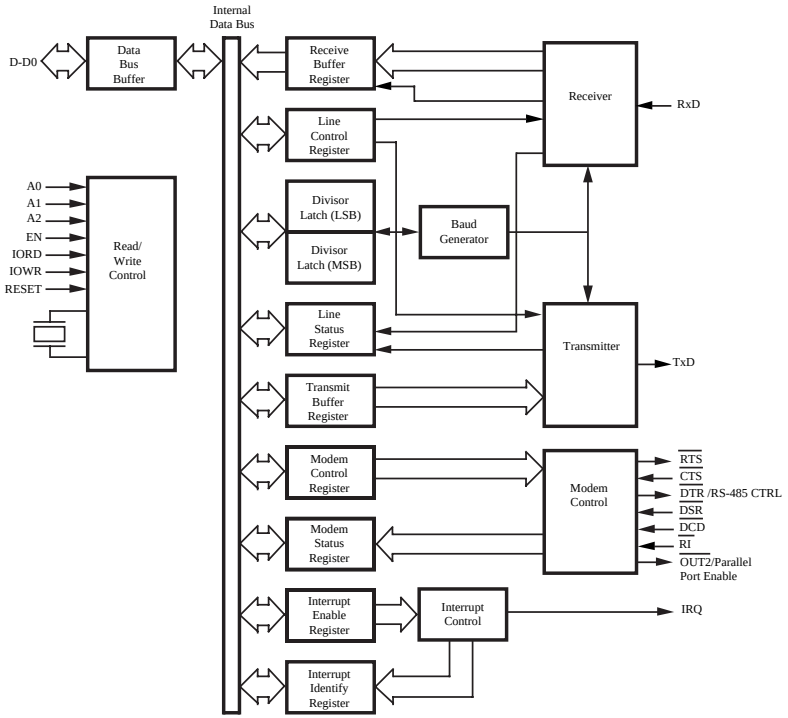



*Figure 11–1. ACC Block Diagram.*

## Read/Write Control

The main function of the Read/Write Control block is to supervise the interface between the ACC internal registers and the CPU. This includes enabling one of the ACC programmable registers onto the system data bus based on control signal inputs.

## Receiver

The Receiver block converts serial data input on the RxD signal to a parallel format with the start, stop, and parity bits removed. The parallel data is placed in the Receive Buffer and the receiver Data Ready bit in the Line Status register is set to a logical 1. Activating the Data Ready bit generates an interrupt if the receive data interrupt is not masked by the Interrupt Enable register. The Receiver also monitors the incoming data for parity, overrun, and framing errors, and reports any such errors to the Line Status register.

## Transmitter

The Transmitter block converts the contents of the Transmit Buffer from a parallel format into a serial string. Start, stop, and parity bits are added, as specified by the programmer in the Line Control register. The serial string is then transmitted out of the TxD signal and the Transmitter Holding Register Empty bit is set to indicate that the ACC is ready to accept a new character for transmission. Activating the Transmitter Holding Register Empty bit will also generate an interrupt if the transmit data interrupt is not masked by the Interrupt Enable register.

## Modem Control

The Modem Control block includes the serial communication handshake, RS-485 buffer control, and parallel port enable signals. The serial handshake signals output by this control block are Request to Send (RTS) and Data Terminal Ready (DTR). The DTR signal also serves as an enable for the Transmit Data and Request To Send drivers when the ACC is configured for RS-485 operation. Both the RTS and DTR signals are programmable through the Modem Control register. The serial handshake signals monitored by this control block are Clear to Send (CTS), Data Set Ready (DSR), Data Carrier Detect (DCD), and Ring Indicator (RI). These signals are monitored through the Modem Status register. The remaining OUT2 signal is programmable through the Modem Control register to enable and disable the 24 bits of parallel I/O on the ZT 8832.

## Interrupt Control

The ACC includes a fully prioritized interrupt system to monitor and report operating conditions such as receiver data available and transmitter empty, and error conditions such as overrun, parity, framing, and break. The Interrupt Control block monitors the interrupt sources defined by the Interrupt Enable register and reports any active interrupts through the Interrupt Identify register.

## PROGRAMMABLE REGISTERS

The remainder of the functional blocks illustrated in Figure 11-1 are programmable registers. Table 11-1 shows the address of each of the registers.

Table 11-5 on pages 11-25 and 11-26 summarizes the ACC register set for programming reference.

Table 11-1
ACC Register Addressing.

| Address | Register | Operation | Page # |
|---|---|---|---|
| 03F8h (DL = 0) | Receive Buffer | Read | 11-8 |
| | Transmit Buffer | Write | 11-8 |
| (DL = 1) | Divisor Latch LSB | Read/Write | 11-17 |
| 03F9h (DL = 0) | Interrupt Enable | Read/Write | 11-19 |
| (DL = 1) | Divisor Latch MSB | Read/Write | 11-17 |
| 03FAh (DL = X) | Interrupt Identify | Read | 11-18 |
| 03FBh (DL = X) | Line Control | Read/Write | 11-8 |
| 03FCh (DL = X) | Modem Control | Read/Write | 11-13 |
| 03FDh (DL = X) | Line Status | Read | 11-11 |
| 03FEh (DL = X) | Modem Status | Read/Write | 11-15 |
| 03FFh (DL = X) | Reserved | -- | -- |

## Transmit and Receive Buffer

The Transmit Buffer, Receive Buffer, and Interrupt Enable registers share the same addresses as the Divisor Latch. Since the Divisor Latch is programmed only once during system initialization, this should cause no problems. During system initialization, the Divisor Latch is selected by programming the Divisor Latch Access Bit of the Line Control register. The Divisor Latch is then initialized to select the appropriate baud rate, and the Divisor Latch Access Bit is reprogrammed to enable the Transmit Buffer, Receive Buffer, and Interrupt Enable register for serial communications.

## Line Control

The main function of the Line Control register, shown in Figure 11-2, is to define the format of the serial data.

The Word Length Select (WLS) field defines the character length, as shown in Table 11-2.

Table 11-2
Serial Character Length.

| Bit 1 | Bit 0 | Character Length |
|---|---|---|
| 0 | 0 | 5 bits |
| 0 | 1 | 6 bits |
| 1 | 0 | 7 bits |
| 1 | 1 | 8 bits |

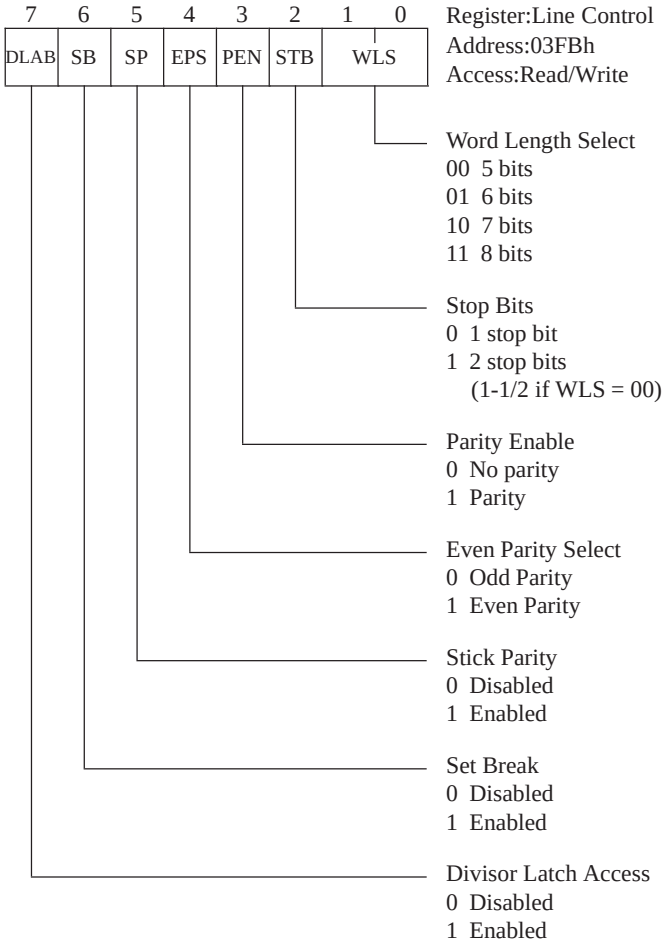



*Figure 11–2. Line Control Register.*

The STB bit selects the number of stop bits added to each character transmitted and removed from each character received. Programming STB with a logical 0 selects one stop bit; a logical 1 selects two stop bits. The exception to this is when the character length is defined as five. In this case, a logical 1 selects one and a half stop bits.

The parity options are programmed through the PEN, EPS, and SP bits. PEN (bit 3) enables and disables parity. A logical 0 in PEN disables parity and a logical 1 enables parity. With parity enabled, the ACC adjusts the parity bit of transmitted data to produce an even or an odd number of 1s when added to the character bits. With parity enabled, the ACC also tests the parity of the received data.

The EPS bit selects even or odd parity. Even parity means that there is an even number of 1s in the character data, including the parity bit. Odd parity means that there is an odd number of 1s in the character data, including the parity bit. Odd parity is selected by programming bit 4 with a logical 0, and even parity is selected by programming bit 4 with a logical 1.

Bit 5 (SP) is the stick parity bit. With bits 3 and 5 a logical 1, the parity bit is transmitted and received in the state opposite that of bit 4.

The SB bit position enables and disables break control. A logical 0 disables break and a logical 1 enables it. Enabling break forces the TxD output to the marking (negative) state.

The DLAB bit controls access to the ACC divisor latch. Programming bit 7 with a logical 1 selects the Divisor Latch; a logical 0 selects the Transmit Buffer, Receive Buffer, and Interrupt Enable register.

## Line Status

The Line Status register, shown in Figure 11-3, provides information to the CPU concerning the data transfer. Reading the Line Status register clears bits 1 through 4 (OE, PE, FE, and BI).

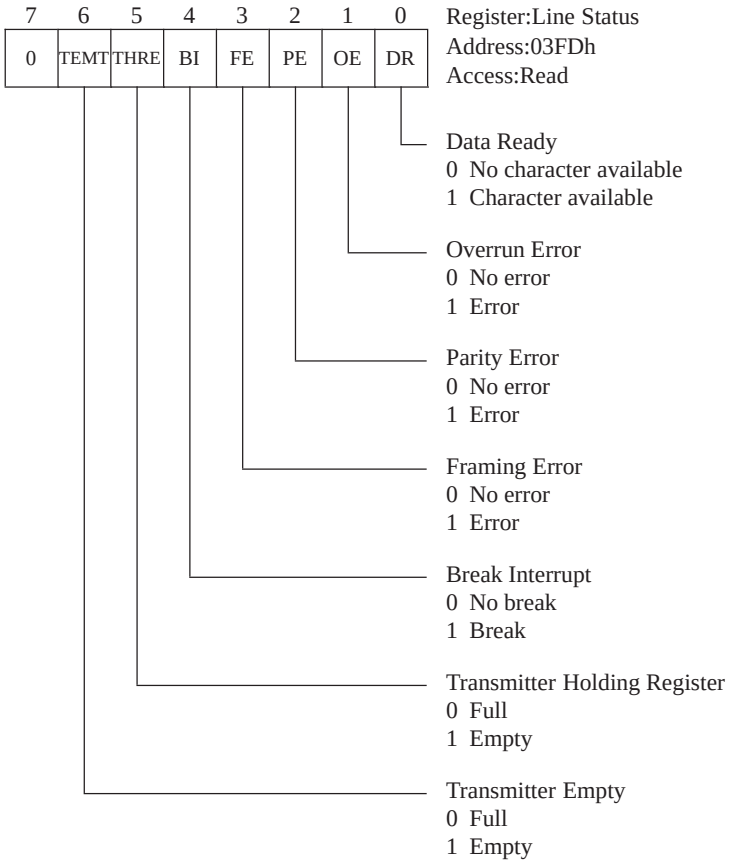



*Figure 11–3. Line Status Register.*

The DR bit indicates the state of the ACC Receive Buffer. A logical 1 in the DR bit signals the availability of a character in the Receive Buffer. The DR bit is automatically reset to a logical 0 when the Receive Buffer is read.

The OE, PE, and FE bits are transmission error indicators. An overrun error is indicated by a logical 1 in the OE bit, a parity error is indicated by a logical 1 in the PE bit, and a framing error is indicated by a logical 1 in the FE bit. In all cases, the error indicator is reset when the Line Status register is read. An overrun error is generated when a character is written into the Receive Buffer before the previous character is read by the CPU. A parity error is generated when the received character does not have the correct even or odd parity. A framing error indicates that the received character does not have a valid stop bit. Any of these error bits are capable of generating an interrupt if enabled through the Interrupt Enable register.

The BI bit is set to a logical 1 when the RxD signal is held in the marking (negative) state for longer than a single character transmission (including start, data, parity, and stop bits). Bit four is reset to a logical 0 when the Line Status register is read. This bit is capable of generating an interrupt if enabled through the Interrupt Enable register.

The THRE and TEMT bits provide Transmitter status. A logical 1 in the THRE bit signals that the Transmit Buffer is empty and ready to receive a new character for transmission. This bit is capable of generating an interrupt if enabled through the Interrupt Enable register. TEMT is similar to bit 5 except that bit 6 is not set to a logical 1 until both the Transmit Buffer and Transmit Shift register are empty. Both THRE and TEMT bits are automatically reset to a logical 0 with the loading of the next character to be transmitted.

Bit 7 is permanently set to logical 0.

## Modem Control

The Modem Control register, shown in Figure 11-4, includes the serial handshake, RS-485 driver enable, and parallel port enable signals.

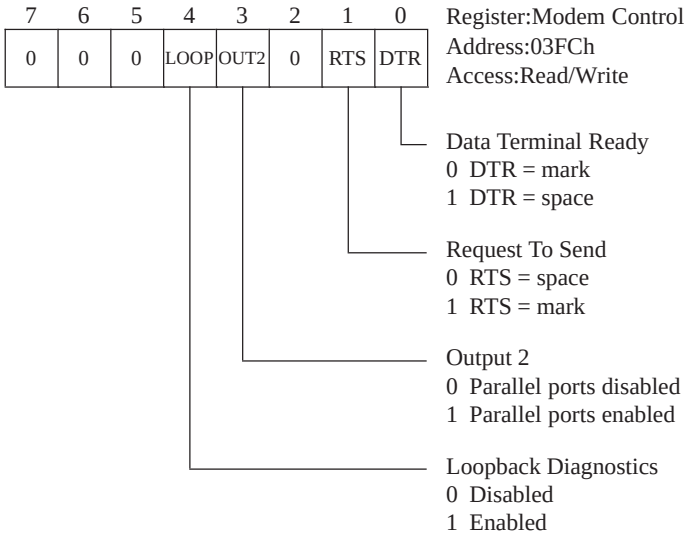



*Figure 11–4. Modem Control Register.*

The DTR bit (bit 0) defines the state of the Data Terminal Ready (DTR) signal. Programming the DTR bit with a logical 0 forces DTR to a marking (negative) state, and programming the DTR bit with a logical 1 forces DTR to a spacing (positive) state.

The DTR signal can be jumpered to enable and disable the RS-485 transmit data and request to send drivers if the ACC serial port is configured for RS-485 operation. In this configuration, programming the DTR bit with a logical 0 disables the drivers, and programming the DTR bit with a logical 1 enables the drivers. The power-on state of DTR disables the drivers.

The RTS bit (bit 1) defines the state of the Request To Send (RTS) signal. Programming the RTS bit with a logical 0 forces RTS to a marking (negative) state, and programming the RTS bit with a logical 1 forces RTS to a spacing (positive) state.

Bit 2 is permanently set to logical 0.

The OUT2 bit (bit 3) defines the state of the Output 2 (OUT2) signal. The OUT2 signal is connected to the output control of the three parallel ports available at connector J1. Programming OUT2 with a logical 0 disables the parallel ports, and programming OUT2 with a logical 1 enables the parallel ports. The power-on state of OUT2 disables the parallel ports.

The LOOP bit (bit 4) selects the internal loopback operation of the ACC. Programming LOOP with a logical 0 disables loopback, and programming LOOP with a logical 1 enables loopback. With loopback enabled, the following takes place:

- The RxD input is ignored.
- The TxD output is externally set to the marking (logical 1) state and internally connected to RxD.
- The modem control inputs (CTS, DSR, DCD, and RI) are ignored.
- The modem control outputs (DTR, RTS, and OUT2) are externally forced to a logical 1 and internally connected to the modem control inputs.
- The receiver and transmitter interrupts are fully functional. During loopback, the data transmitted is immediately received.

Bits 5-7 are permanently set to logical 0.

## Modem Status

Figure 11-5 shows the format of the Modem Status register. All eight bits of this register are used to indicate the status of the serial data transfer.

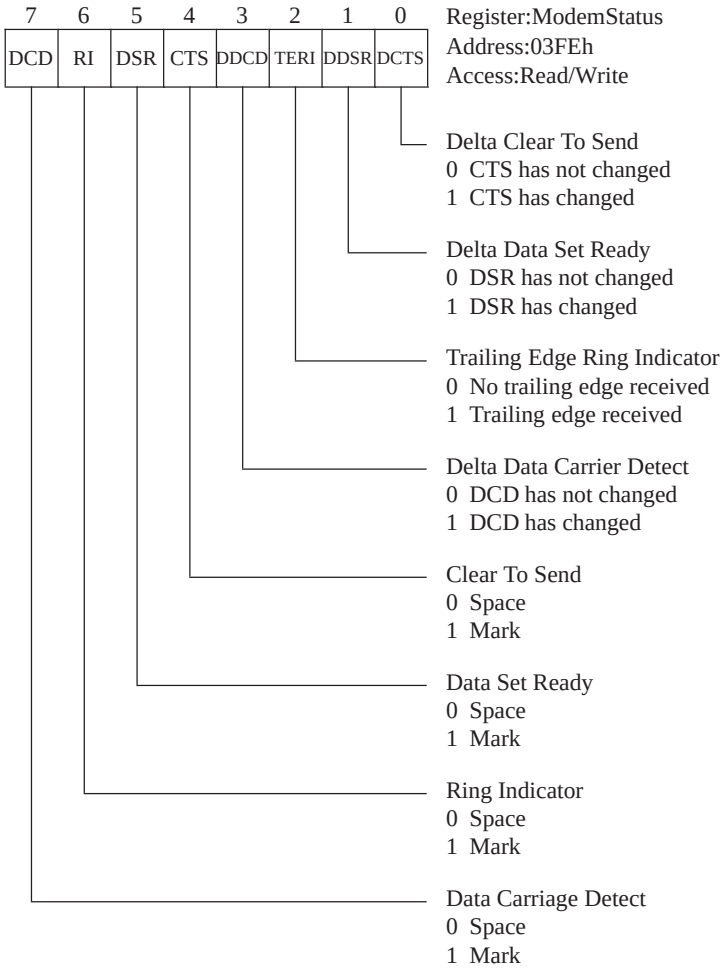



*Figure 11–5. Modem Status Register.*

The DCTS, DDSR, and DDCD bits (bits 0, 1, and 3) are set to a logical 1 to indicate that the CTS, DSR, and DCD bits have changed state since the last time the Modem Status register was read.

The TERI bit (bit 2) signals the trailing edge detection of the Ring Indicator (RI). The TERI bit is a logical 1 if the RI input changed from a spacing (positive) state to a marking (negative) state since the last time the Modem Status register was read.

The CTS bit (bit 4) reflects the state of the Clear To Send (CTS) signal. If CTS is in the marking state, the CTS bit is a logical 0. If CTS is in the spacing state, the CTS bit is a logical 1. During loopback mode, the CTS bit is internally connected to RTS.

The DSR bit (bit 5) reflects the state of the Data Set Ready (DSR) signal. If DSR is in the marking state, the DSR bit is a logical 0. If DSR is in the spacing state, the DSR bit is a logical 1. During loopback mode, the DSR bit is internally connected to DTR.

The RI bit (bit 6) reflects the state of the Ring Indicator (RI) signal. If RI is in the marking state, the RI bit is a logical 0. If RI is in the spacing state, the RI bit is a logical 1.

The DCD bit (bit 7) reflects the state of the Data Carrier Detect (DCD) signal. If DCD is in the marking state, the DCD bit is a logical 0. If DCD is in the spacing state, the DCD bit is a logical 1.

## Divisor Latch

The ACC baud rate is selected by programming the least significant byte (LSB) and most significant byte (MSB) of the Divisor Latch shown in Figure 11-6. The Divisor Latch is programmed with an integer value that divides the ACC oscillator reference frequency of 18.432 MHz into a value of 16 times the baud rate for both transmission and reception of serial data. The most common baud rate divisors are shown in Table 11-4 on page 11-22.

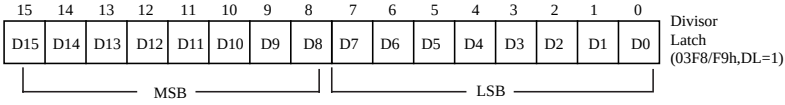



*Figure 11–6. Divisor Latch.*

## Interrupt Identify

The ACC includes 10 sources of interrupts prioritized into four categories. The Interrupt Identify register flags whether or not the ACC has an interrupt pending and, if so, from which of the four categories it originated. Figure 11-7 below shows the format for the Interrupt Identify register.

The IP bit indicates whether an interrupt is pending. A logical 0 in IP signals an active interrupt, and a logical 1 in IP signals no active interrupt. The interrupt sources are prioritized into the following four categories: receiver status, receiver data, transmitter data, and modem status, in that order. The interrupt ID field, bits 1 and 2, identifies the interrupt category if there is an active interrupt.

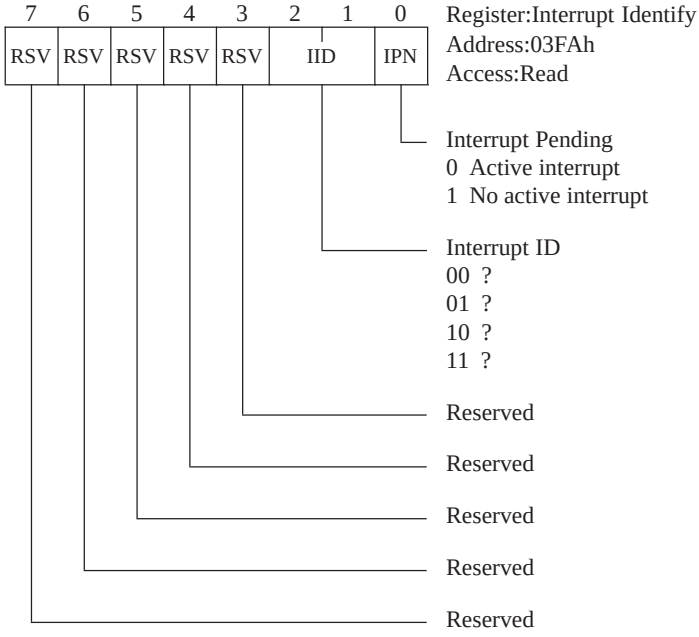



*Figure 11–7. Interrupt Identify Register.*

## Interrupt Enable

The Interrupt Enable register, shown in Figure 11-8, defines which of the four categories of interrupts are enabled and which are not. The Interrupt Enable register includes one bit for each of the four categories of interrupts. Setting bits 0 through 3 all to a logical 0 prevents the ACC from generating an interrupt of any kind and inhibits the Interrupt Identify register.

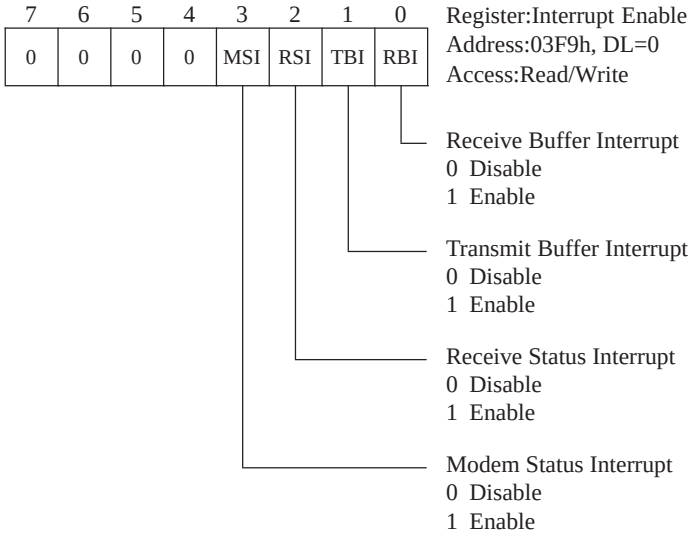



*Figure 11–8. Interrupt Enable Register.*

# OPERATION

## Reset

The ACC registers are automatically initialized to a default state after reset. Table 11-3 shows the default state for these registers.

Table 11-3
ACC Register Defaults.

| Register | Default Bit Value [1] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Transmit Buffer | - | - | - | - | - | - | - | - |
| Receive Buffer | - | - | - | - | - | - | - | - |
| Line Control | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Line Status | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Modem Control | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Modem Status | - | - | - | - | 0 | 0 | 0 | 0 |
| Divisor Latch (LSB) | - | - | - | - | - | - | - | - |
| Divisor Latch (MSB) | - | - | - | - | - | - | - | - |
| Interrupt Identify | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Interrupt Enable | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

[1] Bit positions marked with a dash (-) can default to 1 or 0.

## Serial Data Format

The ACC supports asynchronous data transfers. The format for the asynchronous data includes start, stop, and optional parity bits as well as five, six, seven, or eight data bits. This is illustrated in Figure 11-9.

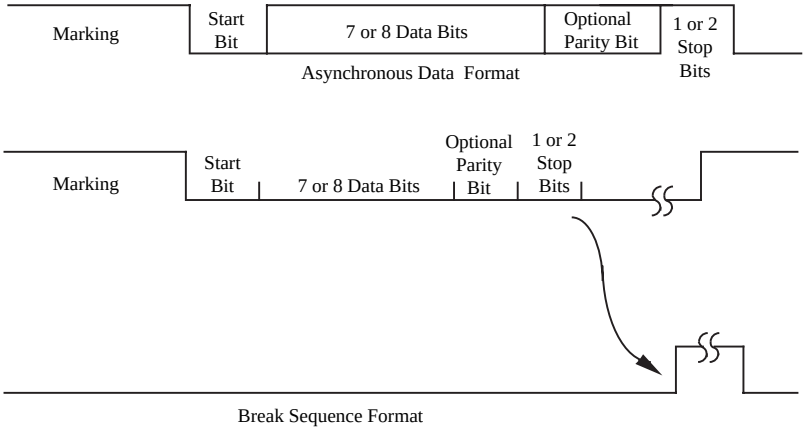



*Figure 11–9. ACC Serial Data Format.*

The TxD signal is in the "mark" state when data is not being transmitted. The start bit indicates the beginning of the serial data or character bits. The parity bit is optionally inserted during transmission and monitored during reception to test data integrity. The stop bit flags the end of the serial data.

The format for the break sequence is also shown in Figure 11-9. The transmission of the break sequence is programmed in the Line Control register, and the reception of the break sequence is flagged in the Line Status register.

## Baud Rate

The ACC includes an internal baud rate generator to divide the 18.432 MHz reference frequency down to a value equal to 16 times the baud rate. The divisor for the reference frequency is a 16-bit integer programmed into the LSB and MSB of the Divisor Latch. The relationship between the baud rate and the count is as follows:

$$\text{Divisor} = 18.432 \times 10^6/(160 \times \text{Baud Rate})$$

Table 11-4 lists the divisor to be used for the more popular baud rates. It also shows the percent error, which is based on the difference between the exact divisor for a specified baud rate and the divisor obtainable with a 16-bit integer format. To guarantee proper operation, the percent error should never be greater than four.

Table 11-4
ACC Baud Rate Divisors.

| Baud Rate | Divisor (dec/hex) | Percent Error |
|---|---|---|
| 50 | 2304/1440 | 0 |
| 75 | 1536/0960 | 0 |
| 150 | 768/0480 | 0 |
| 300 | 384/0240 | 0 |
| 600 | 192/0120 | 0 |
| 1200 | 96/0060 | 0 |
| 1800 | 64/0040 | 0 |
| 2000 | 58/003A | 0.69 |
| 2400 | 48/0030 | 0 |
| 3600 | 32/0020 | 0 |
| 4800 | 24/0018 | 0 |
| 7200 | 16/0010 | 0 |
| 9600 | 12/000C | 0 |
| 19200 | 6/0006 | 0 |
| 38400 | 3/0003 | 0 |
| 56000 | 2/0002 | 2.86 |

## Interrupt and Polled Communication

Serial data can be transferred and received either by polling status bits or with interrupts. In a polled mode, the application program monitors the Line Status register Data Ready and Transmitter Holding Register Empty bits to determine when to transfer a character to the ACC or when a character is available.

The ACC is capable of generating interrupts for applications that cannot afford time wasted in polling for a status change. If enabled in the Interrupt Enable register, a Transmitter Holding Register Empty interrupt is generated when the transmitter is free to receive a character. Also, if enabled in the Interrupt Enable register, a Received Data Available interrupt is generated when a character is received by the ACC.

## RS-485 Serial Communications

The ZT 8832 includes jumper selectable drivers and receivers for both RS-232-C and RS-485 serial communication protocols. The two main advantages of RS-485 over RS-232-C are an increased number of drivers and receivers (32 of each, versus one of each for RS-232-C) and increased transmission distance (4000 feet, as compared to 50 feet for RS-232-C).

A terminated twisted pair should be used to protect the integrity of the RS-485 signals. A typical twisted pair has a characteristic impedance in the range of 100 to 130 Ω, adequate for data transfer rates up to 10 MHz. The RS-485 balanced inputs can be terminated by installing resistors at locations R13 and R14. The surface mount resistors must have a 1206 case size, a power rating of at least 0.125 W, and a resistance value equal to the characteristic impedance of the twisted pair.

Appendix B includes the pin numbers and signal descriptions of the RS-232-C and RS-485 signals available through connector J2. (See page B-13.) When configured for RS-485, Data Terminal Equipment (DTE) is selected by plugging in the serial cable with pin 1 of the cable lined up with pin 1 of the connector. To select Data Communication Equipment (DCE), the serial cable should be plugged in with pin 1 of the cable lined up with pin 14 of the connector.

## Programming

Table 11-5 on the following pages summarizes the ACC register set for programming reference.

Table 11-5
ACC Register Summary.

| Register Address | | | | | |
|---|---|---|---|---|---|
| Bit No. | 0 DLAB = 0 | 0 DLAB = 0 | 1 DLAB = 0 | 2 | 3 |
| | Receive Buffer (Read Only) | Transmit Buffer (Write Only) | Interrupt Enable Register | Interrupt Identify Register (Read Only) | Line Control Register |
| | RBR | THR | IER | IIR | LCR |
| 0 | Data Bit 0* | Data Bit 0 | Enable Receive Buffer Full Interrupt (MSI) | "0" if Interrupt Pending | Word Length Select Bit 0 (WLS0) |
| 1 | Data Bit 1 | Data Bit 1 | Enable Transmit Buffer Empty Interrupt (RSI) | Interupt ID Bit (0) | Word Length Select Bit 1 (WLS1) |
| 2 | Data Bit 2 | Data Bit 2 | Enable Receive Status Interrupt (TBI) | Interrupt ID Bit (1) | Number of Stop Bits (STB) |
| 3 | Data Bit 3 | Data Bit 3 | Enable Modem Status Interrupt (RBI) | 0 | Parity Enable (PEN) |
| 4 | Data Bit 4 | Data Bit 4 | 0 | 0 | Even Parity Select (EPS) |
| 5 | Data Bit 5 | Data Bit 5 | 0 | 0 | Stick Parity |
| 6 | Data Bit 6 | Data Bit 6 | 0 | 0 | Set Break |
| 7 | Data Bit 7 | Data Bit 7 | 0 | 0 | Divisor Latch Access Bit (DLAB) |

*Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Table 11-5
ACC Register Summary (continued).

| Register Address | | | | | |
|---|---|---|---|---|---|
| Bit No. | 4 | 5 | 6 | 0 DLAB = 1 | 1 DLAB = 1 |
| | Modem Control Register | Line Status Register | Modem Status Register | Divisor Latch (LSB) | Divisor Latch (MSB) |
| | MCR | LSR | MSR | DLL | DLM |
| 0 | Data Terminal Ready (DTR) | Data Ready (DR) | Delta Clear to Send (DCTS) | D0 | D8 |
| 1 | Request to Send (RTS) | Overrun Error (OE) | Delta Data Set Ready (DDSR) | D1 | D9 |
| 2 | 0 | Parity Error (PE) | Trailing Edge Ring Indicator (TERI) | D2 | D10 |
| 3 | Out 2 | Framing Error (FE) | Delta Data Carrier Detect (DDCD) | D3 | D11 |
| 4 | Loop | Break Interrupt (BI) | Clear to Send (CTS) | D4 | D12 |
| 5 | 0 | Transmit Holding Register (THRE) | Data Set Ready (DSR) | D5 | D13 |
| 6 | 0 | Transmit Empty (TEMT) | Ring Indicator (RI) | D6 | D14 |
| 7 | 0 | 0 | Data Carrier Detect (DCD) | D7 | D15 |

Chapter 12

# PARALLEL I/O

**Contents** **Page**



## OVERVIEW

The three 8-bit parallel ports provide a total of 24 I/O signals. Each I/O signal can be configured either as an input or as an output with readback. The major features of the parallel I/O are listed below.

- Stable outputs during power-up and reset
- Data transfer rates up to 1 Mbyte/second
- Each I/O signal programmable for input or output with readback
- Direct connection to industry standard I/O module mounting racks

## ZT 8832 SPECIFICS

The three parallel ports are available at connector J1; see page B-11 for pin assignments. In addition to the 24 I/O signals, connector J1 also supports ground and fused +5 V for direct interface to industry standard I/O module mounting racks, such as Ziatech's ZT 2226 or those manufactured by Opto 22. The 1 A fuse is available from Littelfuse (part number 275-001).

The two most significant parallel port signals are connected to other devices in addition to the J1 connector. The most significant parallel port signal (I/O address 220h, bit 7) is connected to J1 and the watchdog timer. It functions as a strobe signal used to prevent the watchdog timer from timing out. This connection is jumper selectable. The second most significant parallel port bit (I/O address 220h, bit 6) is connected to J1 and the Light Emitting Diode (LED) located next to connector J3.

The use of the parallel port with the watchdog timer is discussed in Chapter 13. The use of the parallel port to control the LED is discussed at the end of this chapter.

## FUNCTIONAL DESCRIPTION

A functional diagram of each of the 24 I/O signals is illustrated in Figure 12-1. The diagram includes an Output Latch, an Output Buffer, and an Input Buffer. These functional blocks are described below.

### Output Latch

The Output Latch stores the data present on the internal data bus during a write operation to the parallel port. The data is latched until altered by another parallel port write or until power is turned off. While the Output Latch is not initialized after power-up, the state of the parallel I/O signal at connector J1 is a TTL high because the Output Buffer is disabled and the passive termination is in effect.

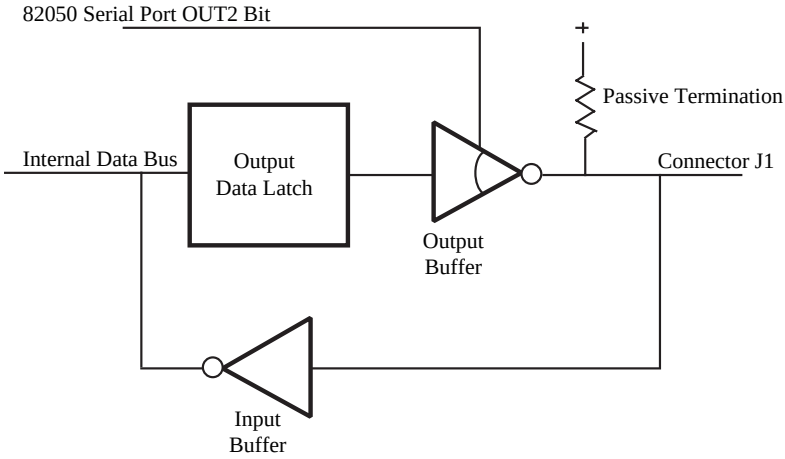



*Figure 12–1. Parallel Port Functional Diagram.*

## Output Buffer

The Output Buffer isolates the Output Latch from connector J1. The Output Buffer is disabled and enabled with the 82050 Serial Port OUT2 bit. The OUT2 connection ensures that the Output Buffer is disabled during and after power-up to prevent the I/O signals at connector J1 from glitching. The OUT2 signal also disables the Output Buffer during and after a reset.

The Output Buffer is an inverting open collector device. The inversion means that a logical 0 written to the parallel port appears as a TTL high at the J1 connector and a logical 1 written to the parallel port appears as a TTL low at the J1 connector. It is the open collector feature that permits each parallel I/O signal to be configured as an input. To use the parallel I/O signal as input, a logical 0 must first be written to the Output Latch to open collect the Output Buffer and prevent contention with the input signal.

## Input Buffer

The Input Buffer is enabled during read operations to transfer the data from connector J1 to the internal data bus. If the parallel port bit is configured as output, the data read is the last data written to the parallel port bit. If the parallel port is configured as input, the data read is the data driven by an external device.

The Input Buffer is an inverting device. This means that data read from the parallel port as a logical 0 is actually a TTL high at connector J1, and data read from the parallel port as a logical 1 is a TTL low at connector J1.

## PROGRAMMABLE REGISTERS

The 24 parallel I/O signals are accessible through three programmable registers. The address of each of the registers is shown in Table 12-1.

Table 12-1
Parallel I/O Register Addressing.

| Address (hex) | Register | Operation |
|---|---|---|
| 0200h | Parallel Port 0 | Read/Write |
| 0210h | Parallel Port 1 | Read/Write |
| 0220h | Parallel Port 2 | Read/Write |

# OPERATION

## Reset

The parallel port outputs are disabled and passively pulled to a TTL high after power up or reset.

## Programming the Parallel Ports

The parallel ports are enabled by writing a logical 1 to the 82050 serial port OUT2 bit (I/O port address 3FCh, bit 3). This operation immediately transfers the contents of the parallel ports to the J1 connector. The contents of the parallel ports are not defined after power-up. Therefore, it is recommended that all 24 bits be initialized with logical 0s before the parallel port outputs are enabled. This initialization does the following:

- Maintains the power-up and reset state of a TTL high at connector J1.
- Prevents contention with external devices driving the parallel I/O signals.
- Ensures that a jumper selected watchdog timer is not armed.

Once enabled, the parallel port bits are programmed with standard input and output instructions. The parallel ports are inverting so that data written to the parallel port appears in the opposite state at connector J1 and data read from the parallel port reflects the inverted state of the data at connector J1.

The V40 supports string I/O instructions for transferring data between the parallel I/O and memory at rates of up to 1 Mbyte/second. The high throughput of these instructions occurs because the memory address is automatically incremented or decremented after each transfer.

## Programming the Light Emitting Diode

- To control the LED, first enable the parallel port by programming the OUT2 bit of the 82050 serial port (I/O port address 3FCh, bit 3) with a logical 1.
- To turn on the LED, write a logical 1 to the second most significant parallel I/O bit (I/O port address 220h, bit 6).
- To turn off the LED, write a logical 0 to the second most significant parallel I/O bit (I/O port address 220h, bit 6).

## Mixing I/O in a Single Port

Each I/O signal is programmable for input or output operation. If inputs and outputs are mixed in a single port, you must take special care to always write logical 0s to the bit locations configured as outputs.

For example, assume that the lower four bits of a port are inputs and the upper four bits of the same port are outputs. For a read-modify-write cycle, you must read the port, modify the output bits as necessary, set the lower four bits to 0, and write the port.

Chapter 13

# WATCHDOG TIMER

**Contents** **Page**

## OVERVIEW

The primary function of the watchdog timer is to monitor ZT 8832 operation and to take corrective action if the ZT 8832 fails to function as programmed. The watchdog timer includes two stages. The first-stage timeout generates a non-maskable interrupt. The second-stage timeout generates a local reset. The major features of the watchdog timer are listed below.

- Enabled and disabled through jumper selection
- Armed and strobed through software control
- Increased programming flexibility with two stages

## ZT 8832 SPECIFICS

The watchdog timer must be jumper selected before it is operational. If the watchdog timer is selected, the most significant bit of the parallel ports becomes dedicated to the watchdog timer and cannot be used for general purpose I/O.

## FUNCTIONAL DESCRIPTION

Figure 13-1 illustrates a functional diagram of the watchdog timer. The diagram includes a timer and a delay for each stage. The functional blocks are described below.

### Stage 1 Timer

The first stage of the watchdog timer generates a non-maskable interrupt to the local CPU if the watchdog timer is jumper selected and armed and a strobe does not occur within the time period defined by the Stage 1 Delay functional block.

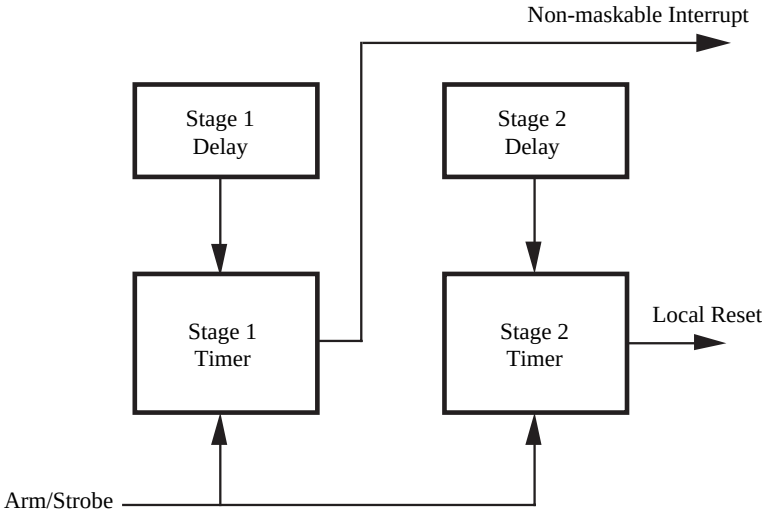



*Figure 13–1. Watchdog Timer Functional Diagram.*

### Stage 1 Delay

The stage 1 delay has a default range of 60 ms minimum and 100 ms maximum. The minimum delay time means that the watchdog timer must be strobed with a period of less than 60 ms to prevent stage 1 from generating a non-maskable interrupt. The maximum delay time means that it could take up to 100 ms after the watchdog timer is strobed before the non-maskable interrupt occurs.

### Stage 2 Timer

The second stage of the watchdog timer generates a reset to the local CPU if the watchdog timer is jumper selected and armed and a strobe does not occur within the time period defined by the Stage 2 Delay functional block.

### Stage 2 Delay

The stage 2 delay has a default range of 250 ms minimum and 1000 ms maximum. The stage 1 delay time must be subtracted from the stage 2 delay time to determine how long the non-maskable interrupt service routine must strobe the stage 2 timer to prevent a local reset. For the default configuration, this time is 150 (250 - 100) ms minimum and 940 (1000 - 60) ms maximum. The minimum delay time means that the watchdog timer must be strobed sooner than 150 ms after the stage 1 timeout to prevent a local reset. The maximum delay time means that it could take up to 940 ms after the non-maskable interrupt before the local reset occurs if the watchdog is not strobed.

## OPERATION

In operation, the local CPU is programmed to strobe the watchdog timer at a periodic rate less than the stage 1 time delay. If the local CPU fails to operate as programmed, stage 1 of the watchdog timer generates a non-maskable interrupt. The non-maskable interrupt service routine takes the necessary corrective action that includes strobing the watchdog timer before the stage 2 time delay to prevent a local reset. If the local reset is desired, the non-maskable interrupt service routine simply does not strobe the watchdog timer.

### Reset

The watchdog timer is disarmed during and after both a power-up and reset condition.

Stage 2 of the watchdog timer generates a local reset if allowed to time out. This reset period lasts for up to 1 second. Since the STD bus CPU is unaffected by this reset, it might attempt an I/O or dual port access to the ZT 8832. If this occurs, the operation of the STD bus CPU is suspended until the reset period is over. This suspension will cause problems in a system having critical interrupt latencies, DMA latencies, asynchronous data transfers, or dynamic RAM that must be refreshed by the STD bus CPU.

## Multiple Stages

Many watchdog timers are implemented with a single stage that generates a reset if allowed to time out. The problem with this implementation is that the CPU does not have advance warning of the reset. Without advance warning, the CPU cannot take corrective action that includes, as a minimum, setting a flag indicating that a system failure has occurred.

Other watchdog timers are implemented with a single stage that generates a non-maskable interrupt if allowed to time out. The problem with this implementation is that the pointer to the non-maskable interrupt service routine is generally stored in RAM that can be overwritten by a CPU not operating as programmed.

The solution to these problems is to implement a two-stage watchdog timer such as the one found on the ZT 8832. If the first stage times out, a non-maskable interrupt is generated to invoke a service routine that takes the necessary corrective action. If the second stage times out, a local reset is generated to put the ZT 8832 in a known operating state.

## Changing Time Delays

Table 13-1 shows the two possibilities for the stage delays. The stage 1 delay is measured from the watchdog strobe to the non-maskable interrupt. The stage 2 delay is measured from the watchdog strobe to the local reset. The stage 2 - stage 1 delay is calculated as the non-maskable interrupt to local reset delay.

The first entry in Table 13-1 shows the default stage delays. The second entry shows an option. The stage 1 delay is implemented with a discrete surface mount resistor and capacitor. This delay cannot be changed. The stage 2 delay can be changed by using cuttable traces, as described on page A-18.

Table 13-1
Watchdog Timer Stage Delays.

| | **Stage 1 (ms)** | **Stage 2 (ms)** | **Stage 2-Stage 1 (ms)** |
|---|---|---|---|
| **Default** | 60 min<br>100 max | 250 min<br>1000 max | 150 min<br>940 max |
| **Optional** | 60 min<br>100 max | 500 min<br>2000 max | 400 min<br>1940 max |

## Programming

The watchdog timer is armed and strobed with the most significant parallel I/O signal. The watchdog timer is armed with the following programming sequence.

1. Initialize the most significant bit of the parallel ports (I/O port address 220h, bit 7) with a logical 0.
2. Enable the parallel ports by writing a logical 1 to the 82050 OUT2 bit (I/O port address 3FCh, bit 3).
3. Arm the watchdog timer by programming the most significant bit of the parallel ports with a logical 1.

The watchdog timer is strobed with the following programming sequence.

1. Disarm the watchdog timer by writing a logical 0 to the most significant bit of the parallel ports.
2. Rearm the watchdog timer by writing a logical 1 to the most significant bit of the parallel ports.

In addition to the watchdog timer non-maskable interrupt, the ZT 8832 also supports 8087 Numeric Data Processor and STD bus control port non-maskable interrupts. If more than one source of non-maskable interrupt is used in an application, the non-maskable interrupt service routine must be able to identify the source of the interrupt.

The watchdog timer non-maskable interrupt request is also routed into IRQ1 of the interrupt controller. This enables the non-maskable interrupt service routine to read the interrupt controller to determine if the watchdog timer has generated a non-maskable interrupt request.

Chapter 14

# SBX EXPANSION MODULE

**Contents** **Page**

## OVERVIEW

The SBX expansion module provides a method for expanding the I/O capabilities of the ZT 8832. The expansion module interface is electrically, mechanically, and functionally compatible with the Intel iSBX MULTIMODULE standard. This level of compatibility ensures that expansion modules produced by other manufacturers will operate with the ZT 8832. Some of the functions available on expansion modules are:

- Serial and parallel I/O
- Stepper and servo motor controllers
- Analog-to-digital and digital-to-analog converters
- Disk and SCSI controllers
- Modems
- Video controllers
- IEEE 488 controllers
- Bar code readers
- Prototyping boards for customized I/O designs

## Features

The major features of the expansion module interface are listed below.

- Standard interface for expanding I/O capabilities
- Compatible to Intel iSBX MULTIMODULE specification
- DMA support for high speed data transfers
- Added address lines for custom designs
- Optional V40 CPU clock for custom synchronous designs
- Supports both single-wide and double-wide expansion modules

## ZT 8832 SPECIFICS

The expansion module interface is supported through connector J4; the pin assignments given on page B-15. The expansion module supports Direct Memory Access (DMA) using the DMA controller discussed in Chapter 9. The DMA controller transfers data between the expansion module and either local or dual port RAM at rates of up to 1 1/3 Mbytes per second. The DMA signals supported are DMA Request (MDRQT) and DMA Acknowledge (MDACK*). The Terminate DMA (TDMA) signal is not supported.

The expansion module standard defines three address lines and two chip selects. This provides a total of 16 I/O port addresses. To overcome this limitation, the ZT 8832 expansion module adds four address lines. These address lines are connected in the default configuration and can be removed using cuttable traces (see page A-19 for details). Without these address lines, the chip select 0 signal (MCS0*) is valid over the I/O address range 2F8 through 2FFh, and the chip select 1 signal (MCS1*) is valid over the I/O address range 300 through 307h. As the added address lines are used, the chip select 0 signal grows downward in I/O address space and the chip select 1 signal grows upward. For example, if all seven address lines are used, chip select 0 is mapped to I/O address 280 through 2FFh and chip select 1 is mapped from I/O address 300 through 37Fh.

The expansion module supports two interrupt request signals for interrupt driven communications. Interrupt Request 0 (MINTR0*) is connected to IRQ2 of the interrupt controller, and Interrupt Request 1 (MINTR1*) is connected to IRQ3 of the interrupt controller. The interrupt controller is explained in detail in Chapter 8.

The expansion Module Present (MPST*) signal is not supported.

## INSTALLATION

The SBX expansion module is installed on the ZT 8832 as shown in Figure 14-1. The module is mechanically secured to the ZT 8832 at the J4 connector and with the threaded spacer shipped with the expansion module.

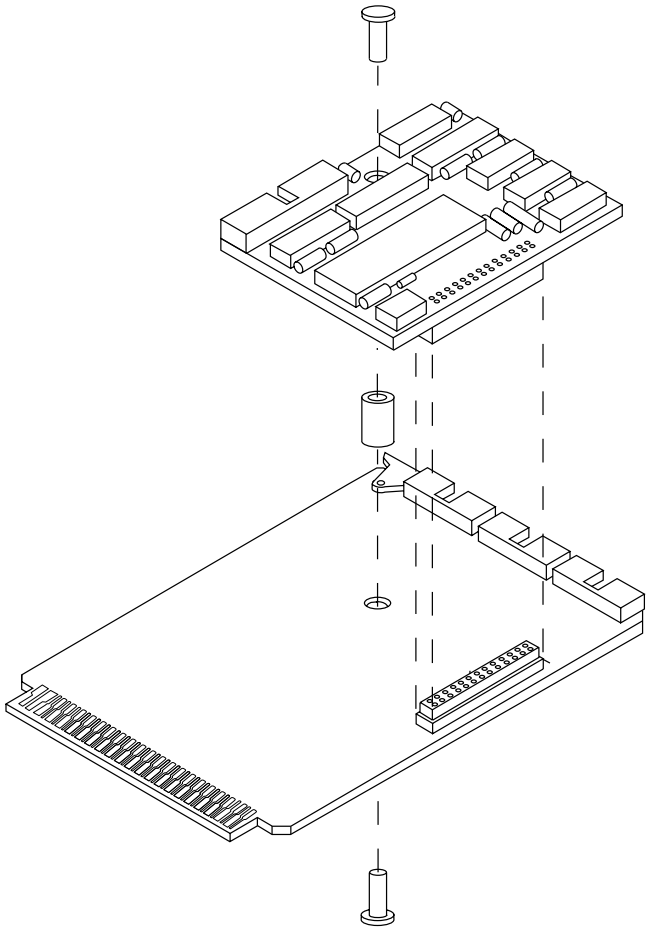


*Figure 14–1. SBX Expansion Module Installation.*

Chapter 15

# NUMERIC DATA PROCESSOR (8087)

**Contents** **Page**

## OVERVIEW

The V40 is a high performance microprocessor designed for a wide variety of applications. The math capabilities of the V40 include addition, subtraction, multiplication, and division of 8-bit and 16-bit numbers. However, the STD bus is often used in numerically intensive applications needing more powerful arithmetic operations and data types than those provided by the V40. The addition of the 8087 Numeric Data Processor (NDP) will provide the following benefits.

- 68 additional instructions for extended arithmetic, trigonometric, exponential, and logarithmic functions
- Seven added data types, including integers (16-, 32-, and 64-bit), floating point (32-, 64-, and 80-bit), and BCD (18-digit)
- A typical performance increase of 100 times over software math routines
- Compatibility with the IEEE 754 Floating Point standard
- Optional automatic error handling

## ZT 8832 SPECIFICS

The NDP provides an interrupt to signal exception errors. This interrupt is optionally connected to a non-maskable interrupt request through jumper selection. The numeric data processor includes a status word that indicates whether or not an interrupt request is active.

## INSTALLATION

The NDP is installed in an empty socket on the ZT 8832 as shown in Figure 15-1. The 8087-1 (10 MHz) is required for proper operation. Be sure power is not applied to the ZT 8832 during installation.

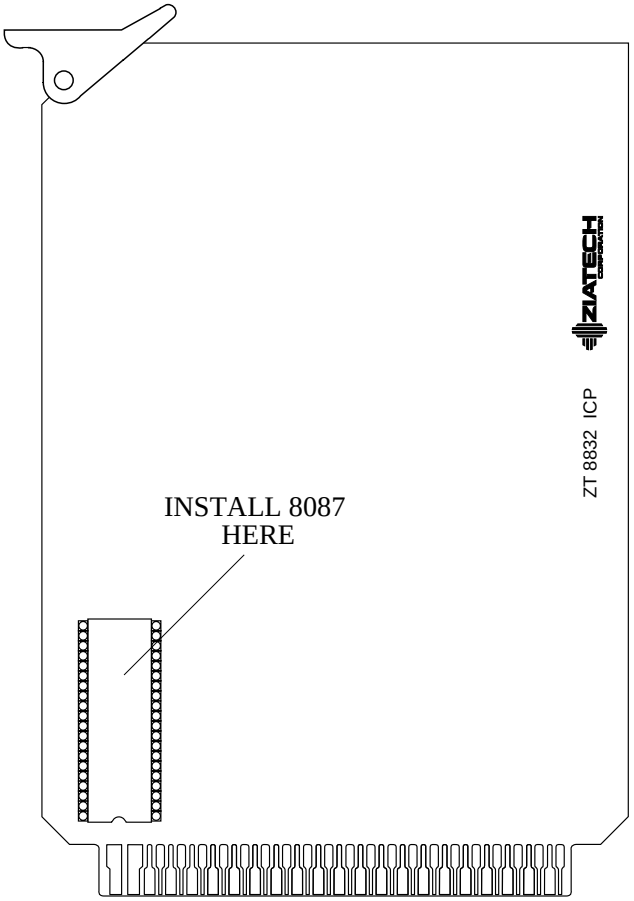



*Figure 15–1. 8087 Numeric Data Processor Installation.*

# OPERATION

The following description is an overview of NDP operation. See the reference section at the end of this chapter for sources with more detailed explanations.

## Coprocessor Interface

### Instruction Level Support

The purpose of the NDP is to extend the math capabilities of the V40. This includes instruction level support for high-precision integer and floating point data types with operations such as add, subtract, multiply, divide, square root, exponent, logarithmic, and trigonometric. For the programmer, the NDP adds 68 numerical instructions and seven data types to the V40 instruction set. The instructions are referred to as ESCAPE instructions because they include an ESCAPE prefix. The ESCAPE prefix signals the V40 and the NDP that a numeric instruction is to be executed.

The NDP maintains a copy of the V40 instruction queue. Instructions fetched by the V40 are also monitored by the NDP. The NDP checks the first byte of each instruction looking for an ESCAPE prefix. If the first byte is not an ESCAPE prefix, the NDP ignores the instruction. If the first byte of an instruction is an ESCAPE prefix, the NDP decodes the numeric instruction in parallel with the V40.

An NDP numerical instruction has one of three options:

1. No reference to memory
2. Read operand from memory
3. Write operand to memory

If no reference to memory is needed, the NDP executes the instruction. If the instruction includes a reference to memory, the V40 executes a "dummy read" cycle, so called because the V40 ignores the data read. Because the NDP does not include an address generator, it must depend on the V40 to generate the operand starting address.

### Read and Write Operations

If a read operation is required, the NDP latches the address and operand as it appears. If the operand is more than one word long, the NDP is granted access to the address, data, and control buses for subsequent read operations. This means that although the NDP does not have the capability of determining the operand starting address, it can latch the starting address and increment it for subsequent operations. If the instruction includes a write to memory operation, the NDP latches the address and, like the V40, ignores the data during the dummy read. The NDP is then granted control of the buses to complete the write operation.

The NDP must get control of the V40 address, data, and control buses to write numerical operands to memory or to read numerical operands longer than one word from memory. A request/grant/release protocol is then used between the V40 and the NDP to carry out this transfer of control:

1. The NDP generates a request to the V40 to gain control of the bus resources.
2. The V40 responds by releasing the bus and granting control to the NDP.
3. The NDP completes the memory operation and releases control back to the V40.

This entire operation is transparent to the programmer since it is done with hardware.

The V40 can continue to fetch and execute instructions while the NDP is performing numerical computations. The V40 WAIT instruction can be used to suspend V40 operation until the NDP has completed the numerical computation.

## Error Handling and Interrupts

A numeric error occurs if an operation is attempted with invalid operands or if the result of a computation cannot be accurately represented. Six defined error conditions can occur during the execution of a numeric instruction:

- Invalid operation
- Overflow
- Zero divisor
- Underflow
- Denormalized operand
- Inexact result

The NDP can be programmed to interrupt the V40 during any or all of these errors. For an interrupt to be recognized, the NDP must be programmed to enable interrupts and remove the mask of the selected error condition. If the NDP interrupts are disabled when an error occurs, the V40 is not interrupted and NDP operation continues. If the NDP interrupts are enabled and an error occurs that is masked, the NDP will flag the error in a status register and automatically execute a default handling procedure. If the NDP interrupts are enabled and an error occurs that is not masked, an interrupt will be generated to the V40. This interrupt must be supported with an interrupt service routine in the application program.

## Further Reference


- Cooner, Jerome, "An Implementation Guide to a Proposed Standard for Floating Point," Computer, Institute of Electrical and Electronic Engineers, Jan. 1980.
- Palmer, John, & Wymore, Charles, "Making Mainframe Mathematics Accessible to MicroComputers," Electronics, 8 May 1982. (or AR-135 from Intel Corporation)
- Rash, Bill, "Getting Started with the Numeric Data Processor," Intel Corporation.
- "The NDP User's Manual Numerics Supplement," Intel Corporation.
- "Microsystem Components Handbook," Volume I, Intel Corporation.

# IV. APPENDICES

Appendix A

# JUMPER CONFIGURATIONS

**Contents** **Page**



## OVERVIEW

The ZT 8832 includes several jumper options that tailor the operation of the board to meet specific application requirements. Options are selected by installing and removing shorting receptacles (jumpers). Several less popular options can also be selected by installing or removing cuttable traces.

This appendix provides detailed descriptions of all jumper and cuttable trace options, as well as an illustration of the board showing the locations of all jumpers and cuttable traces.

## JUMPER OPTIONS

Table A-1 below lists the jumpers associated with each option. It also indicates the pages on which descriptions of these jumpers can be found.

Table A-2 beginning on page A-3 describes each jumper option in detail. A dagger (†) in Table A-2 indicates a standard default jumper configuration.

Figure A-1 on page A-14 shows the factory default jumper configuration. Figure A-2 on page A-15 provides a blank layout to document your custom jumper configuration.

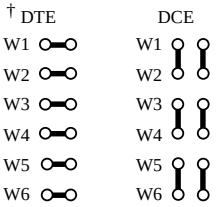
Cuttable trace descriptions begin on page A-16.

Table A-1
Jumper Cross Reference.

| Function | Jumper(s) | Page(s) |
|---|---|---|
| Battery Backup | W7,8 | A-4 |
| Board Select | W41-43 | A-12 |
| Dual Port RAM | W28-36 | A-8 - A-10 |
| Local RAM | W21,22,26,27 | A-6, A-7 |
| Local ROM | W23-25 | A-7 |
| Numeric Data Processor | W20 | A-6 |
| 82050 Serial Port | W1-6, W9-18 | A-3, A-4 |
| STD Bus Interrupts | W44-46 | A-13 |
| STD Bus I/O | W37-40 | A-11 |
| Watchdog Timer | W19 | A-5 |

Table A-2
ZT 8832 Jumper Descriptions.

| JUMPER # | DESCRIPTION |
|---|---|
| **W1-W6** | J2 DCE/DTE Selection - configures the 82050 serial port for RS-232 Data Terminal Equipment (DTE) or Data Communication Equipment (DCE). See also W9-W18 (page A-4). Remove W1-W6 for RS-485 operation. |




†Factory default jumper configuration.

Table A-2
ZT 8832 Jumper Descriptions (continued).

| JUMPER # | DESCRIPTION |
|---|---|
| **W7, W8** | Battery Backup Device Selection - determines whether the local RAM devices, RAM LOW and RAM HIGH, are powered by the battery when the system power is turned off. Note that this operation is valid only if the optional battery is present. The RAM devices must be designed for low power operation (less than 15 µA data retention current). Note that the dual port RAM is always battery-backed if the battery is installed. |

| W7 | W8 | Battery-backed Devices |
|---|---|---|
| In | Out | Dual port, RAM LOW, and RAM HIGH |
| † Out | In | Dual port |

| JUMPER # | DESCRIPTION |
|---|---|
| **W9-15, W18** | J2 RS-232/485 Selection - configures the 82050 serial port for RS-232 or RS-485 operation. See also W1-W6 (page A-3) and W16 and W17 (page A-5). |

| W9, W10, W13, W18 | W11, W12, W14, W15 | Operation |
|---|---|---|
| In | Out | Selects RS-485 |
| † Out | In | Selects RS-232 |

†Factory default jumper configuration.

Table A-2
ZT 8832 Jumper Descriptions (continued).

**JUMPER #** **DESCRIPTION**

**W16, W17** J2 RS-485 Output Enable - selects the method of enabling the RS-485 Transmit Data and Request-To-Send drivers. With W16 installed, the drivers are disabled. With both W16 and W17 removed, the drivers are always enabled. With W17 installed, the drivers are enabled with the DTR bit of the serial port (I/O port address 3FCh, bit 0). Writing a logical 1 to DTR enables the drivers and writing a logical 0 disables them. The power-on state of DTR is a logical 0. See also W1-W6 (page A-3) and W9-15 and W18 (page A-4).

| | W16 | W17 | Operation |
|---|---|---|---|
| † | In | Out | Always disabled |
| | Out | In | DTR control |
| | Out | Out | Always enabled |

**W19** Watchdog Timer.

| | W19 | Operation |
|---|---|---|
| | In | Enables watchdog timer |
| † | Out | Disables watchdog timer |

†Factory default jumper configuration.

Table A-2
ZT 8832 Jumper Descriptions (continued).

**JUMPER #** **DESCRIPTION**

**W20**

Numeric Data Processor Interrupt - enables the Numeric Data Processor (NDP) to generate a non-maskable interrupt in response to an exception error. This jumper must not be installed if the optional NDP is unplugged.

| | **W20** | **Operation** |
|---|---|---|
| | In | Enables non-maskable interrupt |
| † | Out | Disables non-maskable interrupt |

**W21, W22**

RAM Device Size - selects the address range of the RAM devices to be plugged into the RAM LOW and RAM HIGH sockets. Address ranges are given in hexadecimal format. See also W7 and W8 (page A-4) and W26 and W27 (page A-7).

| | **W22** | **W21** | **Device** | **RAM LOW** | **RAM HIGH** |
|---|---|---|---|---|---|
| † | In | In | 32K x 8 | 0-07FFFh | 08000-0FFFFh |
| | In | Out | 128K x 8 | 0-1FFFFh | 20000-3FFFFh |
| | Out | In | 512K x 8 | 0-7FFFFh | --------- |
| | Out | Out | Not used | --------- | --------- |

†Factory default jumper configuration.

Table A-2
ZT 8832 Jumper Descriptions (continued).

**JUMPER #** **DESCRIPTION**

**W23-W25** ROM Device Type - configures the ROM socket for a selected device type. The ROM address range is fixed from 88000h through FFFFFh (480 Kbytes). The 8K, 16K, 32K, 64K, 128K, and 256K devices are redundantly mapped into this address range, and only 480K of the 512K device is used.

| | W23 | W24 | W25 | Device |
|---|---|---|---|---|
| | Out | Out | In | 8K x 8 |
| | Out | Out | In | 16K x 8 |
| | Out | Out | In | 32K x 8 |
| † | In | Out | In | 64K x 8 |
| | In | Out | In | 128K x 8 |
| | In | In | Out | 256K x 8 |
| | In | In | Out | 512K x 8 |

**W26, W27** RAM Device Type - configures the RAM LOW and RAM HIGH sockets for the selected device type. The address range is defined by W21 and W22 (page A-6).

| | W26 | W27 | Device |
|---|---|---|---|
| † | Out | In | 32K x 8 |
| | Out | In | 128K x 8 |
| | In | Out | 512K x 8 (RAM LOW only) |

†Factory default jumper configuration.

Table A-2
ZT 8832 Jumper Descriptions (continued).

| JUMPER # | DESCRIPTION |
|---|---|
| **W28-W32** | STD Bus Dual Port RAM Addressing (20-bit) - defines the address range of the dual port memory as seen by the STD bus CPU. These jumpers map the ZT 8832 dual port RAM into any contiguous 32 Kbyte block within the 1 Mbyte address range for an STD bus supporting 20 address lines.<br><br>The table on page A-9 shows the address range of the dual port memory for each jumper combination. Note that the dual port memory, as seen by the local CPU, is mapped at address 80000h through 87FFFh.<br><br>See jumpers W33-W36 (page A-10) for mapping the ZT 8832 into the 16 Mbyte address space for STD bus CPUs that support 24 address lines. |

Table A-2
ZT 8832 Jumper Descriptions (continued).

| W32 | W31 | W30 | W29 | W28 | Address Range |
|---|---|---|---|---|---|
| In | In | In | In | In | 00000-07FFFh |
| In | In | In | In | Out | 08000-0FFFFh |
| In | In | In | Out | In | 10000-17FFFh |
| In | In | In | Out | Out | 18000-1FFFFh |
| In | In | Out | In | In | 20000-27FFFh |
| In | In | Out | In | Out | 28000-2FFFFh |
| In | In | Out | Out | In | 30000-37FFFh |
| In | In | Out | Out | Out | 38000-3FFFFh |
| In | Out | In | In | In | 40000-47FFFh |
| In | Out | In | In | Out | 48000-4FFFFh |
| In | Out | In | Out | In | 50000-57FFFh |
| In | Out | In | Out | Out | 58000-5FFFFh |
| In | Out | Out | In | In | 60000-67FFFh |
| In | Out | Out | In | Out | 68000-6FFFFh |
| In | Out | Out | Out | In | 70000-77FFFh |
| In | Out | Out | Out | Out | 78000-7FFFFh |
| Out | In | In | In | In | 80000-87FFFh |
| Out | In | In | In | Out | 88000-8FFFFh |
| Out | In | In | Out | In | 90000-97FFFh |
| Out | In | In | Out | Out | 98000-9FFFFh |
| Out | In | Out | In | In | A0000-A7FFFh |
| Out | In | Out | In | Out | A8000-AFFFFh |
| Out | In | Out | Out | In | B0000-B7FFFh |
| Out | In | Out | Out | Out | B8000-BFFFFh |
| Out | Out | In | In | In | C0000-C7FFFh |
| †Out | Out | In | In | Out | C8000-CFFFFh |
| Out | Out | In | Out | In | D0000-D7FFFh |
| Out | Out | In | Out | Out | D8000-DFFFFh |
| Out | Out | Out | In | In | E0000-E7FFFh |
| Out | Out | Out | In | Out | E8000-EFFFFh |
| Out | Out | Out | Out | In | F0000-F7FFFh |
| Out | Out | Out | Out | Out | F8000-FFFFFh |

†Factory default jumper configuration.

Table A-2
ZT 8832 Jumper Descriptions (continued).

| JUMPER # | DESCRIPTION |
|---|---|
| **W33-W36** | STD Bus Dual Port RAM Addressing (24-bit) - defines the address range of the dual port memory as seen by the STD bus CPU. These jumpers, along with W28-W32, map the ZT 8832 dual port RAM into any contiguous 32 Kbyte block in the 16 Mbyte address range for an STD bus supporting 24 address lines. Install cuttable trace CT14 (see page A-20) to select addressing in the upper 8 Mbytes. Cut CT14 to select the lower 8 Mbytes. *CT1 through CT4 must be cut (see page A-17) before W33-W36 can be used.* |

| | CT14 | W36 | W35 | W34 | W33 | Address Range |
|---|---|---|---|---|---|---|
| | Out | Out | In | In | In | 0XXXXXh |
| | Out | Out | In | In | Out | 1XXXXXh |
| | Out | Out | In | Out | In | 2XXXXXh |
| | Out | Out | In | Out | Out | 3XXXXXh |
| | Out | Out | Out | In | In | 4XXXXXh |
| | Out | Out | Out | In | Out | 5XXXXXh |
| | Out | Out | Out | Out | In | 6XXXXXh |
| | Out | Out | Out | Out | Out | 7XXXXXh |
| | In | In | In | In | In | 8XXXXXh |
| | In | In | In | In | Out | 9XXXXXh |
| | In | In | In | Out | In | AXXXXXh |
| | In | In | In | Out | Out | BXXXXXh |
| | In | In | Out | In | In | CXXXXXh |
| | In | In | Out | In | Out | DXXXXXh |
| | In | In | Out | Out | In | EXXXXXh |
| † | In | In | Out | Out | Out | FXXXXXh |

†Factory default jumper configuration.

Table A-2
ZT 8832 Jumper Descriptions (continued).

| JUMPER # | DESCRIPTION |
|---|---|
| **W37-W40** | STD Bus I/O Port Addressing - defines the address range of the 16 contiguous I/O ports as seen by the STD bus CPU. |

| | W40 | W39 | W38 | W37 | Address Range |
|---|---|---|---|---|---|
| | In | In | In | In | FFF0-FFFFh |
| † | In | In | In | Out | 7FF0-7FFFh |
| | In | In | Out | In | 3FF0-3FFFh |
| | In | In | Out | Out | 1FF0-1FFFh |
| | In | Out | In | In | 0FF0-0FFFh |
| | In | Out | In | Out | 07F0-07FFh |
| | In | Out | Out | In | 03F0-03FFh |
| | In | Out | Out | Out | 01F0-01FFh |
| | Out | In | In | In | 00F0-00FFh |
| | Out | In | In | Out | 0070-007Fh |
| | Out | In | Out | In | 0030-003Fh |
| | Out | In | Out | Out | 0010-001Fh |
| | Out | Out | In | In | 00E0-00EFh |
| | Out | Out | In | Out | 0060-006Fh |
| | Out | Out | Out | In | 0020-002Fh |
| | Out | Out | Out | Out | 0000-000Fh |

†Factory default jumper configuration.

Table A-2
ZT 8832 Jumper Descriptions (continued).

| JUMPER # | DESCRIPTION |
|---|---|
| **W41-W43** | Board Select Addressing - defines the board address in a board selection scheme that allows up to seven ZT 8832s to be mapped into the same STD bus memory and I/O address space. |

| | W43 | W42 | W41 | Board Address (hex) |
|---|---|---|---|---|
| † | In | In | In | Board select disabled |
| | In | In | Out | 1 |
| | In | Out | In | 2 |
| | In | Out | Out | 3 |
| | Out | In | In | 4 |
| | Out | In | Out | 5 |
| | Out | Out | In | 6 |
| | Out | Out | Out | 7 |

†Factory default jumper configuration.

Table A-2
ZT 8832 Jumper Descriptions (continued).

**JUMPER #** **DESCRIPTION**

**W44-W46** STD Bus Interrupt Selection - defines which STD bus interrupt request signal is driven by the ZT 8832.

| W44 | W45 | W46 | Interrupt Signal |
|---|---|---|---|
| In | Out | Out | INTRQ2* (STD bus pin 50) |
| Out | In | Out | INTRQ1* (STD bus pin 37) |
| † Out | Out | In | INTRQ* (STD bus pin 44) |

†Factory default jumper configuration.

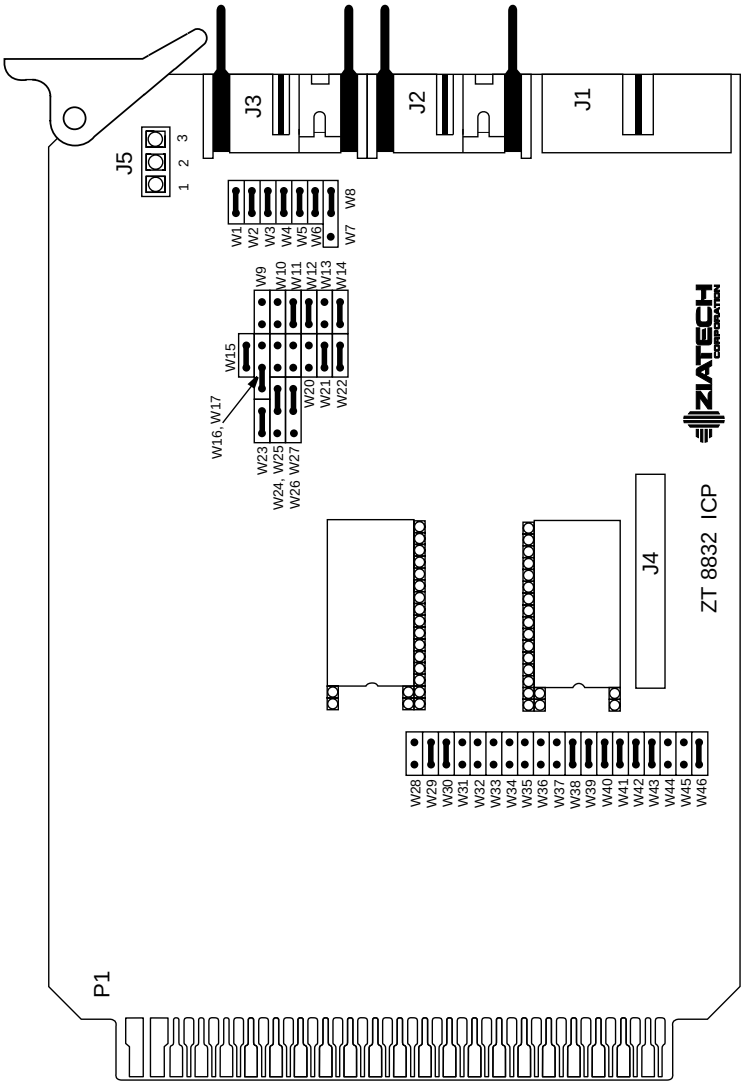



*Figure A–1. ZT 8832 Factory Default Configuration.*

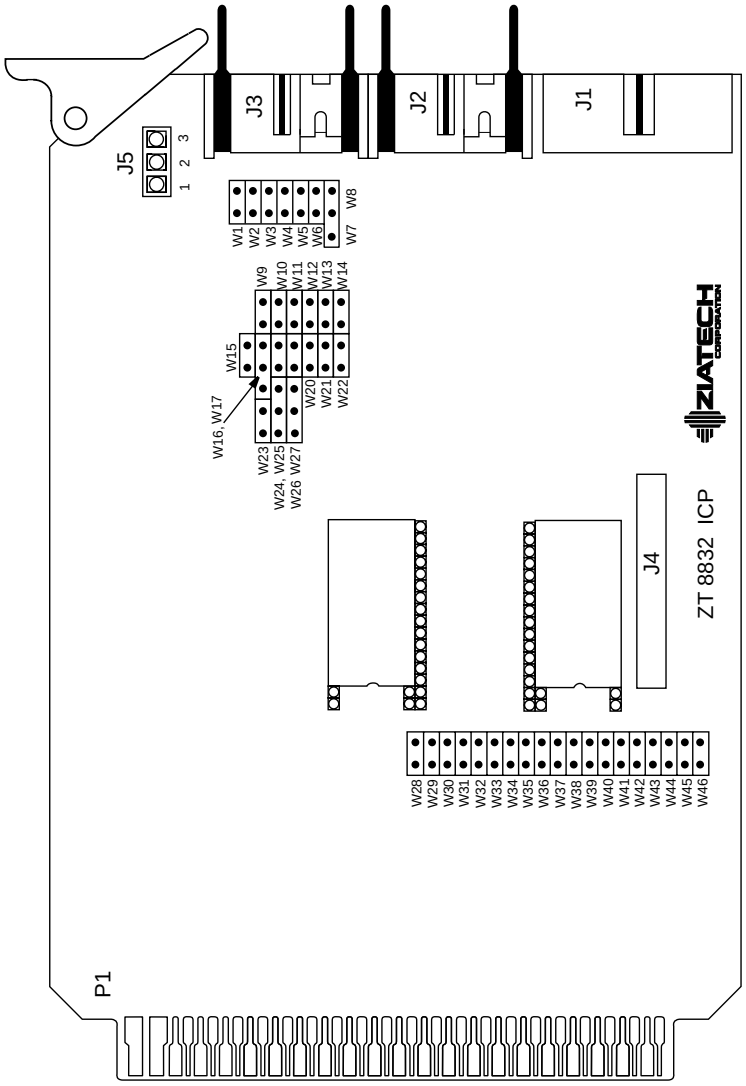



*Figure A–2. Jumper Locations.*

## CUTTABLE TRACES

The ZT 8832 supports several less popular options with cuttable traces. Cuttable traces are similar in function to jumper selections. The difference is that an option change made with a cuttable trace may require a trace cut and/or a short wire to be soldered between two pads. Cuttable traces are labeled CTx on the board, where *x* defines the cuttable trace number.

Figures A-3 and A-4 on pages A-21 and A-22 illustrate cuttable trace locations. You may wish to document your custom configuration using these figures.

Table A-3 below lists the cuttable traces associated with each option. It also indicates the pages on which descriptions of these cuttable traces can be found.

Cuttable trace options are summarized in Table A-4 on the following pages.

"In" next to an entry means the cuttable trace pads are connected. "Out" means the cuttable trace pads are not connected.

A dagger (†) indicates a standard default configuration.

Table A-3
Cuttable Trace Cross Reference.

| Function | Trace(s) | Page(s) |
|---|---|---|
| Dual Port RAM: | | |
| 24-Bit Option | CT3-5 | A-17 |
| Upper or Lower 8 Mbytes | CT14 | A-20 |
| Ground Return | CT10 | A-20 |
| SBX Expansion Module: | | |
| Clock Source | CT1,2 | A-17 |
| Address Expansion | CT9,11-13 | A-19 |
| Watchdog Timer | CT7,8 | A-18 |

Table A-4
ZT 8832 Cuttable Traces.

**TRACE #** **DESCRIPTION**

**CT1, CT2** SBX Expansion Module Clock - selects the clock for the SBX expansion module. The default is a 10 MHz 50% duty cycle signal defined by the Intel SBX expansion module standard. The optional V40 clock is useful for custom SBX expansion module designs that must be synchronized to the local CPU.

| | CT1 | CT2 | Clock |
|---|---|---|---|
| † | In | Out | 10 MHz |
| | Out | In | V40 CPU |

**CT3-CT5** STD Bus Dual Port Memory Addressing (24-bit) - selects the 24-bit addressing option for STD bus CPUs that have a 16 Mbyte addressing range. Once CT3-CT5 are cut, W33 through W36 become functional for selecting the appropriate address range.

| | CT3 | CT4 | CT5 | Operation |
|---|---|---|---|---|
| † | In | In | In | 20-bit dual port RAM addressing |
| | Out | Out | Out | 24-bit dual port RAM addressing |

†Factory default configuration.

Table A-4
ZT 8832 Cuttable Traces (continued).

| TRACE # | DESCRIPTION |
|---|---|
| **CT6** | Reserved for Ziatech use. |
| **CT7, CT8** | Watchdog Timer Time Out - selects the delay from the watchdog timer strobe to the local reset. The watchdog timer includes two stages. The Stage 1 delay is measured from the watchdog strobe to the non-maskable interrupt. The Stage 2 delay is measured from the watchdog strobe to the local CPU reset. The time from the non-maskable interrupt to the local reset is calculated by subtracting the Stage 1 delay from the Stage 2 delay. |

| CT7 | CT8 | Stage 1 Time (ms) | Stage 2 Time (ms) | Stage 2 - Stage 1 Time (ms) |
|---|---|---|---|---|
| † Out | Out | 60 min,<br>100 max | 250 min,<br>1000 max | 150 min,<br>940 max |
| In | Out | 60 min,<br>100 max | 500 min,<br>2000 max | 400 min,<br>1940 max |

†Factory default configuration.

Table A-4
ZT 8832 Cuttable Traces (continued).

| TRACE # | DESCRIPTION |
|---|---|
| **CT9, CT11-CT13** | SBX Expansion Module Address Expansion - increases the number of I/O port addresses available to the SBX expansion module by increasing the number of address lines connected to the expansion module socket. The default connects address lines A0 through A6, providing 128 I/O port addresses for each of the two expansion module chip selects. The Intel SBX MULTIMODULE standard defines only three address lines, A0, A1, A2. The remaining address lines are connected as shown below. |

| CT13 | CT12 | CT11 | CT9 | Operation |
|---|---|---|---|---|
| Out | Out | Out | Out | A0-A2 only |
| Out | Out | Out | In | A0-A2 and A3 (J4 pin 30) |
| Out | Out | In | In | A0-A3 and A4 (J4 pin 28) |
| Out | In | In | In | A0-A4 and A5 (J4 pin 10) |
| † In | In | In | In | A0-A5 and A6 (J4 pin 24) |

†Factory default configuration.

Table A-4
ZT 8832 Cuttable Traces (continued).

| TRACE # | DESCRIPTION |
|---|---|
| **CT10** | STD Bus AUX Ground - connects the STD bus AUX GND signal (P1 pins 53 and 54) to the STD bus logic GND signal (P1 pins 3 and 4). |

| | CT10 | Operation |
|---|---|---|
| † | In | AUX GND connected to logic ground |
| | Out | AUX GND not connected to logic ground |

| TRACE # | DESCRIPTION |
|---|---|
| **CT14** | STD Bus Dual Port RAM Addressing - selects between the upper or lower 8 Mbytes for CPUs that have a 16 Mbyte addressing range. See page A-10. |

| | CT14 | Address Range |
|---|---|---|
| † | In | Upper 8 Mbytes |
| | Out | Lower 8 Mbytes |

†Factory default configuration.

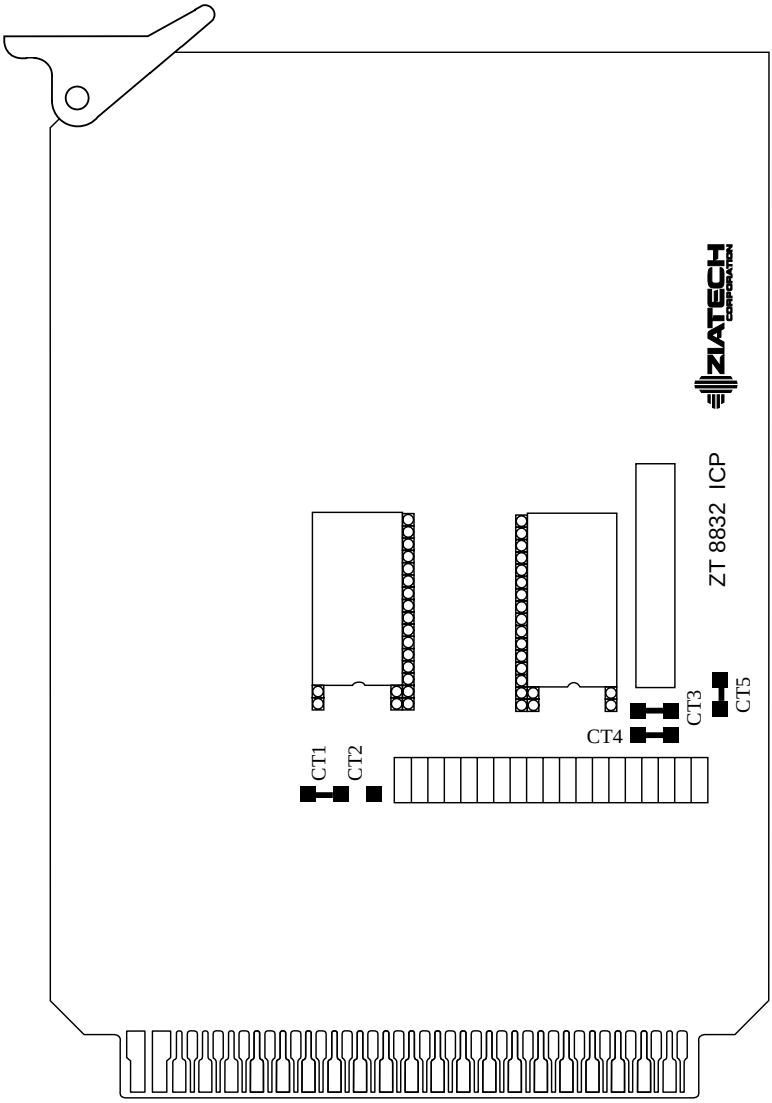



*Figure A–3. Cuttable Trace Locations, Component Side.*

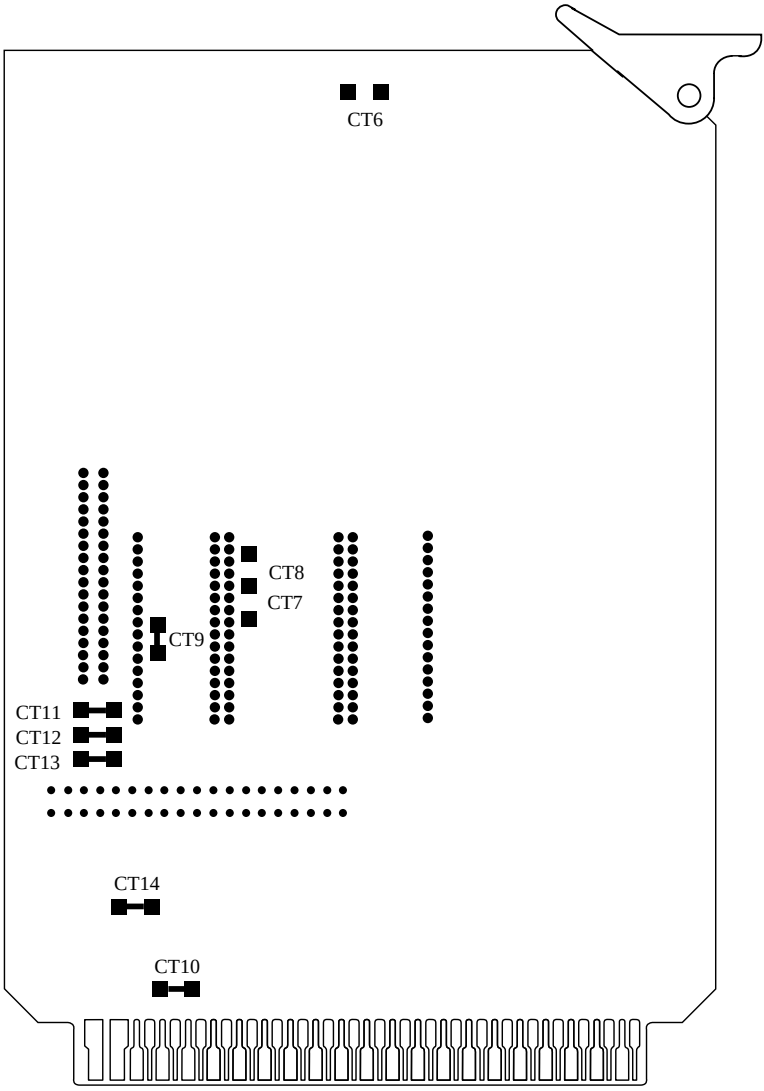



*Figure A–4. Cuttable Trace Locations, Solder Side.*

Appendix B

# SPECIFICATIONS

**Contents** **Page**

# ELECTRICAL AND ENVIRONMENTAL

## Absolute Maximum Ratings

Supply Voltage, Vcc ..... 0 to 7 V
Supply Voltage, AUX +V ..... 0 to 13 V
Supply Voltage, AUX -V ..... 0 to -13 V
Storage Temperature
    ZT 8832 ..... -40˚ to +85˚ C
    ZT 88CT32 ..... -50˚ to +125˚ C
Operating Temperature
    ZT 8832 ..... 0˚ to +65˚ C
    ZT 88CT32 ..... -40˚ to +85˚ C

## DC Operating Characteristics

Supply Voltage, Vcc ..... 4.75 to 5.25 V
Supply Voltage, AUX +V ..... 11.4 to 12.6 V
Supply Voltage, AUX -V ..... -11.4 to -12.6 V
Supply Current, Vcc (without 8087 numeric data processor)
    ZT 8832 ..... 0.9 A typ, 1.5 A max
    ZT 88CT32 ..... 0.3 A typ, 0.5 A max
Supply Current, AUX +V ..... 0.05 A typ, 0.10 A max
Supply Current, AUX -V ..... 0.05 A typ, 0.10 A max
Non-condensing relative humidity ..... <95% at 40˚ C

## Battery Backup Characteristics

Supply Voltage, Vcc ..... 4.49 V max
Retention Time
    Dual Port RAM only ..... 6 years min, 10 years typ
    Dual Port and Local RAM ..... 2 years min, 5 years typ
(Assuming Local RAMs each have 15 μA max retention current)

## STD Bus Loading Characteristics

The unit load is a convenient method for specifying the input and output drive capability of STD bus cards. In the STD bus systems, one unit load is equal to one LSTTL load as follows:

- Maximum high level input current: 20 µA
- Maximum low level input current: -400 µA

The STD bus unit load reflects input current requirements at worst case conditions over the recommended supply voltage and ambient temperature ranges. An output rate at 60 unit loads can drive 60 STD bus cards having input rated at one unit load.

Table B-1, "ZT 8832 STD Bus Loading, P Connector," includes load values for STD 32 P pins. Table B-2, "ZT 8832 STD Bus Loading, E Connector," includes load values for STD 32 E pins.

**Notes for Table B-1:**

REQ indicates required connection

[1]High order address bits multiplexed over databus

[2]PCI connected to PCO

**Notes for Table B-2:**

REQ indicates required connection

Table B-1
ZT 8832 STD Bus Loading, P Connector.

| PIN (CIRCUIT SIDE) / OUTPUT DRIVE / INPUT LOAD / MNEMONIC | | | | | | | PIN (COMPONENT SIDE) / OUTPUT DRIVE / INPUT LOAD / MNEMONIC |
|---|---|---|---|---|---|---|---|
| +5 VDC | REQ | | 2 | 1 | | REQ | +5 VDC |
| GND | REQ | | 4 | 3 | | REQ | GND |
| DCPDN* | | | 6 | 5 | | | VBAT |
| D7/A13 [1] | 4 | 55 | 8 | 7 | 55 | 4 | D3/A19 [1] |
| D6/A22 [1] | 4 | 55 | 10 | 9 | 55 | 4 | D2/A18 [1] |
| D5/A21 [1] | 4 | 55 | 12 | 11 | 55 | 4 | D1/A17 [1] |
| D4/A20 [1] | 4 | 55 | 14 | 13 | 55 | 4 | D0/A16 [1] |
| A15 | 2 | | 16 | 15 | | 2 | A7 |
| A14 | 2 | | 18 | 17 | | 2 | A6 |
| A13 | 2 | | 20 | 19 | | 2 | A5 |
| A12 | 2 | | 22 | 21 | | 2 | A4 |
| A11 | 2 | | 24 | 23 | | 2 | A3 |
| A10 | 2 | | 26 | 25 | | 2 | A2 |
| A9 | 2 | | 28 | 27 | | 2 | A1 |
| A8 | 2 | | 30 | 29 | | 2 | A0 |
| RD* | 2 | | 32 | 31 | | 2 | WR* |
| MEMRQ* | 2 | | 34 | 33 | | 2 | IORQ* |
| BHE | | | 36 | 35 | | | IOEXP |
| ALE* | 2 | | 38 | 37 | 15 | | INTRQ1* |
| STATUS 0* | 2 | | 40 | 39 | | 2 | STATUS 1* |
| BUSRQ* | | | 42 | 41 | | | BUSAK* [2] |
| INTRQ* | | 15 | 44 | 43 | | | INTAK* |
| NMIRQ* | | 15 | 46 | 45 | 35 | | WAITRQ* |
| PBRESET* | | 55 | 48 | 47 | | 2 | SYSRESET* [3] |
| INTRQ2* (CNTRL*) | | 15 | 50 | 49 | | | CLOCK* [3] |
| PCI [4] | [2] | | 52 | 51 | [2] | | PCO [4] |
| AUX GND | REQ | | 54 | 53 | | REQ | AUX GND |
| AUX-V | REQ | | 56 | 55 | | REQ | AUX+V |

Table B-2
ZT 8832 STD Bus Loading, E Connector.

| PIN (CIRCUIT SIDE) / OUTPUT DRIVE / INPUT LOAD / MNEMONIC | | | | | | | PIN (COMPONENT SIDE) / OUTPUT DRIVE / INPUT LOAD / MNEMONIC |
|---|---|---|---|---|---|---|---|
| RSVD | | | E2 | E1 | | | GND |
| XA23 | | | E4 | E3 | | | XA19 |
| XA22 | | | E6 | E5 | | | XA18 |
| XA21 | | | E8 | E7 | | | XA17 |
| XA20 | | | E10 | E9 | | | XA16 |
| RSVD | | | E12 | E11 | | | NOWS* |
| +5 VDC | REQ | | E14 | E13 | | REQ | +5 VDC |
| DREQx* | | | E16 | E15 | | | DAKx* |
| GND | REQ | | E18 | E17 | | REQ | GND |
| D31 | | | E20 | E19 | | | D27 |
| D30 | | | E22 | E21 | | | D26 |
| D29 | | | E24 | E23 | | | D25 |
| D28 | | | E26 | E25 | | | D24 |
| GND | REQ | | E28 | E27 | | | D23 |
| D15 | | | E30 | E29 | | | D22 |
| D14 | | | E32 | E31 | | | D21 |
| D13 | | | E34 | E33 | | | D20 |
| D12 | | | E36 | E35 | | REQ | GND |
| D11 | | | E38 | E37 | | | D19 |
| D10 | | | E40 | E39 | | | D18 |
| D9 | | | E42 | E41 | | | D17 |
| D8 | | | E44 | E43 | | | D16 |
| MASTER16* | | | E46 | E45 | | REQ | GND |
| AENx* | | | E48 | E47 | | | IRQx |
| BE3* | | | E50 | E49 | | | BE1* |
| BE2* | | | E52 | E51 | | | BE0* |
| GND | REQ | | E54 | E53 | | | MEM16* |
| W-R | | | E56 | E55 | | | M-IO |
| DMAIOR* | | | E58 | E57 | | | DMAIOW* |
| EX8* | | | E60 | E59 | | | IO16* |
| START* | | | E62 | E61 | | | CMD* |
| EX32* | | | E64 | E63 | | | EX16* |
| T-C | | | E66 | E65 | | | EXRDY |
| +5 VDC | REQ | | E68 | E67 | | | LOCK* |
| MREQx* | | | E70 | E69 | | | MAKx* |
| MSBURST* | | | E72 | E71 | | | SLBURST* |
| XA31* | | | E74 | E73 | | | XA27* |
| XA30* | | | E76 | E75 | | | XA26* |
| XA29* | | | E78 | E77 | | | XA25* |
| XA28* | | | E80 | E79 | | | XA24* |

# MECHANICAL

## Card Dimensions & Weight

The ZT 8832 meets the STD 32 bus specification for all mechanical parameters except for the component lead length protruding from the back of the board. The specification requires a maximum lead length of 0.093 inches, but the battery socket pins extend a maximum of 0.150 inches. Be sure the battery socket pins do not touch the adjacent board when installing the ZT 8832 into the STD bus card cage. In a card cage with 0.625 inch spacing, the ZT 8832 requires one card slot without an SBX expansion module installed and two card slots with an SBX expansion module installed.

Board Length ..... 16.51 cm ±0.063 cm
(6.500 in ±0.025 in)

Board Width ..... 11.43 cm ±0.038 cm
(4.500 in ±0.015 in)

Board Thickness ..... 0.158 cm ±0.013 cm
(0.062 in ±0.005 in)

Board Weight ..... 155.9 g (5.5 oz)

Board Height From Top Surface

- Without following items ..... 9.652 mm (0.380 in) max
- Including Battery ..... 10.922 mm (0.430 in) max
- Including SBX Expansion Module 29.21 mm (1.150 in) max

Board Height From Bottom Surface ..... 3.81 mm (0.150 in) max

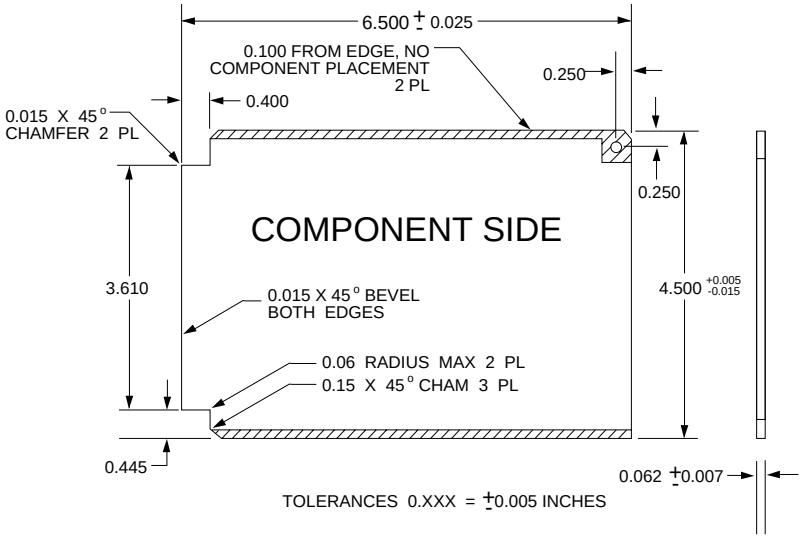



*Figure B–1. Board Dimensions.*

## Connectors

The ZT 8832 has two card-edge connectors (P and E) and five headers to interface to the STD bus and application-specific devices (see Figure B-3). The following pages provide connector descriptions and pin assignments.

**P:** The P connector is the interface between the ZT 8832 and the STD-80 bus. This connector is a 56-pin (dual 28-pin) card-edge connector with fingers on 0.125 inch centers. The mating connector is a Viking 3VH28/1CND5 or equivalent for a three-level wire wrap, or a Viking 3VH28/1CNK5 or equivalent for the solder tail. Pin assignments are shown on page B-4.

**E:** The E connector extends the P connector to interface the ZT 8832 to the STD 32 bus. This connector combines with the P connector to make a 114-pin (dual 57-pin) card-edge connector with fingers on 0.062 inch centers. The mating connector is a Viking S3VT68/5DE12 or equivalent for the card extender, or a Viking S3VT68/5DP12 or equivalent for the solder tail. Pin assignments are shown on page B-5.

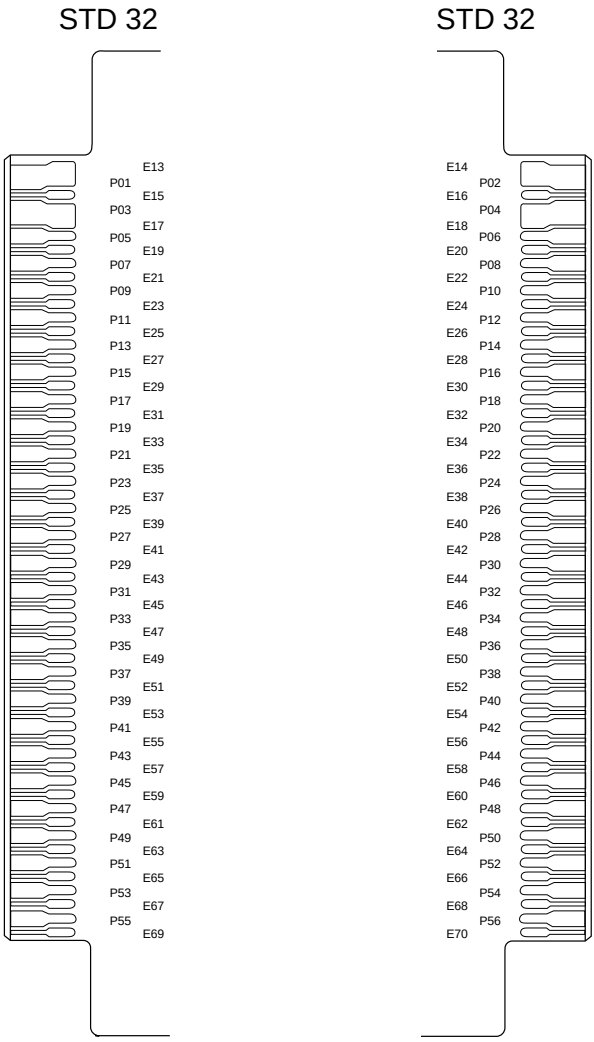



*Figure B–2. STD 32 P/E Connector Pinout.*

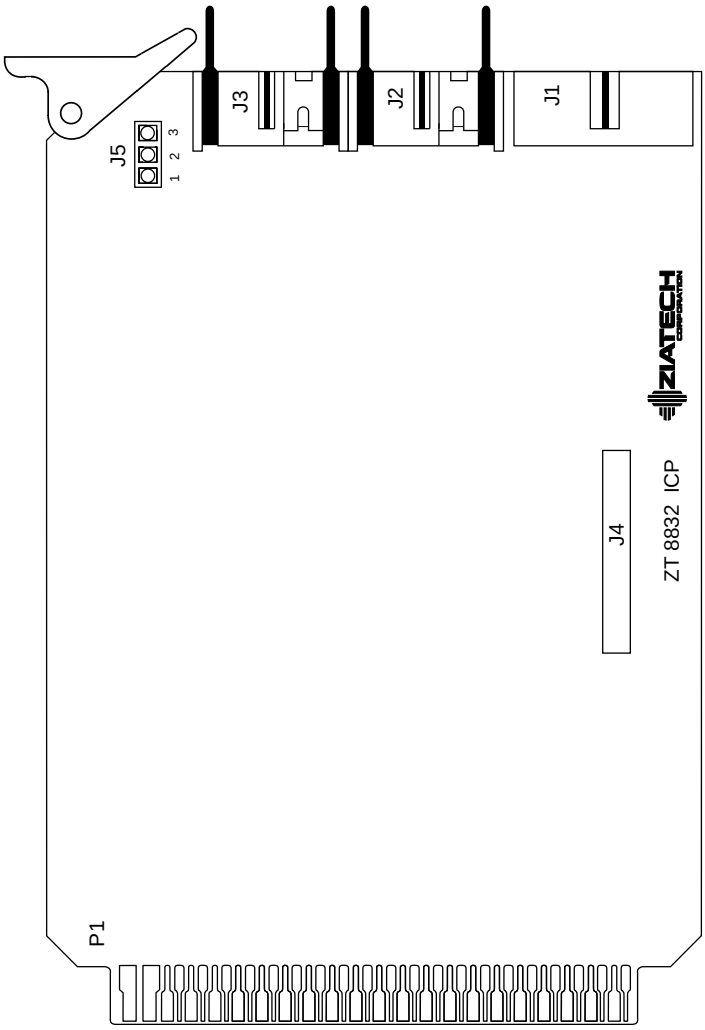



*Figure B–3. Connector Locations.*

**J1:** J1 is a nonlatching 26-pin (dual 13-pin) male transition connector with 0.1 inch lead spacing. J1 provides 24 digital I/O lines, fused +5 V ±10%, and ground. Table B-3 on page B-11 lists the pin assignments. These pin assignments enable the ZT 90068 cable to connect J1 directly to an I/O module mounting rack with 8, 16, or 24 positions. For applications not using this cable, the mating connector is a T&B Ansley #622-2630 or equivalent.

**J2:** J2 is a latching 14-pin (dual 7-pin) male transition connector with 0.1 inch lead spacing. The 82050 serial port is available as RS-232-C or RS-422/485 through connector J2. The pin assignments for RS-232-C are given in Table B-4 on page B-12, and the pin assignments for RS-422/485 are given in Table B-5 on page B-13. The mating connector is a T&B Ansley #622-1430 or equivalent.

**J3:** J3 is a latching 10-pin (dual 5-pin) male transition connector with 0.1 inch lead spacing. The V40 counter/timer and interrupt inputs are available through connector J3. Table B-6 on page B-14 lists the pin assignments. The mating connector is a T&B Ansley #622-1030 or equivalent.

**J4:** J4 is a 36-pin (dual 18-pin) female SBX connector with 0.1 inch lead spacing. This connector includes the power, address, data, and control signals needed to add custom I/O to the ZT 8832. Table B-7 on page B-15 lists the pin assignments. The mating connector is a Viking 000292-00003 or equivalent.

**J5:** J5 is a 3-pin male low profile header with 0.1 inch lead spacing. The V40 serial port is available through this connector. Table B-8 on page B-16 lists the pin assignments. The mating connector is a Molex 39-01-0033 or equivalent. The mating connector also requires three Molex 39-00-0031 terminals or equivalent.

Table B-3
J1 Parallel Port Pinout.

| Pin | Signal | Port Address [hex] |
|---|---|---|
| 1 | MOD24 | 220h bit 7 |
| 2 | MOD11 | 210h bit 2 |
| 3 | MOD23 | 220h bit 6 |
| 4 | MOD10 | 210h bit 1 |
| 5 | MOD22 | 220h bit 5 |
| 6 | MOD09 | 210h bit 0 |
| 7 | MOD21 | 220h bit 4 |
| 8 | MOD08 | 200h bit 7 |
| 9 | MOD20 | 220h bit 3 |
| 10 | MOD07 | 200h bit 6 |
| 11 | MOD19 | 220h bit 2 |
| 12 | MOD06 | 200h bit 5 |
| 13 | MOD18 | 220h bit 1 |
| 14 | MOD05 | 200h bit 4 |
| 15 | MOD17 | 220h bit 0 |
| 16 | MOD04 | 200h bit 3 |
| 17 | MOD16 | 210h bit 7 |
| 18 | MOD03 | 200h bit 2 |
| 19 | MOD15 | 210h bit 6 |
| 20 | MOD02 | 200h bit 1 |
| 21 | MOD14 | 210h bit 5 |
| 22 | MOD01 | 200h bit 0 |
| 23 | MOD13 | 210h bit 4 |
| 24 | Fused +5 V | —— |
| 25 | MOD12 | 210h bit 3 |
| 26 | Ground | —— |

**Note:** The pin assignments enable the ZT 90068 cable to connect J1 directly to the 50-pin connector of an I/O module mounting rack.

Table B-4
J2 Serial Port (RS-232-C) Pinout.

| Pin | | | |
|---|---|---|---|
| **DTE** | **DCE** | **Signal** | **Description** |
| 3 | 5 | TxD | Transmit Data |
| 5 | 3 | RxD | Receive Data |
| 7 | 9 | RTS | Request To Send |
| 9 | 7 | CTS | Clear To Send |
| 10 | 10 | DCD | Data Carrier Detect |
| 11 | 14 | DSR | Data Set Ready |
| 12 | 12 | RI | Ring Indicator |
| 13 | 13 | GND | Ground |
| 14 | 11 | DTR | Data Terminal Ready |
| 1,2,4 | 1,2,4 | --- | High Impedance |
| 6,8 | 6,8 | --- | No Connection |

**Note:** The Data Communication Equipment (DCE) and Data Terminal Equipment (DTE) options are jumper selectable for RS-232-C. The pin assignments enable the ZT 90014 and ZT 90027 cables to connect J2 directly to a standard 25-pin D shell connector.

Table B-5
J2 Serial Port (RS-422/485) Pinout.

| Pin | Signal | Description |
|---|---|---|
| 1 | SDA | Send Data (negative) |
| 2 | SDB | Send Data (positive) |
| 3 | RSA | Request To Send (negative) |
| 4 | RSB | Request To Send (positive) |
| 11 | CSB | Clear To Send (positive) |
| 12 | CSA | Clear To Send (negative) |
| 13 | RDB | Receive Data (positive) |
| 14 | RDA | Receive Data (negative) |
| 7,8 | GND | Ground |
| 5,6,9,10 | --- | No Connection |

**Note:** The J2 pin assignments permit ZT 8832s to be connected together using a straight cable and rotating one of the connectors 180˚. This same pin assignment is used on all Ziatech products that support RS-422/485.

Table B-6
J3 Counter/Timer and Interrupt Pinout.

| Pin | Signal | Description |
|---|---|---|
| 2 | TCLK | V40 Counter/Timer Clock Input |
| 4 | TCTL | V40 Counter/Timer 2 Control Input |
| 6 | TOUT | V40 Counter/Timer 2 Output |
| 8 | IRQ6 | V40 Interrupt Controller Input 6 |
| 10 | IRQ7 | V40 Interrupt Controller Input 7 |
| 1,3,5,7,9 | GND | Ground |

Table B-7
J4 SBX Expansion Module Pinout.

| Pin | Signal[1] | Description |
|---|---|---|
| 1 | +12V | +12 Volts |
| 2 | -12V | -12 Volts |
| 3 | GND | Signal Ground |
| 4 | +5V | +5 Volts |
| 5 | RESET | Reset |
| 6 | MCLK | 10 MHz Clock[2] |
| 7 | MA2 | Address 2 |
| 8 | MPST* | Module present[3] |
| 9 | MA1 | Address 1 |
| 10 | RSVD | Reserved - Address 5[4] |
| 11 | MA0 | Address 0 |
| 12 | MINTR1 | Interrupt 1[5] |
| 13 | IOWRT* | I/O Write |
| 14 | MINTR0 | Interrupt 0[5] |
| 15 | IORD* | I/O Read |
| 16 | MWAIT* | Wait Request |
| 17 | GND | Ground |
| 18 | +5 V | +5 Volts |
| 19 | MD7 | Data Bit 7 |
| 20 | MCS1* | Chip Select 1[6] |
| 21 | MD6 | Data Bit 6 |
| 22 | MCS0* | Chip Select 0[6] |
| 23 | MD5 | Data Bit 5 |
| 24 | RSVD | Reserved - Address 6[4] |
| 25 | MD4 | Data Bit 4 |
| 26 | TDMA | Terminate DMA[3] |
| 27 | MD3 | Data Bit 3 |
| 28 | OPT1 | Option 1 - Address 4[4] |
| 29 | MD2 | Data Bit 2 |
| 30 | OPT0 | Option 0- Address 3[4] |
| 31 | MD1 | Data Bit 1 |
| 32 | MDACK* | DMA Acknowledge[7] |
| 33 | MD0 | Data Bit 0 |
| 34 | MDRQT | DMA Request[7] |
| 35 | GND | Ground |
| 36 | +5V | +5 Volts |

## Notes:

[1] Signals ending with an asterisk are active low and signals without an asterisk are active high.

[2] The V40 clock is optionally connected to MCLK with CT1 and CT2 (refer to the cuttable trace description on page A-17). This option is useful for designing SBX expansion modules synchronous to the ZT 8832 CPU.

[3] These signals are not supported. The MPST* is a no-connect and TDMA is grounded.

[4] These signals provide additional address lines to the three supported in the expansion module specification. This feature is supported with CT5 through CT8 as defined in the cuttable trace table starting on page A-16.

[5] Interrupt 1 is routed to IRQ3 of the V40 interrupt controller and Interrupt 0 is routed to IRQ2 of the V40 interrupt controller.

[6] The I/O address range for Chip Select 0 is 2F8h through 2FFh and the I/O address range for Chip Select 1 is 300h through 307h. If more than the three standard I/O address lines are used, Chip Select 0 grows downward and Chip Select 1 grows upward.

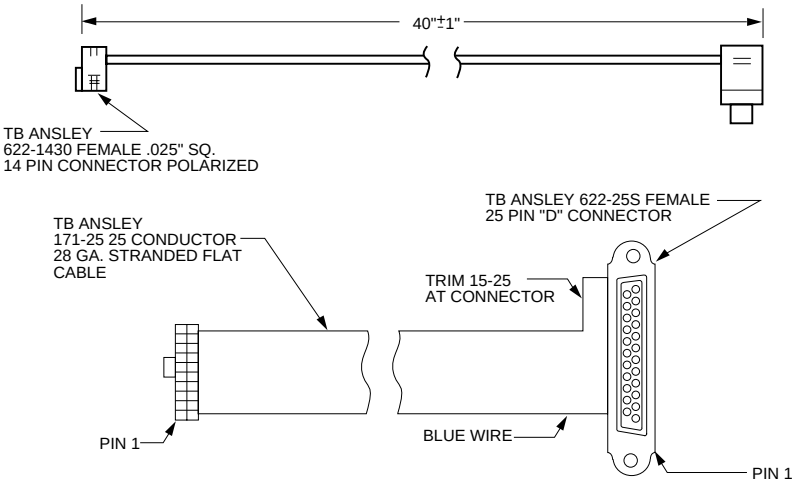
[7] DMA is supported through channel 0 of the V40 DMA controller.

Table B-8
J5 Serial Port Pinout.

| Pin | Signal | Description |
|---|---|---|
| 1 | RxD | Receive Data |
| 2 | TxD | Transmit Data |
| 3 | GND | Ground |

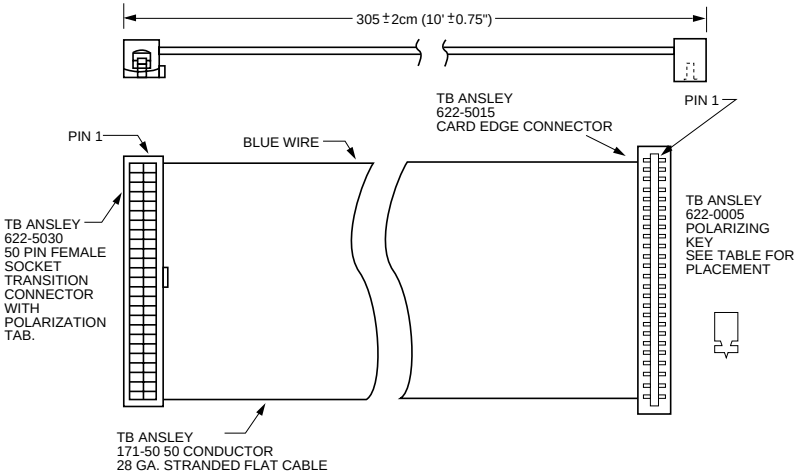
**Note:** The pin assignments enable the ZT 90069 cable to connect J5 directly to a male 25-pin D shell connector.

## Cables




| P1 | J1 | P1 | J1 |
|---|---|---|---|
| 1 | 1 | 8 | 17 |
| 2 | 14 | 9 | 5 |
| 3 | 2 | 10 | 18 |
| 4 | 15 | 11 | 6 |
| 5 | 3 | 12 | 19 |
| 6 | 16 | 13 | 7 |
| 7 | 4 | 14 | 20 |

*Figure B–4. ZT 90014 Rev _ Serial I/O Cable.*




| NO. OF MODULES | KEY LOCATION, BETWEEN PINS |
|---|---|
| 8 | 29 AND 31 OR 17 AND 19 |
| 16 | 11 AND 13 |
| 24 | 23 AND 25 |

*Figure B–5. ZT 90021 Rev _ Parallel I/O Cable.*

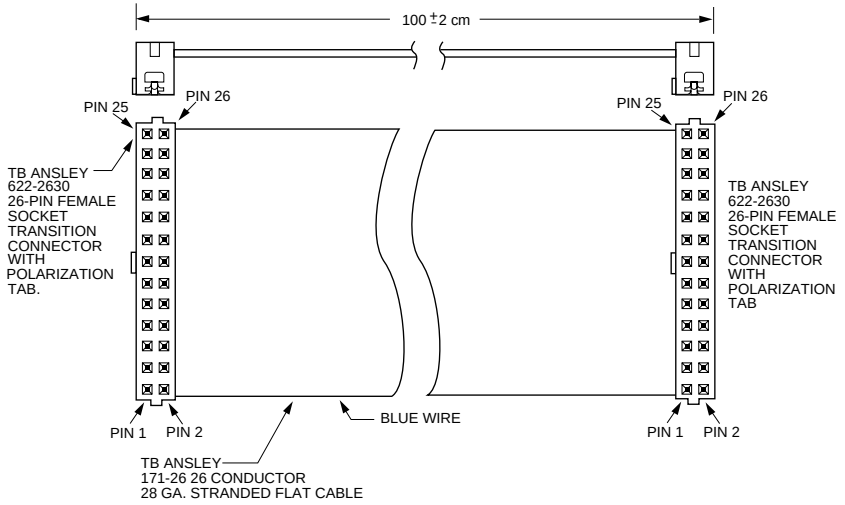
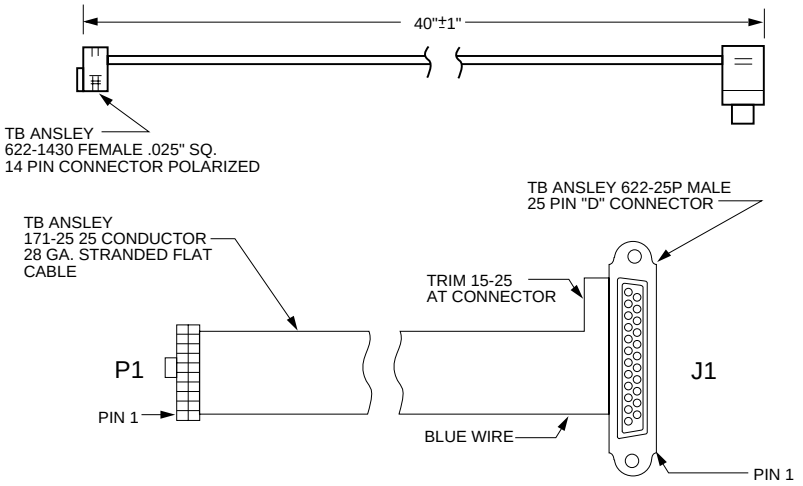
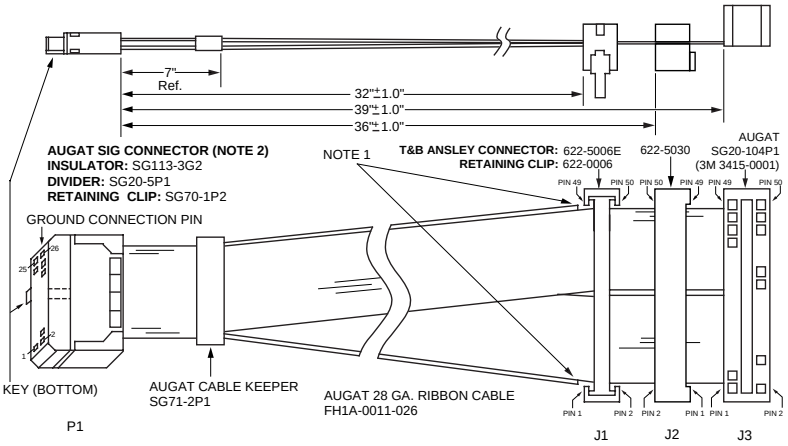



*Figure B–6. ZT 90090 Rev _ Parallel I/O Cable.*




| P1 | J1 | P1 | J1 |
|---|---|---|---|
| 1 | 1 | 8 | 17 |
| 2 | 14 | 9 | 5 |
| 3 | 2 | 10 | 18 |
| 4 | 15 | 11 | 6 |
| 5 | 3 | 12 | 19 |
| 6 | 16 | 13 | 7 |
| 7 | 4 | 14 | 20 |

*Figure B–7. ZT 90027 Rev _ Serial I/O Cable.*

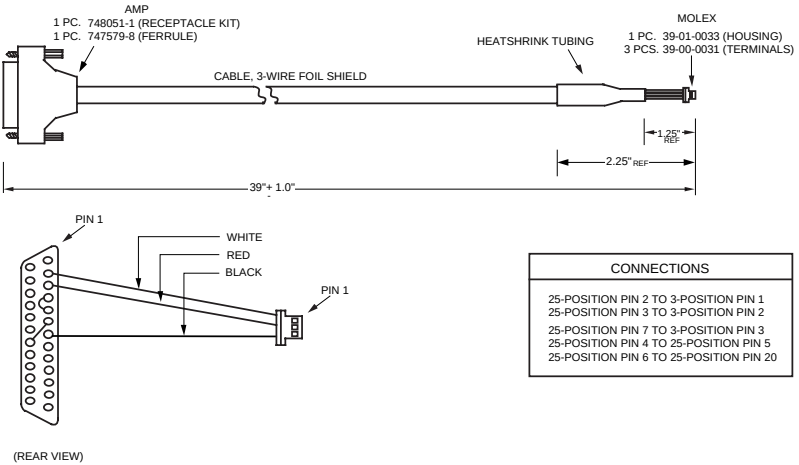



NOTE 1: OUTSIDE WIRES REMOVED AT THIS POINT

NOTE 2: CONNECT THE GROUND BUS BARS TOGETHER

NOTE 3: FIRST ARTICLE REQUIRED FOR NEW VENDOR QUALIFICATION OR PART SUB-STITUTION

*Figure B–8. ZT 90068 Rev _ Parallel I/O Cable.*




(REAR VIEW)

*Figure B–9. ZT 90069 Rev _ Serial I/O Cable.*

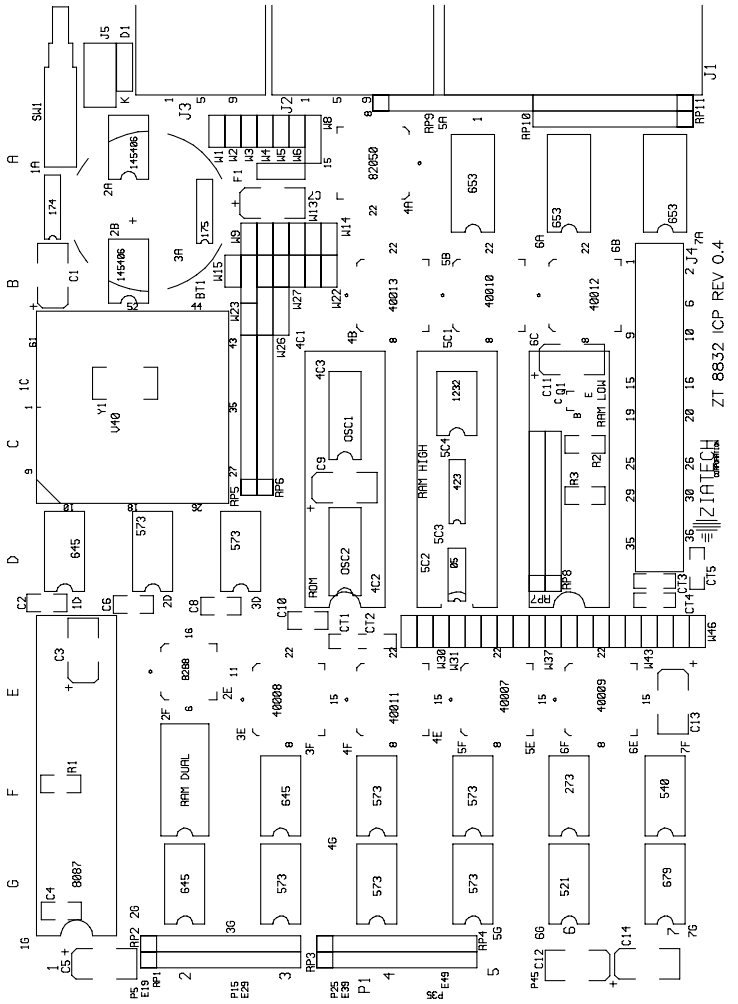
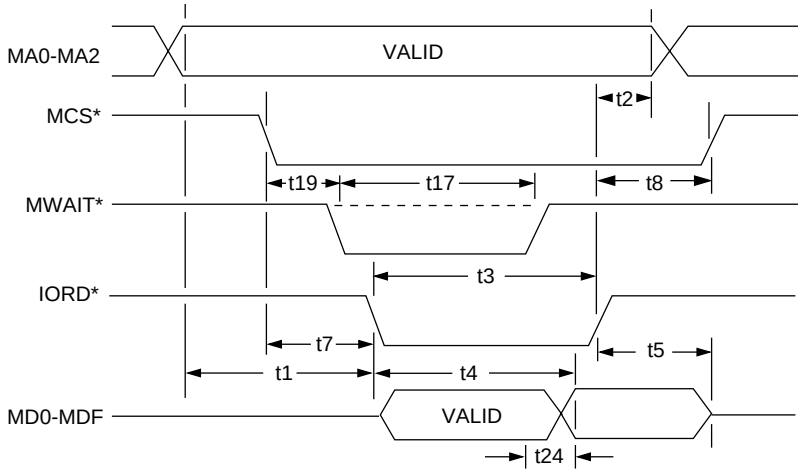



*Figure B–10. Standard Assembly Diagram.*

## TIMING

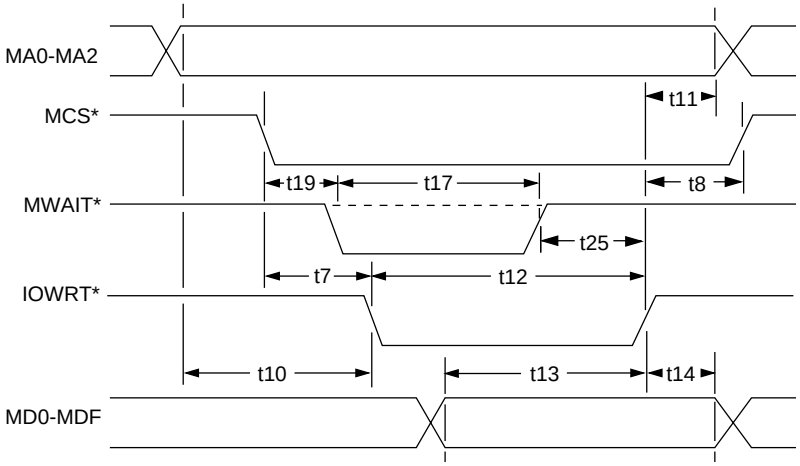
The ZT 8832 meets the timing requirements outlined in the STD 32 bus specification. The SBX expansion module timings are given on the following pages. These pages assume the WCY1 and WCY2 V40 configuration registers are programmed to insert two wait states for all I/O and DMA transfers.




| Symbol | Parameter | Min | Max |
|---|---|---|---|
| t1 | Address setup to read low | 50 | |
| t2 | Address hold from read high | 30 | |
| t3 | Read pulse width | 300 | |
| t4 | Data delay from read low | 0 | 250 |
| t5 | Data float after read high | 0 | 80 |
| t7 | Chip select setup to read low | 25 | |
| t8 | Chip select hold from read high | 30 | |
| t17 | Wait request pulse width | 0 | 4 ms |
| t19 | Wait request delay from chip select | 0 | 75 |
| t24 | Data hold from wait request | | 0 |

All times given in nanoseconds except where otherwise indicated

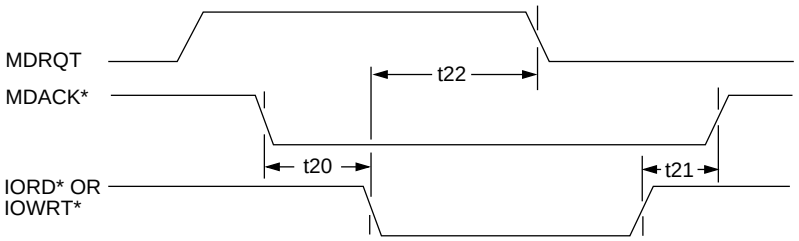
*Figure B–11. SBX Expansion Module Read Timing.*




| Symbol | Parameter | Min | Max |
|---|---|---|---|
| t7 | Chip select setup to write low | 25 | |
| t8 | Chip select hold from write high | 30 | |
| t10 | Address setup to write low | 50 | |
| t11 | Address hold from write high | 30 | |
| t12 | Write pulse width | 300 | |
| t13 | Data setup to write high | 250 | |
| t14 | Data hold from write high | 30 | |
| t17 | Wait request pulse width | 0 | 4 ms |
| t19 | Wait request delay from chip select | 0 | 75 |
| t25 | Write delay from wait request | | 0 |

All times given in nanoseconds except where otherwise indicated

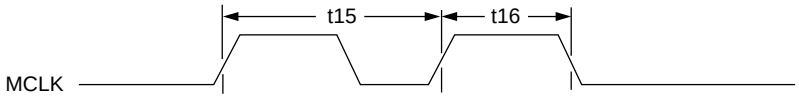
*Figure B–12. SBX Expansion Module Write Timing.*




| Symbol | Parameter | Min | Max |
|---|---|---|---|
| t20 | DMA acknowledge setup to read or write low | 25 | |
| t21 | DMA acknowledge hold from read or write high | 30 | |
| t22 | DMA request hold from read or write low | | 150 |

All times given in nanoseconds

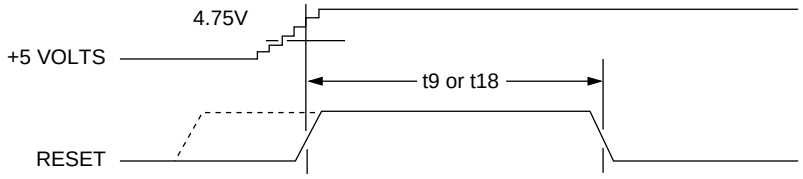
*Figure B–13. SBX Expansion Module DMA Timing.*




| Symbol | Parameter | Min | Max |
|---|---|---|---|
| t15 | Standard clock period | 100 | 110 |
| | V40 synchronous clock period | 125 | |
| t16 | Standard clock period high width | 35 | 75 |
| | V40 synchronous clock high width | 55 | |

All times given in nanoseconds

*Figure B–14. SBX Expansion Module Clock Timing.*




| Symbol | Parameter | Min | Max |
|---|---|---|---|
| t9 | Power up reset pulse width | 50 | |
| t18 | Reset pulse width | 50 µs | |

All times given in nanoseconds except where otherwise indicated

*Figure B–15. SBX Expansion Module Reset Timing.*

Appendix C

# CUSTOMER SUPPORT

**Contents** **Page**



## OVERVIEW

This appendix offers a product revision history, technical assistance for the ZT 8832, and the necessary information should you need to return your ZT 8832 for repair.

# REVISION HISTORY

## ZT 8832

### Revision 0

The ZT 8832 was originally released on 7/3/89 as Revision 0.

### Revision 0.1

There were no functional changes at this revision.

### Revision 0.2

A change was made to the ZT 8832 for Revision 0.2 that affects when the STD bus control port and local control port are reset. The Revision 0.1 board reset these control ports in response to an STD bus SYSRESET* only. This implementation caused problems when the ZT 8832 was used by itself because SYSRESET* was not driven. The solution implemented in Revision 0.2 was to expand the control port reset to include the ZT 8832 power monitor, pushbutton, and Stage 2 watchdog timeout.

### Revision 0.3

No functional changes.

### Revision 0.4

The STD-80 fingers were replaced with STD 32 fingers.

A 10 kΩ pulldown resistor, R19, was added to the POLL* input on the V40 processor. The POLL* input is now active low with the 8087 removed instead of floating.

Support was added for 24-bit addressing in either the upper or the lower 8 Mbyte region. See page A-10 for more information.

**Revision 0.5**

No functional changes.

**ZT 88CT32**

**Revision 0**

The ZT 88CT32 was originally released as Revision 0.

**Revision 0.1**

Pullup resistor packs RP7 and RP8 were changed from 100 kΩ to 10 kΩ. This eliminates problems experienced with SBX expansion modules that drive signals such as wait request with an undetermined open collector device.

**Revision 0.2**

No functional changes.

**Revision 0.3**

See the description of changes for the ZT 8832 Revision 0.4.

**Revision 0.4**

No functional changes.

## TECHNICAL ASSISTANCE

You can reach Ziatech's Customer Support Service at the following number:

Corporate Headquarters: (805) 541-0488
(805) 541-5088 (FAX)

You can also use your modem to leave a message on the 24-hour Ziatech Bulletin Board Service (BBS) by calling (805) 541-8218. The BBS will also provide you with current Ziatech product revision and upgrade information.

## RELIABILITY

Ziatech has taken extra care in the design of the ZT 8832 to ensure reliability. The four major ways in which reliability is achieved are:

1. The product was designed in top-down fashion, using the latest in hardware and software design techniques, so that unwanted side effects and unclean interactions between parts of the system are eliminated.
2. The advanced low-power Schottky TTL devices and the high speed CMOS TTL devices used in the ZT 8832 are high-reliability parts available from several manufacturers.
3. Ziatech tests each board under power to ensure that the infant mortality phase is passed before the product is shipped.
4. Each ZT 8832 has an identification number. Ziatech maintains a lifetime data base on each board and on the parts used. Any negative trends in reliability are spotted and Ziatech's suppliers are informed and/or changed.

## RETURNING FOR SERVICE

Before returning any of Ziatech's products, you must obtain a Returned Material Authorization (RMA) number by calling (805) 541-0488. We will need the following information to expedite the shipment of a replacement to you:

1. Your company name and address for invoice
2. Shipping address and phone number
3. Product ID number
4. If possible, the name of a technically qualified individual at your company familiar with the mode of failure on the board

If the unit is out of warranty, service is available at a predesignated service charge. Contact Ziatech for pricing and please supply a purchase order number for invoicing the repair.

Pack the ZT 8832 in *anti-static* material and ship in a sturdy cardboard box with enough packing material to adequately cushion the board. *Any product returned to Ziatech improperly packed will immediately void the warranty for that particular product!* Mark the RMA number clearly on the outside of the box before returning.

# ZIATECH 5+5 WARRANTY

## FIVE-YEAR LIMITED WARRANTY

Products manufactured by Ziatech Corporation are covered from the date of purchase by a five-year warranty against defects in materials, workmanship, and published specifications applicable to the date of manufacture. During the warranty period, Ziatech will repair or replace, solely at its option, defective units provided they are returned at customer expense to an authorized Ziatech repair facility. Products which have been subjected to misuse, abuse, neglect, alteration, or unauthorized repair, determined at the sole discretion of Ziatech, whether by accident or otherwise, are excluded from warranty. The warranty on fans and disk drives is limited to two years and the warranty on flat panel displays is limited to nine months from date of purchase. Other products and accessories not manufactured by Ziatech are limited to the warranty provided by the original manufacturer. Consumable items (fuses, batteries, etc.) and software are not covered by this warranty. Within 90 days of shipping date, Ziatech will replace software disk media should it prove defective.

Ziatech may offer, where applicable and available, replacement products; otherwise, repairs requiring components, assemblies, and other purchased materials may be limited by market availability.

Ziatech assumes no liability resulting from changes to government regulations affecting use of materials, equipment, safety, and methods of repair. Ziatech may, at its discretion, offer replacement products.

THE ABOVE WARRANTY IS IN LIEU OF ANY OTHER WARRANTY, WHETHER EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY FOR FITNESS OF PURPOSE, MERCHANTABILITY, OR FREEDOM FROM INFRINGEMENT OR THE LIKE, AND ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATIONS, OR SAMPLE.

Ziatech neither assumes nor authorizes any person to assume for it any other liability. The liability of Ziatech under this warranty agreement is limited to a refund of the purchase price. In no event shall Ziatech be liable for loss of profits, use, incidental, consequential, or other damage, under this agreement.

## SPECIAL EXTENDED WARRANTY OPTION

In addition to the standard five-year warranty, Ziatech offers, for a nominal fee, an extended period of warranty up to five extra years. This extended warranty period provides similar coverage and conditions as stated above in the five-year limited warranty agreement.

## LIFE SUPPORT POLICY

Ziatech products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Ziatech Corporation. As used herein:

1. Life support devices or systems are devices or systems which support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be expected to cause the failure of the life support device or system, affect its safety, or limit its effectiveness.

Appendix D

# GLOSSARY

**backplane** The edge of the board that inserts into the STD bus connector. This term is generally used to define the location of signals that are routed across the STD bus.

**BAU** Bus Arbitration Unit. Section of the CPU that controls which internal or external bus master has access to the buses at any given time.

**BCD** Binary Coded Decimal. Representation of the cardinal numbers 0 through 9 by ten binary codes. Each binary code is 4 binary digits long.

**BIU** Bus Interface Unit. Section of the CPU that controls the external address, data, and control buses.

**CGU** Clock Generator Unit. Section of the CPU that provides a clock reference with a 50% duty cycle to the CPU.

**CMOS** Complementary Metal Oxide Semiconductor. Provides low power density and high noise immunity.

**CNTRL*** Control. This STD bus signal (pin 50) was used in previous designs for special clock timing on peripheral boards. It may also be used as an interrupt request, INTRQ2*, on the backplane.

**DCE** Data Communication Equipment. One of two possible orientations (DCE or DTE) for drivers and receivers in the RS-232-C serial communications protocol.

**DCU** DMA Control Unit. Section of the CPU that controls high speed data transfer between I/O and memory devices.

**DMA** Direct Memory Access. Used for faster data transfer rate when processing of I/O data on a byte-by-byte basis is not required.

**DTE** Data Terminal Equipment. One of two possible orientations (DCE or DTE) for drivers and receivers in the RS-232-C serial communications protocol.

**EOI** End Of Interrupt. A byte sent to the interrupt controller that signifies the interrupt level has been serviced.

**EPROM** Erasable Programmable Read-Only Memory. Memory is programmed and erased with ultraviolet light.

**frontplane** The edge of the board on which the extractor is located, opposite to the backplane. This term is generally used to define the location of user interface signals.

**ICU** Interrupt Control Unit, on the V40 CPU.

**INTRQ*** **INTRQ1*** **INTRQ2*** Interrupt Requests. These STD-80 signals are processor card input signals that conditionally interrupt the program when enabled by a specific program instruction. INTRQ2* was formerly called CNTRL*; see *CNTRL**.

**LSTTL** Low Power Schottky Transistor Transistor Logic. See *Schottky TTL*.

**mark** A negative voltage on a serial link.

**NMI** Non-Maskable Interrupt. Interrupt request input that cannot be disabled through software control. Generally used to signal events such as power failure and parity error.

**NMIRQ*** Non-Maskable Interrupt Request. STD bus signal (pin 46) that generates an NMI. See *NMI*.

**PIC** Programmable Interrupt Controller (on the ZT 8802, equivalent to the Intel 8259A). This device prioritizes and handles interrupt requests from the STD bus.

**pop** A stack operation that retrieves one byte from the top of the processor stack.

**prefetch** Instructions are fetched and stored into a queue on the microprocessor prior to execution in order to optimize performance.

**push** A stack operation that stores one byte onto the top of the processor stack.

**RS-232-C** An acronym for Required Standard 232 of the Electronics Industry Association. Interface standard between Data Terminal Equipment and Data Communication Equipment, employing serial binary data exchange.

**RS-485** Standard for electrical characteristics of hardware drivers and receivers for use in balanced multipoint systems.

**Schottky TTL** Provides high speed, low noise, and compatibility with standard TTL. See *TTL*.

**SCU** Serial Control Unit. Single asynchronous serial channel used for serial communication between the CPU and a serial device external to the CPU.

**space** A positive voltage on a serial link.

**TCU** Timer/Counter Control Unit on the V40 CPU.

**TTL** Transistor-Transistor Logic.

**VCR** V40 Configuration Registers. Twelve programmable registers used to configure the CPU to meet the needs of varying applications.

**WCU** Wait Control Unit. Section of the CPU that can define a different number of wait states for each of the three areas of the memory space.

# INDEX

## *- A -*

## *- B -*



## *- C -*

## *- D -*

## *- E -*



## *- F -*

## *- G -*



## *- I -*

## *- J -*

## *- L -*



## *- M -*

## *- N -*



## *- O -*



## *- P -*

## *- R -*

## - *S* -

## - *T* -



## - *U* -



## - *V* -

## *- W -*

## - Z -