**Advanced**
**Micro**
**Computers**
A subsidiary of
Advanced Micro Devices

# Am96/4018
# AmZ8000 Evaluation Board

## User's Manual

059910622-001

$10.00

| REVISION RECORD | |
|---|---|
| **REVISION** | **DESCRIPTION** |
| 01 | Preliminary Issue |
| (7/13/81) | |
| A | Manual Released |
| (8/21/81) | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

Publication No.
059910622-001

REVISION LETTERS I, O, Q AND X ARE NOT USED



Address comments concerning
this manual to:

ADVANCED MICRO COMPUTERS

Publications Department
3340 Scott Boulevard
Santa Clara, CA 95051

# PREFACE

This manual provides general information, an installation and interface guide, and programming information,and principles of operation for the Advanced Micro Computers Am96/4018 AmZ8000† Evaluation Board. Additional information concerning components of the Am96/4018 is available in the the following documents:

- AmZ8001/2 Processor Instruction Set
- AmZ8000 User's Manual
- AmZ8000 Family Data Book
- AmZ8001/AmZ8002 Processor Interface
- AmZ8000 Microprocessor Specification
- Zilog, Z8001 CPU/8002 CPU Product Spec.
- AmZ8001 Data Sheet
- AmZ8002 Data Sheet
- AMD Shottky and Low Power Shottky Data Book
- Am8251-Am9551 Data Sheet
- Am8255/Am8255-5 Data Sheet
- Am8253 Data Sheet
- 8259A Data Sheet (Intel)

†Z8000 is a trademark of Zilog, Inc.

# TABLE OF CONTENTS

FIGURES



TABLES

# CHAPTER 1
# INTRODUCTION

## DESCRIPTION

The Am96/4018 Evaluation Board is a complete single-board micro computer built around the new 16-bit AmZ8000 microprocessor. Used with a standard CRT or the low-cost keyboard/display console available as an option, it provides an excellent means of testing the advanced capabilities of this remarkably versatile CPU. Used with Advanced Micro Computer's 16-bit Development Systems, the entire AmZ8000 instruction set can be macroassembled to create very powerful programs for execution on the Evaluation Board.

The fully assembled and tested board has the following features:

Standard Features:

- AmZ8002 (non-segmented) CPU.
- 8K bytes of dynamic RAM, expandable off-board.
- Up to 16K bytes of ROM or E-PROM, expandable off-board.
- 8K byte ROM Monitor, with breakpoint, single-step and up/down-load commands.
- Buffered CPU bus available at edge of board.
- Two programmable serial I/O ports (RS232C and 20mA).
- A programmable counter/timer channel.
- 24 programmable parallel I/O lines (three 8-bit ports).
- Fits in Multibus† or iSBC† card cage.

Optional Features:

- AmZ8001 (segmented) CPU with 8K byte ROM monitor (Am96/4018/600).
- Keyboard/Display console (Am96/4018/100).
- CRT terminal.
- Universal prototyping board.
- 3-slot CPU-bus backplane (Am96/4018/400).
- Six-board card cage.
- 64-kilobyte memory expansion (up to 4 M bytes for AmZ8001).
- Complete 16-bit Development Systems, with diskette storage, interface utilities and macro-assembler for the AmZ8000 series.
- Line-by-line assembler in E-PROM (Am96/4018/200).
- Parallel I/O cable and diskette for use with Development System (Am96/4018/500).

Programs can be entered and executed in either the stand-alone or Development System configuration. Specialized circuits or external expansion boards can also be attached in either mode. The entire CPU bus is buffered and brought to the edge of the board for interconnection, as are extra I/O ports for serial and parallel transfers on the data bus.

†Trademarks of Intel Corporation

Figure 1-1 illustrates the board layout. The lower two edge connectors are physically compatible with the Multibus standard, although the P1 edge connector uses only power and ground in the standard Multibus card cage.

All four top edge connectors on the board are used for I/O. The P3 connector has 24 programmable lines, each with an associated ground, divided into three 8-bit ports for data and handshaking. The P4 connector is also parallel but designed specifically for the optional keyboard/display console that can be attached to the Evaluation Board. The P5 and P6 connectors both carry programmable lines for asynchronous or synchronous serial data. The P5 connector also carries control I/O for an on-board programmable counter/timer channel.

The CPUs are in the bottom right corner of the board. Near the top center is the Am8253 counter/timer and the two Am9551 serial I/O devices. The Am8255A parallel I/O device is in the upper left near P3 and the six driver/terminator sockets. The 8K bytes of user memory (RAM) and sockets for the 16K bytes of ROM are near the left edge of the board (the upper two sockets are reserved for 2732-type E-PROMS).

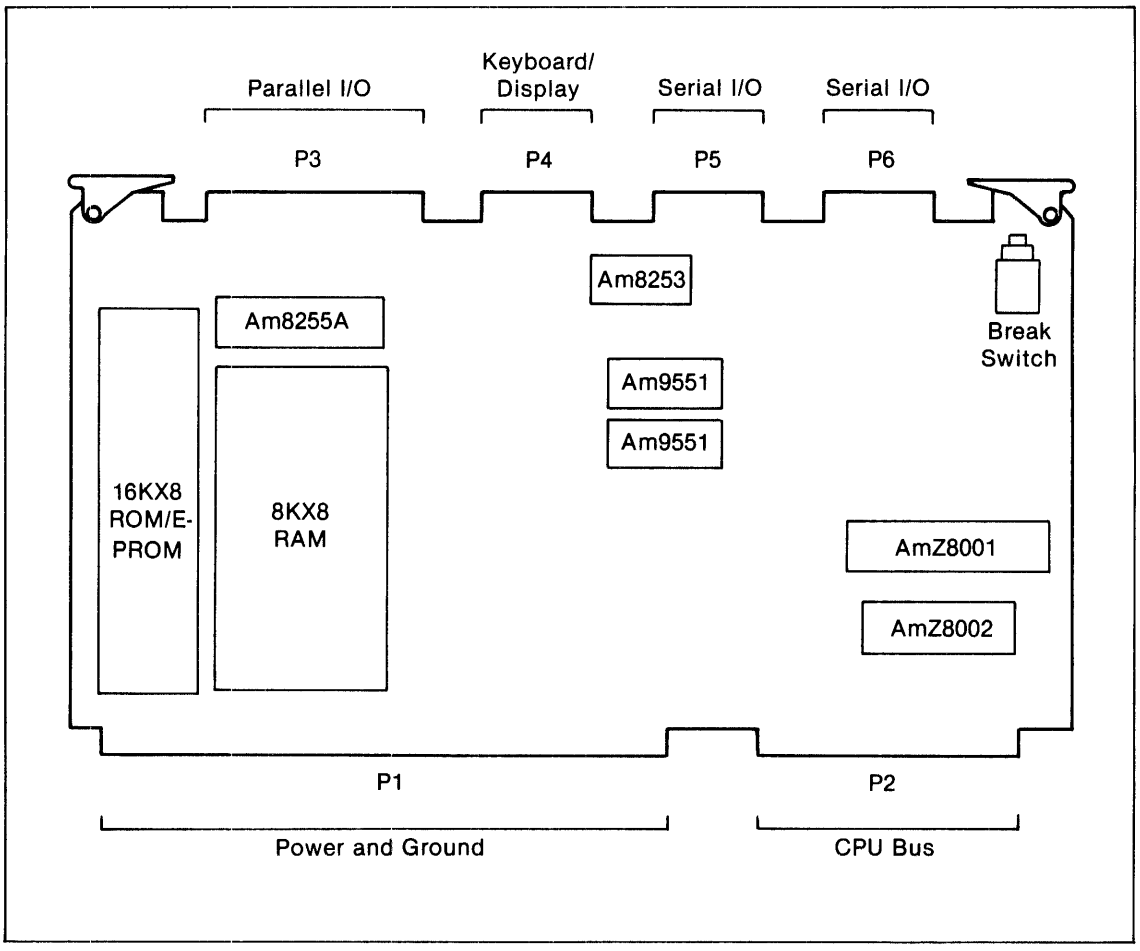



Figure 1-1. Board Layout

Figure 1-2 is a block diagram of the board. The system buses (address, data, and control) are buffered and available at connector P2. The multiplexed address/data is also available via the DATA pins on P2. The address and data buses, which are multiplexed in the CPU are separated into two separate 16-bit buses on the board. Figure 1-2 shows more clearly the difference between P5 and P6 serial I/O ports; one has a counter/timer available and the other can be either RS232C or 20mA current loop. Six of the seven AmZ8001 segment outputs are available at connector P2.

Memory-expansion boards and prototyping boards carrying any AmZ8000-compatible circuits (such as DMAs or even multiple CPUs) can be attached at P2. Specialized circuits can also be attached at P3, P5 and P6. P5 is useful for event-counting, process control circuits, or foreign host computers.

Figure 1-3 illustrates the range of standard plug-in configurations supported by the board's Monitor program. If attachments to the board are limited to any one or more of these three choices, application programming and execution can begin immediately without having to program the I/O circuits.

When a Development System is connected for up/down loading of programs, the Development System's console can communicate with the Evaluation Board to control all functions.

The Development System contains a comprehensive set of hardware and software resources to fully utilize AmZ8000 capabilities. The system includes 64K bytes of RAM, serial and parallel ports, and a multimaster bus. Existing programming support includes a CP/M-compatible operating system with linking loader, editor and debugger and an AmZ8000 macroassembler, 8080 macro assembler and AmZ8000 translator.

Table 1-1 gives a summary list of specifications for the Evaluation Board

## THE CPU

The AmZ8001 and the AmZ8002 microprocessors are register-oriented CPU's with minicomputer-like architecture. Sixteen general-purpose 16-bit registers, are available to the user. Over 100 instructions, and 400 combinations of instructions, can be used to manipulate data between the CPU registers, memory and I/O. Only one CPU can be used on-board, never both at the same time.

The CPU can operate in two modes, System and Normal, with separate relocatable stacks for each mode. This allows a distinction between privileged and protected instructions, as well as flexibility in allocating the system's use of memory. In either mode, 14 status conditions are continuously reported:

Internal operation
Memory refresh
I/O reference
Special I/O reference
Segment trap acknowledge
Nonmaskable interrupt acknowledge
Nonvectored interrupt acknowledge
Vectored interrupt acknowledge
Data memory request
Stack memory request
Data memory request (EPU)
Stack memory request (EPU)
Instruction space access
Instruction fetch, first word
Extension processor transfer
Reserved

Only three of these status outputs (2, 6, and 7) are needed by the Evaluation Board's internal functions, but all are available, by decoding the status lines, at the P2 edge connector.

The interrupt and trap structure is particularly powerful, with very fast response to external devices and illegal conditions.

## MEMORY

The Evaluation Board's dynamic RAM is refreshed by the CPU. Additional memory, either dynamic or static, can be added off-board through the P2 edge connector.

The AmZ8002 can directly address 64K bytes of memory using 16 address bits. The 23-bit address structure of the AmZ8001, is made up of 16 address bits and seven segment select outputs. Of the seven segment outputs available from the AmZ8001, only six are available on the Am96/4018, permitting memory addressing capability of 4 megabytes. If the distinctions between operating mode (System and Normal), and memory access status are used to differentiate memory resources, up to 384K bytes are available with the AmZ8002 and up to 24 Megabytes for the AmZ8001. Figure 1-4 illustrates the expanded memory capability for the AmZ8002.

## INPUT/OUTPUT

Memory space and I/O space are differentiated by a memory-request line and other status outputs from the CPU. The CPU can use 16-bit I/O port addresses, although in the Evaluation board implementation the upper four address bits are not decoded. This gives 4K I/O ports for both on-board and off-board I/O.

TABLE 1-1. SPECIFICATIONS

| | |
|---|---|
| CPU | AmZ8001 (segmented) or AmZ8002 (non-segmented) |
| Time Base | 4MHz crystal oscillator |
| Serial I/O | Two RS232C serial ports with software-programmable baud rates. One port jumper- selectable for 20mA current-loop (TTY) operation. |
| Parallel I/O | 24 parallel I/O lines (three 8-bit ports). Also provides interconnection to 16-bit Development Systems. |
| RAM Memory | 8K bytes of on-board dynamic memory; CPU refreshed (transparent). |
| ROM Space | 16K bytes of ROM/EPROM space provided in six sockets; ROM monitor occupies four sockets. |
| Counter/Timer | Three 16-bit programmable interval counter; two counters used for serial I/O baud rate control; third counter available to user. |
| Power | -12Vdc at 0.085A, +12Vdc at 0.06A, +5Vdc at 1.65A (without optional keyboard/display console), +5Vdc at 2.0A (with keyboard/ display console). |
| Dimensions | 12.0" (305 mm) x 6.75" (172 mm); MULTIBUS form factor with six edge connectors (P1 through P6). |
| Memory | ROM space: 0-3FFFH Addressing<br>RAM space: 4000-5FFFH |
| Environmental | 0 to 55C ambient in free-air space with Conditions relative humidity to 90% without condensation. |
| Edge-of-Card Connectors | P1: 86-Pin for power, ground, and initialize.<br>P2: 60-Pin CPU bus<br>P3: 50-Pin parallel I/O for up/down-load from 16-bit Development Systems.<br>P4: 26-Pin interface for optional keyboard/display board.<br>P5: 26-Pin RS232 and counter/timer interface.<br>P6: 26-Pin RS232 or 20mA current loop for CRT or TTY console. |
| Monitor | 8K ROM monitor included at addresses 0-1FFFH |

TABLE 1-1. SPECIFICATIONS (continued)

| | |
|---|---|
| Up/Down-Load | Can be plugged into 16-bit Development Systems to provide up-load and down-load capability. Can also by used with other host computer systems to execute AmZ8000 code. |
| Optional Keyboard/ Display | 56-key keyboard with 20-character alphanumeric LED display. Same physical form as Am96/4016 Evaluation Board with attaching standoff connectors and interconnection ribbon cable. |

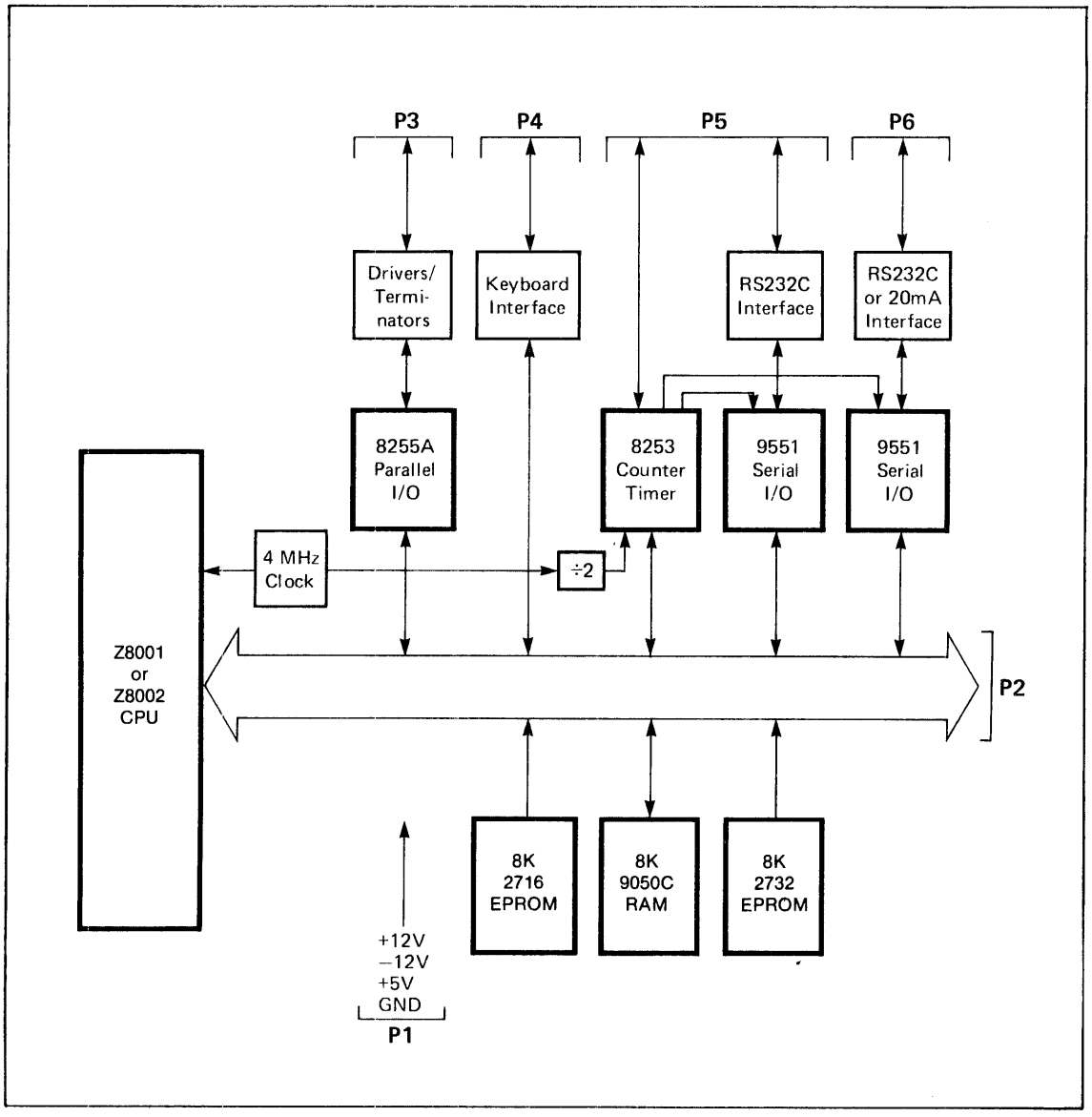



Figure 1-2. Am96/4018 Block Diagram

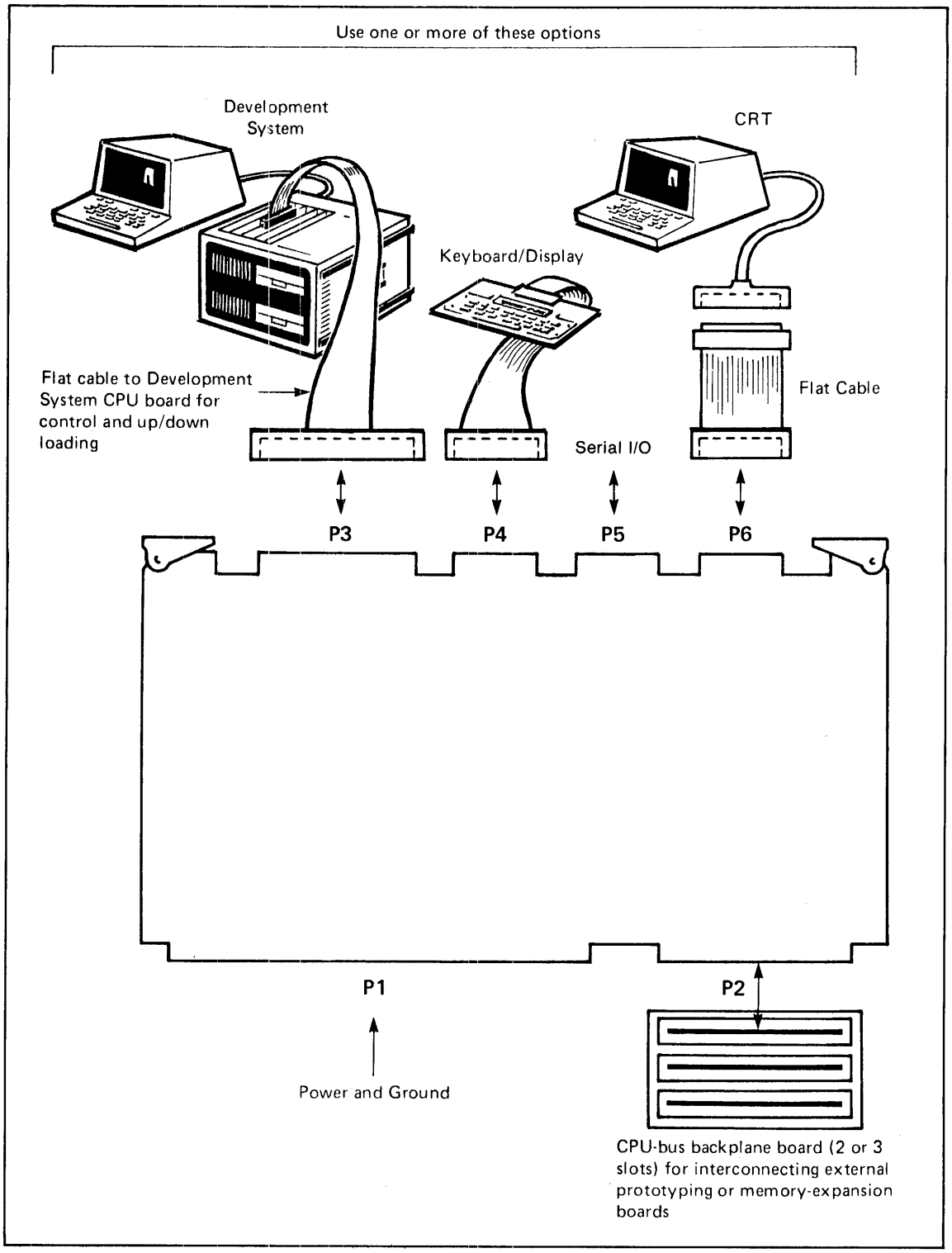



Figure 1-3. Standard Configurations

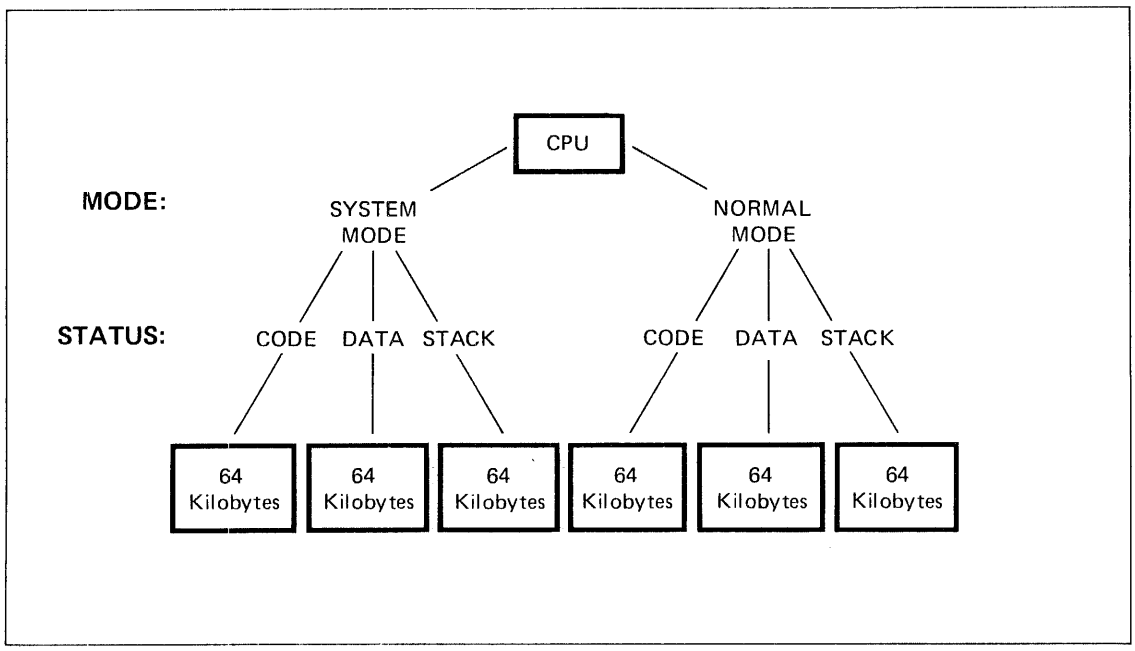



Figure 1-4. Memory Addressing Potential

The Evaluation Board implements only 8-bit I/O transfers on-board. These are the lower 8 bits on the data bus. Off-board I/O transfers can be 8 or 16 bits.

The 24 parallel I/O lines of the Am8255A circuit at P3 can be configured to transfer data in a variety of ways, with or without handshaking. In the standard configuration, this edge connector is used for up/down loading between the 16-bit Development Systems and the Evaluation Board. The Am8255A has a bit set/reset function that increases the efficiency of handshake software. The sockets provided for driver or terminator circuits allow further characterization of these ports for special applications. Ground lines are interleaved with signal lines for better noise immunity at this edge connector.

Two Am9551 Programmable Communications Interface devices can be programmed for a broad range of full-duplex, double-buffered serial communication protocols. The two serial ports are available at connectors P5(RS232C) and P6(RS232C or 20mA current loop). The data (baud) rate is controlled by two of three programmable counters in the Am8253 counter/timer device.

As mentioned above, no programming of I/O circuits is necessary when the Evaluation Board is used in one of the standard configurations illustrated in figure 1-3. The Monitor program will initialize these circuits.

## SOFTWARE

The Evaluation Board's ROM-based Monitor program provides complete system initialization functions and over two dozen user commands for access to CPU registers and memory. It features breakpoint, single step and trace functions, as well as up-save (Am96/4018 to host) and down-load (host to Am96/4018) for use with the 16-bit Micro Computer Development Systems as the host computer. Any AmZ8000 instruction whether privileged (system mode) or protected (normal mode) can be entered in hex format using only the Monitor.

The full CPU instruction set can be used to create very powerful application programs. The 110 basic instructions have permutations around operand addressing modes, plus autoincrement and autodecrement facilities. Code written for the AmZ8002 is compatible with the AmZ8001 microprocessor.

Most instructions are between one and three 16-bit words in length. Instructions and hardware are available to operate on bits, 4-bit digits (BCD), 8-bit bytes, 16-bit words, 32-bit long words, 64-bit quad words, byte strings and word strings.

String instructions are auto-incrementing or auto-decrementing block transfers. The strings can be up to 64 kilobytes in length, the entire directly-addressable memory, and the transfers are interruptable.

# CHAPTER 2
# INSTALLATION AND INTERFACING

## UNPACKING AND INSPECTION

Upon receipt of this equipment, inspect both the equipment and the shipping carton immediately for evidence of damage during transit. If the shipping carton is damaged or water-stained, request the carrier's agent to be present when the carton is opened. If the carrier's agent is not present when the carton is opened and the contents of the carton are damaged, save the carton and packing material for the agent's inspection. Shipping damages should be reported immediately to the carrier.

NOTE

Do not attempt to service the board yourself, as this will void the warranty.

## INSTALLATION OVERVIEW

Figure 2-1 illustrates the hardware connectors and sockets that might require setup before the board can be used. These are the six edge connectors, P1 through P6, plus sockets for ROM or E-PROM and the sockets for parallel I/O driver or terminator circuits. Each of these groups is discussed individually below.

NOTE

When one of the standard configurations illustrated in figure 1-3 has been ordered from AMC, only the following installation connections need be considered.

| EDGE CONNECTION | EXTERNAL CONNECTION |
|---|---|
| P3 | 16-bit Development Systems CPU board (P3), as illustrated in figure 2-2, or a 8/8630 Communication Controller package. |
| P4 | Keyboard/LED Display Board (Am96/4016-KBD), as illustrated in figure 2-3. |
| P5 | Serial I/O RS232C. |
| P6 | CRT or other RS232C terminal, or 20mA current loop. |

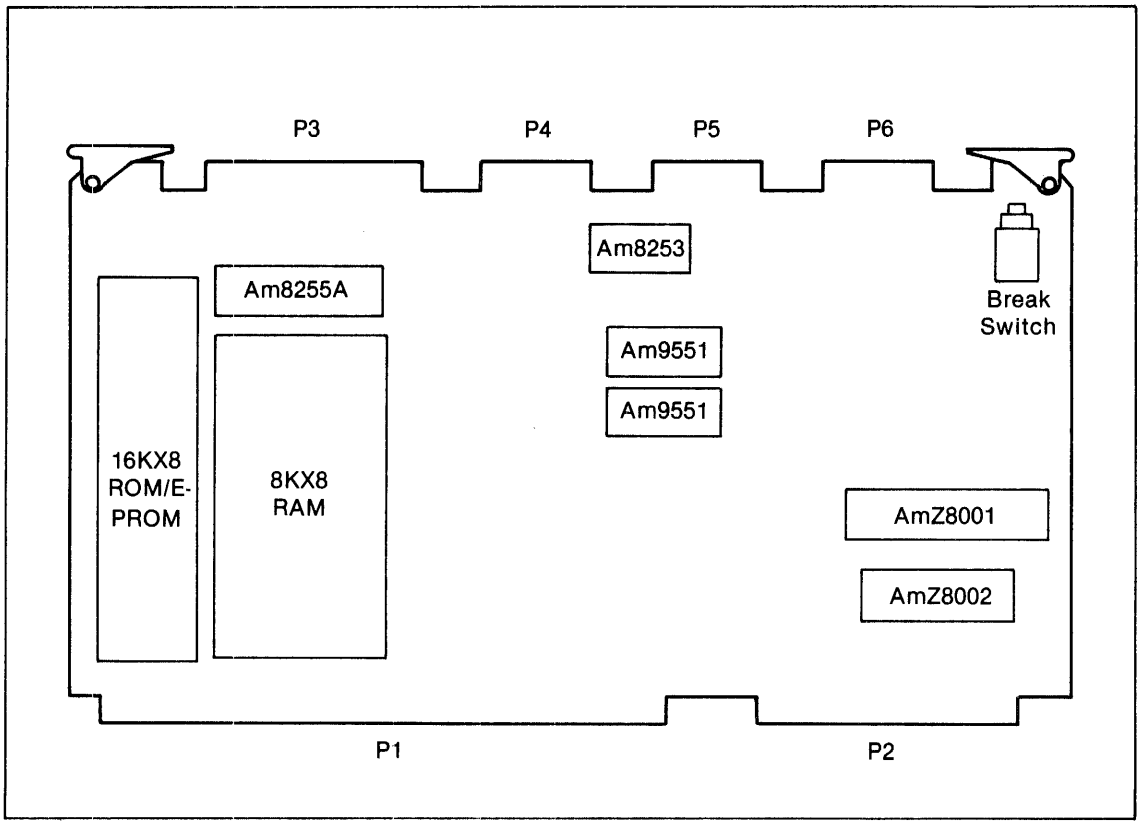



Figure 2-1. Edge Connectors and IC Sockets

## ROM/E-PROM SOCKETS

The board contains six sockets for fixed-address ROM or E-PROM. All boards are shipped with at least four E-PROMs installed, which contain the 8-kilobyte Monitor program. Table 2-1 lists the memory space assigned to ROM and the device reference designators.

TABLE 2-1. ROM/E-PROM SOCKETS

| PROGRAM | ADDRESSES (HEX) | SOCKET AND DEVICE |
|---|---|---|
| Monitor | 000000 to 000FFE (Even) | U23 (2716) |
| Monitor | 000001 to 000FFF (odd) | U20 (2716) |
| Monitor | 001000 to 001FFE (Even) | U17 (2716) |
| Monitor | 001001 to 001FFF (Odd) | U12 (2716) |
| User or ASM | 002000 to 003FFE (Even) | U7 (2732) |
| User or ASM | 002001 to 003FFF (Odd) | U5 (2732) |

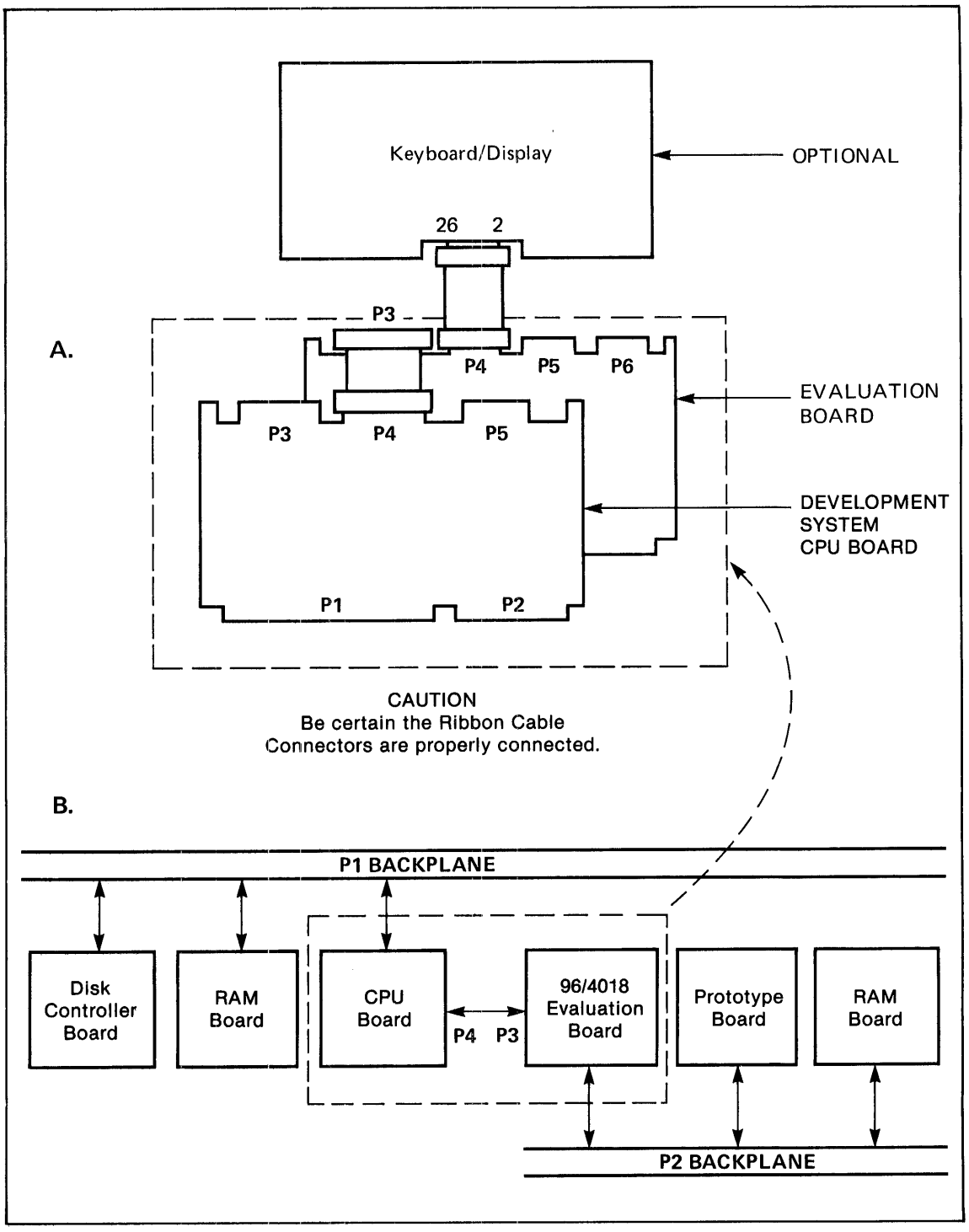



Figure 2-2. Connecting to the Development System

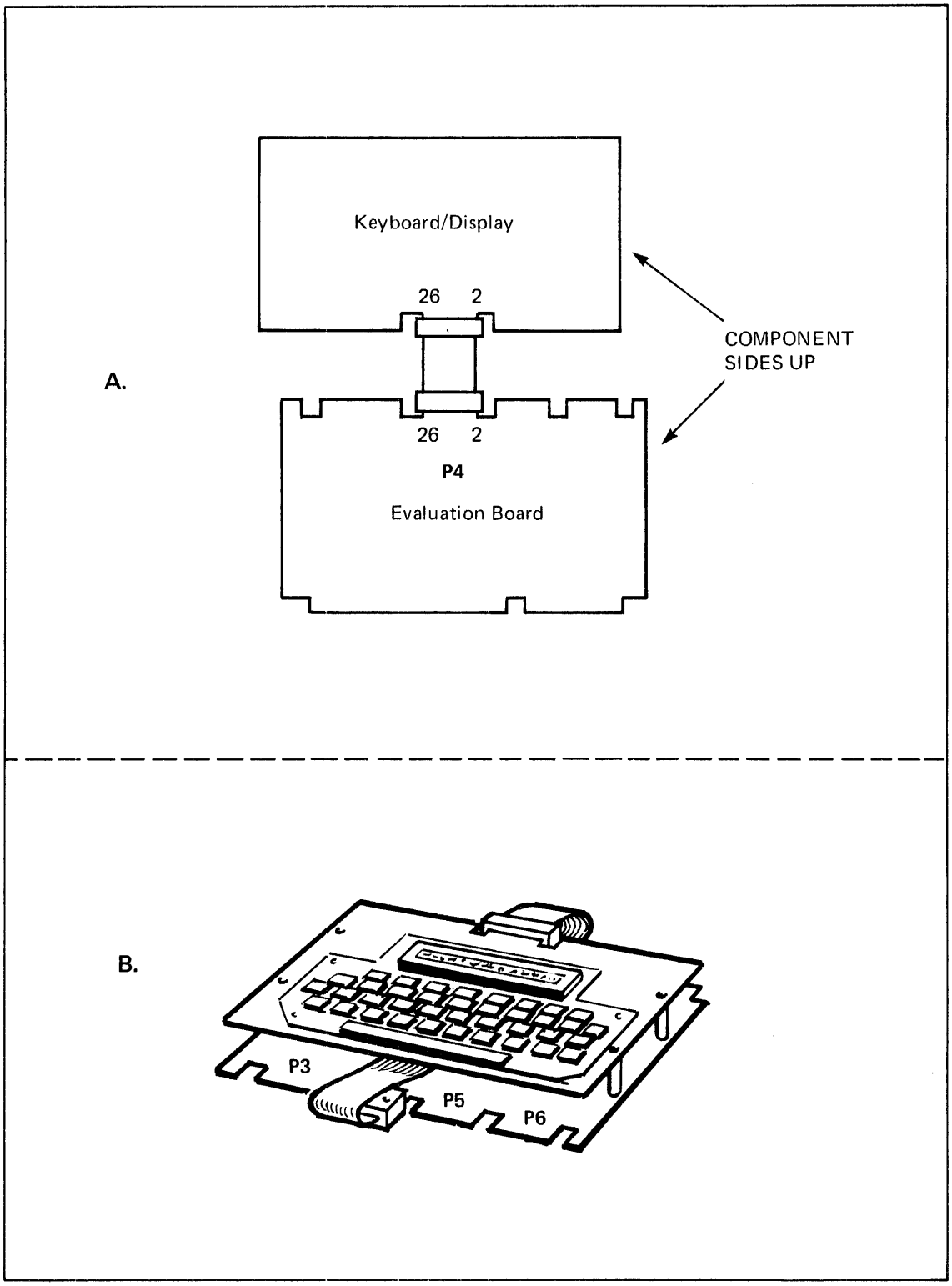



Figure 2-3. Connecting the Keyboard/Display Console

## PARALLEL I/O DRIVER/TERMINATORS

When the board is ordered, the driver and terminator circuits listed in table 2-2 are provided.

Various uses of the P3 edge connector may require different devices for the driver/terminator sockets. Some of the possible devices are listed in table 2-3. Alternatively, the sockets can be jumpered for direct connection to the Am8255A pins.

TABLE 2-2. DRIVER/TERMINATORS FOR PARALLEL PORT (P3)

| DEVICE | FUNCTION | SOCKET NO. |
|---|---|---|
| 74LS37 | Driver | U1 |
| 74LS37 | Driver | U2 |
| 74LS37 | Driver | U3 |
| iSBC902 | Terminator | U4 |
| iSBC902 | Terminator | U28 |
| iSBC902 | Terminator | U30 |

TABLE 2-3. OTHER DRIVER/TERMINATOR CIRCUITS FOR P3

| DRIVERS | TERMINATORS |
|---|---|
| 7438, 74LS38 | Intel iSBC 901 |
| 7437, 74LS37 | Intel iSBC 902 |
| 7432, 74LS32 | National BLC 901 |
| 7426, 74LS26 | National BLC 902 |
| 7409, 74LS09 | National BLC 903 |
| 7408, 74LS08 | |
| 7403, 74LS03 | |
| 7400, 74LS00 | |

## EDGE CONNECTORS

A minimum of two edge connectors must be used to operate the board. Power (+12V, -12V, +5V) and ground must be supplied through the P1 connector. A system console must also be connected on one of the remaining connectors: either P4 for the AMC-supplied keyboard/display console or P6 for a user-supplied terminal.

Many additions to these required connections are possible, as indicated in the list of edge-connector characteristics and applications below. The list is not needed if you are using the standard configuration illustrated in figure 1-3.

Note that connectors P1 and P2 have their odd-numbered pins on the component side of the board, whereas the remaining connectors have them on the solder side. The abbreviation N/C in the pin-connection lists means no connection.

P1 - an 86-pin connector physically (but not electrically) compatible to the Multibus and iSBC-80 formats. It is used only for power and ground, except that pin 14 can be used for external initializing (reset) when jumpered for this function (see figure 2-4). The pin connections are shown in table 2-4.

P2 - a 60-pin connector physically compatible with the Multibus and iSBC-80 formats. It carries buffered signals from all CPU lines except MREQ*, uI*, uO*, and +5V, which is on the P1 connector, and DECOUPLE, which is not used on the AmZ8000. The multiplexed address/ data bus from the CPU is available on the data pins, and is also demultiplexed into separate address and data buses. The CPU's RESET* line is connected directly to the RST* line on P2, although RST* also resets the Am9551 USARTs and the Am8255A PIO. One additional line, INH* is used to inhibit addressing of off-board memory when on-board memory is addressed, (INH* low will inhibit off-board memory).

The P2 connector is used to interface a prototyping board and/or external memory boards. An optional backplane is available for this purpose.

If the CPU is disabled via the BUSRQ* input, an off-board device can take control over most of the signals on this connector to access memory and peripherals on the Evaluation Board. The pin connections are shown in table 2-5.

P3 - a 50-pin connector to the six driver/terminator sockets of the Am8255A parallel I/O circuit. The 24 active line are divided into three 8-bit ports (addresses FF0, FF1 and FF2 hex where the upper four address bits are not decoded). Each port can be programmed independently for input or output to match the functions of the driver or terminator ICs that you insert in sockets U1 through U4, U28, and U30.

When the Development System is used, P3 of the Evaluation Board is connected to the Development System's CPU board for up/down loading and control from the Development System's console, or to the 8/8630. This is illustrated in figure 2-2a. The pin connections are shown in table 2-6.

NOTE

Be certain the connectors between the Evaluation Board and CPU are connected properly. Damage to the Evaluation Board may result if P3 is reversed.

P4 - 26-pin connector used to attach the optional keyboard/display console.

The cable-connection method for the keyboard/display console is illustrated in figure 2-3.

The pin connections listed in table 2-7 include keyboard return and scan lines (R0-R2, S0-S2) and port addresses for LED characters (Ports FC0 through FD0 hex).

P5 - 26-pin connector providing access to the RS232C interface to one of the two Am9551 serial I/O circuits, plus three three pins connected to channel 0 of the Am8253 counter/timer circuit.

This connector can be used for time-controlled serial I/O (e.g., process controllers), a host computer other than the 16-bit Development Systems, or any other RS232C serial device.

The I/O port address for data through this connector is FEC hex.

The pin connections are given in table 2-8. Parenthesized numbers are the corresponding EIA pin numbers for RS232C. See the Am9551 and Am8253 Data Sheets for more details.

P6 - 26-pin connector providing access to the RS232C or 20mA interface to one of the two Am9551 serial I/O circuits.

Refer to figure C-6 for jumpering to change P6 from RS232 to 20mZ current loop.

This connector can be used for a system console (terminal) or any other serial I/O device.

The I/O port address for data through this connector is FE8 hex.

The pin connections are given in table 2-9. Parenthesized numbers are the corresponding EIA pin numbers for RS232C.

## KEYBOARD/DISPLAY INSTALLATION

When using the optional keyboard/display console, the following installation method should be used.

Orient the Evaluation Board and the keyboard/display board so that their component sides are facing up and the P4 connector of the Evaluation Board faces the only edge connector of the keyboard/display board. Place the flat cable between the two connectors so that the even pins of the Evaluation Board connect to the even pins of the keyboard/display board, as shown in figure 2-3a. Fold the cable between the boards so that the board spacers lock into the holes on the boards, as shown in figure 2-3b. Both boards will then have their component sides facing down.

TABLE 2-4. P1 CONNECTOR PINS

| COMPONENT SIDE | | SOLDER SIDE | |
|---|---|---|---|
| 1 | GND | 2 | GND |
| 3 | +5V | 4 | +5V |
| 5 | +5V | 6 | +5V |
| 7 | +12V | 8 | +12V |
| 9 | N/C | 10 | N/C |
| 11 | GND | 12 | GND |
| 13 | N/C | 14 | INIT* |
| 15 | N/C | 16 | N/C |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| 73 | N/C | 74 | N/C |
| 75 | GND | 76 | GND |
| 77 | N/C | 78 | N/C |
| 79 | -12V | 80 | -12V |
| 81 | +5V | 82 | +5V |

TABLE 2-5. P2 CONNECTOR PINS

| PIN | SIGNAL | PIN | SIGNAL | PIN | SIGNAL |
|---|---|---|---|---|---|
| 1 | ADR10 | 21 | PHI(CLK) | 41 | AC |
| 2 | ADR11 | 22 | STOP* | 42 | AD |
| 3 | ADR12 | 23 | NMI* | 43 | AE |
| 4 | ADR13 | 24 | VI* | 44 | AF |
| 5 | ADR14 | 25 | ST0 | 45 | A0/D0 |
| 6 | ADR15 | 26 | ST1 | 46 | A1/D1 |
| 7 | NVI* | 27 | ST2 | 47 | A2/D2 |
| 8 | INH* | 28 | ST3 | 48 | A3/D3 |
| 9 | AS* | 29 | A0 | 49 | A4/D4 |
| 10 | DS* | 30 | A1 | 50 | A5/D5 |
| 11 | R/W* | 31 | A2 | 51 | A6/D6 |
| 12 | SEGT* | 32 | A3 | 52 | A7/D7 |
| 13 | N/S* | 33 | A4 | 53 | A8/D8 |
| 14 | B/W* | 34 | A5 | 54 | A9/D9 |
| 15 | BUSRQ* | 35 | A6 | 55 | AA/DA |
| 16 | BUSAK* | 36 | A7 | 56 | AB/DB |
| 17 | GND | 37 | A8 | 57 | AC/DC |
| 18 | GND | 38 | A9 | 58 | AD/DD |
| 19 | RST* | 39 | AA | 59 | AE/DE |
| 20 | WAIT* | 40 | AB | 60 | AF/DF |

TABLE 2-6. P3 CONNECTOR PINS

| SOLDER SIDE | | COMPONENT SIDE | |
|---|---|---|---|
| 1 | PB7 (Port FF1) | 2 | GND |
| 3 | PB6 | 4 | GND |
| 5 | PB5 | 6 | GND |
| 7 | PB4 | 8 | GND |
| 9 | PB3 | 10 | GND |
| 11 | PB2 | 12 | GND |
| 13 | PB1 | 14 | GND |
| 15 | PB0 | 16 | GND |
| 17 | PC3 (Port FF2) | 18 | GND |
| 19 | PC2 | 20 | GND |
| 21 | PC1 | 22 | GND |
| 23 | PC0 | 24 | GND |
| 25 | PC4 | 26 | GND |
| 27 | PC5 | 28 | GND |
| 29 | PC6 | 30 | GND |
| 31 | PC7 | 32 | GND |
| 33 | PA7 (Port FF0) | 34 | GND |
| 35 | PA6 | 36 | GND |
| 37 | PA5 | 38 | GND |
| 39 | PA4 | 40 | GND |
| 41 | PA3 | 42 | GND |
| 43 | PA2 | 44 | GND |
| 45 | PA1 | 46 | GND |
| 47 | PA0 | 48 | GND |
| 49 | N/C | 50 | GND |

TABLE 2-7. P4 CONNECTOR PINS

| SOLDER SIDE | | COMPONENT SIDE | |
|---|---|---|---|
| 1 | GND | 2 | KEYBOARD DETECT |
| 3 | CTL | 4 | SHIFT |
| 5 | R0 | 6 | R1 |
| 7 | R2 | 8 | S0 |
| 9 | S1 | 10 | S2 |
| 11 | PFC0* | 12 | PFC4* |
| 13 | PFC8* | 14 | PFCC* |
| 15 | PFD0* | 16 | A0 |
| 17 | A1 | 18 | ID0 |
| 19 | ID1 | 20 | ID2 |
| 21 | ID3 | 22 | ID4 |
| 23 | ID5 | 24 | ID6 |
| 25 | GND | 26 | +5V |

TABLE 2-8. P5 CONNECTOR PINS

| SOLDER SIDE | | (RS232) | COMPONENT SIDE | | (RS232) |
|---|---|---|---|---|---|
| 1 | CHASSIS GND | | 2 | N/C | |
| 3 | TRANSMITTED DATA | (2) | 4 | N/C | |
| 5 | RECEIVED DATA | (3) | 6 | N/C | |
| 7 | REQUEST TO SEND | (4) | 8 | N/C | |
| 9 | CLEAR TO SEND | (5) | 10 | N/C | |
| 11 | DATA SET READY | (6) | 12 | N/C | |
| 13 | SIGNAL GND | (7) | 14 | DATA TERM RDY | (20) |
| 15 | N/C | | 16 | N/C | |
| 17 | N/C | | 18 | N/C | |
| 19 | N/C | | 20 | N/C | |
| 21 | N/C | | 22 | N/C | |
| 23 | CLKO | | 24 | GATEO | |
| 25 | OUTO | | 26 | SIGNAL GND | |

TABLE 2-9. P6 CONNECTOR PINS

| SOLDER SIDE | | (RS232) | COMPONENT SIDE | | (RS232) |
|---|---|---|---|---|---|
| 1 | CHASSIS GND | ( 1) | 2 | N/C | |
| 3 | TRANSMITTED DATA | ( 2) | 4 | N/C | |
| 5 | RECEIVED DATA | ( 3) | 6 | TTY RDR CONTROL | (16) |
| 7 | REQUEST TO SEND | ( 4) | 8 | N/C | |
| 9 | CLEAR TO SEND | ( 5) | 10 | N/C | |
| 11 | DATA SET READY | ( 6) | 12 | N/C | |
| 13 | SIGNAL GND | ( 7) | 14 | DATA TERM RDY | (20) |
| 15 | DATA CARRIER RTN | ( 8) | 16 | TTY RDR CONTROL RTN | (21) |
| 17 | N/C | | 18 | N/C | |
| 19 | N/C | | 20 | N/C | |
| 21 | N/C | | 22 | TTY RX TRN | (24) |
| 23 | TTY RX | (12) | 24 | TTY TX RN | (25) |
| 25 | TTY TX | (13) | 26 | SIGNAL GND | |

## PROTOTYPING AND MEMORY-EXPANSION BOARDS

Prototyping and memory-expansion boards can be connected to the CPU bus via P2. AMC offers the following optional products to accommodate these functions:

- Universal Prototyping Board for Multibus (Am96/9410)
- 64-kilobyte dynamic RAM board for standard Multibus and Evaluation Board with AmZ8002 (Am96/1064) or 128 kilobytes RAM (Am96/1128) for the AmZ8001.
- Three-board backplane for P2 edge connector (Am96/4016/400).
- Six-board Multibus-compatible card cage with cooling fan (Am95/6440). A card cage with power supplies is also available (Am95/6448)

Prototyping boards can contain any type of circuitry, including circuits that capture the system buses. The AmZ8002 bus is described further in the section entitled Principles of Operations.

Expansion memory can be of any type compatible with the available pins and the buffered electrical characteristics at P2 (see appendix A). Dynamic-memory refresh may be done on the external board or it may use the CPU refresh cycle, as does the on-board RAM. This is accomplished by decoding the AmZ8000 status lines, available on the P2 bus.

The 64K and 128K RAM board supplied by AMC are compatible with both the Evaluation Board (on P2), and with standard Multibus slave applications (on P1). Due to this dual use, dynamic refresh is handled on the memory board itself rather than relying on the AmZ8002. The P2 connector of the memory board supports 24 address bits whereas its P1 connector supports 20 bits.

## DEVELOPMENT SYSTEM INTERCONNECTION

When a 16-bit Development System is used, P3 of the Evaluation Board is connected to P4 of the Development System's CPU board as illustrated in figure 2-2. A cable (Am96/4016/500) for connecting P3 to P4 is available from AMC. Alternatively, a cable can be fabricated by using two Scotchflex card edge connectors (3M part #3415-0001) and one foot of Scotchflex flat cable (3M part #3306/50).

A diskette-based program named GHOST is supplied with the Development System for communication between the Development System and the Evaluation Board. Execution of this utility on the Development System allows Evaluation Board operations to be controlled through the Development-system console.

The Development System can be used for programming work simultaneously with execution of programs on the Evaluation Board. The P1 bus serves the Development System while the P2 bus serves the Evaluation Board. In this case, separate consoles may be needed. However, communication or transfers between the Evaluation Board and the Development System require the GHOST program to be in execution.

## JUMPER OPTIONS

Several jumpers, illustrated in figure 2-4, can be connected to modify the standard functions of the Evaluation Board. There is a Break switch for regaining CPU control, two jumpers for grounding external I/O devices to the board, one for timing an external device, one for selecting RS232C or TTY serial interface, one for enabling an initialization line, one for enabling an inhibit line, and one for disabling the memory-cycle Wait state.

The jumper configuration when the board is shipped is as follows: 8-9, 11-12, 18-19, 21-22, and 23-24. Wire-wrapping between the jumper posts will affect the board's functions as described below.

BREAK SWITCH - When pressed, it asynchronously generates a non-maskable interrupt (NMI*) to the CPU, which forces control back to the Monitor program. This circuit is connected via jumpers 18 and 19. If the NMI* line on P2 is to be used, the Break switch must be disabled by removing jumper 18 and 19, and placing a jumper between 17 and 18.

Jumper 1,2 - When jumpered, it connects the 2MHz clock (4MHz divided by 2) to counter 0 of the Am8253 counter/timer circuit, instead of using the external clock via P5.

Jumper 3,4 - When jumpered, it connects the chassis ground of the external serial I/O device at P5 to the Evaluation-Board ground.

Jumper 5,6 - When jumpered, it connects the chassis ground of the external serial I/O device at P6 to the Evaluation-Board ground.

Jumper 8,9 - Must be jumpered for Z8002 CPU. For Z8001 CPU, jumper 7 and 8 instead.

Jumper 11,12 - Must be jumpered for Z8002 CPU. For Z8001 CPU, jumper 10 and 11 instead.

Jumper 14,15,16,17 - Used by the Monitor to determine baud rate at P6 as follows:

9600 baud - all open
2400 baud - connect 14 to 17
300 baud - connect 15 to 16
110 baud - connect 14 to 17 and 15 to 16

Jumper 18,19 - These jumpers enable the Break switch. If NMI* is used, at P2, connect jumper between 17 and 18 instead.

Jumper 21,22 - Must be jumpered for Z8002 CPU. For Z8001 CPU, jumper 20 and 21 instead.

Jumper 23,24 - When jumpered, it enables pin 14 on the P1 Multibus connector to be an INIT* input. When the INIT* line is low, it resets the CPU, USARTs, and parallel I/O circuit.

Jumper 25,26 - When jumpered, it connects the OUT0/ output of the Am8253 at P5 to the NVI* line, thereby generating periodic non-vectored interrupts.

RS232/20mA (P6) The Evaluation Board is shipped with the local console I/O port configured for RS232C. To configure for use with a 20mA current-loop terminal, refer to figure C-6.

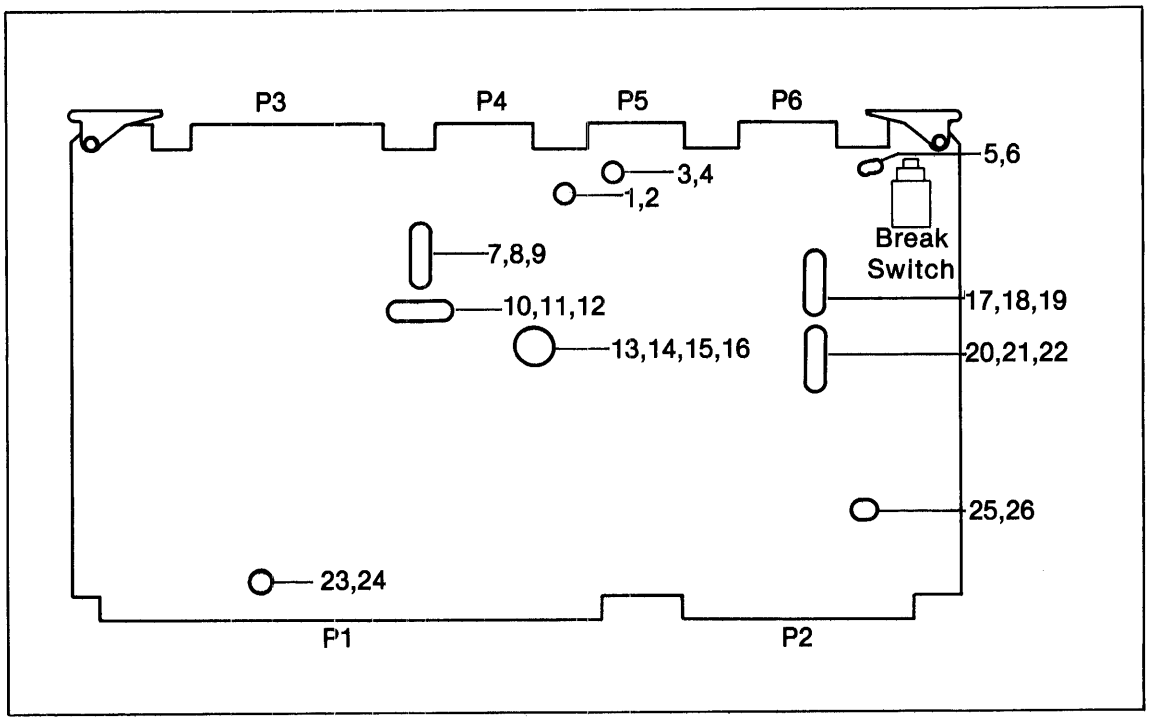



Figure 2-4. Jumper Option

# CHAPTER 3
# PERIPHERAL PROGRAMMING

## PARALLEL I/O

The Am8255A provides three 8-bit parallel I/O ports into or out of the lower eight lines of the internal data bus. In the standard Evaluation Board configuration, the three ports (A, B and C) are used for up/down program loading and remote console control from the 16-bit Development Systems. Port A is used for data output to the Development System, port B for data input, and port C for software-controlled handshaking (4 bits in, 4 bits out).

During power-up initialization, the Monitor writes an 8-bit Operation Control Word to the Am8255A control register for this purpose. In addition, a Bit Set/Reset handshaking byte is written to the same I/O address by the Monitor for each byte of data sent to port A. The port C lines are read for a response from AmSYS 8/8.

For non-standard uses of the P3 edge connector, the 8-bit Operation Control Word must be rewritten after initialization by the Monitor. There are three modes of operation possible:

Mode 0 - Basic input or output without strobed handshaking (but allowing software handshake routines). Ports A and B are 8 bits each and port C consists of two 4-bit ports. Any port can be input or output. Outputs are latched; inputs are not. In the standard Evaluation Board, the Am8255A is initialized in this mode.

Mode 1 - Input or output in conjunction with strobed handshaking signals. Port A consists of 8 data lines in conjunction with the upper four handshake lines of port C. Port B is similar but uses the lower four lines of port C for handshake. All data lines are latched, whether input or

Mode 2 - Bidirectional data transfers on the eight lines of port A, controlled by the upper five monodirectional lines of of port C. Port A is latched on both input and output. The lower three lines of port C and the eight eight lines of port B are monodirectional input or output data lines.

The form of the 8-bit Operation Control Word is shown in figure 3-1.

Port C contains an 8-bit output latch/buffer and an 8-bit input buffer (no latch). Any of the output bits can be set or reset with an output instruction to the same address according to the 8-bit Bit Set/Reset Control Word shown in figure 3-2.

After initialization, data or control bytes can be transferred at ports A, B and C using the I/O addresses shown in table 3-1.

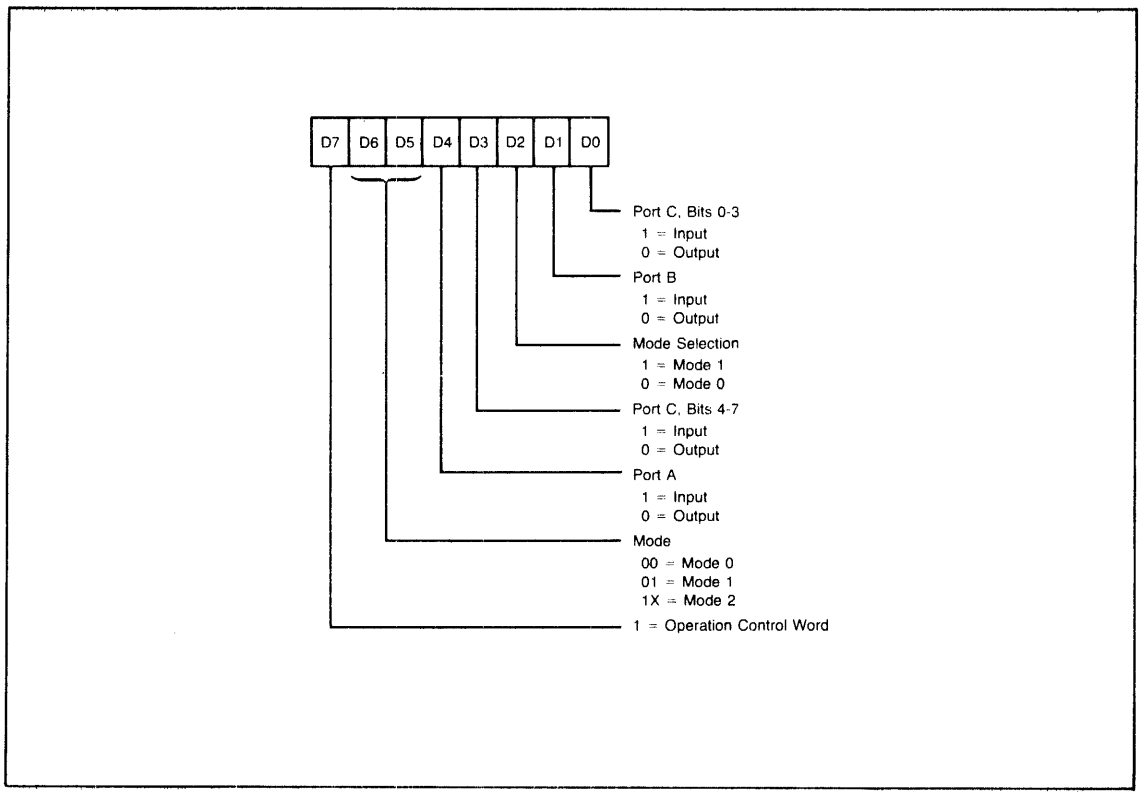



Figure 3-1. Am8255A Operation Control Word Format

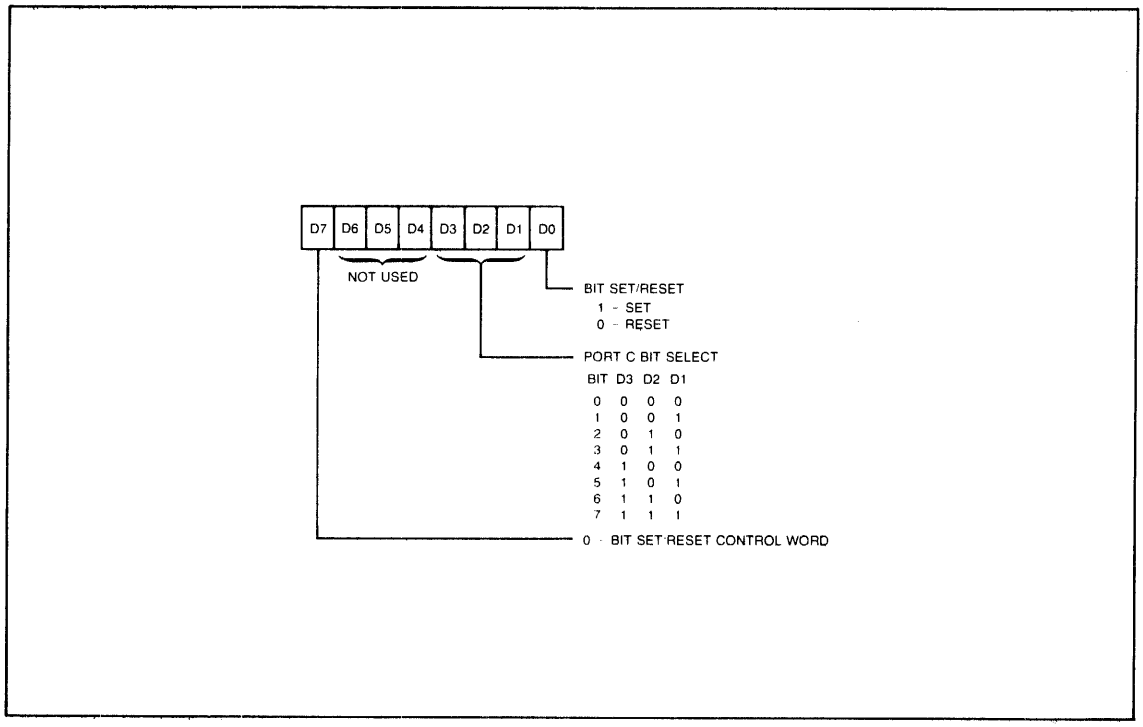



Figure 3-2. Am8255A Bit Set/Reset Control Word Format

TABLE 3-1. Am8255A ADDRESSES

| Port | Data | Control |
|---|---|---|
| A | FF0 | FF3 |
| B | FF1 | FF3 |
| C | FF2 | FF3 |

The six 14-pin sockets provided for drivers and terminators between the Am8255A and the P3 edge connector limit the use of all three ports to either input or output. Bi-directional capability, which is normally possible with port A in Mode 2 is not possible, with the drivers used with the Evaluation Board. However, if there is only a short distance between the P3 edge connector and the external device, the driver/terminator sockets can be shorted with headers and jumpers to use the bi-directional capability of port A.

## SERIAL I/O

Two Am9551 USARTs provide full-duplex serial transmission between the lower eight lines of the internal data bus and the P5 and P6 edge connectors. Both the transmitter and receiver clock inputs to these circuits come from the Am8253 counter/timer.

In the standard Evaluation Board configuration, the Am9551 at P6 (U42) is initialized by the Monitor with control bytes at power-up, independently of whether the ports are jumpered for RS232C or 20mA compatibility. The devices are configured for 8-bit asynchronous characters with two stop bits, parity disabled and 16x baud rate factor. They are also enabled for transmission and reception, with the Request-To-Send line forced active and without Hunt Mode. This is accomplished by first writing a hex CE Mode Control Code to I/O addresses FE9H and FEDH, followed by a hex 27 Control Command to the same address (11001110 followed by 00100111). The format for the Asynchronous Mode Control Code is shown in figure 3-3. The format for the Control Command that follows is shown in figure 3-4.

Synchronous transmission/reception is obtained by first writing a Synchronous Mode Control Code followed by one or two 8-bit sync characters, followed by the same Control Command format illustrated above. The format for the Synchronous Mode Control Code is shown in figure 3-5.

If you wish to change the Monitor's initialization of the P6 USART after power-up, or if you intend to use the P5 USART, the following steps are recommended:

1. Write three successive null bytes to the control port (FE9H for P6 or FEDH for P5).

2. Write a hex 40 to the port for a software reset.

3. Initialize the device by writing new control information as described above.

4. Execute a dummy Read to clear data in buffer and RxRDY line.

After initialization, data can be transferred using the I/O addresses shown in table 3-2.

A readable status register maintains information on the current operational status of the device. Its format is shown in figure 3-6.

The definition of the status bits is as follows:

TxRDY — Transmitter Ready indicates the Am9551 is ready to accept a data character or command.

RxRDY — Receiver Ready indicates the Am9551 has received a character on its serial input and is ready to transfer it to the CPU.

TxE — Transmitter Empty signals the processor that the transmit register is empty.

PE — Parity Error indicates the character stored in the receiver character buffer was received with an incorrect number of binary 1 bits.

OE — Overrun flag is set when a byte stored in the receiver character register is overwritten with a new byte before being transferred to the processor.

FE — Framing Error indicates the asynchronous mode byte stored in the receiver character buffer was received with incorrect character bit format.

SYNDET — When Sync Detect is set for internal sync detect, this bit indicates character sync has been achieved and the Am9551 is ready for data.

After initialization, always check the status of the TxRDY bit prior to writing data or a new command word to the Am9551. The TxRDY bit must be true to prevent overwriting and subsequent loss of commands or data. The TxRDY is inactive until initialization has been completed.

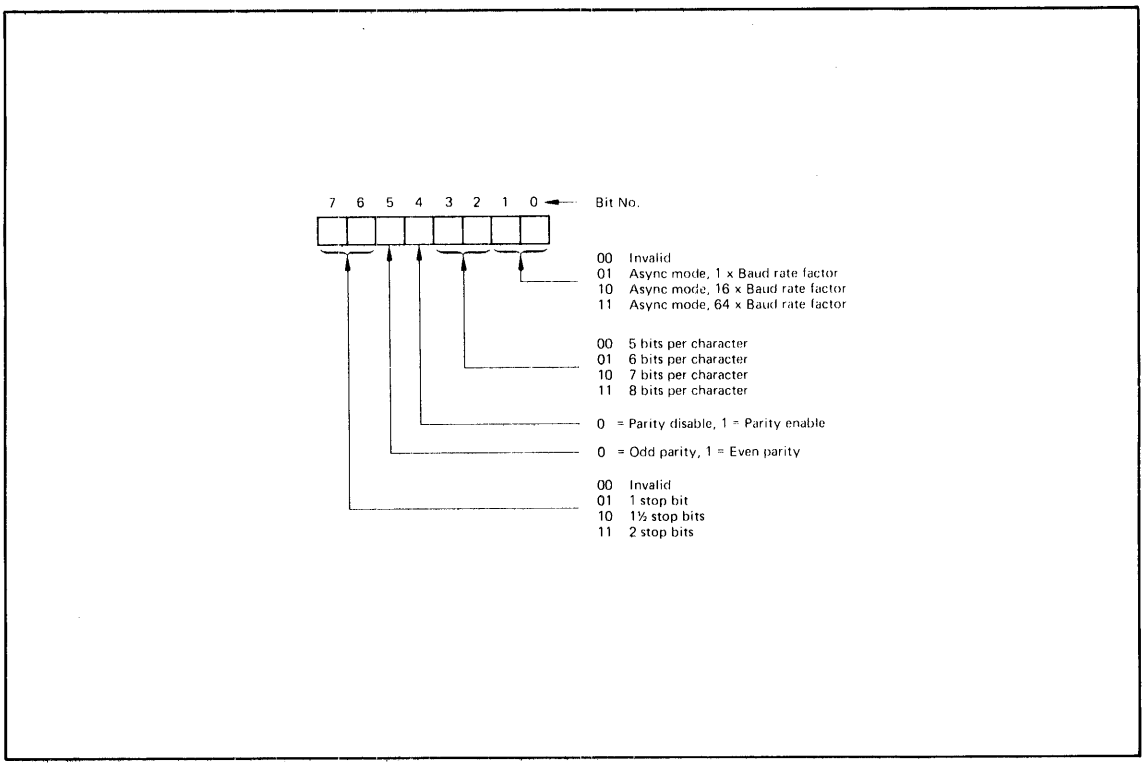



Figure 3-3. Am9551 Asynchronous Mode Control Code

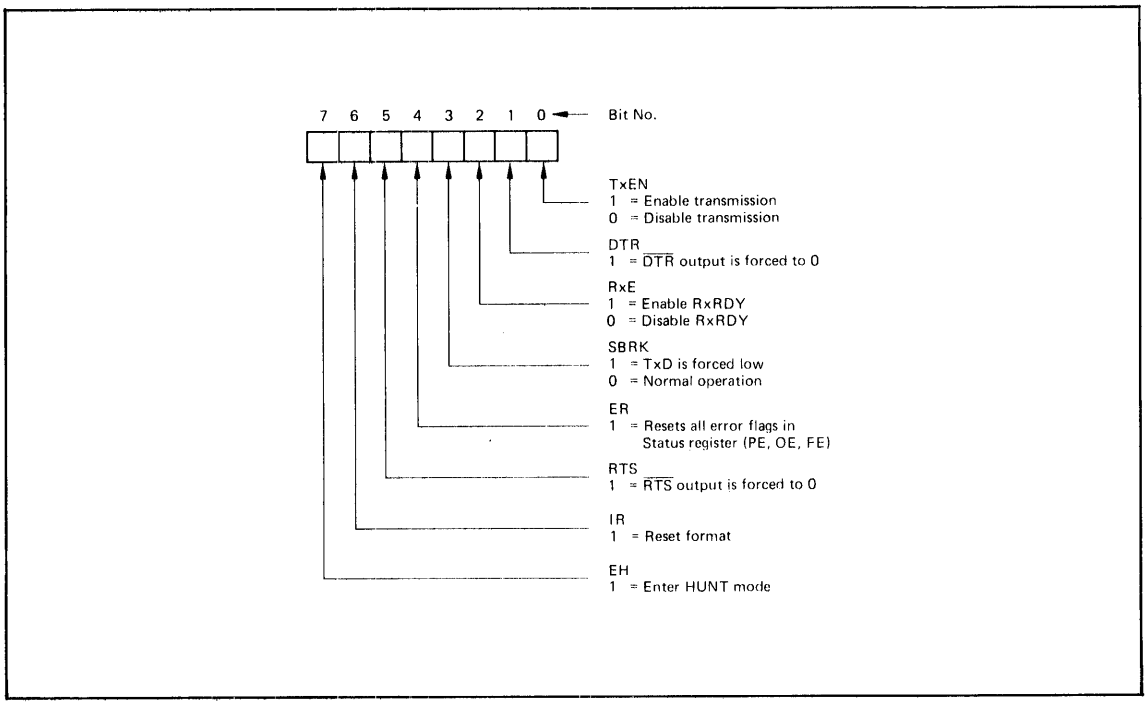



Figure 3-4. Am9551 Control Command

TABLE 3-2. Am9551 ADDRESSES

| PORT | DATA | CONTROL |
|---|---|---|
| P5 | FEE#<br>FEC | FEF#<br>FED |
| P6 | FEA<br>FE8 | FEB<br>FE9 |

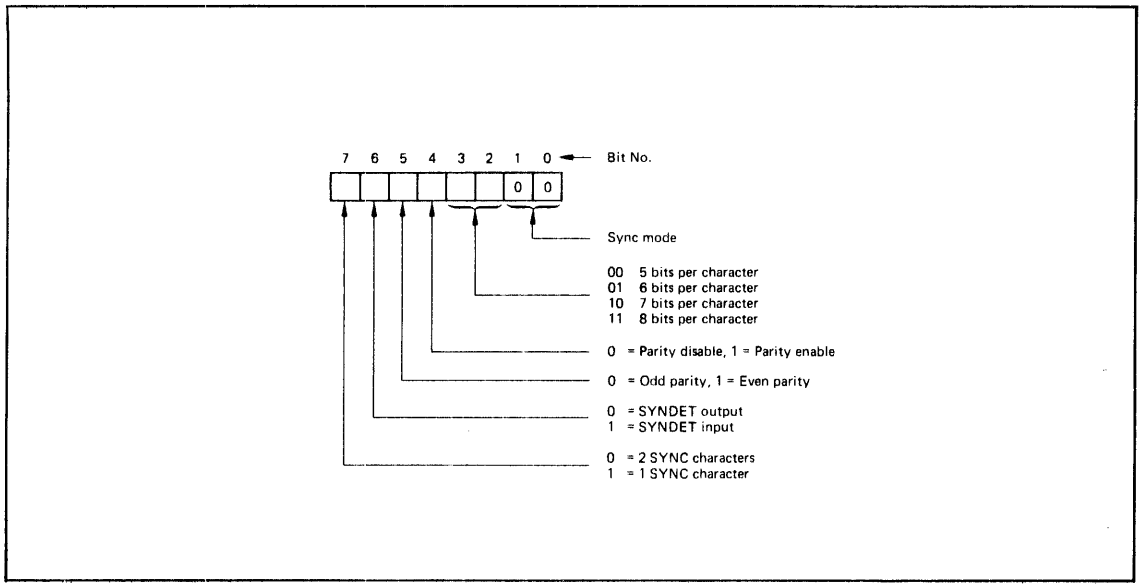



Figure 3-5. Synchronous Mode Control Code

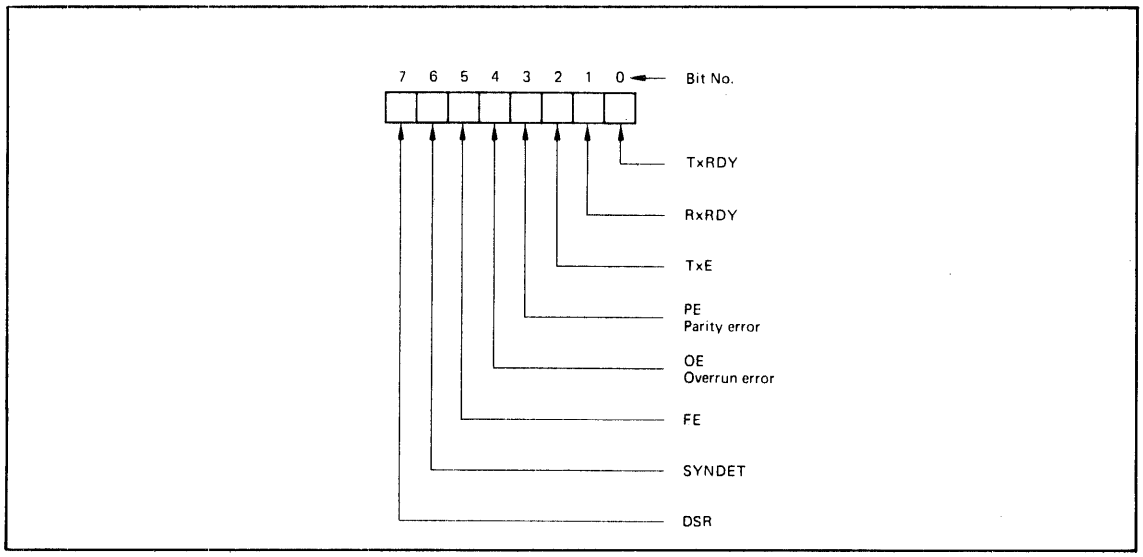



Figure 3-6. Am9551 Status Register

## COUNTER/TIMER

The Am8253 provides three counter/timer channels using a 2MHz clock input derived from the 4MHz CPU clock. Two of the three counter outputs drive the transmit and receive clock inputs of the two Am9551 USARTs up to 9600 baud; counter 1 is attached to the USART at P6, and counter 2 is attached to the USART at P5. The third counter output, together with its associated gate and clock inputs, are available for external use on P5 together with the serial port.

A 16-bit binary counter in each channel counts down at the clock input rate and generates a signal when reaching zero. Alternatively, a 4-decade BCD counter can be used.

There are five modes associated with the manner in which signals are output:

Mode 0 - Interrupt on terminal count

The output will be initially low after the mode set operation. After the count is loaded into the selected count register, the output will remain low and the counter will count. When terminal count is reached, the output will go high and remain high until the selected count register is reloaded with the mode.

Reloading a counter register during counting results in the following:

(1) Write 1st byte stops the current counting.

(2) Write 2nd byte starts the new count.

The GATE input will enable the counting when high and inhibit counting when low.

Mode 1 - Programmable One-Shot

The output will go low on the count following the rising edge of the GATE input.

The output will go high on the terminal count. If a new count value is loaded while th output is low it will not affect the of the one shot pulse until the succeeding trigger. The current count can be read at any time without affecting the one-shot pulse.

The one-shot is retriggerable, hence the output will remain low for the full count after any rising edge of the gate input.

Mode 2 - Rate Generator

Divide by N counter. The output will be low for one period of the input clock. The period from one output pulse to the next equals the number of input counts in the count register. If the count register is reloaded between output pulses, the present period will not be affected, but the subsequent period will reflect the new value.

The GATE input, when low, will force the output high. When the GATE input goes high, the counter will start from the initial count. Thus, the GATE input can be used to synchronize the counter. When this mode is set, the output will remain high until after the count register is loaded. The output then can also be synchronized by software.

Mode 3 - Square Wave Rate Generator.

Similar to Mode 2 except that the output will remain high until one-half the count has been completed (for even numbers) and go low for the other half of the count. If the count is odd, the output will be high for (N+1)/2 counts and low for (N-1)/2 counts.

If the count register is reloaded with a new value during counting, this new value will be reflected immediately after the output transition of the current count.

Mode 4 - Software-triggered strobe.

After the mode is set, the output will be high. When the count is loaded, the counter will begin counting. On terminal count, the output will go low for one input clock period, then will go high again.

If the count register is reloaded between output pulses the present period will not be affected, but the subsequent period will reflect the new value. The count will be inhibited while the gate input is low. Reloading the counter register will restart counting beginning with the new number.

Mode 5 - Hardware-triggered strobe

The counter will start counting after the rising edge of the trigger input and will go low for one clock period when the terminal count is reached. The counter is retriggerable. The output will not go low until the full count after the rising edge of any trigger.

The format of the control byte is shown in figure 3-7; the various bits in the control byte are defined by figure 3-8. Counter 0 is not initialized by the Monitor during power-up. It is available for use at P5 and must be programmed, if used, in the same manner described

above. There are two inputs (clock and gate) and one output available. A jumper is provided on the Evaluation Board for tying the clock input to the same 2MHz clock used in channels 1 and 2.

The current counter values at each channel can be read while actively counting in the following manner: first, the control byte shown in figures 3-7, 3-8, and 3-9, is written to the control register (FE7H) in order to latch the current value. Then the counter is read by addressing it as shown in table 3-3.

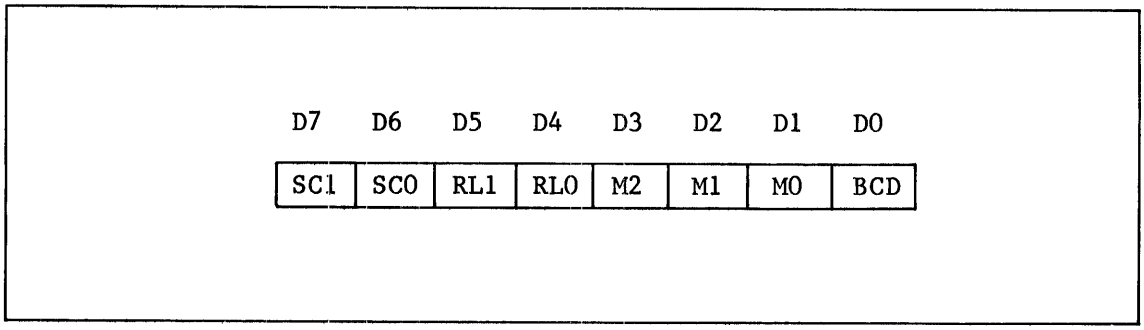



Figure 3-7. Am8253 Control Byte Format

TABLE 3-3. Am8253 ADDRESSES

| Counter | Status | Control |
|---|---|---|
| 0 | FE4 | FE7 |
| 1 | FE5 | FE7 |
| 2 | FE6 | FE7 |

| SC1 | SC0 | |
|---|---|---|
| 0 | 0 | Select Counter 0 |
| 0 | 1 | Select Counter 1 |
| 1 | 0 | Select Counter 2 |
| 1 | 1 | Illegal |

| RL1 | RL0 | |
|---|---|---|
| 0 | 0 | Counter Latching operation |
| 1 | 0 | Read/Load most significant byte only. |
| 0 | 1 | Read/Load least significant byte only. |
| 1 | 1 | Read/Load least significant byte first, than most significant byte. |

| M2 | M1 | M0 | |
|---|---|---|---|
| 0 | 0 | 0 | Mode 0 |
| 0 | 0 | 1 | Mode 1 |
| X | 1 | 0 | Mode 2 |
| X | 1 | 1 | Mode 3 |
| 1 | 0 | 0 | Mode 4 |
| 1 | 0 | 1 | Mode 5 |

| | |
|---|---|
| 0 | Binary Counter 16-bits |
| 1 | Binary Coded Decimal (BCD) Counter (4 Decades) |

Figure 3-8. Am8253 Control Byte Definition

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| SC1 | SC0 | 0 | 0 | X | X | X | X |

Figure 3-9. Am8253 Control Byte to Latch Count

# CHAPTER 4

# AmZ8001 AND AmZ8002 MONITORS

## INTRODUCTION

The Monitors provide a basic program development tool for the Am96/4018 Evaluation Board. Two Monitors are available, one for the AmZ8001 segmented CPU, and one for the AmZ8002 non-segmented CPU. With the exception of the unique AmZ8001 modes, both monitors use the same command set. The Monitors provide the following capabilities:

- Memory inspection and modification.
- Display and alteration for all registers, program counter, and flag bits.
- Selection of System or Normal mode.
- Selection of segmented or non-segmented mode (AmZ8001).
- Trace facility and hardware or software breakpoint insertion.
- Saving and Loading programs using an 16-bit (System 8) Microcomputer Development System executing GHOST.

## INSTALLATION

The Am96/4018 Monitors are supplied on four 2716-type E-PROMs. Two devices are for odd addresses, the other two for even address. Table 4-1 lists the part numbers and board locations for installing the E-PROMs.

TABLE 4-1. E-PROM LOCATIONS

| MONITOR | PART NO. | BOARD LOCATION |
|---|---|---|
| AmZ8001 | 002510682-001 | U23(Even) |
| AmZ8001 | 002510682-002 | U20(Odd) |
| AmZ8001 | 002510682-003 | U17(Even) |
| AmZ8001 | 002510682-004 | U12(Odd) |
| AmZ8002 | 002510683-001 | U23(Even) |
| AmZ8002 | 002510683-002 | U20(Odd) |
| AmZ8002 | 002510683-003 | U17(Even) |
| AmZ8002 | 002510683-004 | U12(Odd) |

Interface with the Monitor is through a standard RS232C or 20mA current loop terminal connected to connector P6. The Keyboard/Display console designed for the Evaluation Board connects to connector P4. An

Development System, executing GHOST, can also be used through either connector P3 (parallel) or P5 (serial). Figure 4-1 is a memory map of the evaluation board.

When power is supplied to the Am96/4018, the Monitor initializes all ports, and outputs ENTER ANY KEY to all I/O ports. The message is repeated every five seconds until the Monitor receives a response through one of the I/O ports. The port that receives the first character becomes the control port. An identification message and prompt is then output to the control port:

```
AMZ800x EVALUATION BOARD MONITOR V1.0
->
```

Where AMZ800x can be either AMZ8002 or AMZ8001.

## MONITOR COMMANDS

Command mode directives are specified in the following manner:

- All commands are specified with alphabetic characters followed by arguments described in the following pages.
- The alphabetic command can be followed by a space or no space, and the arguments separated by commas only.
- Address parameters can be 1 to 4 hex digits for the AmZ8002 Monitor (FFFF hex), and 1 to 6 hex digits for the AmZ8001 Monitor (3FFFFF Hex). Data parameters for both Monitors can be 1 to 4 hex digits.
- Arguments enclosed in brackets [ ] are optional.
- A Carriage Return <CR> must end each command input line.

When a command is not recognized by the Monitor, or the command parameters do not conform to the required format, the Monitor responds with a question mark (?). Table 4-2 is a summary of the Monitor commands.

The backspace character <CTRL-H> can be used to erase the current character on a line. Escape <ESC> can be used to erase entire current command line. In all examples of Monitor command usage, the user input is underlined, to distinguish it from Monitor response.

## KEYBOARD/DISPLAY CONSOLE

Since the optional Keyboard/Display console has only a single 20-character display, certain of the commands produce a display different from that on a standard CRT console. In particular, the Help

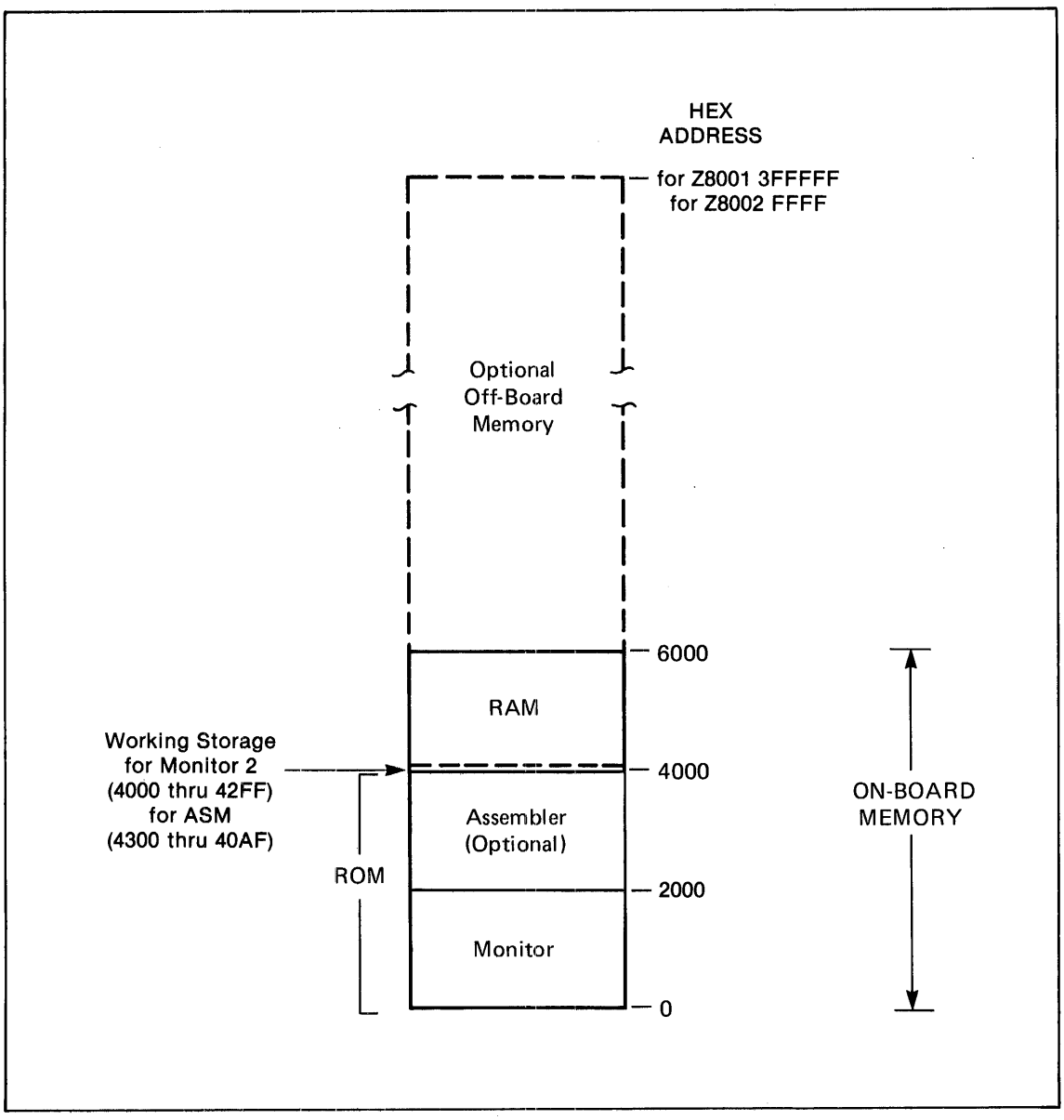



Figure 4-1. Memory Address

command does not output a list of available commands. Also, the Save commands cannot display the entire address parameters as they are typed in, but will function properly if given the required data. The X (Register display) command outputs program counter address, and next instruction data only. The XF command is two displays, where <CR> is used to move from display to display and back to command mode. To clear Monitor displays, enter a <CR>. Entry of an <ESC> character will clear the Line-by-Line Assembler display. Entry of an <ESC> character always cancels the current command line.

TABLE 4-2. MONITOR COMMAND SUMMARY

| COMMAND FORMAT | FUNCTION |
|---|---|
| ASM | Calls the Line-by-line Assembler |
| B | Displays and permits setting or clearing up to three software breakpoints. |
| D <start-addr>,<end-addr> | Displays the contents of memory from <start-addr> through <end-addr>. |
| E | Displays and permits setting one hardware breakpoint. |
| F <start-addr>,<end-addr>, <data> | Fills memory from <start-addr> through <end-addr> with 16-bit value of <data>. |
| G [<execute-addr>] [,<stop-addr>] | Execute program beginning at <execute-addr>. |
| H | Help. Displays a list of commands available with Monitor. |
| I [<port-addr>] | Input one word of data from I/O port at <port-addr>. |
| LOADP (n:)filename.typ | Load a file from host computer using the parallel I/O port. |
| LOADS (n:)filename.typ | Load a file from host computer using the serial I/O port. |
| M <start-addr>,<end-addr>, <new-addr> | Move contents of memory from <start-addr> through <end-addr>, to new location at <new-addr>. |
| NM | Set CPU Normal Mode. |
| O [<port-addr>],<data> | Output one word of <data> to I/O port at <port-addr>. |
| P | Displays and permits setting the user Program Counter. |
| R n (n=0 to 15 decimal) | Displays and permits altering the contents of registers 0 through 15. |
| SAVEP (n:)filename[.BIN] <start-addr>,<end-addr> [,<entry-addr>] | Save contents of memory from <start-addr> through <end-addr> as a file on host computer system, using the parallel I/O port. |

TABLE 4-2. MONITOR COMMAND SUMMARY (continued)

| COMMAND FORMAT | FUNCTION |
|---|---|
| SAVES (n:)filename[.BIN] <start-addr>,<end-addr> [,<entry-addr>] | Same as SAVEP, except uses the serial I/O port. |
| S <address> | Displays and permits alteration of data at <address>. |
| SM | Set CPU System Mode. |
| T n (n-1 to 65,535 decimal) | Trace mode. Permits single-step execution of instructions, with a register display for each instruction. |
| U n (n=1 to 65,535 decimal) | Untrace mode. Same as Trace, except n instructions are single-stepped before a register display. |
| X | Display of all CPU registers, Program Counter, Next Instruction, and flag bits. |
| XF | Displays Flag and Control Word bits. |
| C=1 or 0 | Set/reset Carry flag. |
| Z=1 or 0 | Set/reset Zero flag. |
| S=1 or 0 | Set/reset Sign flag. |
| P=1 or 0 | Set/reset Parity/Overflow flag. |
| D=1 or 0 | Set/reset Decimal Adjust flag. |
| H=1 or 0 | Set/reset Half-carry flag. |
| SEG=1 or 0 | Set/reset Z8001 segmented addressing mode. |
| EPE=1 or 0 | Set/reset EPE bit. |
| VI=1 or 0 | Set/reset VI bit. |
| NVI=1 or 0 | Set/reset NVI bit. |
| Carriage Return <CR> | Used to terminate a command line. Can also be used for single-step execution of a program. |
| Escape <ESC> | Cancels current command line, and issues new prompt. |

## LINE-BY-LINE ASSEMBLER

Format: ASM

Calls the optional Line-by-line Assembler. Refer to Chapter 5 for details of assembler functions. The assembler generates only non-segmented code. If programs are to be executed on the AmZ8001, the non-segmented mode must be set. If the assembler is not installed, the Monitor will respond with an error (?) display, and return to the Command mode.

## BREAKPOINTS

Format: B

Permits display and alteration of each of three software breakpoints, beginning with BKPT1. Breakpoint addresses are set by entering 1 to 4 hex digits (6 for AmZ8001). If a breakpoint is not set, it will be indicated by asterisks. To clear a previously set breakpoint, enter an asterisk and <CR>. Entry of <CR> only, will advance to the next breakpoint, up to BKPT3. After BKPT3, a <CR> will exit to the Monitor Command mode. Entry of a period (.) and <CR> will also exit to the Command mode. The least-significant bit of each address is cleared to assure operation on word boundaries. Upon initialization, the Monitor clears all breakpoints. Also, if a breakpoint is not set at the first word of an instruction, the Monitor will not sense the presence of the breakpoint, resulting in unpredictable performance.

NOTE

> Software breakpoints cannot be set in ROM space 0 to 3FFF hex, any non-existing RAM space, or RAM space from 4000 to 42FF hex. An error (?) will be generated and no breakpoint set.

Examples:

| | |
|---|---|
| ->B<CR> | Enter breakpoint routine. |
| BKPT1=****-<CR> | BKPT1 not set. |
| BKPT2=4F00-<CR> | BKPT2 set at 4F00 hex. |
| BKPT3=****-CF20<CR> | BKPT3 not previously set, then set to |
| -> | CF20 hex. |

## DISPLAY MEMORY CONTENTS

Format: D <start-addr>,<end-addr>

Displays the contents of memory from <start-addr> through <end-addr>. If more than 4 hex digits (6 for AmZ8001) are entered, an error (?) will be generated, and the Monitor will exit to the Command mode. Each display line is address, 16 bytes of data and the ASCII equivalent for

printable characters. To temporarily halt the display, enter a Control S <CTRL-S>. Another <CTRL-S> resumes display. Entering on an Escape <ESC> character causes an exit to the Command mode. When using the Keyboard/Display console, use of the Substitute command is preferred over the Display command.

Example:

```
->D 5000,501F<CR>              Display memory contents from
                               5000 to 501F hex.

5000=0054 0045 0051 0054 0020 0050 0047 004D  .T.E.S.T. .P.G.M
5010=0000 0000 0000 0000 0000 0000 0000 0000  ................
->
```

## HARDWARE BREAKPOINT

Format: E

Permits a hardware breakpoint to be set. The hardware breakpoint can be set by entering a 1 to 4 digit hex address (6 for the AmZ8001). Entering an asterisk and <CR> clears the breakpoint. The least significant bit of the address parameter is cleared to assure operation on word boundaries.

NOTE

Hardware breakpoints can be set in ROM space, software breakpoints cannot. For the Z8001 Monitor, it is recommended that the hardware breakpoint be cleared when not in use. Hardware compares the low-order 16-bits, and software tests the segment address bits. A Non-Maskable Interrupt (NMI) is generated each time the low-order 16-bits match that portion of the breakpoint address. The NMI service routine then compares the segment address. If no match is found, the Monitor ignores NMI and returns to the user program. Clearing the hardware breakpoint, when not required, will prevent unneeded NMI interrupts.

Examples:

```
->E<CR>
BKPTE=******-112233<CR>        Set breakpoint at 112232 hex
->                             for the AmZ8001.

->E<CR>
BKPTE=1FFF-*<CR>               Clear breakpoint for AmZ8002.
->
```

## FILL MEMORY

Format: F <start-addr>,<end-addr>,<data>

Memory from <start-addr> through <end-addr> is be filled with <data>. The least significant bit of each address is be cleared to assure operation on word boundaries. The <end-addr> must be equal to or greater than <start-addr>, or an error (?) will be generated.

NOTE

Addresses from 0 to 3FFF hex and 4000 to 42FF hex are restricted. Any attempt to use Fill command will generate an error (?).

Example:

```
->F 5000,5FFF,55<CR>
->
```

Fill memory from 5000 to 5FFF hex with 16-bit data 0055 hex.

## HELP

Format: H

Entering an H and <CR> will cause a list of Monitor commands and functions to be displayed. The optional Keyboard/Display unit does not output a list of Commands.

Example:

```
->H<CR>
AMZ8002 MONITOR COMMANDS

LOAD  LOAD FILE FROM HOST
SAVE  SAVE FILE ON HOST
SM    SET SYSTEM MODE
NM    SET NORMAL MODE
B     SET SOFTWARE BREAKPOINT(S)
C     SET/RESET CARRY FLAG
D     SET/RESET DECIMAL ADJUST FLAG OR DISPLAY MEMORY
E     SET HARDWARE BREAKPOINT
F     FILL MEMORY
G     GO FROM PC (TO BREAKPOINT)
H     SET/RESET HALF CARRY FLAG OR HELP
I     INPUT FROM PORT
M     MOVE MEMORY
O     OUTPUT TO PORT
P     SET/RESET PARITY FLAG OR PC REGISTER
R     SET REGISTER CONTENT
S     SET/RESET SIGN FLAG OR SUBSTITUTE MEMORY
T     TRACE
U     UNTRACE
X     DISPLAY ALL REGISTERS OR FLAG CONTROL WORD (XF)
Z     SET/RESET ZERO FLAG
->
```

## GO TO USER PROGRAM

Format: G [<execute-addr>][,<stop-addr>]

Permits execution of a program in memory, beginning at <execute-addr>. The execute address is optional, and if omitted, execution begins at the current user Program Counter address. The <stop-addr> is an optional breakpoint address. When the <stop-addr> is encountered during program execution, execution stops, the stop breakpoint is reset, and a display of all CPU registers, next instruction in hex, program counter and flag bits is output to the control port. However, the software and hardware breakpoints are not affected. Encountering a hardware or software breakpoint during program execution, also resets the <stop-addr> breakpoint.

Example:

```
->G 4500<CR>
```

Execute a program beginning at address 4500 hex.

## INPUT FROM I/O PORT

Format: I [<port-addr>]

Permits reading data from an I/O port at an address specified by <port-addr>. The Monitor executes a CPU IN instruction to input 16-bit data. The <port-addr> value remains unchanged until changed by another I command. Entering I alone reads from the previously set address.

Example:

```
->I FFE8<CR>                    Read data from address FFE8 hex.
DATA=FF0D   PORT=FFE8
->I<CR>                         Read data from previously set address
DATA=FF0D   PORT=FFE8           FFE8 hex.
```

## LOAD FILE THROUGH PARALLEL I/O PORT

Format: LOADP [n:]filename[.typ]

Permits loading a file from a host computer to the Evaluation Board. The drive specification n: is optional, and if omitted, the command defaults to the current logged-in drive. The Load command must be entered as shown, with a space between LOADP and the rest of the command. The .typ must either be .BIN or .HEX to load a file from a Development system.

Examples:

```
->LOADP A:TEST<CR>          Loads file named TEST.BIN from drive A.
->
```

## LOAD FILE THROUGH SERIAL I/O PORT

Format: LOADS [n:]filename[.typ]

Same as LOADP command, except through the serial I/O port.

NOTE

When a local console is being used, and the communication link is broken, the Monitor waits for 90 seconds for a response from GHOST before timing out After time out occurs, the Monitor outputs an I/O ERROR message.

## MOVE MEMORY CONTENTS

Format: M <start-addr>,<end-addr>,<new-addr>

Permits moving a block of 16-bit words beginning at <start-addr> through <end-addr> to new location beginning at <new-addr>. As in the Display command, <end-addr> must be <start-addr>+2 or greater. The least significant bit of address parameters is cleared to assure operation on word boundaries. Otherwise an error (?) is generated, and no move will take place.

NOTE

Addresses 0 to 3FFF and 4000 to 42FF are restricted. Any attempt to use the Move command causes an error (?) to be generated.

Example:

```
->M 1000,1FFF,C000<CR>
->
```

Move contents of memory from 1000 to through 1FFF hex to new location beginning at C000 hex.

## SET NORMAL MODE

Format: NM

Set the CPU Normal Mode (no privileged instructions can be executed).

## OUTPUT DATA TO I/O PORT

Format: O [<port-addr>],<data>

Permits <data> to be written to an I/O port at an address specified by <port-addr>. The monitor executes a CPU OUT instruction to output 16-bit data. As in the I command, the port address remains unchanged until an new address is specified. Entry of O,<data> only will write to the previously set address.

Examples:

```
->O FFEC,41<CR>                 Output 16-bit word 0041 hex
DATA=0041        PORT=FFEC      as ASCII A to address FFEC hex.
->O,42<CR>                      Output 16-bit word 0042 hex
DATA=0042        PORT=FFEC      as ASCII B to previously set address.
->O FFE8,61<CR>                 Output 16-bit word 0061 hex
aDATA=0062 PORT=FFE8            as ASCII a to address FFE8 hex.
->062<CR>                       Output 16-bit word 0062 hex
bDATA=0062 PORT=FFE8            as ASCII b to previously set address.
```

## DISPLAY/SET PROGRAM COUNTER

Format: P

Displays the current user Program Counter (PC) address in hexadecimal, and waits for input. Entering a 1 to 4 hex digit address (6 for AmZ8001) and <CR> changes the PC address to the new value. The least-significant bit of the address parameter is cleared to assure operation on word boundaries. Entering more than four or six digits causes an error (?) output, and leave the PC address unchanged. Entering <CR> only leaves the PC unchanged, and return to the Command mode.

Examples:

```
->P<CR>
PC=4522-4500<CR>                Change PC address to 4500 hex.
->P<CR>
PC=4500                         Check that PC address was changed.
```

## DISPLAY/ALTER REGISTERS

Format: R n (n=0 to 15)

Permits displaying and altering each of the 16 CPU registers. In addition, the AmZ8001 R14′ and R15′ registers, and the AmZ8002 R15′ register can be modified, depending on CPU mode. After displaying the contents of the specified register, the Monitor waits for input. Entering 1 to 4 hex digits and <CR> changes the register contents, and

advance to the next higher register. The registers will wrap around to R0 from R15 (Normal Mode) or R15' (System Mode). Entering more than four characters generate an error (?), and repeat the current register display. Entering <CR> only advances to the next register. To exit to the Command mode, enter a period (.) and <CR>.

Examples:

| | |
|---|---|
| ->R14<CR> | Select register R14 |
| R14=01FF-200<CR> | R14 contents changed to 0200 hex. |
| R15=0000-<CR> | R15 no change. |
| R0 =FFFF-.<CR> | R0 no change. Exit to Command mode. |
| ->SM<CR> | Set System mode. |
| ->R15<CR> | Select register R15'(System mode stack pointer). |
| R15'=41F4-8000<CR> | R15' contents changed to 8000 hex. |
| R0= FFFF-.<CR> | R0 no change, exit to Command mode. |

## SUBSTITUTE MEMORY CONTENTS

Format: S <address>

Permits displaying and altering one word at the address specified. Entering 1 to 4 hex digits and <CR> changes the contents of the address displayed on the console. Entering more than four digits causes an error (?) output, and the current address and contents are re-displayed. A <CR> alone advances to the next word address. Entering a period (.) and <CR> exits to the Command mode.

Example:

| | |
|---|---|
| ->S 5000<CR> | Select memory address 5000 hex. |
| 5000=0555-05556<CR> | Too many digits entered. |
| ? | Error output. |
| 5000=0555-0556<CR> | Location 5000 contents changed to 0556 hex. |
| 5002=0557-.<CR> | No change at 5002 hex, exit to |
| -> | Command mode. |

## SAVE MEMORY CONTENTS THROUGH PARALLEL I/O PORT

Format: SAVEP [n:]filename[.BIN] <start-addr>,<end-addr>[,<entry-addr>]

Saves the contents of memory from <start-addr> through <end-addr> as a file on a host computer. The optional <entry-addr>, if omitted, becomes the same as <start-addr>. The <entry-addr> is used to set the Program Counter whenever the Load commands are used. Address parameters can be specified as byte (odd or even) addresses. A space must be included between SAVEP and either the optional drive specification n: or the filename. A space must also separate the

filename.BIN and address parameters. A comma must separate the two address parameters. When any filetype other than .BIN is specified, an error (?) is generated. When the drive specification is omitted, the current logged-in drive is used.

Example:

```
->SAVEP A:MOVEIT 4500,4600<CR>
->
```

Save contents of memory from 4500 through 4600 hex, using filename MOVEIT.BIN on drive

## SAVE MEMORY CONTENTS THROUGH SERIAL I/O PORT

Format: SAVES (n:)filename[.BIN] <start-addr>,<end-addr>[,<entry-addr>]

Same as SAVEP, except using the serial I/O port.

NOTE

When a local console is being used, and the communication link is broken, the Monitor waits for 90 seconds for a response from GHOST before timing out. After time out occurs, the Monitor outputs I/O ERROR message.

## SYSTEM MODE

Format: SM

Sets the CPU system mode. Upon initialization, the Evaluation Board is in the System mode.

## TRACE

Format: T n (n=1 to 65,535)

The Trace command permits execution of n instructions beginning at the current PC address. As each instruction executes, a display of all registers, program counter, next instruction, and flag bits is output to the console. The Trace command cannot cross segment boundaries (AmZ8001). When n instructions have executed, the Monitor exits to the Command mode. A <CTRL-S> halts a display in progress. Entering another <CTRL-S> resumes the display. An escape <ESC> character will force an exit to the Command mode. Entering a <CR> only executes one instruction (single-step) each time <CR> is pressed.

Example:

| | |
|---|---|
| ->T 10<CR> | Causes ten instructions to be executed beginning at the current PC address. |

## UNTRACE

Format: U n (n=1 to 65,535)

Similar to the Trace command, except that no register display occurs until n instructions have executed.

## REGISTER DISPLAY

Format: X

Causes a display of all registers, Program Counter, Flag bits, and the next instruction in hexadecimal, then returns to the Command mode. The optional Keyboard/Display console displays only the PC address and next instruction. A <CR> causes a return to the Command mode.

Example:

```
->X<CR>
PC=4500 NI=8D07 C=0 Z=1 S=0 P=0 D=0 H=0
00=0FFF 01=0000 02=0100 03=0200 04=0FFF 05=0FFD 06=0FFB 07=0FFA
08=0000 09=0000 10=0000 11=0000 12=0000 13=0000 14=0000 15=4174 15'=41F4
->
```

## DISPLAY FLAG AND CONTROL WORD

Format: XF

Displays contents of the Flag and Control Word. When using the optional Keyboard/Display console, the high half of the Flag and Control Word is displayed first. Entry of a <CR> will display the lower half of the FCW (status flags). Another <CR> is required to exit to the Command mode.

Example:

```
->XF<CR>
FLAG     S S E   N             P
CONTROL  E / P V V             / D
WORD     G N E I I       C Z S V A H
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        !0!1!0!0!0!     !0!0!0!0!0!0!   !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
->
```

## SET/RESET FLAG CONTROL WORDS BITS

Formats:

| | |
|---|---|
| C=1 or 0 | Set/reset Carry flag bit. |
| Z=1 or 0 | Set/reset Zero flag bit. |
| S=1 or 0 | Set/reset Sign flag bit. |
| P=1 or 0 | Set/reset Parity/Overflow flag bit. |
| D=1 or 0 | Set/reset Decimal Adjust flag bit. |
| H=1 or 0 | Set/reset Half-carry flag bit. |
| SEG=1 or 0 | Set/reset Z8001 segmented addressing mode. |
| EPE=1 or 0 | Set/reset EPM bit. |
| VI =1 or 0 | Set/reset Vectored Interrupt bit. |
| NVI=1 or 0 | Set/reset Non-Vectored Interrupt bit. |

Example:

```
->C=1<CR>

FLAG     S S E   N             P
CONTROL  E / P V V             / D
WORD     G N E I I       C Z S V A H
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        !0!1!0!0!0!     !1!0!0!0!0!0!   !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
->
```

## MONITOR CONSOLE I/O SUPPORT

Console I/O can be done either by user written I/O programs, through the CPU's I/O instructions in system mode, or through the Monitor. To use the Monitor for console I/O, use the following general steps:

1. Store a block of four 16-bit control words in memory.

2. Store the beginning address of this block in register R1. (Use register pair RR2 with the AmZ8001 Monitor.)

3. Execute the SC instruction using a value of zero. A value of 04 hex with SC will be interpreted as a program exit which will return control to the Monitor.

CAUTION

The Monitor uses SC with a value of FF hex to set Software Breakpoints.

The block of four 16-bit control words is ordered as shown in table 4-3. The function code is stored in the most significant byte of the first word; the least significant byte of the first word contains the response code after an I/O operation. Word two is used for the AmZ8001 Monitor. Words three and four provide the beginning address of data, and the number of bytes to be written or read.

Read Control Statement - Function Code 0 - This command is intended for use by the line-by-line assembler. It transfers the user call command line to the specified area in the same form as the Read Console (Function Code 1).

Read Console - Function Code 1 - Reads a single line from the console. The data length represents a maximum and is replaced by the actual length. A console message is always terminated by a carriage return. The response code is normally zero unless the message was truncated, in which case the response code is 1.

Write Console - Function Code 2 - Writes a single line to the console. Data length represents the write length. The system provides no automatic end of line, so any required carriage returns or line feed codes must be included in the message. The response code is zero.

TABLE 4-3. MONITOR CONSOLE I/O CONTROL BLOCK

| Word | Byte | Contents |
|---|---|---|
| 1 | High | Function Code<br>0 = Read Control statement<br>1 = Read from console<br>2 = Write to console |
| | Low | No entry. Response code is stored here after I/O. |
| 2 | High | 6-bit segment number. (Z8001 only) |
| | Low | Low byte must be zero. |
| 3 | High | Beginning address of data to be written or read. Data written must include any necessary carrige returns, line feeds, or other control characters. Data read will include them also. |
| 4 | High | Length in number of bytes. It indicates either the total number of bytes to be written, or the number of bytes to be read. The total number of bytes read up to and including the carriage return is stored here upon return to the user program. |

## GHOST - Am96/4018 HOST FACILITY

GHOST provides a means of communication between the Am96/4018 Evaluation Board Monitor and a Microcomputer Development System. The following is a description of GHOST on the Development System. The facility allows:

- User interaction directly with the Evaluation Board Monitor program using the terminal assigned to GHOST on the Development System.

- Loading of program or data files from the Development System to the Evaluation Board (Download mode).

- Saving of program or data files from the Evaluation Board to the Development System (Upsave mode).

Communication can occur through either a parallel or serial port. Various configurations of interface hardware in the development system are accommodated by optionally specifying devices or port addresses when GHOST is invoked.

The Transparent mode is to allow the use of the development system console as a remote console for the board Monitor. The Monitor also allows, however, the use of a local terminal as its console. In that case, the Transparent mode of GHOST is not utilized. Uploading and downloading can still be accomplished, however, regardless of the location of the Monitor console. Any reference to Monitor console in this document refers to either the local or remote console (depending on which has been selected.)

### TRANSPARENT MODE

When the development system and the one-board computer are connected properly (refer to Chapter 2), communication can begin by starting GHOST. The GHOST command (invocation) line can include any or none of the parameters. These optional parameters are used to specify either device names or port addresses to be used for the serial and parallel ports. The allowable parameters are P and S for parallel and serial ports respectively, and T or TP for serial or parallel printer respectively. Each parameter is followed by an equal sign (=) and a valid parameter value. The parameter values are:

| Parallel port | Serial port | |
|---|---|---|
| ULl (Port address E4 ) | S=TTY | (Port address DC |
| #xx | UCl | DC |
| | LPT | 40 |
| | PTR | 42 |
| | PTP | 42 |
| | URl | 44 |
| | UPl | 44 |
| | UR2 | 46 |
| where xx is a valid | UP2 | 46) |
| port address | #xx | |

When any parameters are omitted, the default assignments are P=#4C, S=TTY, and T=LPT.

Example:

A>GHOST P=ULl S=LPT TP=#E8

The program will begin execution in the Transparent mode. Resetting the Evaluation Board Monitor will cause the Monitor to look for the first key input from one of the four I/O ports, and request ENTER ANY KEY. When the input comes from the remote (system) console, the remote console becomes the Monitor console. When the key input comes from the local (Evaluation Board) port, the local terminal becomes the Monitor console.

When the remote terminal is selected as the Monitor console, all characters typed there are sent directly to the Monitor when GHOST is in the Transparent mode. Characters echoed by the Monitor are displayed on the remote terminal. When the LINE FEED key is pressed, GHOST issues the Command> prompt. All other characters are passed to the Monitor and echoed.

NOTE

When no parameters or both S and P parameters are supplied on the command line, GHOST defaults to using the parallel port for communication. When only one parameter is supplied, GHOST defaults to communicating on that port. This can be changed with the P and S commands which are described next.

When the Command> prompt appears, the next keystroke is directed to GHOST rather than being sent to the board Monitor. The user can enter one of five different commands: P, Q, T, -T, or S. After completion of the command, GHOST returns to Transparent mode. The commands are as follows:

P Use the parallel port to communicate with the board Monitor. This refers to the port selected with the P parameter when GHOST was started.

S Use the serial port for communication. (The one selected with the S command line parameter.)

T Use the printer to type all remote console displays from the board Monitor, when in transparent mode.

-T Cancel printer function.

Q Quit (return to AMDOS.)

## DOWNLOAD MODE

This mode allows the board Monitor to request that a data file on the development system disk be downloaded to the board's memory. It is activated by typing one of the following commands on the Monitor console:

```
LOADS filename.typ        (for serial port)
```
or
```
LOADP filename.typ        (for parallel port)
```

The filename.typ parameter must be any file of .BIN or .HEX type. If the named file can be opened successfully the download will run to completion. When the file does not exist or is of another type than the two mentioned above, an error display is generated and downloading. In either case, GHOST automatically returns to Transparent mode (which will be indicated by a prompt from the board Monitor.)

## UPSAVE MODE

This mode allows the Monitor to save a range of its memory in a development system disk file of type BIN. It is activated by typing one of the following commands on the Monitor console:

```
SAVES filename.BIN <start-addr>, <end-addr>   (for serial port)
```
or
```
SAVEP filename.BIN <start-addr>, <end-addr>   (for parallel port)
```

The filename parameter can be any valid file name. The BIN parameter is optional. The <start-addr>,<end-addr> parameters denote a hex address range to be saved. If the file can be successfully created the upsave operation will run to completion. If not, an error message will be displayed and the Upsave will not occur. In either case GHOST will return to Transparent mode automatically.

NOTE

When the BIN file supplied in the SAVES command already exits, GHOST renames it to <filename>.BKN and create a new file with a .BIN extension. When <filename>.BKN already exists, it is deleted first.

## GHOST ERROR MESSAGES

These messages are displayed at the development system console.

Illegal parameters - This means that a parameter type other than P, S, T, or -T was included in the command line.

Illegal device:ddd - An undefined device name (ddd) was included in a command line parameter. (e.g. P=UL4:)

Illegal port address:xx - A non-hexadecimal number (xx) was included in a command line parameter. (e.g. S=#G9)

Illegal command - Something other than P, S, or Q was typed in response to a Command> prompt.

Illegal function code - This means that a function code other than UPSAVE or DOWNLOAD was received in a message block from the Monitor. This error should not normally occur. Try restarting both the Monitor and GHOST. If this does not work there may be a hardware problem.

No response from Monitor - GHOST is expecting a character from the Monitor and the time-out period has elapsed. Check the cabling between the target and the host. Try restarting both the Monitor and GHOST.

Function terminated--noisy line - During a download operation, GHOST has received four NAKs to a data block and four NAKs to an abnormal termination block. Check the cabling.

Illegal file type - A file type other than .BIN in an UPSAVE or other than .HEX or . BIN in a DOWNLOAD was specified.

BDOS error on file read - A disk I/O error was detected by AMDOS. Check the integrity of the diskette.

Unexpected EOF - A physical EOF was detected in a download file

Bad BIN file format - An error was detected in the contents of a

Checksum error in HEX file at record <ddd> - the calculated checksum in a HEX file disagree with the checksum recorded for that record. The <ddd> indicates the decimal record number where the error occurred. The integrity of the download file should be suspected.

File renamed to <filename>.ERR - If an abnormal termination block shouldn't be used.

No response from printer - GHOST is attempting to write a character to the printer, and the time-out period (approximately 25 seconds) has elapsed. Check the cabling between the host and printer. Also check the printer port address and type (serial or parallel).

These messages are displayed at local Evaluation Board Monitor console:

I/O ERROR FUNCOD=FF - Data transfer complete; however, normal termination of communication was not satisfactory.

I/O ERROR FUNCOD=FE - Abnormal termination during data transfer.

I/O ERROR FUNCOD=FD - Abnormal termination of UPSAVE because Monitor was not successful at initial transfer of Filename to GHOST.

I/O ERROR FUNCOD=FC - Abnormal termination of DOWNLOAD because Monitor was not successful at initial transfer of Filename to GHOST.

I/O ERROR FUNCOD=FA - Abnormal termination during entry address transfer.

I/O ERROR FUNCOD=F0 - Abnormal termination of GHOST encounted file access error and could not recover.

# CHAPTER 5
# LINE-BY-LINE ASSEMBLER (ASM)

## ENVIRONMENT

The Assembler resides in two E-PROM circuits that must be inserted in the following sockets on the Am96/4018 Evaluation Board:

| Part No. | Socket |
|---|---|
| 002510684-001 | U7 (Even) |
| 002510684-002 | U5 (Odd) |

The Assembler requires service from the Evaluation Board E-PROM Monitor for execution. Optionally, the Monitor can handle I/O for the Assembler (see Chapter 4). The absolute code generated by the Assembler can be uploaded to a Development System for permanent Storage.

## FUNCTIONS

The Assembler provides the mechanism for entering symbolic programs into the RAM memory of the Evaluation Board. The Assembler translates mnemonic operation codes, symbolic labels, and symbolic or absolute (hex) operands directly into machine code in a single pass and enters them directly into memory.

This is a limited one-pass assembler which does not generate object code files or save source code after entry (hence, no listing files are generated by the Assembler). Nor are macros, modules, arithmetic or logical expressions or high-level constructs supported. It is, however, upward source-compatible with the MACRO8000 Assembler, which runs on the AmSYS 8 Development System, and the AmZ8000 instruction set supported by the MACRO8000 Assembler. In this regard, it provides an excellent alternative to the conventional method of entering evaluation programs in hex code.

The Assembler reads user-supplied symbolic assembly statements from the user's console and assembles each statement as received. Forward symbolic references are satisfied when the forward reference is defined. If the statement is in error, a diagnostic is immediately available, requiring re-entry of the corrected statement. The user's program image is created in memory, as is the symbol (label) table. If the two components exceed available memory, a diagnostic is provided. At the conclusion of assembly (END statement) a diagnostic is provided for each unsatisfied reference (label). Unsatisfied references can be satisfied by entering more code through the assembler Evaluation Board Monitor.

Because of space constraints, a sub-set of the AmZ8002 instructions are implemented. The additional instructions not implemented may be inserted in hex format through the use of byte or word Directive statements.

The assembler produces only nonsegmented 16-bit addresses for the AmZ8002 nonsegmented processor.

## ASM CALL

The Assembler is invoked by one of two forms of the following Evaluation Board Monitor command:

```
ASM
ASM xxxx
```

where xxxx is a four-digit hex address indicating the beginning address for storing the absolute code generated by the Assembler. If the address is not specified, the default is hex 42B0.

Example:

```
ASM 4500
```

Due to the display constraints on the optional Am96/4016-KBD Keyboard/ Display Console, a prompt is never displayed for inputs to the Assembler, even when a CRT is used as the system console.

## UP/DOWN LOADING

Absolute code generated by the Assembler can be uploaded to an AmSYS 8 Development System for permanent storage on diskette. It can also be subsequently downloaded back to Evaluation Board memory. The Evaluation Board Monitor's SAVE and LOAD commands are used for this. See chapter 4.

## OUTPUT

While the Assembler does not generate listing files, it will display assembled code for each line of input immediately after you press the carriage return key. On a CRT or printer console, this assembled out put line will appear just below your corresponding input statement. On an LED Keyboard/Display console, the line might be truncated on the right. The column format for the output is:

| | |
|---|---|
| Columns 1-4: | Memory location of assembled code |
| Columns 7-20: | Assembled code (hex) |
| Columns 21 on: | Input statement |

## SAMPLE PROGRAM

The sample program shown in Figure 5-1 takes data bytes from the message buffer, transfers them to the line buffer and makes a system call to the AmZ8000 Evaluation Board Monitor to display the contents of the line buffer at the console. Program execution is initiated from the monitor by the G <address> command, where <address>is the starting address of the assembled program.

## STATEMENTS

A statement must be a simple statement; compound statements are not al lowed. A simple statement has one opcode per line and is terminated with a semicolon. For example:

```
ADD  R4,1;
```

You must space over two (2) spaces before entering non-labeled statements.

## SPECIAL CHARACTERS

Certain special characters are used within statements to further de scribe the instruction or operand being entered. These characters are as follows:

| Name | Character | Meaning |
|---|---|---|
| Number symbol | # | Denotes a hex constant |
| Parentheses | () | Enclose a subscript index register of the form Rn, where n must be in the range 1-15. |
| Circumflex | ^ | Denotes an address constant if it precedes a label. If the circumflex follows a word register, it denotes an indirect address. If the circumflex follows a constant and is followed by a left parentheses, it denotes the base for an indexed address. If a circumflex alone follows a constant, it denotes a direct address (the contents at the address). |
| Single Quote | ' | Denotes an ASCII character string. The string must be enclosed by single quotes. The quote may occur within the string by its appearance twice in succession. |

```
A>B:GHOST
G-G-GHOST -- V1.0

The first character received by the evaluation board monitor
will establish which terminal (local or remote) is the
monitor console.  If you wish this terminal to be the (remote)
console, first reset the board monitor and then type any key.

The PARALLEL port is active

ENTER ANY KEY
AMZ8002 MONITOR V1.0 (96/4018)
->ASM
AMZ8002 ASM VER 1.0

  LD R8,^MSG; ADDR OF MESSAGE IN REG 8
45B0  2108000         LD R8,^MSG; ADDR OF MESSAGE IN REG 8

  LD R9,^LINE; ADDR OF LINE IN REG 9
45B4  21090000        LD R9,^LINE; ADDR OF LINE IN REG 9

  LD R7,64; NO. OF BYTES IN MSG BUFFER
45B8  21070040        LD R7,64; NO. OF BYTES IN MSG BUFFER

  LDIRB R9^,R8^,R7; MOVE MSG DATA TO LINE BUFFER
45BC  BA810790        LDIRB R9^,R8^,R7; MOVE MSG DATA TO LINE BUFFER

  LD R1,^CBLK; ADDR OF CALL BLOCK
45C0  21010000        LD R1,^CBLK; ADDR OF CALL BLOCK

  LD R2,^FILL; PUT ADDR OF FILL IN REG 2
45C4  21020000        LD R2,^FILL; PUT ADDR OF FILL IN REG 2

  LD R2^,^LINE; ADDR OF LINE IN FILL
45C8  0D2542B6        LD R2^,^LINE; ADDR OF LINE IN FILL

  SC 0; PRINT LINE - CALL TO MONITOR
45CC  7F00            SC 0; PRINT LINE - CALL TO MONITOR

  SC 4;
45CE  7F04            SC 4;

MSG:  WORD: #0D0A,#0D0A,#5448,#4953,#2049,#5320,#5448,#4520,#4E45;
45D0 0D0A0D0A544849MSG:  WORD: #0D0A,#0D0A,#5448,#4953,#2049,#5320,#5448,#4520,#4E45;

      WORD: #5720,#414D,#5A38,#3030,#3220,#0D0A,#4556;
45E2  5720414D5A3830      WORD: #5720,#414D,#5A38,#3030,#3220,#0D0A,#4556;

      WORD: #414C,#5541,#5449,#4F4E,#2042,#4F41,#5244;
45F0  414C554154494F      WORD: #414C,#5541,#5449,#4F4E,#2042,#4F41,#5244;

      WORD: #2041,#5353,#454D,#424C,#4552,#2021,#0D0A, #0D0A;
45FE  20415353454D42      WORD: #2041,#5353,#454D,#424C,#4552,#2021,#0D0A,#0D0A;

LINE: BYTE (64); INTERMEDIATE BUFFER
460E                LINE: BYTE (64); INTERMEDIATE BUFFER

CBLK: WORD: #0200,0; OUTPUT DATA
464E  02000000      CBLK: WORD: #0200,0; OUTPUT DATA

FILL: WORD: 0,64; FOR SYSTEM CALL
4652  00000040      FILL: WORD: 0,64; FOR SYSTEM CALL

  END.
4652                 END.

->G45B0


THIS IS THE NEW AMZ8002
EVALUATION BOARD ASSEMBLER!


->
```

**Figure 5-1. Sample Program**

| Name | Character | Meaning |
|---|---|---|
| Comma | , | Separates multiple operands |
| Space | | Element separator between label, operation code and operands |
| Plus or minus | + - | Unary operators that may optionally precede immediate addresses used as operand values. Counts such as re peat and shift do not permit unary operators. |
| Colon | : | Denotes end of label |
| Semi-colon | ; | Denotes end of statement |

## DELIMITERS

Within statements, the possible delimiters are blanks, commas, and parentheses.

# SYMBOLS

The basic classes of symbols are opcodes, labels, and symbolic con stants. Symbols can be as long as 6 characters. The upper-case characters A through Z and 0/ through 9 are legal in a symbol; lower-case alpha characters are not allowed. A symbol cannot start with a digit and it cannot have embedded spaces. For example, valid symbols are:

```
LOOP
LABEL5
```

# NUMERIC CONSTANTS

Numeric constants are represented internally as signed 32-bit con stants. The notation for different types of numeric constants is as follows:

| Form | Base | Example |
|---|---|---|
| nnnn | Decimal | 12 |
| #nnnn | Hexadecimal | #A5 |

## OPCODES

An opcode is one of the AmZ8000 instruction mnemonics and can follow a label after one or more spaces, or start a statement if preceded by two or more spaces.

## LABELS

A label is a symbol that is prefixed to a statement and followed by a colon. It must begin in columns 1 or 2, and be no longer than 6 characters in length with no embedded space. A label is not declared explicitly as a label; usage of the label serves to declare the label. For example, the label NBT1 is defined in the following statement:

```
NBT1:  LD  R10/,0/;
```

Therefore, a statement such as:

```
JR  ZR,NBT1;
```

will cause a jump to the statement labeled NBT1.

## ADDRESS CONSTANTS

A label can be used as an address constant when preceded by a circum flex (). The preceding circumflex is interpreted as meaning address of or pointer to. For example:

```
LD   R2,L1;
     (Load address of L1 into R2)
```

## ABSOLUTE ADDRESS CONSTANTS

Absolute addresses can be used as operands when expressed in the form:

```
#4000
```

as in:

```
 LD R1,#4000
```

which specifies the item at address 4000 hexadecimal.

## SYMBOLIC CONSTANTS

A symbolic constant is a symbol that represents a constant. Symbolic constants are declared by the CONST directive.

A symbolic constant must be defined before being referenced. For example:

```
CONST   LF = #0/A;
```

substitutes the value #0/A for all subsequent occurrences of the name LF.

## STRINGS

A string is defined to be zero or more characters delimited by apostro phes. Each character is represented in memory by its 8-bit ASCII code. The maximum number of characters in a string is 255. The apostrophe itself is represented within a string by a double apostrophe. If there are zero characters between apostrophes, then the string is empty. For example, valid strings are:

```
'ABC1234'
'12AO/'
'IT''S'
''
```

## EXPRESSIONS

Expressions can be used in directives and instructions. The simplest form of an expression is a numeric constant, such as 5. Expressions can be numeric constants or symbolic constants. Opcodes and arithmetic or logical operators cannot be used in expressions.

## DIRECTIVES

Directives are all the statements in a program that do not create exe cutable machine instructions; they control the operation of the As sembler, allocate storage, or associate symbolic names with constant values.

## END DIRECTIVE

The END directive is required at the end of each program. The END directive has the form:

```
END.
```

END is followed by a period. Note that the END directive is only used at the end of a module.

## BYTE DIRECTIVE

The BYTE directive reserves or defines one or more bytes; it can be preceded by a label. Without a label, The byte directive has one of two forms:

```
BYTE (n);

BYTE: exp ,... exp;
```

where:

| | |
|---|---|
| n | Is an expression for the number of bytes to be reserved. |
| exp | Is a numeric expression or string expression. One or more values can be specified, separated by commas. |

The first form of the BYTE directive reserves successive memory locations beginning with the current location counter. The second form reserves memory locations that are defined to contain the specified expression values.

A numeric expression evaluates to a single byte. A string expression evaluates to a sequence of bytes, one for each character. For example:

```
BYTE   (3);            Reserves 3 bytes
BYTE:  5,#A;           Defines  2 bytes with values of 5 and A
BYTE:  'STRING';       Defines  6 bytes with ASCII values of STRING
BYTE:  3,'AB',4;       Defines  4 bytes with values of 03, 41, 42
                       and 04
```

A symbolic constant can be used to define byte values. For example:

```
CONST  P = 5;
BYTE: P;               Defines 1 byte with value 5
```

## WORD DIRECTIVE

The WORD directive reserves or defines one or more 16-bit words. It can be preceded by a label. Without a label it has one of two forms:

```
WORD  (n);

WORD: exp ,... exp;
```

The WORD directive is similar to the BYTE directive, except that words, rather than bytes, are reserved or defined and only one word per expression (exp) can be reserved. For example:

```
CONST CRLF = #0/D0/A;          Declares a symbolic constant
                               named CRLF
WORD: 'VA';                    Defines  1 Word with
                               the values 'VA'
WORD  (3);                     Reserves 3 Words
WORD: CRLF;                    Defines  1 Word with
                               the value #0/D0/A
```

String expressions are left-justified and right-filled with blanks (ASCII 20). Constant expressions are right-justified and left-filled with zeros.

## LONG DIRECTIVE

The LONG directive reserves or defines one or more long words (or 32-bit word pairs). It can be preceded by a label. Without a label it has one of two forms:

```
LONG  (n);

LONG: exp ,... exp;
```

The LONG directive is similar to the BYTE directive, except that word pairs, rather than bytes, are reserved or defined and only one word pair per expression (exp) can be reserved. For example:

```
CONST CRLF = #0/D0/A;       Declares a symbolic constant
LONG: 'BR';               Defines 1 word pair
                          with the value 'BR'
                          named CRLF
LONG  (3);                Reserves 3 word pairs
LONG: CRLF, CRLF;         Defines 2 word pairs
                          with the value
                          '0/D0/A' and '0/D0/A'
```

## CONST DIRECTIVE

The CONST directive declares a name which is to be associated with a constant value. The CONST directive has the form:

```
CONST name = exp;
```

where:

name  Is the symbolic name. One or more names can only be declared as symbolic constants in separate declarations.

exp  Is a numeric or ACSII-string expression which evaluates to no more than one word in length.

When a symbolic constant is used, the expression is evaluated and the value of the expression is associated with the name. For example:

```
CONST  LINESZ = 80/;
.
..
LD R14,LINESZ;        Value of LINESZ is 80
```

Once defined, the value of a symbolic constant cannot be changed later in the program.

## QUIT DIRECTIVE

The QUIT directive unconditionally terminates assembly. It has the form:

```
QUIT;
```

This directive allows you to exit the Assembler without completing the Assembly process. A semicolon is required after the command.

## INSTRUCTIONS

The AmZ8000 instructions available in the Assembler constitute a true source-compatible subset of the full instruction set available in the MACRO8000 Assembler, which runs on AMC's AmSYS 8 Development System. Instead of an object file, the Evaluation Board Assembler produces ab solute code which is written directly into the memory. This means that there are no listings available from the assembler, since the source code is discarded immediately.

The full instruction set is described in detail in the AmZ8001/2 Processor Instruction Set book published by Advanced Micro Devices.

## OPCODES

An opcode is an instruction mnemonic. Each instruction requires a specific number of operands. Zero, one, two, or three operands are re quired, depending on the instruction. Each instruction has the general form:

```
opcode  operands;
```

There is not a one-to-one correspondence between opcode mnemonics and hex equivalents. The operands themselves define the exact operation indicated by the generic opcode.

## OPERANDS

An operand shown as src or dst is a source or destination value. An src or dst operand in an instruction must utilize one of the addressing modes listed for the instruction. The addressing modes are listed in table 5-1.

An operand shown as r is a word register. An rr operand is a register pair, and an rq operand is a register quadruple. See the R addressing mode in table 5-1.

An operand shown as im is an immediate operand. See the IM addressing mode in table 5-1.

An operand shown as ir is an indirect operand. See the IR addressing mode in table 5-1.

An operand shown as exp is a numeric expression. The simplest form of an expression is a constant.

An operand shown as cc is an AmZ8000 condition code. The condition codes are listed in table 5-2. The condition bits C, Z, S, and P/V are in the flag and control word of the program status registers.

## INSTRUCTION SUMMARY

The relevant AmZ8000 instructions are listed in alphabetic order within the following groups:

- Load and exchange instructions are in table 5-3.
- Arithmetic instructions are in table 5-4.
- Logical instructions are in table 5-5.
- Program control instructions are in table 5-6.
- Bit manipulation instructions are in table 5-7.
- Rotate and shift instructions are in table 5-8.
- Block transfer and string manipulation instructions are in table 5-9.
- Input/output instructions are in table 5-10.
- CPU control instructions are in table 5-11.
- Complete instruction is in table 5-12.

NOTE

Certain instructions are noted as privileged. A user running in normal mode on the AmZ8000 is prevented from executing privileged instructions.

TABLE 5-1. ADDRESSING MODES FOR SRC AND DST OPERANDS

| Mode | Assembly Notation | Examples |
|---|---|---|
| IM<br>Immediate | exp<br>(Numeric expression) | ADD R0/,5;<br>(Add 5 into register 0/) |
| R<br>Register | RLn<br>(Lower byte reg., n=0/ to 7) | ADDB RL0/,RL4;<br>(Add RL4 into RL0/) |
| | RHn<br>(Upper byte reg., n=0/ to 7) | ADDB RH0/,RL4;<br>(Add RL4 into RH0/) |
| | Rn<br>(Word reg., n=0/ to 15) | ADD R0/,R4;<br>(Add R4 into R0/) |
| | RRn<br>(Reg. pair, n=0/ by 2 to 14) | ADDL RR0/,RR4;<br>(Add RR4 into RR0/) |
| | RQn<br>(Reg. quad, n=0/, 4, 8, 12) | MULTL RQ0/,RR4;<br>(Multiply RR2 by RR4, result in RQ0/) |
| IR<br>Indirect<br>Register | Rn<br>(n = 1 to 15) | ADD R0/,R4;<br>(Add contents of word that R4 points to into R0/) |
| DA<br>Direct<br>Address | label<br>(Program label) | ADD R0/,FLAG;<br>(Add contents of FLAG into R0/. Equivalent to loading address of FLAG into Rx, then adding what Rx points to into R0/) LD R2, #4320; (Load contents at address 4320 into R2.) |
| X<br>Indexed | label(Rn)<br>(Program label, with offset contained in a register) | ADD R0/,FLAG(R4);<br>(Add contents of FLAG, offset by the contents contents of R4, into R0/) LD R2,#4320(R4);<br>(LD address 4320 offset by contents of R4, into R2.) |
| RA<br>Relative<br>Address | label<br>(Program label) | JR L1;<br>(Jump relative to L1) |

TABLE 5-1. ADDRESSING MODES FOR SRC AND DST OPERANDS (continued)

| Mode | Assembly Notation | Examples |
|---|---|---|
| PA<br>Port<br>Address | exp<br>(Numeric expression that specifies a 16-bit port adress) | OUT #FFC0/, R4;<br>(Output contents of R4 to port #FFC0/) |
| PR<br>Port<br>Register | Rn<br>(Word register that contains a 16-bit port address) | OUT R2,R4;<br>(Output contents of R4 to port specified by R2) |

TABLE 5-2. CONDITION CODES

| Code | Meaning | If used, test for |
|---|---|---|
| NZ | Not Zero | Z = 0/ |
| ZR | Zero | Z = 1 |
| NC | No Carry | C = 0/ |
| CY | Carry | C = 1 |
| PO | Parity odd | P/V = 0/ |
| PE | Parity even | P/V = 1 |
| PL | Plus | S = 0/ |
| MI | Minus | S = 1 |
| NE | Not equal | Z = 0/ |
| EQ | Equal | Z = 1 |
| NOV | Overflow is reset | P/V = 0/ |
| OV | Overflow is set | P/V = 1 |
| GE | Greater than or equal | (S XOR P/V) = 0/ |
| LT | Less than | (S XOR P/V) = 1 |
| GT | Greater than | (Z OR (S XOR P/V)) = 0/ |
| LE | Less than or equal | (Z OR (S XOR P/V)) = 1 |
| LGE | Logical greater than or equal | C = 0/ |
| LLT | Logical less than | C = 1 |
| LGT | Logical greater than | ((C = 0/) AND (Z = 0/)) = 1 |
| LLE | Logical less than or equal | (C or Z) = 1 |

TABLE 5-3. LOAD AND EXCHANGE INSTRUCTIONS

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| CLRB<br>CLR | dst | R,IR,DA,X | Clear<br>dst <== 0/ |
| EXB<br>EX | r,src | R,IR,DA,X | Exchange<br>r <====> src |
| LDB<br>LD | r,src | IM,R,IR,DA,X | Load<br>r <== src LDL |
| LDB<br>LD<br>LDL | dst,r | IR,DA,X | Load to Memory<br>dst <== r |
| LDB<br>LD | dst,im | IR,DA,X | Load to Memory Immediate<br>dst <== im |
| LD | r,src | DA,X | Load Address<br>r <== src<br>(src means address of source) |
| LDM | r,src,exp | IR,DA,X | Load Multiple<br>r <== src<br>(starting at r and src, load exp consecutive words; exp is 1 to 16) |
| LDM | dst,r,exp | IR,DA,X | Load Multiple to Memory<br>dst <== r<br>(starting at dst and r, load exp consecutive words; exp is 1 to 16) |
| LDRB<br>LDR | r,src | RA | Load Relative<br>r <== src |
| LDRB<br>LDR | dst,r | RA | Load Relative to Memory<br>dst <== r |

TABLE 5-3. LOAD AND EXCHANGE INSTRUCTIONS (continued)

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| POP<br>POPL | dst,ir | R,IR,DA,X | Pop<br>dst <== ir<br>(autoincrement contents of register after pop) |
| PUSH<br>PUSHL | ir,src | IM,R,IR,DA,X | Push<br>ir <== src<br>(autodecrement contents of register before push) |

TABLE 5-4. ARITHMETIC INSTRUCTIONS

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| ADCB<br>ADC | r,src | R | Add with Carry<br>r <== r + src + carry |
| ADDB<br>ADD<br>ADDL | r,src | IM,R,IR,DA,X | Add<br>r <== r + src, set carry |
| CPB<br>CP<br>CPL | r,src | IM,R,IR,DA,X | Compare<br>r - src<br>(affects flags) |
| CPB<br>CP | dst,im | IR,DA,X | Compare Memory with Immediate<br>dst - im<br>(affects flags) |
| DAB | r | -- | Decimal Adjust<br>(decimal adjust of r) |
| DECB<br>DEC | dst,exp | R,IR,DA,X | Decrement<br>dst <== dst - exp<br>(exp is 1 to 16) |

TABLE 5-4. ARITHMETIC INSTRUCTIONS (continued)

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| DIV<br>DIVL | rr,src<br>rq,src | IM,R,IR,DA,X | Signed Divide<br>rrn+1 <== rrn,n+1<br>/ src rrn<br><== remainder (register pair) rqn+2,n+3<br><== rqn,n+1,n+2,n+3<br>/ src rqn,n+1<br><== remainder (register quad) |
| EXTSB<br>EXTS<br>EXTSL | dst | R | Extend Sign<br>(extend sign of dstlow to dsthigh) |
| INCB<br>INC | dst,exp | R,IR,DA,X | Increment<br>dst <== dst + exp<br>(exp is 1 to 16) |
| MULT<br>MULTL | rr,src<br>rq,src | IM,R,IR,DA,X | Signed Multiply<br>rrn,n+1 <== rrn+1<br>* src (register pair) rqn,n+1,n+2,n+3<br><== rqn+2,n+3 * src<br>(register quad) |
| NEGB<br>NEG | dst | R,IR,DA,X | Negate (two's complement)<br>dst <== 0/ - dst |
| SBCB<br>SBC | r,src | R | Subtract with Carry |

TABLE 5-5. LOGICAL INSTRUCTIONS

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| ANDB<br>AND | r,src | IM,R,IR,DA,X | AND<br>r <== r AND src |
| COMB<br>COM | dst | R,IR,DA,X | Complement<br>dst <== NOT dst |
| ORB<br>OR | r,src | IM,R,IR,DA,X | OR<br>r <== r OR src |
| TESTB<br>TEST | dst | R,IR,DA,X | Test<br>dst OR 0/ TESTL |
| TCCB<br>TCC | cc,r | R | Test Condition Code<br>If cc is true:<br>rlsb <== 1<br>otherwise: rlsb<br><== 0/ (lsb is least significant bit) |
| XORB<br>XOR | r,src | IM,R,IR,DA,X | Exclusive OR<br>R <== R XOR src |

TABLE 5-6. PROGRAM CONTROL INSTRUCTIONS

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| CALL | dst | IR,DA,X | Call Subroutine<br>Autodecrement SP<br>SP <== PC<br>PC <== dst |
| CALR | dst | RA | Call Relative<br>Autodecrement SP<br>SP <== PC<br>PC <== PC + dst<br>(dst is -4092 to +4098) |
| DBJNZ | r,dst | RA | Decrement and Jump if Nonzero<br>R <== R - 1<br>If R =/ 0/: PC <== PC<br>+ dst (dst is -252 to +2;<br>flags are not affected) |
| IRET | - | - | *Interrupt return<br>PS <== SP<br>Autoincrement SP |
| JP | cc,dst | IR,DA,X | Jump<br>If cc is true: PC <== dst<br>(cc is optional) |
| JR | cc,dst | RA | Jump Relative<br>If cc is true: PC <== PC<br>+ dst (cc is optional; dst<br>is -254 to +256) |
| RET | cc | - | Return Conditional<br>If cc is true: PC <== SP<br>Autoincrement SP |
| SC | exp | - | System Call<br>Autodecrement SP<br>SP <== old PS<br>Push instruction<br>PS <== system call PS<br>(exp is 0/ to 255; if<br>exp is 0/, the Evaluation<br>Board Monitor will<br>perform an I/O operation) |
| *Privileged instruction | | | |

TABLE 5-7. BIT MANIPULATION INSTRUCTIONS

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| BITB<br>BIT | dst,exp | R,IR,DA,X | Test Bit Static<br>Z flag <== NOT dstexp |
| BITB<br>BIT | dst,r | R | Test Bit Dynamic<br>Z flag <== NOT dstr |
| RESB<br>RES | dst,exp | R,IR,DA,X | Reset Bit Static<br>dstexp <== 0/ |
| RESB<br>RES | dst,r | R | Reset Bit Dynamic<br>dstr <== 0/ |
| SETB<br>SET | dst,exp | R,IR,DA,X | Set Bit Static<br>dstexp <== 1 |
| SETB<br>SET | dst,r | R | Set Bit Dynamic<br>dstr <== 1 |
| TSETB<br>TSET | dst | R,IR,DA,X | Test and Set<br>S flag <== dstmsb<br>(all bits in dst are set to 1; msb is most significant bit) |

TABLE 5-8. ROTATE AND SHIFT INSTRUCTIONS

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| RLDB | r,src | R | Rotate Digit Left (4-bit digit) |
| RRDB | r,src | R | Rotate Digit Right (4-bit digit) |
| RLB<br>RL | dst,exp | R | Rotate Left (rotate dst left; exp is 1 or 2, default 1) |
| RLCB<br>RLC | dst,exp | R | Rotate Left through Carry (rotate dst left; exp is 1 or 2, default 1) |
| RRB<br>RR | dst,exp | R | Rotate Right (rotate dst right; exp is 1 or 2, default 1) |
| RRCB<br>RRC | dst,exp | R | Rotate Right through Carry (rotate dst right; exp is 1 or 2, default 1) |
| SDAB<br>SDA<br>SDAL | dst,r | R | Shift Dynamic Arithmetic (shift dst left or right by r bits; positive left, negative right) |
| SDLB<br>SDL<br>SDLL | dst,r | R | Shift Dynamic Logical (shift dst left or right by r bits; positive left, negative right) |
| SLAB<br>SLA<br>SLAL | dst,exp | R | Shift Left Arithmetic (shift dst left by exp bits) |
| SLLB<br>SLL<br>SLLL | dst,exp | R | Shift Left Logical (shift dst left by exp bits) |
| SRAB<br>SRA<br>SRAL | dst,exp | R | Shift Right Arithmetic (shift dst right by exp bits) |
| SRLB<br>SRL<br>SRLL | dst,exp | R | Shift Right Logical (shift dst right by exp bits) |

TABLE 5-9. BLOCK TRANSFER AND STRING MANIPULATION INSTRUCTIONS

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| CPDB<br>CPD | rl,src,r2,cc | IR | Compare and Decrement<br>rl - src<br>Autodecrement src<br>r2 <== r2 - 1 |
| CPDRB<br>CPDR | rl,src,r2,cc | IR | Compare, Decrement and Repeat<br>rl - src<br>Autodecrement src<br>r2 <== r2 - 1<br>(repeat until cc is true<br>or r2 = 0/) |
| CPIB<br>CPI | rl,src,r2,cc | IR | Compare and Increment<br>rl - src<br>Autoincrement src<br>r2 <== r2 - 1 |
| CPIRB<br>CPIR | rl,src,r2,cc | IR | Compare, Increment and Repeat<br>rl - src<br>Autoincrement src<br>r2 <== r2 - 1<br>(repeat until cc is true<br>or r2 = 0/) |
| CPSDB<br>CPSD | dst,src,r,cc | IR | Compare String and Decrement<br>dst - src<br>Autodecrement dst and src<br>r <== r - 1 |
| CPSDRB<br>CPSDR | dst,src,r,cc | IR | Compare String, Decrement<br>and Repeat<br>dst - src<br>Autodecrement dst and src<br>r <== r - 1<br>(repeat until cc is true<br>or r = 0/) |
| CPSIB<br>CPSI | dst,src,r,cc | IR | Compare String and Increment<br>dst - src<br>Autoincrement dst and src<br>r <== r - 1 |
| CPSIRB<br>CPSIR | dst,src,r,cc | IR | Compare String, Increment<br>and Repeat<br>dst - src<br>Autoincrement dst and src<br>r <== r - 1<br>(repeat until cc is true<br>or r = 0/) |

TABLE 5-9. BLOCK TRANSFER AND STRING MANIPULATION INSTRUCTIONS (continued)

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| LDDB<br>LDD | dst,src,r | IR | Load and Decrement<br>dst <== src<br>Autodecrement dst and src<br>r <== r - 1 |
| LDDRB<br>LDDR | dst,src,r | IR | Load, Decrement and Repeat<br>dst <== src<br>Autodecrement dst and src<br>r <== r - 1<br>(repeat until r = 0/) |
| LDIB<br>LDI | dst,scr,r | IR | Load and Increment<br>dst <== src<br>Autoincrement dst and src<br>r <== r - 1 |
| LDIRB<br>LDIR | dst,src,r | IR | Load, Increment and Repeat<br>dst <== src<br>Autoincrement dst and src<br>r <== r - 1<br>(repeat until r = 0/) |
| TRDB | dst,src,r | IR | Translate and Decrement<br>dst <== src(dst)<br>Autoincrement dst<br>r <== r - 1 |
| TRDRB | dst,src,r | IR | Translate, Decrement and<br>Repeat dst <== src(dst)<br>Autodecrement dst<br>r <== r - 1<br>(repeat until r = 0/) |
| TRIB | dst,src,r | IR | Translate and Increment<br>dst <== src(dst)<br>Autoincrement dst<br>r <== r - 1 |
| TRIRB | dst,src,r | IR | Translate, Increment and<br>Repeat dst <== src(dst)<br>Autoincrement dst<br>r <== r - 1<br>(repeat until r = 0/) |

TABLE 5-9. BLOCK TRANSFER AND STRING MANIPULATION INSTRUCTIONS (continued)

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| TRTDB | src1,src2,r | IR | Translate and Test, Decrement RH1 <== src2(src1) Autodecrement src1 r <== r - 1 |
| TRTDRB | src1,src2,r | IR | Translate and Test, Decrement and Repeat RH1 <== src2(src1) Autodecrement src1 r <== r - 1 (repeat until r = 0/ or RH1 = 0/) |
| TRTIB | src1,src2,r | IR | Translate and Test, Increment RH1 <== src2(src1) Autoincrement src1 r <== r - 1 |
| TRTIRB | src1,src2,r | IR | Translate and Test, Increment and Repeat RH1 <== src2(src1) Autoincrement src1 r <== r - 1 (repeat until r = 0/ or RH1 = 0/) |

TABLE 5-10. INPUT/OUTPUT INSTRUCTIONS

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| INB<br>IN | r,src | PA,PR | *Input<br>r <== src<br>(DA src indicates 16-bit port address) |
| INDB<br>IND | dst,pr,r | IR | *Input and Decrement<br>dst <== pr<br>Autodecrement dst<br>r <== r - 1 |
| INDRB<br>INDR | dst,pr,r | IR | *Input, Decrement and Repeat<br>dst <== pr<br>Autodecrement dst<br>r <== r - 1<br>(repeat until r = 0/) |
| INIB<br>INI | dst,pr,r | IR | *Input and Increment<br>dst <== pr<br>Autoincrement dst<br>r <== r - 1 |
| INIRB<br>INIR | dst,pr,r | IR | *Input, Increment and Repeat<br>dst <== pr<br>Autoincrement dst<br>r <== r - 1<br>(repeat until r = 0/) |
| OUTB<br>OUT | dst,r | PA,PR | *Output<br>dst <== r<br>(DA dst indicates 16-bit port address) |
| OUTDB<br>OUTD | pr,src,r | IR | *Output and Decrement<br>pr <== src<br>Autodecrement src<br>r <== r - 1 |
| OTDRB<br>OTDR | pr,src,r | IR | *Output, Decrement and Repeat<br>pr <== src<br>Autodecrement src<br>r <== r - 1<br>(repeat until r = 0/) |
| OUTIB<br>OUTI | pr,src,r | IR | *Output and Increment<br>pr <== src<br>Autoincrement src<br>r <== r - 1 |
| *Privileged instruction | | | |

TABLE 5-10. INPUT/OUTPUT INSTRUCTIONS (continued)

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| OTIRB | pr,src,r | IR | *Output, Increment and OTIR Repeat<br>pr <== src<br>Autoincrement src<br>r <== r - 1<br>(repeat until r = 0/) |
| SINDB<br>SIND | dst,pr,r | IR | *Special Input and Decrement<br>dst <== pr<br>Autodecrement dst<br>r <== r - 1 |
| SINDRB<br>SINDR | dst,pr,r | IR | *Special Input, Decrement and Repeat<br>dst <== pr<br>Autodecrement dst<br>r <== r - 1<br>(repeat until r = 0/) |
| SINIB<br>SINI | dst,pr,r | IR | *Special Input and Increment<br>dst <== pr<br>Autoincrement dst<br>r <== r - 1 |
| SINIRB<br>SINIR | dst,pr,r | IR | *Special Input, Increment and Repeat<br>dst <== pr<br>Autoincrement dst<br>r <== r - 1<br>(repeat until r = 0/) |
| SOUTDB<br>SOUTD | pr,src,r | IR | *Special Output and Decrement<br>pr <== src<br>Autodecrement src<br>r <== r - 1 |
| SOTDRB<br>SOTDR | pr,src,r | IR | *Special Output, Decrement and Repeat<br>pr <== src<br>Autodecrement src<br>r <== r - 1<br>(repeat until r = 0/) |
| SOUTIB<br>SOUTI | pr,src,r | IR | *Special Output and Increment<br>pr <== src<br>Autoincrement src<br>r <== r - 1 |
| * Privileged instruction | | | |

TABLE 5-10. INPUT/OUTPUT INSTRUCTIONS (continued)

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| SOTIRB<br>SOTIR | pr,src,r | IR | *Special Output, Increment and Repeat<br>pr <== src<br>Autoincrement src<br>r <== r - 1<br>(repeat until r = 0/) |
| *Privileged instruction | | | |

TABLE 5-11. CPU CONTROL INSTRUCTIONS

| Opcode | Operands | Addressing Modes for src or dst | Description |
|---|---|---|---|
| COMFLG | flags | - | Complement Flags (flags are CY, ZR, SGN, PY, OV) |
| DI | ints | - | *Disable Interrupt (interrupts are NVI and VI) |
| EI | ints | - | *Enable Interrupt (interrupts are NVI and VI) |
| HALT | - | - | *Halt |
| LDCTLB | FLAGS,src | R | *Load Flag Byte<br>FLAGS <== src |
| LDCTLB | dst,FLAGS | R | *Load from Flag Byte<br>dst <== FLAGS |
| NOP | - | - | No Operation |
| RESFLG | flags | - | Reset Flags (flags are CY, ZR, SGN, PY, OV) |
| SETFLG | flags | - | Set Flags (flags are CY, ZR, SGN, PY, OV) |
| *Privileged instruction | | | |

## ALPHABETICAL LISTING OF INSTRUCTIONS

The following table lists permissible AmZ8000 instructions in alphabetical order. Under addressing modes, value indicates that the operand is immediate. Values in parentheses are hexadecimal equivalents for operation codes.

TABLE 5-12. ALPHABETICAL LISTING

| Mnemonic | Address Modes | Description |
|---|---|---|
| ADC | R(B5) | Add word with carry |
| ADCB | R(B4) | Add byte with carry |
| ADD | R(81), IM(01), IR(01), DA(41), X(41) | Add Word |
| ADDB | R(80), IM(00), IR(00), DA(40), X(40) | Add byte |
| ADDL | R(96), IM(16), IR(16), DA(56), X(56) | Add long word |
| AND | R(87), IM(07), IR(07), DA(47), X(47) | Logical AND word |
| ANDB | R(86), IM(06), IR(06), DA(46), X(46) | Logical AND byte |
| BIT | DY(27), R(A7), IR(27), DA(67), X(67) | Test word bit |
| BITB | DY(26), R(A6), IR(26), DA(66), X(66) | Test byte bit |
| BYTE | | (See Directives) |
| CALL | IR(1F), DA(5F), X(5F) | Call subroutine |
| CALR | RA(D) | Call subroutine relative |
| CLR | R(8D), IR(0D), DA(4D), X(4D) | Clear word |
| *Privileged (system) instructions | | |

TABLE 5-12. ALPHABETICAL LISTING (continued)

| Mnemonic | Address Modes | Description |
|---|---|---|
| CLRB | R(8C), IR(0C), DA(4C), X(4C) | Clear byte |
| COM | R(8D), IR(0D), DA(4D), X(4D) | Complement word |
| COMB | R(8C), IR(0C), DA(4C), X(4C) | Complement byte |
| COMFLG | C, Z, S, P, V(8D) | Complement flags |
| CONST | | (See Directive) |
| CP | R(8B), IM(0B), IR(0B), DA(4B), X(4B), IM-IR(0D), IM-DA(4D), IM-X(4D) | Compare word |
| CPB | R(8A), IM(0A), IR(0A), DA(4A), X(4A), IM-IR(0C), IM-DA(4C), IM-X(4C) | Compare byte |
| CPD | IR(BB) | Compare word and decrement |
| CPDB | IR(BA) | Compare byte and decrement |
| CPDR | IR(BB) | Compare word decrement and repeat |
| CPDRB | IR(BA) | Compare byte decrement and repeat |
| CPI | IR(BB) | Compare word and increment |
| CPIB | IR(BA) | Compare byte and increment |
| CPIR | IR(BB) | Compare word increment repeat |
| CPIRB | IR(BA) | Compare byte increment repeat |
| CPL | R(90), IM(10), IR(10), DA(50), X(50) | Compare long word |
| CPSD | IR(BB) | Compare word string and decrement |

*Privileged (system) instructions

TABLE 5-12. ALPHABETICAL LISTING (continued)

| Mnemonic | Address Modes | Description |
|---|---|---|
| CPSDB | IR(BA) | Compare byte string and decrement |
| CPSDR | IR(BB) | Compare word string de crement and repeat |
| CPSDRB | IR(BA) | Compare byte string de crement and repeat |
| CPSI | IR(BB) | Compare word string and increment |
| CPSIB | IR(BA) | Compare byte string and increment |
| CPSIR | IR(BB) | Compare word string in crement and repeat |
| CPSIRB | IR(BA) | Compare byte string in crement and repeat |
| DAB | R(B0) | Decimal adjust |
| DBJNZ | RA(F) | Decrement byte and jump non-zero |
| DEC | R(AB), IR(2B), DA(6B), X(6B) | Decrement word |
| DECB | R(AA), IR(2A), DA(6A), X(6A) | Decrement byte |
| *DI | VI, NVI (7C) | Disable interrupts |
| DIV | R(9B), IM(1B), IR(1B), DA(5B), X(5B) | Divide word |
| DIVL | R(9A), IM(1A), IR(1A), DA(5A), X(5A) | Divide long word |
| *EI | VI,NVI(7C) | Enable interrupts |
| END | | (See Directive) |
| EX | R(AD), IR(2D), DA(6D), X(6D) | Exchange words |
| EXB | R(AC), IR(2C), DA(6C), X(6C) | Exchanges bytes |
| EXTS | R(B1) | Extend word sign |
| *Privileged (system) instructions | | |

TABLE 5-12. ALPHABETICAL LISTING (continued)

| Mnemonic | Address Modes | Description |
|---|---|---|
| EXTSB | R(B1) | Extend byte sign |
| EXTSL | R(B1) | Extent long word sign |
| *HALT | (7A) | Halt |
| *IN | PR(3D), PA(3B) | Input word |
| *INB | PR(3C), PA(3C) | Input byte |
| INC | R(A9), IR(29), DA(69), X(69) | Increment word |
| INCB | R(A8), IR(28), DA(68), X(68) | Increment byte |
| *IND | IR(3B) | Input word and decrement |
| *INDB | IR(3A) | Input byte and decrement |
| *INDR | IR(3B) | Input word decrement and repeat |
| *INDRB | IR(3A) | Input byte decrement and repeat |
| *INI | IR(3B) | Input word and increment |
| *INIB | IR(3A) | Input byte and increment |
| *INIR | IR(3B) | Input word increment and repeat |
| *INIRB | IR(3A) | Input byte increment and repeat |
| *IRET | (7B) | System call return |
| JP | IR(1E), DA(5E), X(5E) | Conditional jump |
| JR | RA(E) | Jump relative condition |
| LD | R(A1), IM(21), IR(21), DA(61), X(61) | Load word register |
| LD | IR(2F), DA(6F), X(6F), IM-IR(OD), IM-DA(4D), IM-X(4D) | Load word memory |

*Privileged (system) instructions

TABLE 5-12. ALPHABETICAL LISTING (continued)

| Mnemonic | Address Modes | Description |
|---|---|---|
| LDB | R(A0), IM(C), IR(20), DA(60), X(60) | Load byte register |
| LDB | IR(2E), DA(6E), X(6E), IM-IR(0C), IM-DA(4C), IM-X(4C) | Load byte memory |
| LDCTLB | R,FLAGS(8C), FLAGS,R(8C) | Load flag register |
| LDD | IR(BB) | Load word and decrement |
| LDDB | IR(BA) | Load byte and decrement |
| LDDR | IR(BB) | Load word decrement and repeat |
| LDDRB | IR(BA) | Load byte decrement and repeat |
| LDI | IR(BB) | Load word and increment |
| LDIB | IR(BA) | Load byte and increment |
| LDIR | IR(BB) | Load word increment and repeat |
| LDIRB | IR(BA) | Load byte increment and repeat |
| LDL | R(94), IM(14), IR(14), DA(54), X(54) | Load long word register |
| LDL | IR(1D), DA(5D), X(5D) | Load long word memory |
| LDM | IR(1C), DA(5C), X(5C) | Load multiple registers |
| LDM | IR(1C), DA(5C), X(5C) | Load multiple memory |
| LDR | RA-R(31), R-RA(33) | Load relative |
| LDRB | RA-R(30), R-RA(32) | Load relative byte |
| LONG | | (See Directive) |
| MULT | R(99), IM(19), IR(19), DA(59), X(59) | Multiply word |
| *Privileged (system) instructions | | |

TABLE 5-12. ALPHABETICAL LISTING (continued)

| Mnemonic | Address Modes | Description |
|---|---|---|
| MULTL | R(98), IM(18), IR(18), DA(58), X(58) | Multiply long word |
| NEG | R(8D), IR(0D), DA(4D), X(4D) | Negate word |
| NEGB | R(8C), IR(0C), DA(4C), X(4C) | Negate byte |
| NOP | (8D07) | No operation |
| OR | R(85), IM(05), IR(05), DA(45), X(45) | Logical OR word |
| ORB | R(84), IM(04), IR(04), DA(44), X(44) | Logical OR byte |
| OTDR | IR(3B) | Output word decrement and repeat |
| *OTDRB | IR(3A) | Output byte decrement and repeat |
| *OTIR | IR(3B) | Output word increment and repeat |
| *OTIRB | IR(3A) | Output byte increment and repeat |
| *OUT | PR(3F), PA(3B), | Output word |
| *OUTB | PR(3E), PA(3A), | Output byte |
| *OUTD | IR(3B) | Output word and decrement |
| *OUTDB | IR(3A) | Output byte and decrement |
| *OUTI | IR(3B) | Output word and increment |
| *OUTIB | IR(3A) | Output byte and increment |
| POP | R(97), IR(17), DA(57), X(57) | Pop stack word |
| POPL | R(95), IR(15), DA(55), X(55) | Pop stack long word |
| *Privileged (system) instructions | | |

TABLE 5-12. ALPHABETICAL LISTING (continued)

| Mnemonic | Address Modes | Description |
|---|---|---|
| PUSH | R(93), IM(OD), IR(13), DA(53), X(53) | Push stack word |
| PUSHL | R(91), IR(11), DA(51), X(51) | Push stack long word |
| QUIT | | (See Directives) |
| RES | DY(23), R(A3), IR(23), DA(63), X(63) | Reset word bit |
| RESFLG | C, Z, S, P, V(8D) | Reset flags |
| RET | (9E) | Conditional return |
| RL | Value (B3) | Rotate word left |
| RESB | DY(22), R(A2), IR(22), DA(62), X(62) | Reset byte bit |
| RLB | Value (B2) | Rotate byte left |
| RLC | Value (B3) | Rotate word left through carry |
| RLCB | Value (B2) | Rotate byte left through carry |
| RLDB | R(BE) | Rotate digit left |
| RR | Value (B3) | Rotate word right |
| RRB | Value (B2) | Rotate byte right |
| RRC | Value (B3) | Rotate word right through carry |
| RRCB | Value (B2) | Rotate byte right through carry |
| RRDB | R(BC) | Rotate digit right |
| SBC | R(B7) | Subtract word with carry |
| SBCB | R(B6) | Subtract byte with carry |
| SC | Value (7F) | System call |
| *Privileged (system) instructions | | |

TABLE 5-12. ALPHABETICAL LISTING (continued)

| Mnemonic | Address Modes | Description |
|---|---|---|
| SDA | R(B3) | Shift dynamic arithmetic word |
| SDAB | R(B2) | Shift dynamic arithmetic byte |
| SDAL | R(B3) | Shift dynamic arithmetic long word |
| SDL | R(B3) | Shift dynamic long word |
| SDLB | R(B2) | Shift dynamic long byte |
| SDLL | R(B3) | Shift dynamic logical long word |
| SET | DY(25), R(A5), IR(25), DA(65), X(65) | Set word bit |
| SETB | DY(24), R(A4), IR(24), DA(64), X(64) | Set byte bit |
| SETFLG | C, Z, S, P, V(8D) | Set flags |
| *SIND | IR(3B) | Special input |
| *SINDB | IR(3A) | Special input and decrement |
| *SINDR | IR(3B) | Special input, decrement and repeat |
| *SINDRB | IR(3A) | Special input, decrement and repeat |
| *SINI | IR(3B) | Special input and increment |
| *SINIB | IR(3A) | Special input and increment |
| *SINIR | IR(3B) | Special input, increment and repeat |
| *SINIRB | IR(3A) | Special input, increment and repeat |
| *Privileged (system) instructions | | |

TABLE 5-12. ALPHABETICAL LISTING (continued)

| Mnemonic | Address Modes | Description |
|---|---|---|
| SLA | Value (B3) | Shift left arithmetic word |
| SLAB | Value (B2) | Shift left arithmetic byte |
| SLAL | Value (B3) | Shift left arithmetic long word |
| SLL | Value (B3) | Shift left logical word |
| SLLB | Value (B2) | Shift left logical byte |
| SLLL | Value (B3) | Shift left logical long word |
| *SOTDR | IR(3B) | Special output, decrement and repeat |
| *SOTDRB | IR(3A) | Special output, decrement and repeat |
| *SOTIR | IR(3B) | Special output, increment and repeat |
| *SOTIRB | IR(3A) | Special output, increment and repeat |
| *SOUTD | IR(3B) | Special output and decrement |
| *SOUTDB | IR(3A) | Special output and decrement |
| *SOUTI | IR(3B) | Special output and increment |
| *SOUTIB | IR(3A) | Special output and increment |
| SRA | Value (B3) | Shift right arithmetic word |
| SRAB | Value (B2) | Shift right arithmetic byte |

*Privileged (system) instructions

TABLE 5-12. ALPHABETICAL LISTING (continued)

| Mnemonic | Address Modes | Description |
|---|---|---|
| SRAL | Value (B3) | Shift right arithmetic long word |
| SRL | Value (B3) | Shift right logical word |
| SRLB | Value (B2) | Shift right logical byte |
| SRLL | Value (B3) | Shift right logical long word |
| SUB | R(83), IM(03), IR(03), DA(43), X(43) | Subtract word |
| SUBB | R(82), IM(02), IR(02), DA(42), X(42) | Subtract byte |
| SUBL | R(92), IM(12), IR(12), DA(52), X(52) | Subtract long word |
| TCC | R(AF) | Test condition and set word |
| TCCB | R(AE) | Test condition and set byte |
| TEST | R(8D), IR(0D), DA(4D), X(4D) | Test word |
| TESTB | R(8C), IR(0C), DA(4C), X(4C) | Test byte |
| TESTL | R(9C), IR(1C), DA(5C), X(5C) | Test long word |
| TRDB | IR(B8) | Translate byte and decrement |
| TRDRB | IR(B8) | Translate byte decrement and repeat |
| TRIB | IR(B8) | Translate byte and increment |
| TRIRB | IR(B8) | Translate byte increment and repeat |
| TRTDB | IR(B8) | Translate test byte and decrement |
| *Privileged (system) instructions | | |

TABLE 5-12. ALPHABETICAL LISTING (continued)

| Mnemonic | Address Modes | Description |
|---|---|---|
| TRTDRB | IR(B8) | Translate test byte decrement and repeat |
| TRTIB | IR(B8) | Translate test byte and increment |
| TRTIRB | IR(B8) | Translate test byte increment and repeat |
| TSET | R(8D), IR(0D), DA(4D), X(4D) | Test word and set |
| TSETB | R(8C), IR(0C), DA(4C), X(4C) | Test byte and set |
| WORD | | (See Directives) |
| XOR | R(89), IM(09), IR(09), DA(49), X(49) | Exclusive OR word |
| XORB | R(88), IM(08), IR(08), DA(48), X(48) | Exclusive OR byte |
| *Privileged (system) instructions | | |

## ERROR MESSAGES

During the line by line assembly, statements are completely evaluated before any assembly occurs. If any error exists, a diagnostic is displayed and the assembler awaits re-entry of the instruction. The following codes are displayed:

NOTE

The error display on the keyboard/display can only be cleared with the ESCape key.

L - A syntax error occurred during label processing, a label is required, or one is present and is not permitted.

D - A duplicate label has been encountered.

O - A syntax error occurred during opcode processing, or an undefined operation code was encountered.

X - A system error occurred. This is the result of a software or hardware malfunction.

S - A syntax error occurred in statement processing.

U - An equivalence operand was not previously defined.

V - A memory overflow occurred. The program plus the number of labels exceeds machine capacity.

R - Occurs only after an END pseudo-operation and specifies an undefined label reference.

# CHAPTER 6
# PRINCIPLES OF OPERATION

## POWER-UP SEQUENCE

The Am96/4018 Evaluation Board requires three input voltages at the P1 edge connector. Their use is shown below:

+12V - RAM and serial I/O circuits

-12V - RS232C and TTY interfaces.
-5V for RAM is derived from this voltage.

+5V - TTL

The ground (GND) pins at P1, P2, P3 and P4 are tied together with the Signal Ground pins at P5 and P6. They require external ground at P1. The Chassis ground pins at P5 and P6 can be jumpered into this ground plane as shown in figure 2-4.

When the board is powered up, an RC network generates a reset to the CPU and to the programmable parallel and serial I/O devices. The CPU's flag and control word (FCW) is initialized, with the System/Normal flag set to System Mode. The Monitor program then begins execution.

The Monitor writes a value to the CPU's refresh counter that causes refresh at 30-microsecond intervals. The program status area pointer (interrupt vector table) is also written. Then, the Monitor writes control bytes to the Am8253 counter/timer, the two Am9551 serial I/O circuits, and the Am8255A parallel I/O circuit. It also samples a line to see if the optional keyboard/display console is present at P4 (when present, it receives all output).

The Monitor then sets the user's program counter to 1000 hex and sends a command prompt to either the keyboard/ display (if attached) or to the console at P6. If consoles are attached to both P5 and P6, without the keyboard/display at P4, the first console to respond becomes the system console and gets all output.

## CPU FUNCTIONS

The list of CPU pins below describes which pins are used for the internal functions of the Evaluation Board.

AS* - (output, three-state).
Used in the normal way to indicate valid addresses, and to demultiplex the address/data buses.

DS* - (output, three-state).
Used in the normal way to indicate valid data, and to demultiplex the address/data buses. Figure 6-1 shows how it is also used with R/W* and I/O* (a line decoded from the ST0-ST3 lines) to decode the IOR* and IOW* lines which gate the Am8255A, Am9551 and Am8253 circuits.

MREQ* - (output, three-state).
Used in the normal way to gate memory (not available on the P2 bus).

R/W* - (output, three-state).
Used with DS* and I/O*. See figure 6-1. RAM R/W*, which is forced high during memory refresh, is derived from this line.

N/S* - (output, three-state).
Not used. Trapping of System Mode instructions is done independently of this output, which is designed primarily to distinguish physical memory spaces for Normal and System Modes.

B/W* - (output, three-state).
Used in the normal way to distinguish byte or word memory references.

ST0-ST3 - (outputs, three-state).
Only three of the ten possible conditions meaningful for the AmZ8002 are used on-board:

1) I/O reference is used with R/W* and DS* as described under DS*

2) memory refresh becomes RFSH* and is used on the RAM

3) instruction fetch, first word, becomes IF1* and is used in the hardware breakpoint and single step functions.

(input).
WAIT* Used to generate Wait states between the T2 and T3 clock cycles of a memory cycle during memory references to address 0-5FFF hex.

STOP* (input).
Not used. It stops the CPU entirely, which prevents the Monitor program from executing.

BUSRQ* (input).
Not used on-board since only the CPU controls the system buses.

BUSAK* (output).
Not used. It is connected to disable P2 buffers if the BUSRQ* is used by an external device.

NMI* - (input).
Used by the break point, single-step and break switch functions to force control back to the Monitor program.

VI* - (input).
Not used.

NVI* - (input).
Not used.

RESET* - (input).
Connected to the RST* line on the board, which resets the CPU and all other programmable circuits during power-up.

## BUS STRUCTURE

The CPU's 16-bit multiplexed address/ data pins are demultiplexed into separate 16-bit data and address buses, as shown in figure 6-1. The address bus is latched and the data bus is buffered before use on the board. A transparent latch instead of a register is used on the buffered address bus so that addresses may become valid before the trailing edge of the address strobe (AS*).

The status lines and control strobes are also buffered to P2. All of these lines and buses float to the high impedance state when an external device's bus request (BUSRQ*) is acknowledged (BUSAK*).

The CPU's data bus is separated from a 16-bit internal data bus (ID0-ID16) by a transceiver which is enabled during memory references to the on-board memory space (0 to 5FFF hex) or I/O references to on-board I/O space (FC0-FFF hex). The INH* line is an output on P2, which is used to disable off-board memory during on- board accesses.

The internal data bus services on-board memory (16 bits) and I/O ports (lower 8 bits). A 16-bit hardware breakpoint register sits between the internal data bus and the address bus. When loaded through the data bus with a hardware breakpoint address (the Bx command), it generates a non-maskable interrupt to the CPU upon seeing that address on the address bus.

## MEMORY

Figure 4-1 shows the allocation of hex memory space on the board. The ROM Monitor occupies addresses 0-1FFF. The optional ROM Assembler occupies 2000-3FFF hex.

RAM occupies 4000-5FFF hex, with up to hex 300 low bytes used by working storage for the ROM programs and CPU stack pointer (i.e., up to address 42FF hex).

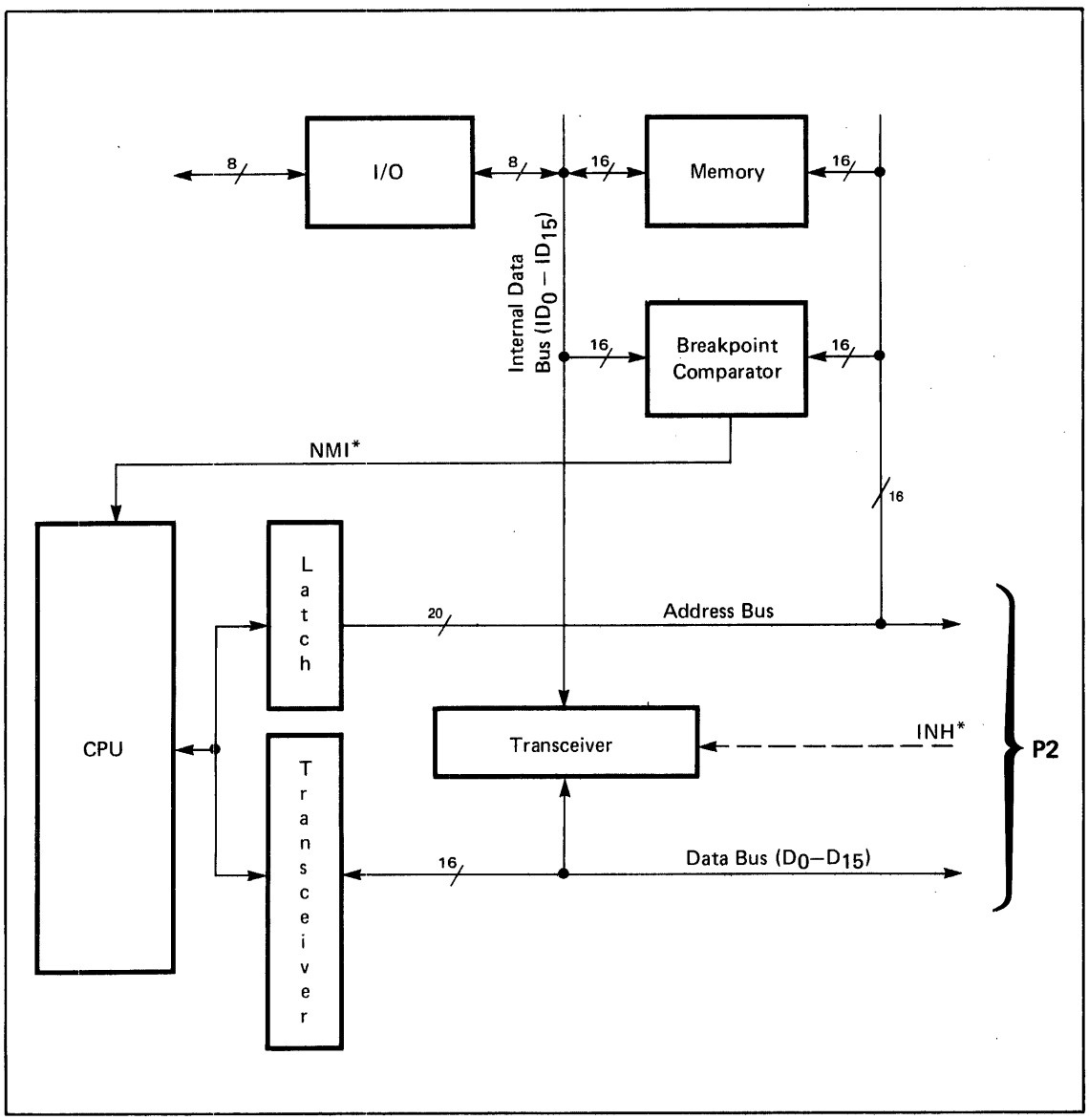



Figure 6-1. Bus Structure

RAM memory is refreshed at 30-microsecond intervals by the CPU, according to the Monitor's initialization of the CPU's refresh counter. ROMs are disabled during this refresh. Care should be taken if the INIT* input on P1 is enabled by jumpers 14 and 15 (figure 2-4): this causes a CPU reset, during which memory refresh ceases.

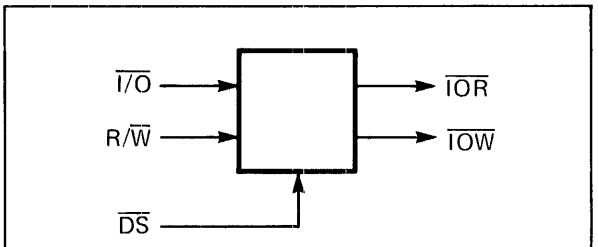
Each memory address contains an 8-bit byte. For a RAM read, both high and low byte RAMs of the word-oriented memory are enabled, irrespective of whether the operation is on a byte or a word. In a byte write to RAM, however, only the high or low byte of RAM memory is enabled (internal data bus lines ID15 -ID8 or ID7-ID0, respectively). By contrast, all ROM transfers enable the entire word.

## PERIPHERAL DECODING

Only the low-order 12 address bits on the Evaluation Board are decoded for peripherals, giving a total of up to 4K I/O ports. Some peripheral circuits use A0 and A1 to directly address internal channels or registers. Therefore, these two bits are uniformly treated as don't-care bits for device addressing as a result, each decoded I/O port consists of a block of four contiguous addresses.

These address lines are gated by the decoded status signal I/O*, indicating an input or output cycle (either one). In order to be compatible with the peripheral chips, two control strobes, IOR* and IOW* are generated from the R/W* line when I/O* is active, and they are gated by DS* as illustrated in the CPU Functions discussion above. Furthermore, the last eight groups are decoded as write-only ports by gating with the IOW* strobe. (see figure 6-2)

The complete list of I/O port addresses is shown in table 6-1. The range of four contiguous addresses in each group is first shown. If the device uses the A0 or A1 lines for direct addressing of channels or registers within the circuit, these are listed below the group range.




| DS | I/O | R/W | IOR | IOW |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 1 | X | X | 1 | 1 |
| X | 1 | X | 1 | 1 |

Figure 6-2. IOR* and IOW* Decoding

TABLE 6-1. PERIPHERAL ADDRESSES

```
FFC-FFF : N/C

FF8-FFB : N/C

FF4-FF7 : N/C

FF3-FF0 : Am8255A buffered user ports
    FF3 : Control port
    FF2 : Port C
    FF1 : Port B
    FF0 : Port A

FEF-FEC : Am9551 RS232 port (P5)
    FEF : Control
    FEE : Data
    FED : Control (same as FEF)
    FEC : Data (same as FEE)

FEB-FE8 : Am9551 RS232 or TTY port (P6)
    FEB : Control
    FEA : Data
    FE9 : Control (same as FEB)
    FE8 : Data (same as FEA)

FE7-FE4 : Am8253 counter time
    FE7 : Mode control
    FE6 : Counter 2
    FE5 : Counter 1
    FE4 : Counter 0

FE3-FE0 : Keyboard return lines

FDF-FEC : Keyboard scan lines

FD8-FDB : Single-step control
```

TABLE 6-1. PERIPHERAL ADDRESSES (continued)

```
FD7-FD4 : Breakpoint register


FD3-FD0 : LED Display
    FD3 : 20th character (left most)
    FD2 : 19th character
    FD1 : 18th character
    FD0 : 17th character


FCF-FCC : LED Display
    FCF : 16th character
    FCE : 15th character
    FCD : 14th character
    FCC : 13th character


FCB-FC8 : LED Display
    FCB : 12th character
    FCA : 11th character
    FC9 : 10th character
    FC8 :  9th character


FC7-FC4 : LED Display
    FC7 :  8th character
    FC6 :  7th character
    FC5 :  6th character
    FC4 :  5th character


FC3-FC0 : LED Display
    FC3 :  4th character
    FC2 :  3rd character
    FC1 :  2nd character
    FC0 :  1st character (right-most)
```

# APPENDIX A
# CPU BUS BUFFERING CHARACTERISTICS AT P2

The following output delay and input setup times need to be considered when interfacing external logic through the P2 edge connector.

| P2 Pin | From Z8000 to P2 output | From P2 Input to Z8000 |
|---|---|---|
| BUSREQ* | | 0ns |
| VI* | | 0ns |
| NVI* | | 0ns |
| STOP* | | 0ns |
| WAIT* | | 8ns |
| NMI | | 37ns |
| BUSAK* | 15ns | |
| R/W* | 9ns | |
| N/S* | 9ns | |
| B/W* | 9ns | |
| AS* | 9ns | |
| DS* | 9ns | |
| ST0-ST3 | 9ns | |
| A0-AF | Valid at least 55ns before rising edge of AS*. | |
| A0-AF/D0-DF | Address output valid at least 55 ns before rising edge of AS*. Data output valid on bus 46ns before falling edge of DS*. Input must be valid at least 93ns before falling edge of T3. It must remain valid until rising edge of DS*. | |

# APPENDIX B

# ASCII CHARACTER SET

The ASCII internal character set used with the Evaluation Board is the ANSI X3.4 1968 version. The following is a summary.

Table B-1. ASCII

| HEX | DEC | CHAR | HEX | DEC | CHAR | HEX | DEC | CHAR | HEX | DEC | CHAR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 0 | NUL | 20 | 32 | SP | 40 | 64 | @ | 60 | 96 | ` |
| 01 | 1 | SOH | 21 | 33 | ! | 41 | 65 | A | 61 | 97 | a |
| 02 | 2 | STX | 22 | 34 | " | 42 | 66 | B | 62 | 98 | b |
| 03 | 3 | ETX | 23 | 35 | # | 43 | 67 | C | 63 | 99 | c |
| 04 | 4 | EOT | 24 | 36 | $ | 44 | 68 | D | 64 | 100 | d |
| 05 | 5 | ENQ | 25 | 37 | % | 45 | 69 | E | 65 | 101 | e |
| 06 | 6 | ACK | 26 | 38 | & | 46 | 70 | F | 66 | 102 | f |
| 07 | 7 | BEL | 27 | 39 | ' | 47 | 71 | G | 67 | 103 | g |
| 08 | 8 | BS | 28 | 40 | ( | 48 | 72 | H | 68 | 104 | h |
| 09 | 9 | HT | 29 | 41 | ) | 49 | 73 | I | 69 | 105 | i |
| 0A | 10 | LF | 2A | 42 | * | 4A | 74 | J | 6A | 106 | j |
| 0B | 11 | VT | 2B | 43 | + | 4B | 75 | K | 6B | 107 | k |
| 0C | 12 | FF | 2C | 44 | , | 4C | 76 | L | 6C | 108 | l |
| 0D | 13 | CR | 2D | 45 | - | 4D | 77 | M | 6D | 109 | m |
| 0E | 14 | SO | 2E | 46 | . | 4E | 78 | N | 6E | 110 | n |
| 0F | 15 | SI | 2F | 47 | / | 4F | 79 | O | 6F | 111 | o |
| 10 | 16 | DLE | 30 | 48 | 0 | 50 | 80 | P | 70 | 112 | p |
| 11 | 17 | DC1 | 31 | 49 | 1 | 51 | 81 | Q | 71 | 113 | q |
| 12 | 18 | DC2 | 32 | 50 | 2 | 52 | 82 | R | 72 | 114 | r |
| 13 | 19 | DC3 | 33 | 51 | 3 | 53 | 83 | S | 73 | 115 | s |
| 14 | 20 | DC4 | 34 | 52 | 4 | 54 | 84 | T | 74 | 116 | t |
| 15 | 21 | NAK | 35 | 53 | 5 | 55 | 85 | U | 75 | 117 | u |
| 16 | 22 | SYN | 36 | 54 | 6 | 56 | 86 | V | 76 | 118 | v |
| 17 | 23 | ETB | 37 | 55 | 7 | 57 | 87 | W | 77 | 119 | w |
| 18 | 24 | CAN | 38 | 56 | 8 | 58 | 88 | X | 78 | 120 | x |
| 19 | 25 | EM | 39 | 57 | 9 | 59 | 89 | Y | 79 | 121 | y |
| 1A | 26 | SUB | 3A | 58 | : | 5A | 90 | Z | 7A | 122 | z |
| 1B | 27 | ESC | 3B | 59 | ; | 5B | 91 | [ | 7B | 123 | { |
| 1C | 28 | FS | 3C | 60 | < | 5C | 92 | \ | 7C | 124 | \| |
| 1D | 29 | GS | 3D | 61 | = | 5D | 93 | ] | 7D | 125 | } |
| 1E | 30 | RS | 3E | 62 | > | 5E | 94 | ^ | 7E | 126 | ~ |
| 1F | 31 | US | 3F | 63 | ? | 5F | 95 | _ | 7F | 127 | DEL |

# APPENDIX C

# SERVICE INFORMATION

## INTRODUCTION

This chapter provides service diagrams and information on service and repair assistance for AMC product lines.

## SERVICE AND REPAIR ASSISTANCE

Service and repair assistance can be obtained from Advanced Micro Computers by contacting the AMC Field Service Department in Santa Clara, California at one of the following numbers:

Telephone: (408) 988-7777

Toll Free: (800) 672-3548 (California)
(800) 538-9791 U.S.A. (except California)

If it is necessary to return a product to Advanced Micro Computers for service or repair, contact the Field Service Department at the previously listed telephone number. A Return Material Authorization number will be provided along with shipping instructions and other important information that will help AMC provide you with fast, efficient service. When reshipment is due to the product being damaged during shipment from AMC, or when the product is out of warranty, a purchase order is required for the AMC Field Service Department to initiate the repair.

Prepare the product for shipment by repackaging it in the original factory packaging material, if available. When the original packaging is not available, wrap the product in a cushioning material (such as Air Cap TH-240, manufactured by the Sealed Air Corporation, Hawthorne, New Jersey) and enclose in a heavy-duty corrugated shipping carton. Seal the shipping carton securely, mark it FRAGILE, and ship it to the address specified by the AMC Field Service Department.

Customers outside of the United States can contact an AMC Sales Office or Authorized AMC Distributor for directions on obtaining service or repair assisstance.

## SERVICE DIAGRAMS

The Am96/4018 assembly drawing is shown as figure C-1.

Schematic diagrams of the Am96/4018 are shown in figures C-1 through C-10. Active-low (logical 0) signals are indicated by an asterisk (*) following the signal name.

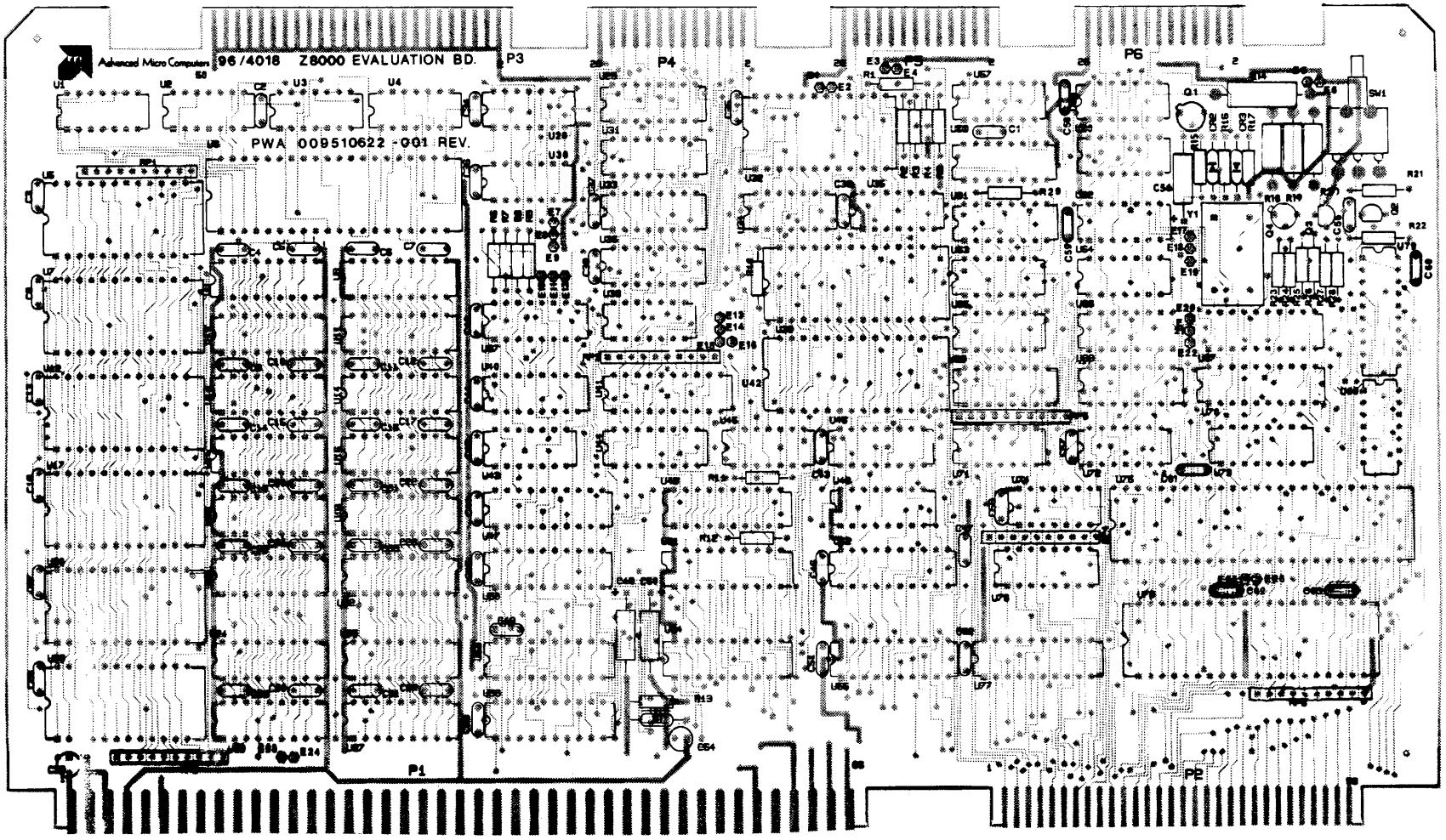



Figure C-1. Component Location Diagram

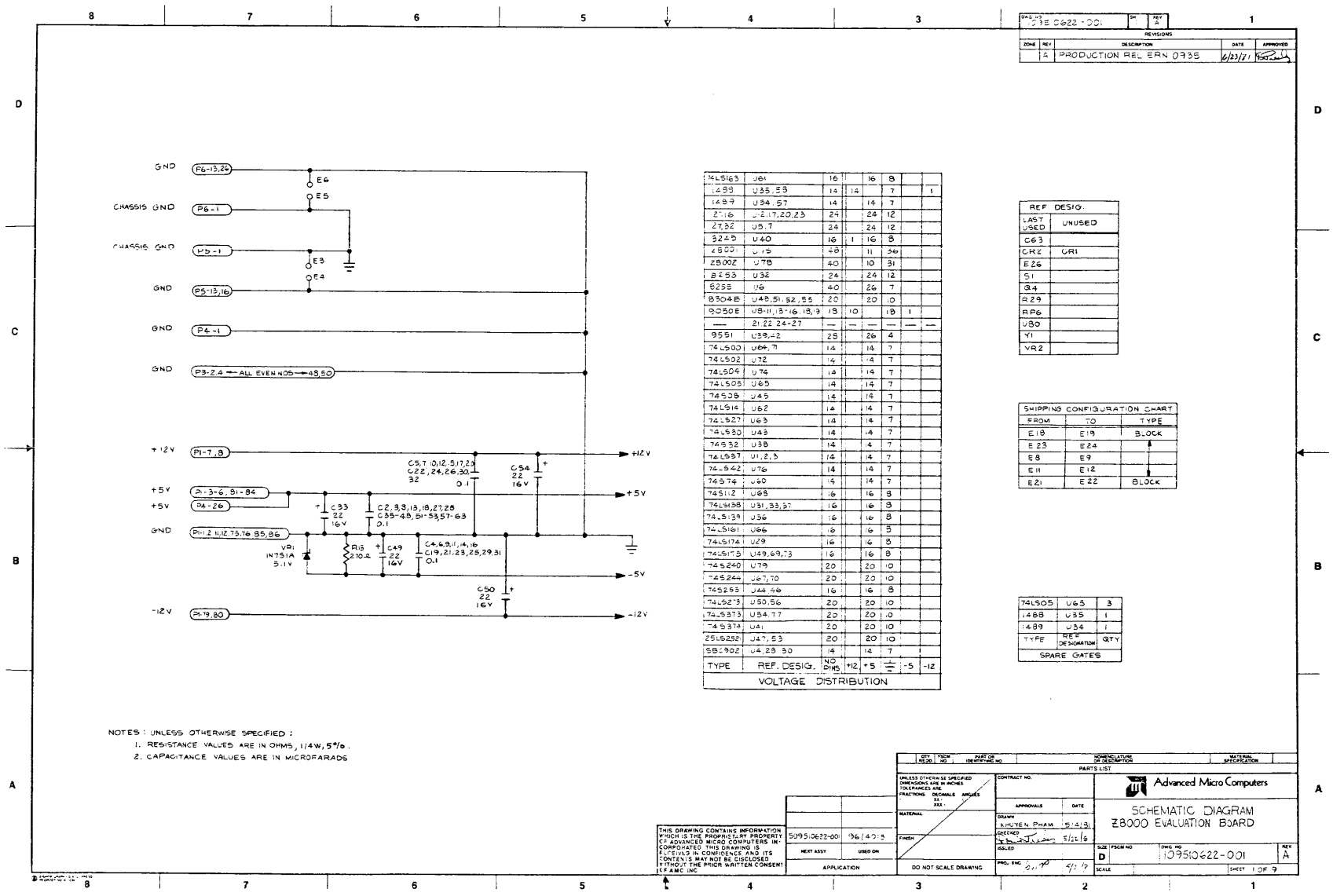



Figure C-2. Am96/4018 Schematic Diagram Sheet 1

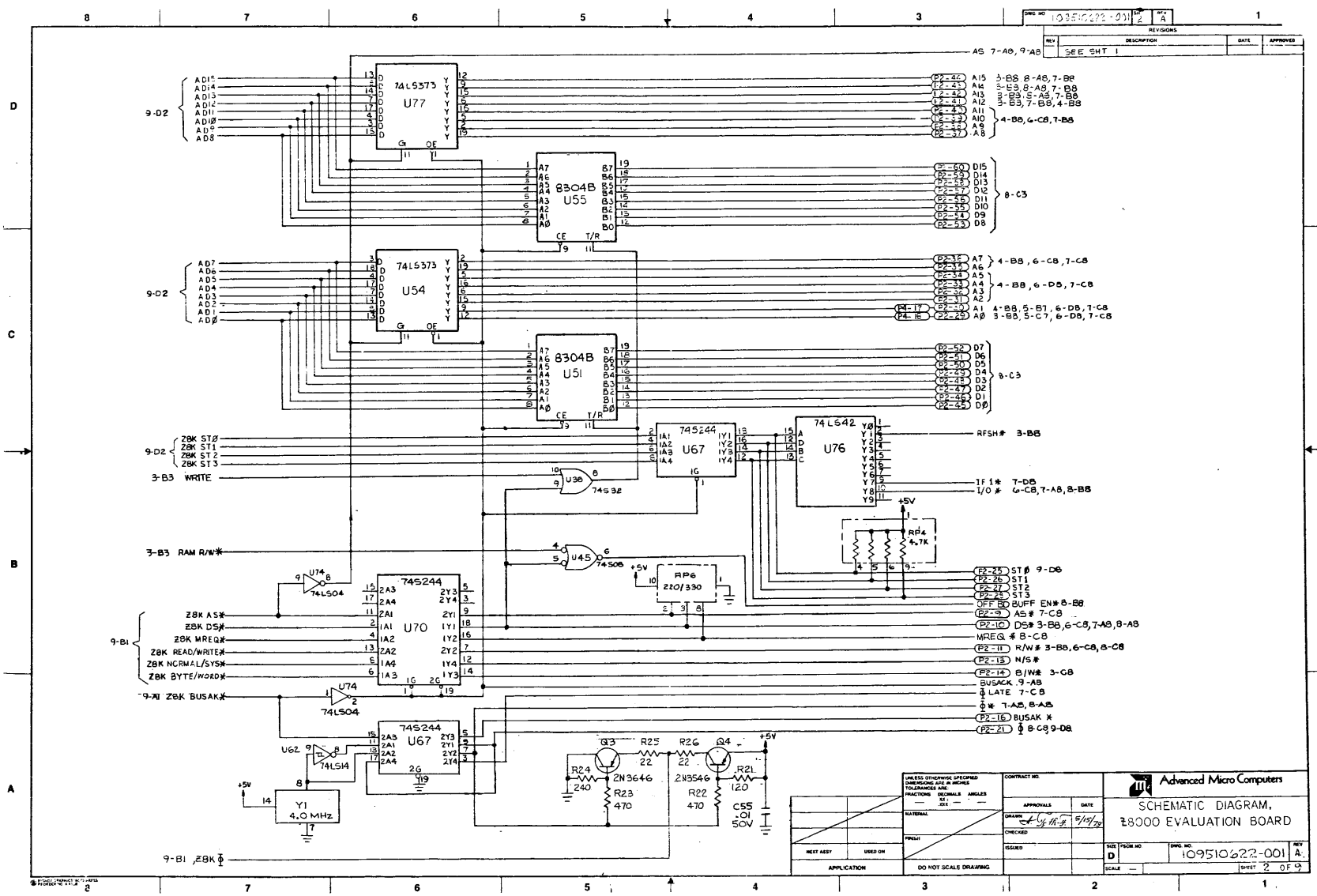



Figure C-3. Am96/4018 Schematic Diagram Sheet 2

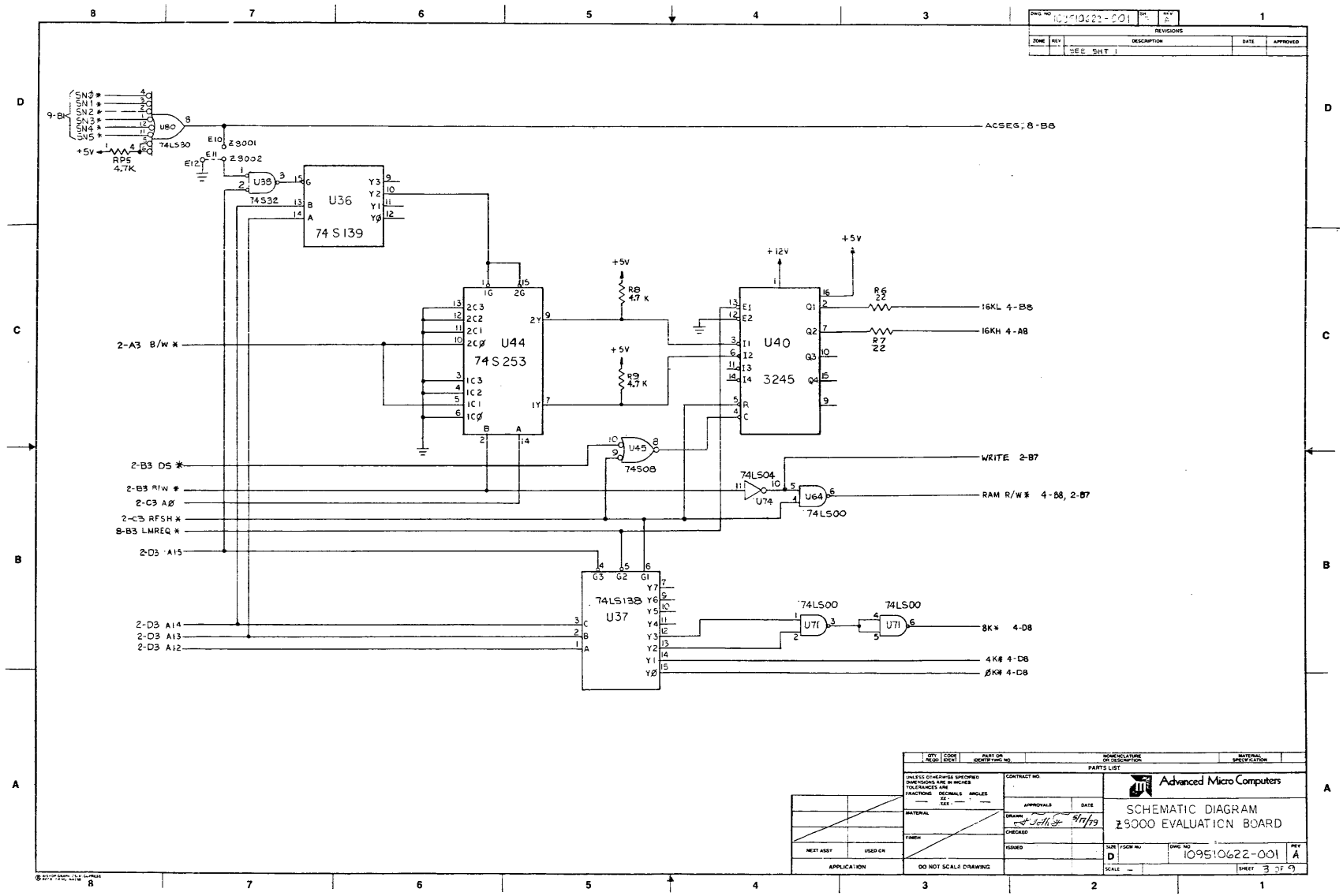



Figure C-4. Am96/4018 Schematic Diagram Sheet 3

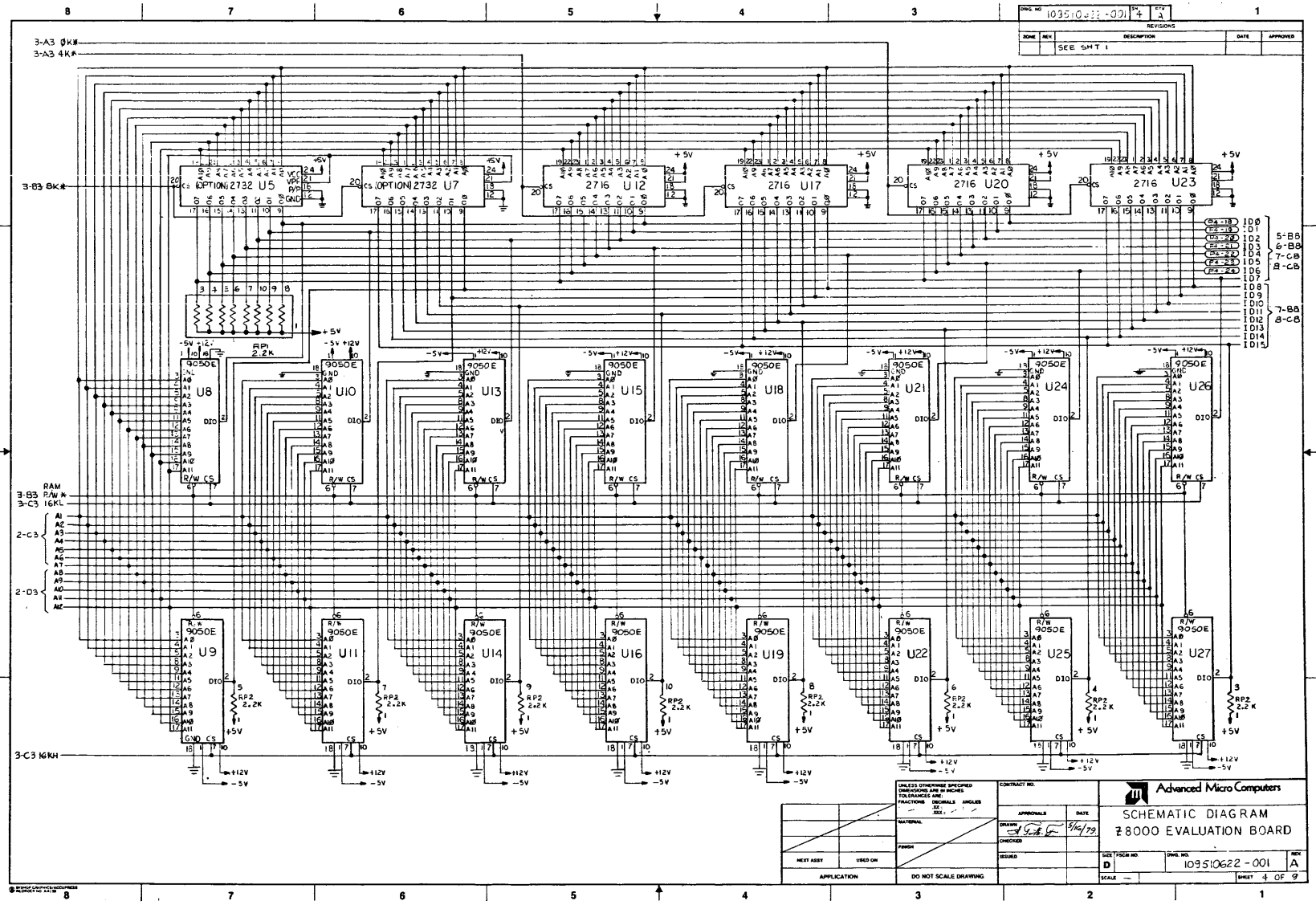



Figure C-5. Am96/4018 Schematic Diagram Sheet 4

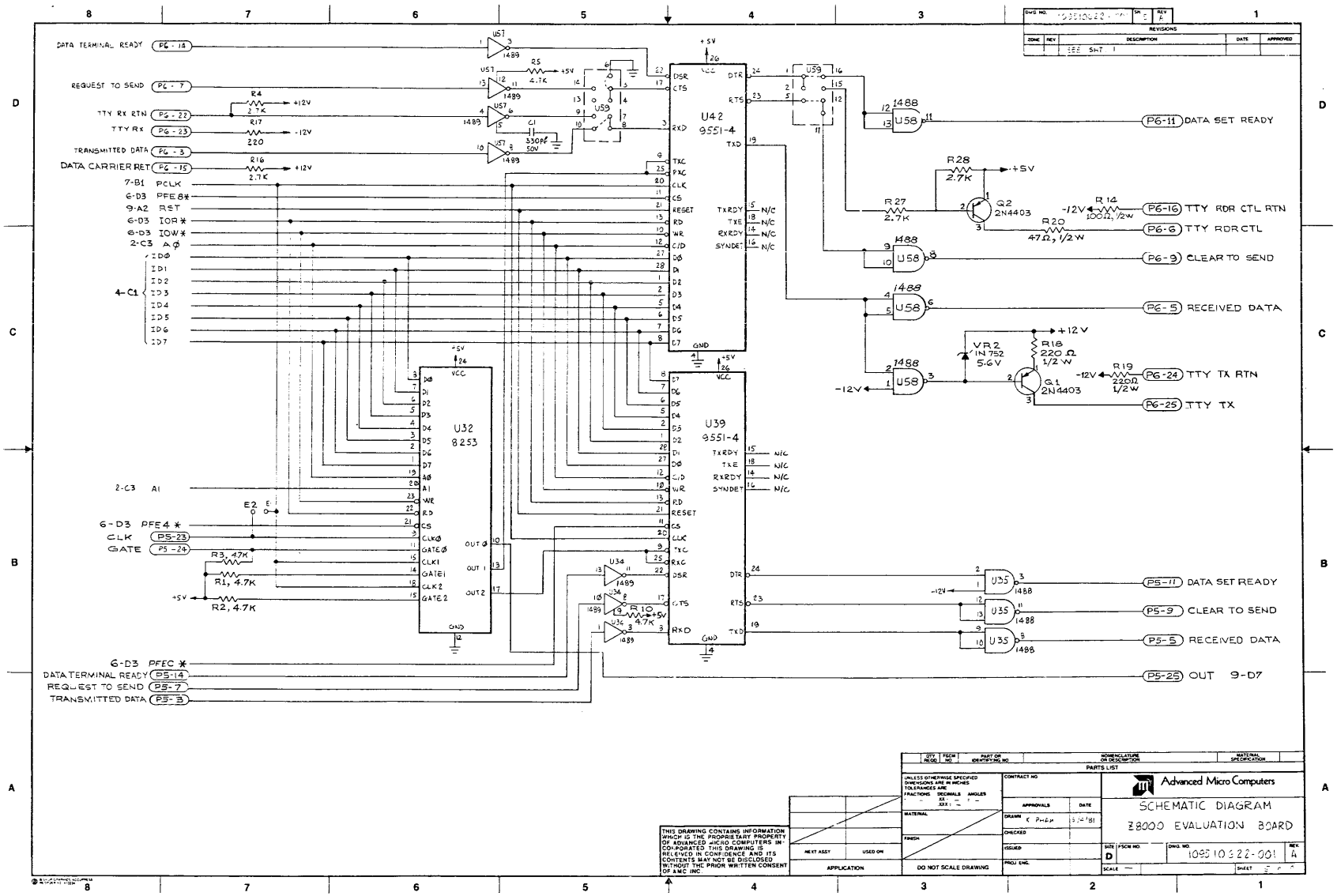



Figure C-6. Am96/4018 Schematic Diagram Sheet 5

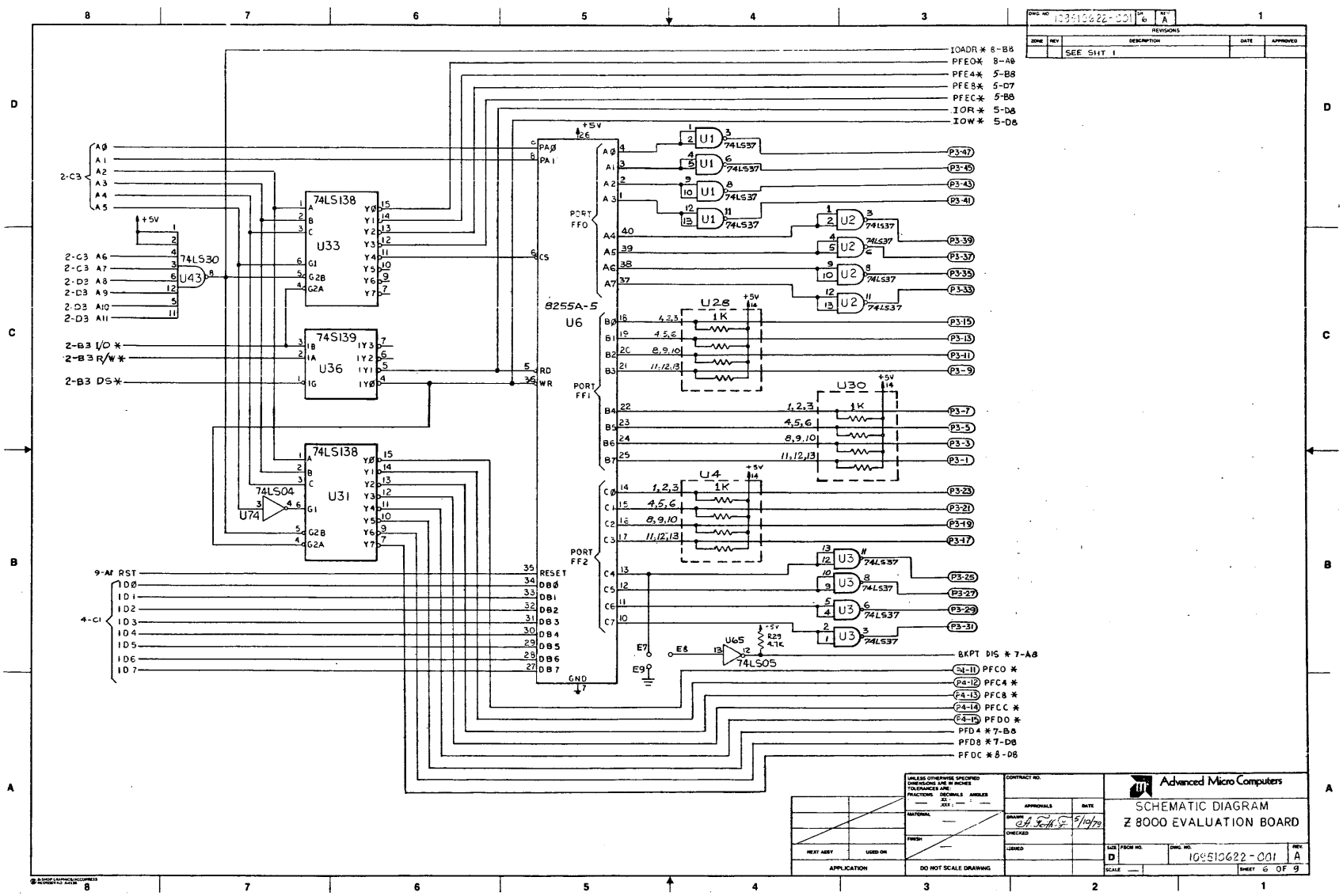



Figure C-7. Am96/4018 Schematic Diagram Sheet 6

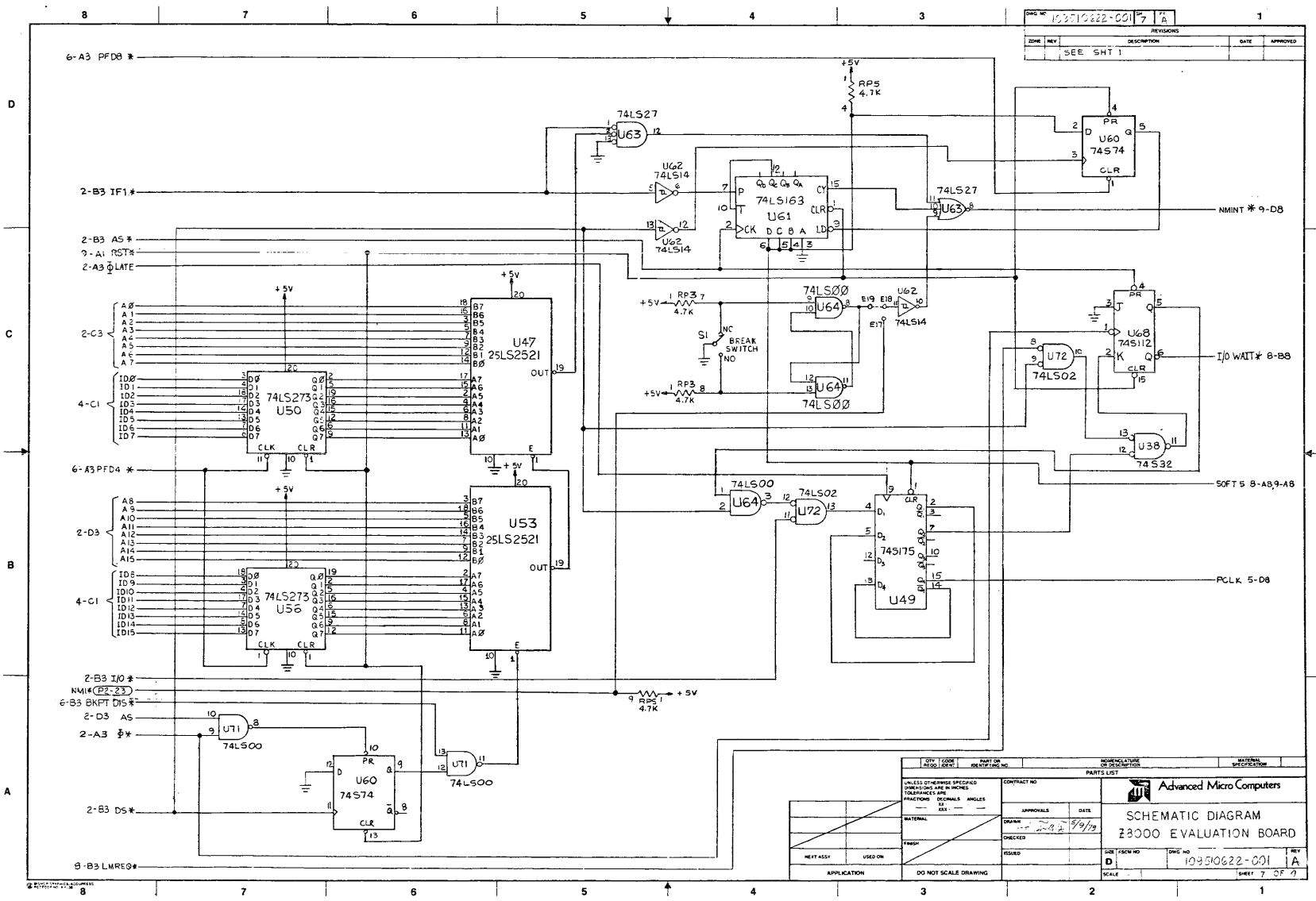



Figure C-8. Am96/4018 Schematic Diagram Sheet 7

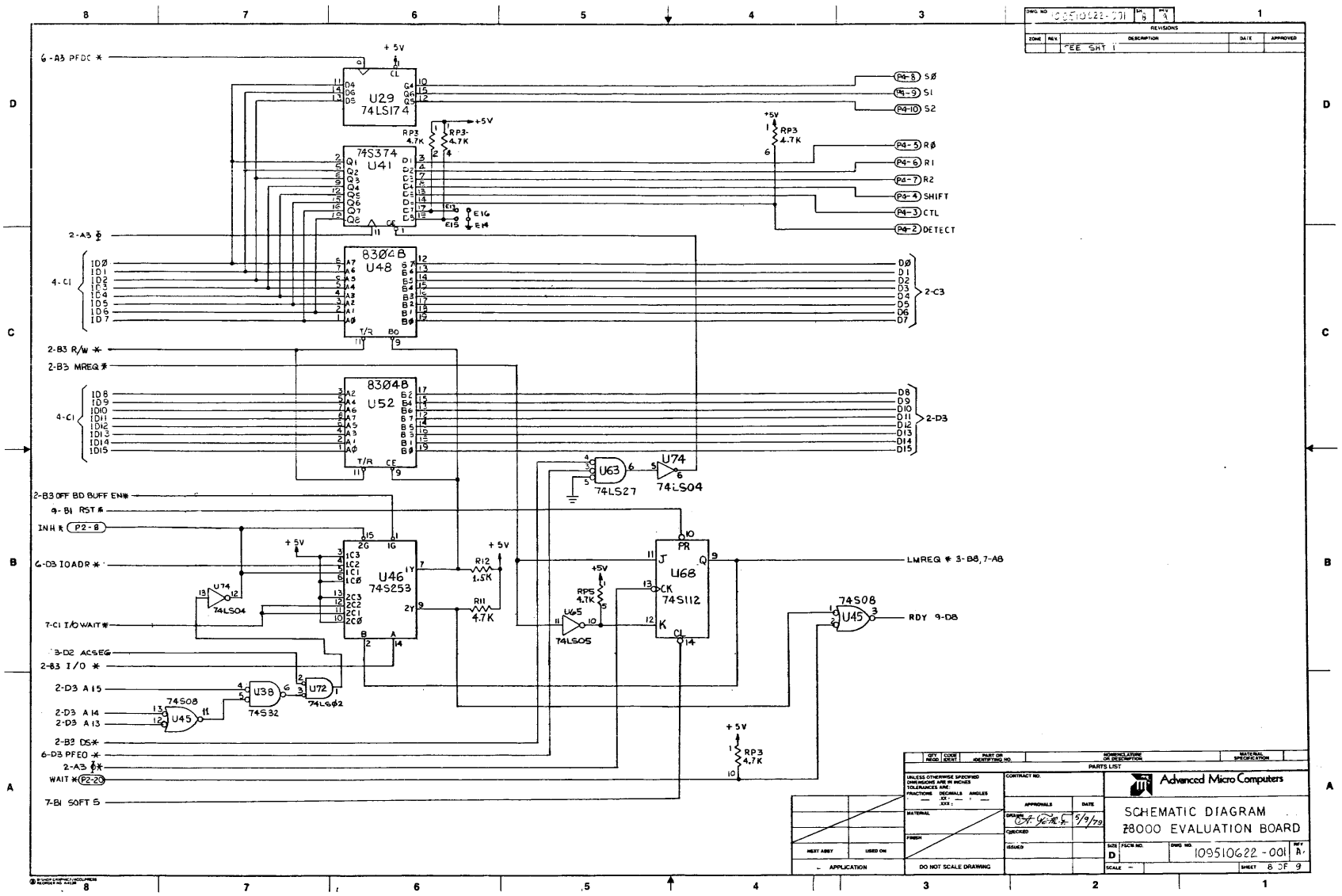



Figure C-9. Am96/4018 Schematic Diagram Sheet 8

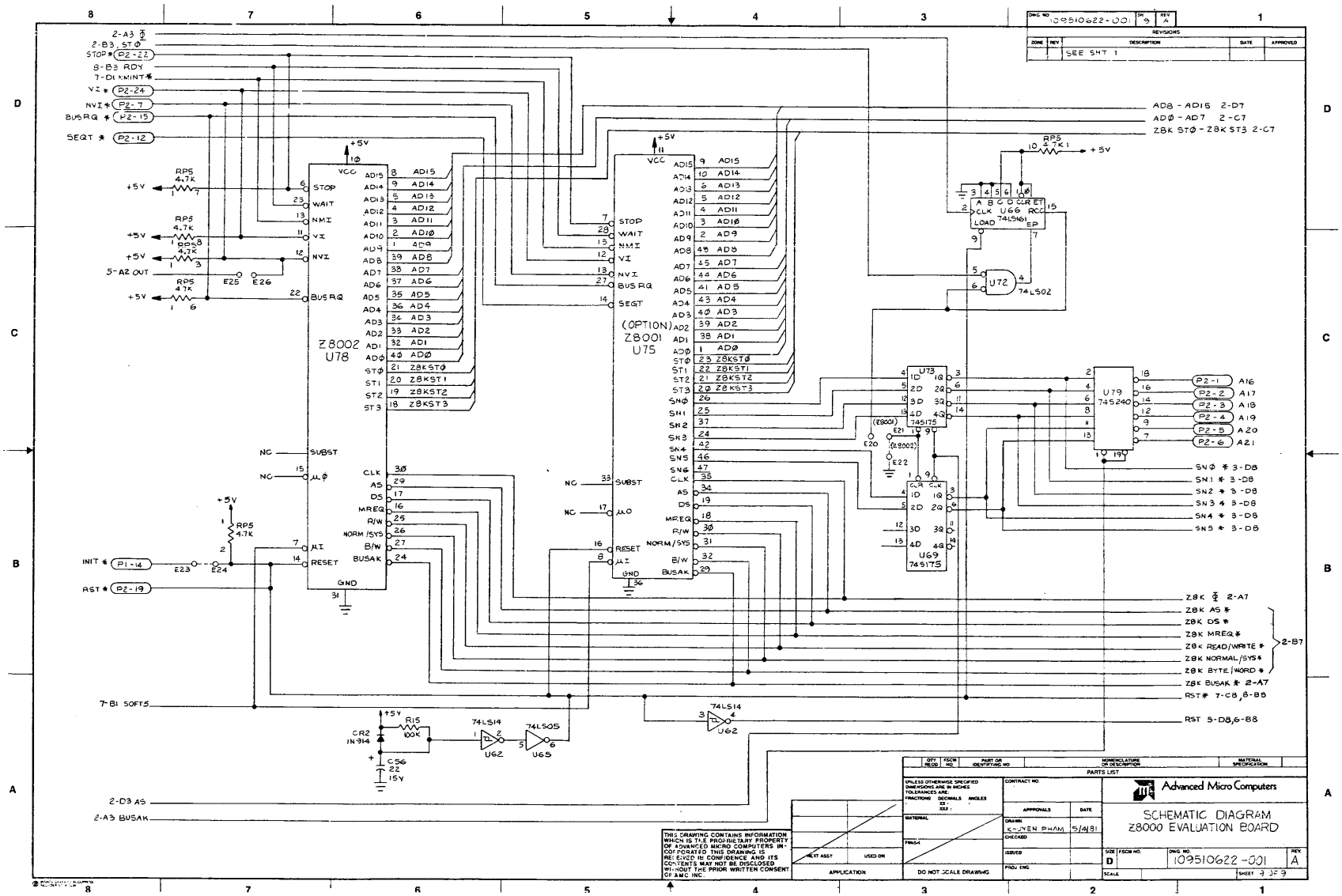



Figure C-10. Am96/4018 Schematic Diagram Sheet 9

**COMMENT SHEET**

Address comments to:

Advanced Micro Computers
Publications Department
3340 Scott Boulevard
Santa Clara, CA 95051

TITLE: Am96/4018 AmZ8000 Evaluation Board
PUBLICATION NO: 05510622-001 Revision A

COMMENTS: (Describe errors, suggested additions or deletions, and include page numbers, etc.)

From: Name: _____ Position: _____

Company: _____

Address: _____

***Advanced***
***Micro***
***Computers***

A subsidiary of
Advanced Micro Devices
3340 Scott Boulevard
Santa Clara,
California 95051
(408) 988-7777
TELEX: 171 142