Heurikon MLZ-91/92 Microcomputer
User's Manual

HEURIKON CORP.

Microcomputers For Industry

MLZ-91 User Manual

HEURIKON CORPORATION

Contents

INTRODUCTION

The purpose of this manual is to document the features and present implementation examples of the MLZ-91 microcomputer board.

The MLZ-91 contains a number of special features not commonly available on single board microcomputers. This manual covers these unique features in detail. General information, such as FDIO, CTC, and SIO commands are discussed, however, the following documents and manuals should be consulted to obtain complete information on the chip set and external peripherals:

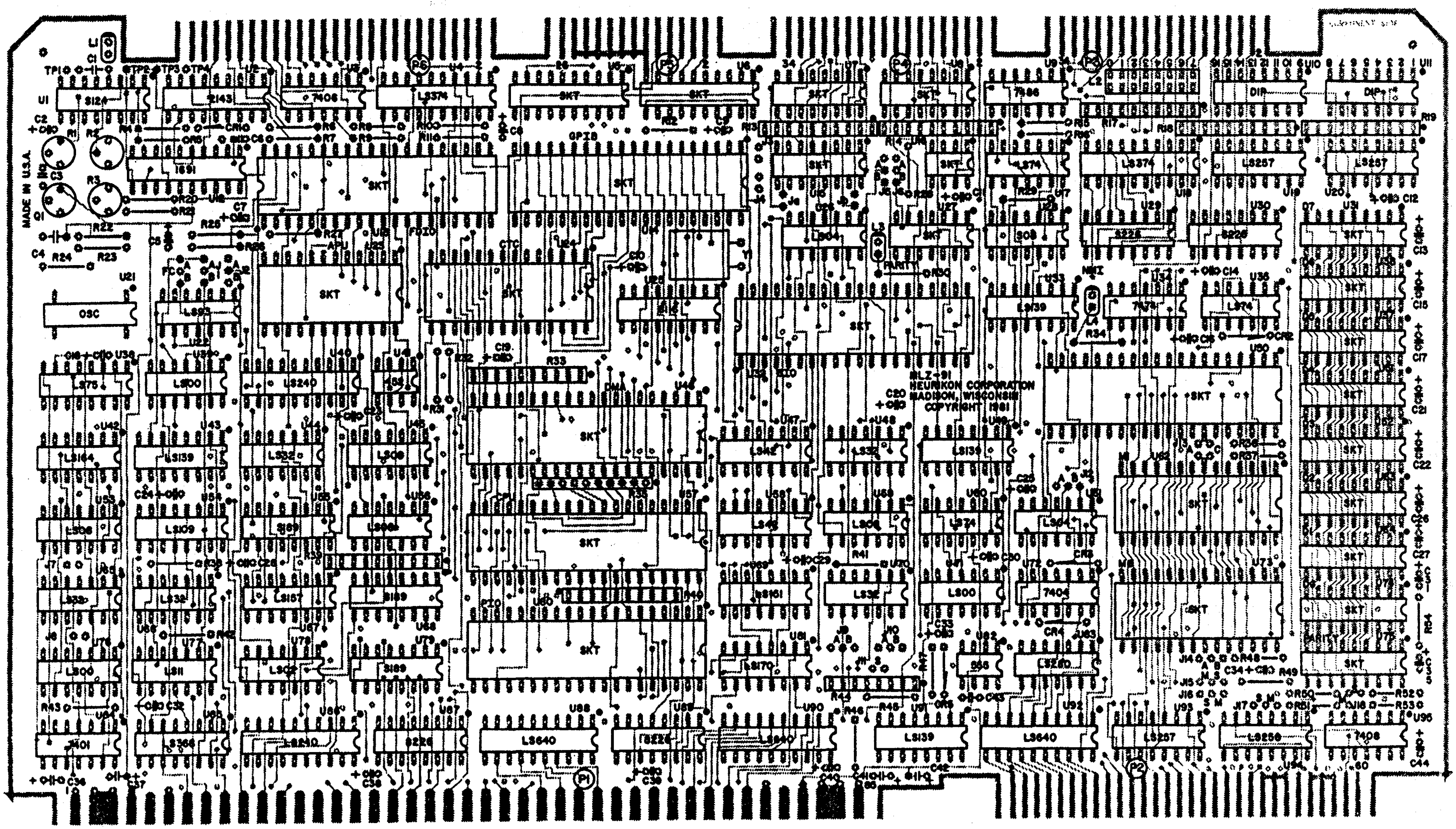
MLZ-91 Schematic Diagrams (Heurikon)
Z-80 Chip Set Product Specification booklet (Zilog, Mostek)
Z80 chip set Technical Manuals (Zilog, Mostek)
APU/FPU Technical Manuals (Advanced Micro Devices)
FD1793 (FDIO) Controller Manual (Western Digital)
TMS 9914 GPIB Controller (Texas Instruments)
ZRAID Monitor Manual (Heurikon)
ZRAID Monitor source code listing (Heurikon)
Floppy disk drive specifications (Shugart)
Streamer Tape drive specification (Archive)
Winchester Controller Manuals (Priam, Shugart, Micropolis)
CP/M Operating System Manuals (Digital Research)

Heurikon; Madison, WI (608) 271-8700
AMD; Sunnyvale, CA (408) 732-2400
Zilog; Cupertino, CA (408) 446-4666
Western Digital; Newport Beach, CA (714) 557-3550
Texas Instruments; Dallas, TX (214) 238-6531
Archive Corporation; Costa Mesa, CA (714) 641-0279
Priam Corporation; San Jose, CA (408) 946-4600
Micropolis Corporation; Chatsworth, CA (213) 709-3300
Shugart; Sunnyvale, CA (408) 733-0100
Digital Research Corporation; Pacific Grove, CA (408) 375-6262

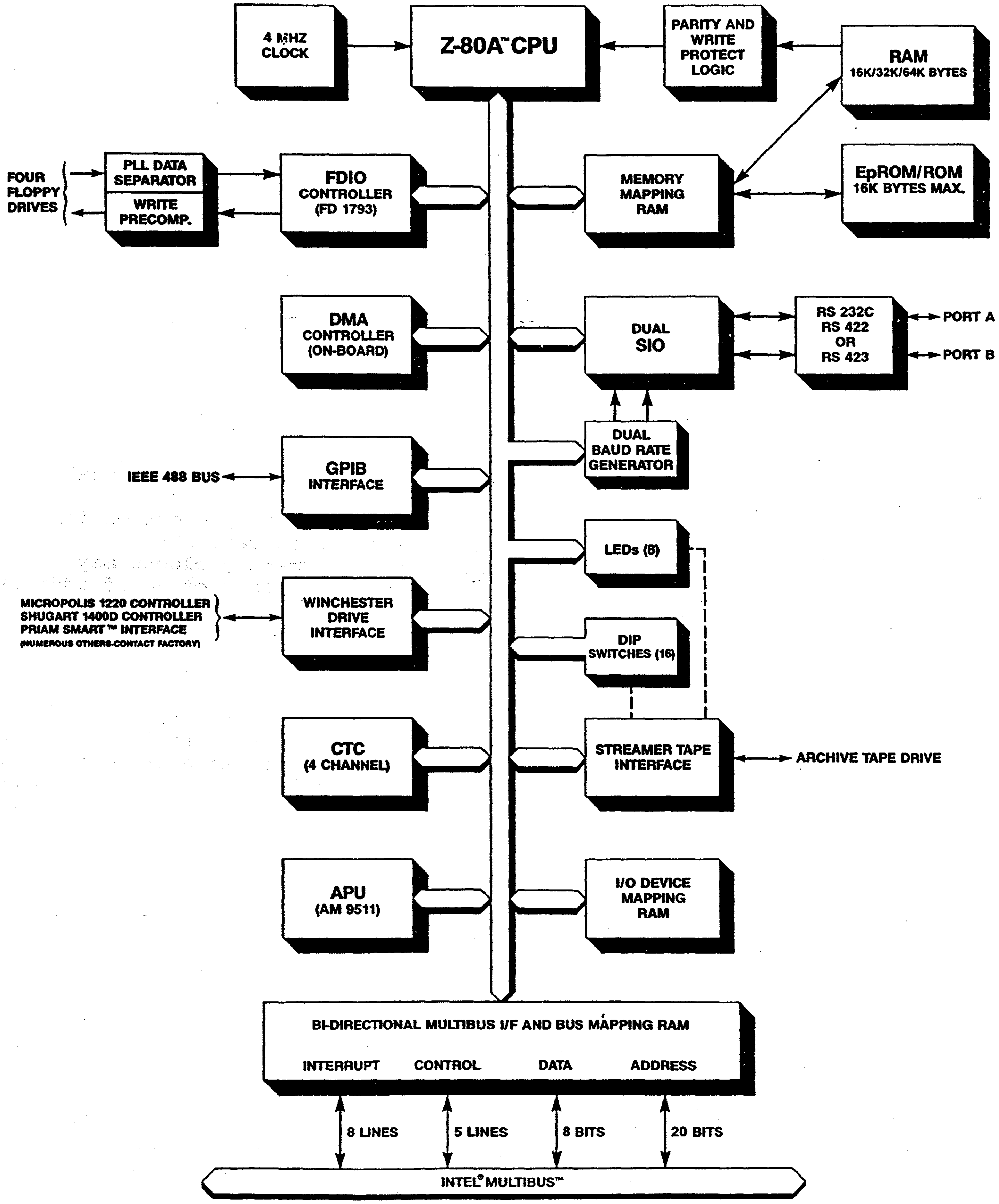
Z-80 is a trademark of ZILOG Corp.
MULTIBUS is a trademark of INTEL Corp.
CP/M is a trademark of DIGITAL RESEARCH

Some portions of the Z-80 chip literature are reprinted herein courtesy of Zilog Corporation.

The information furnished by Heurikon is believed to be accurate and reliable. However, no responsibility is assumed by Heurikon for its use. Specifications are subject to change without notice.


MLZ-91
HEURIKON CORPORATION
MADISON, WISCONSIN
COPYRIGHT 1981
MADE IN U.S.A.
TP1
TP2
TP3
TP4
L1
C1
U1
S124
2143
7406
LS374
SKT
GPIB
7486
DIP
LS91
FDIO
APU
CTC
OSC
LS93
LS75
LS00
LS240
LS164
LS139
LS32
LS08
LS109
S189
LS157
LS11
LS02
LS366
8226
LS640
DMA
CPU
PIO
LS42
LS161
LS170
LS04
LS257
LS174
LS138
LS260
7404
555
LS259
PARITY
NMI
P1
P2
P3
P4
P5
P6
Q1
R1
R2
R3
C2
C3
C4
U21
U50
U96
C44

HEURIKON MLZ-91A MICROCOMPUTER



FACILITIES DESCRIPTION

CPU

Z-80A or equivalent.
Standard Z-80 instruction set - 158 instructions.
16 bit address bus, 8 bit data bus.
8, 16, 24, and 32 bit instruction lengths.

APU

AMD 9511 arithmetic processing unit (APU).
Stack oriented data transfers.
43 microprogrammed "macro commands" including cos, sin, tan, mult, div, sqrt, log, ln, exp, asin, etc.
Results computed to 32 bit precision.

DMA

Z-80 compatible I/O and memory data transfers.
Multi-mode operation (byte-at-time, burst, and continuous).
Transfers data from I/O or memory to I/O or memory. Also has data search mode.
Vectored interrupt generated on completion.

MEMORY

Two memory sockets for ROM (types 2716, 2732, or 2764 with maximum capacity of 8K bytes per socket).
16K/32K/or64K on-card RAM with optional parity bit.
Software controlled write-protection for each 4K block of on-card RAM.
Any (or all) on-card memory blocks may be disabled to increase off-card address space.

BUS I/F

Intel Multibus compatible
20 bit memory address space, fully addressable by CPU or DMA.
1 megabyte addressing capability.
Bi-directional I/F (Master/Slave)
Software programmable master mode operation:

- Mode 0 - Release bus after each transaction
- Mode 1 - Release bus for any other card (uses CBREQ-)
- Mode 2 - Release bus only for higher priority cards (uses BAI-)
- Mode 3 - Never release bus (override)

Software programmable slave mode operation:

- Board location on bus
- Inhibited bus operations (Mem RD, Mem WR, all I/O)

Eight bus interrupts. (Bi-directional)

MEMORY MANAGEMENT

Programmable address mapping RAM.
Memory map completely under software control.
Allows full use of 20 bit address space on bus

Any 8K block of bus address space may be mapped into any 8K block of CPU/DMA address space.
Any 4K block of on-card address space may be mapped into any 4K block of CPU/DMA address space.

FDIO
Double density floppy-disk interface (uses WD1793).
On-card data separation (PLL) and write pre-composition logic.
Four drive select lines plus side select for dual sided drives. (4 megabyte capacity).
Also supports single density and 5¼" formats.
Software controllable status LED.

OTHER I/O
On-card:
- I/O device base addresses mappable under software control.
- Winchester controller I/F (Priam, Micropolis or Shugart)
- GPIB (IEEE-488) controller, talker and listener
- 8 bit parallel port for Streamer tape I/F
- 16 position user DIP switches
- 8 bit user LED display
- Two SIO ports.
  - RS232/423/422 interface.
  - Asynchronous or sychronous modes (including SCLC).
- Software controlled dual baud rate generator.
- Two utility 8-bit PIO ports for on-card control and bus interrupt functions.
- Four counter/timer channels.

Off-card:
- Entire I/O device address space available for off-card use.

POWER-ON JUMP
Provided via memory mapping RAM, above.

BOARD
Multilayer, 6.75 inch by 12 inch

OPTIONS
Hardware jumpers:
- System clock (2 or 4 MHZ)
- APU clock (2 or 4 MHz)
- Highest priority board designation
- Memory configuration (ROM type, RAM size)
- Memory wait state select (all cycles, M1 only, ROM only)
- Drive type (8" or 5¼")
- SIO clock control (1 port)
- SIO I/F select (RS232/423/422)
- Winchester type select (Priam, Micropolis or Shugart)

Software controlled:
- Bus mode (control release)
- Memory map contents
- Bus map contents and inhibit states
- I/O map contents
- DMA "data ready" source selection
- Bus interrupts
- Baud rates
- FDIO single/double density select
- Drive select and side select
- User status LED

GETTING GOING

This section is an outline of the minimum work necessary to get the MLZ-91 "on the air":

Items required: (See diagram)

MLZ-91 Microcomputer Board
ZRAID-91 Software Monitor program (ROM)
RS232 Interactive Terminal and cable with male "D" connector
MLZ-P4N Serial Interface cable and connectors
Power supply (+5, +12, -12, volts)
Card Rack

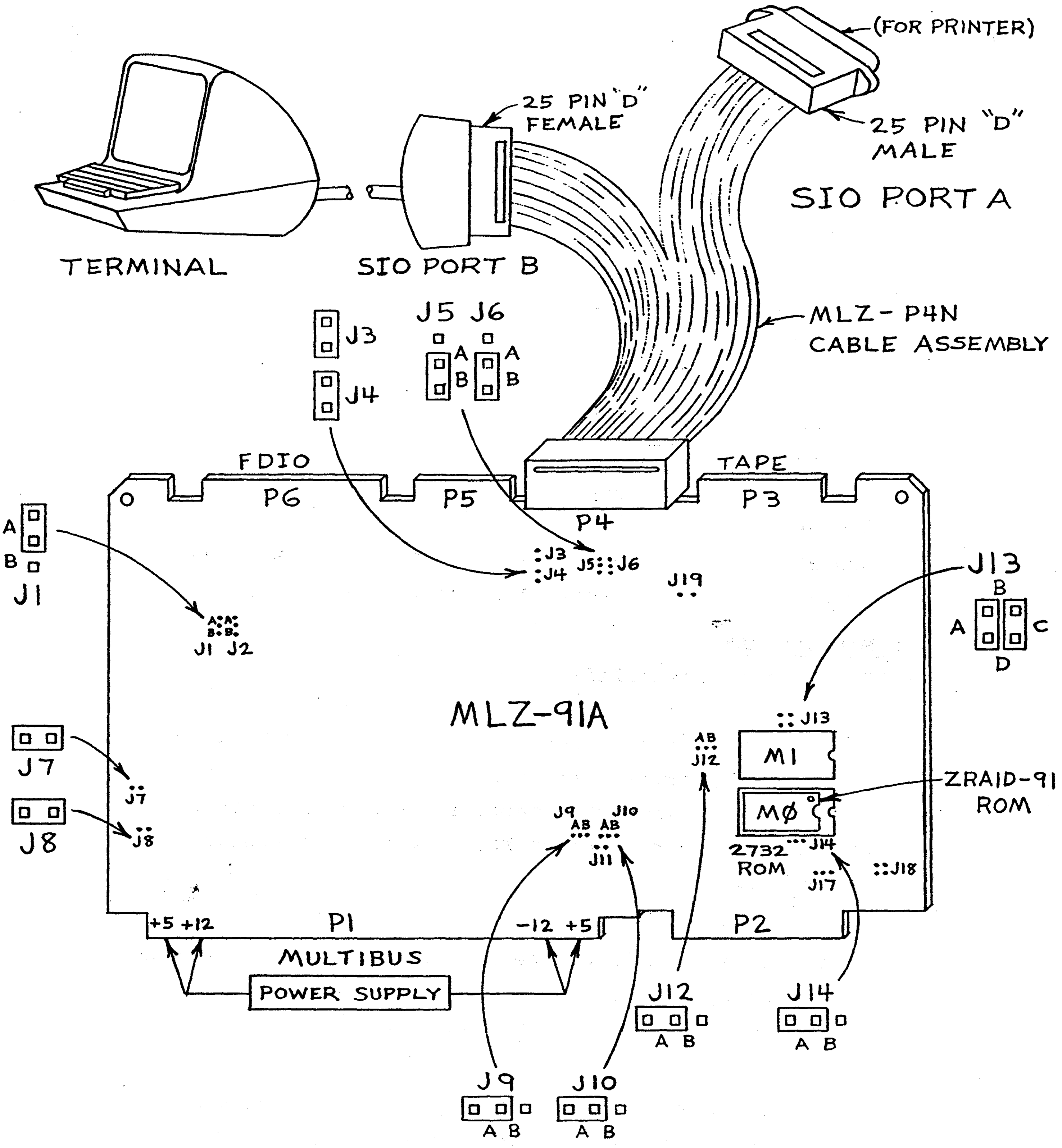
1. Insert ZRAID ROM in socket MØ (See diagram for position detail.)
2. Install Jumpers as follows:

J1-A	(2MHz clock)
J3,J4	(SIO I/F)
J5-B, J6-B	(SIO Port A Receive clock)
J7	(Upper address enable)
J8	(Processor priority)
J9-A, J10-A	(Wait states for ROM)
J12-A, J14-A	(ROM type 2732)
J13-A,C	(RAM type) (Assumes 4164 or 4532-2)

3. Connect console terminal to SIO port B via MLZ-P4N cable and P4 connector. Use the female D connector on the cable.

"D" Pin #	
2	Data from terminal
3	Data to terminal
4-5	Jumper (RTS-CTS)
6-20	Jumper (DSR-DTR)
7	Ground

4. Turn all DIP switches on MLZ-91, if installed, to "OFF".
5. Set baud rate on terminal for 9600 baud. Set terminal options, if available, to eight bits, no parity, two stop bits.
6. Apply power to MLZ-91 and terminal.
7. Activate RESET (momentarily ground P1-14).
8. ZRAID sign-on message should appear on terminal.
9. Consult ZRAID manual for further details. ZRAID automatically sets up the MLZ-91 mapping RAMs and allows access to all memory and I/O devices from the terminal.



MLZ-91A WITH ZRAID - SETUP DIAGRAM

To load the CP/M operating system, follow the above steps but also connect a floppy disk drive to P6. Set the floppy disk configuration jumpers for "8" or "5" as appropriate. (See page 128). After turning on power and resetting the system, insert the CP/M system diskette in drive "A" and enter apostrophe-space on the terminal. Refer to the ZRAID manual for details.

ZRAID-91 Initialization State

The ZRAID-91 monitor initializes the mapping RAMs and on-card I/O derives as follows:

A. Memory Mapping RAM (See diagram, next page)

ROM socket MØ at CPU address FØØØ (hex)

On-card RAM allocated from address ØØØØ through address EFFF.

B. I/O Mapping RAM

I/O Addresses	Assignment
ØØ thru 3F	Off-card
4Ø thru 7F	Off-card
8Ø thru BF	On-card I/O Group A (e.g. Baud Gen)
CØ thru FF	On-card I/O Group B (e.g. CTC)

C. Bus Mapping RAM

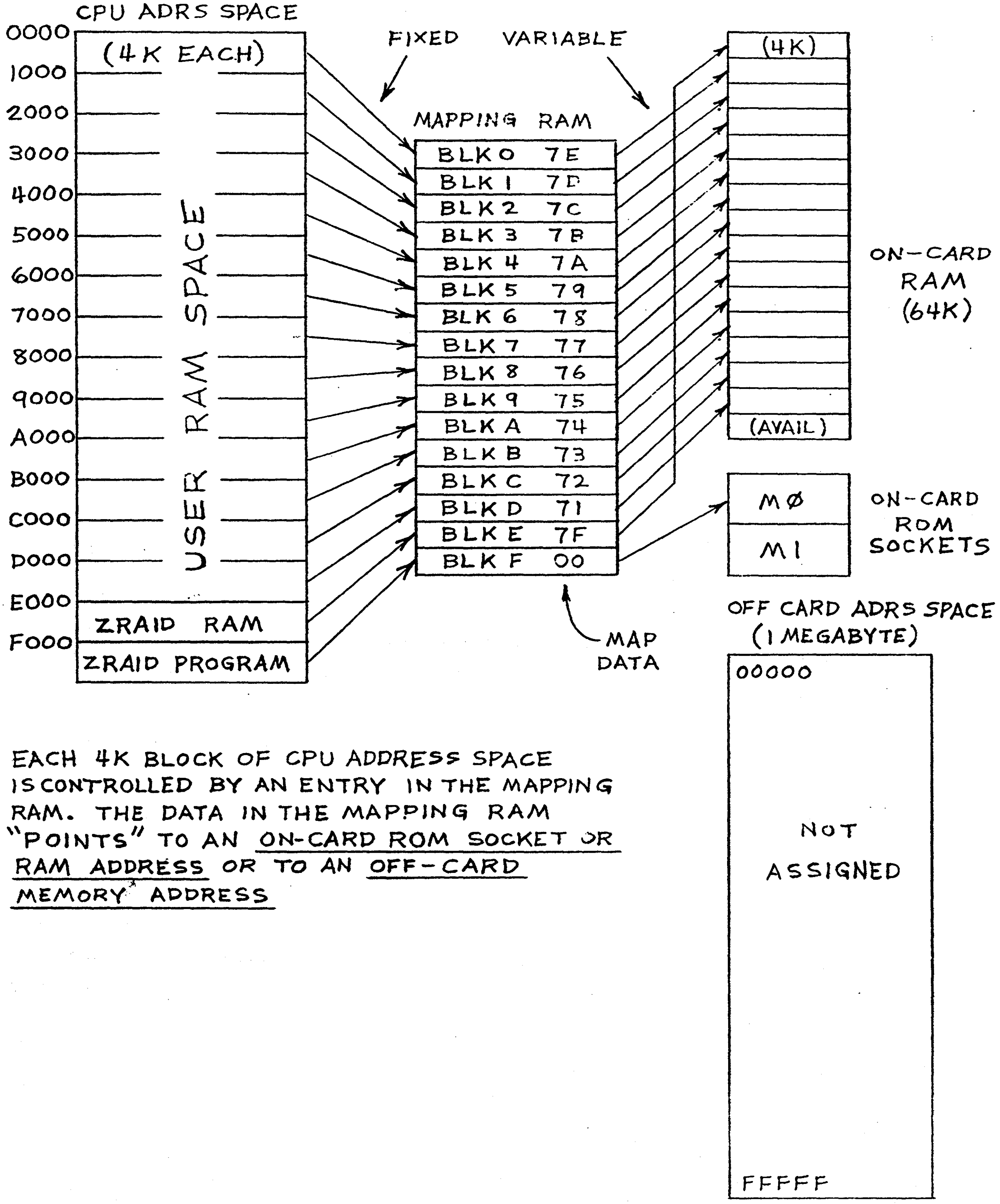
If DIP switches installed:

Board is assigned to the bus block (Ø - F) as specified by switches 5, 6, 7 and 8 of DIP switch group Ø. Otherwise, the board is assigned to block Ø (default). In either case, all board operations are enabled (i.e., Memory RD, WR and I/O operations are valid.)

D. SIO Baud rates:

If DIP switches installed, the baud rates are set according to DIP switch group 1 as shown on page 93. (See also the ZRAID manual.) Otherwise, both SIO port baud rates are set at 9600 baud (default).

Note: These values may be modified by special ZRAID commands or the initial values may be changed in the ZRAID ROM.



MAPPING RAM	
BLK 0	7E
BLK 1	7D
BLK 2	7C
BLK 3	7B
BLK 4	7A
BLK 5	79
BLK 6	78
BLK 7	77
BLK 8	76
BLK 9	75
BLK A	74
BLK B	73
BLK C	72
BLK D	71
BLK E	7F
BLK F	00

EACH 4K BLOCK OF CPU ADDRESS SPACE IS CONTROLLED BY AN ENTRY IN THE MAPPING RAM. THE DATA IN THE MAPPING RAM "POINTS" TO AN ON-CARD ROM SOCKET OR RAM ADDRESS OR TO AN OFF-CARD MEMORY ADDRESS

ZRAID-91 INITIAL MEMORY MAP

ZRAID Command Summary (Partial listing)

Command	Function	Example
Hnn	Set upper eight bits of POINTER	H45
Lnn	Set lower eight bits of POINTER	LA5
A	Print POINTER value in H, L, format	A
Snnnn	Set POINTER using 4 character hex value (or 16-bit octal value)	SC709
.nn	Set addressed location	.5A
W	Print contents of addressed location	W
I	Increment POINTER, print location	I
D	Decrement POINTER, print location	D
*	Transfer control to POINTER address via JUMP	*
C	Transfer control to POINTER address via CALL	C
J	Indirect CALL	J
Pnn	Print-nn lines (nn in hex or octal)	P2
X	Set/Reset octal/hex I/O mode	X
Rn	Insert a RST instruction	R2
U	Remove the last RST instruction	U
'	(Apostrophe) CP/M Bootstrap	'
Fnn	Output to I/O device L	F12
Y	Input from I/O device L	Y
E	Blink USER LED	E
=	Set contents of memory mapping RAM	=ØØ
@	Set contents of I/O mapping RAM	@7
B	Set contents of bus mapping RAM	BFØ
"	Set system bus mode	"3
/	Cancel previous input	H152/

All commands except "/" must be followed by a space or carriage return to cause execution to begin.

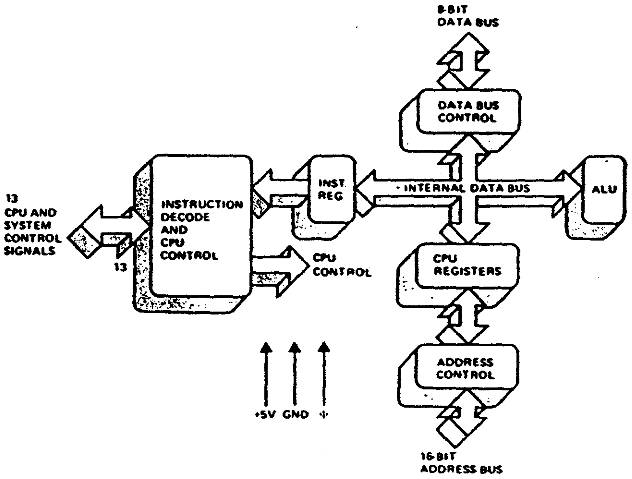
The above list shows only the most used commands. Additional commands are provided to allow disk I/O and other functions. Since all commands must be used in the proper sequence, refer to the ZRAID Manual for details before attempting to use ZRAID.

CPU DESCRIPTION

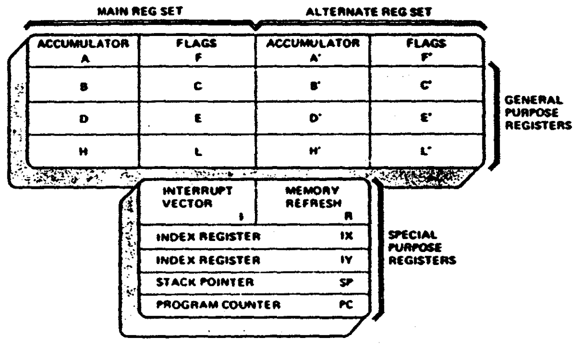
The Zilog Z-80 CPU is a powerful single-chip 8-bit microprocessor. All of the original Intel 8080 instructions are executable by the Z-80 while adding numerous other instructions and internal registers, which give added versatility. A block diagram of the CPU and the internal register detail are shown below.

The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or as 16-bit register pairs. There are also two sets of accumulator and flag registers. The programmer has access to either set of main or alternate registers through a group of exchange instructions. This alternate set allows foreground/background mode of operation or may be reserved for very fast Interrupt response. The CPU also contains a 16-bit stack pointer which permits simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data handling.

The two 16-bit index registers allow tabular data manipulation and easy implementation of relocatable code. The I register is used in a powerful interrupt response mode to form the upper 8 bits of a pointer to a interrupt service address table, while the interrupting device supplies the lower 8 bits of the pointer. An indirect call is then made to this service address.



Z80, Z80A CPU BLOCK DIAGRAM



MAIN REG SET		ALTERNATE REG SET	
ACCUMULATOR A	FLAGS F	ACCUMULATOR A'	FLAGS F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

INTERRUPT VECTOR I	MEMORY REFRESH R
INDEX REGISTER IX	
INDEX REGISTER IY	
STACK POINTER SP	
PROGRAM COUNTER PC	

Z80, Z80A CPU REGISTERS

Z-80® CPU Z-80A CPU

Zilog

Instruction Set

The following is a summary of the Z80, Z80A instruction set showing the assembly language mnemonic and the symbolic operation performed by the instruction. A more detailed listing appears in the Z80-CPU technical manual, and assembly language programming manual. The instructions are divided into the following categories:

- 8-bit loads
- 16-bit loads
- Exchanges
- Memory Block Moves
- Memory Block Searches
- 8-bit arithmetic and logic
- 16-bit arithmetic
- General purpose Accumulator & Flag Operations
- Miscellaneous Group
- Rotates and Shifts
- Bit Set, Reset and Test
- Input and Output
- Jumps
- Calls
- Restarts
- Returns

In the table the following terminology is used.

b ≡ a bit number in any 8-bit register or memory location

cc ≡ flag condition code

NZ	≡	non zero
Z	≡	zero
NC	≡	non carry
C	≡	carry
PO	≡	Parity odd or no over flow
PE	≡	Parity even or over flow
P	≡	Positive
M	≡	Negative (minus)

d	≡	any 8-bit destination register or memory location
dd	≡	any 16-bit destination register or memory location
e	≡	8-bit signed 2's complement displacement used in relative jumps and indexed addressing
L	≡	8 special call locations in page zero. In decimal notation these are 0, 8, 16, 24, 32, 40, 48 and 56
n	≡	any 8-bit binary number
nn	≡	any 16-bit binary number
r	≡	any 8-bit general purpose register (A, B, C, D, E, H, or L)
s	≡	any 8-bit source register or memory location
s_b	≡	a bit in a specific 8-bit register or memory location
ss	≡	any 16-bit source register or memory location
subscript "L"	≡	the low order 8 bits of a 16-bit register
subscript "H"	≡	the high order 8 bits of a 16-bit register
( )	≡	the contents within the ( ) are to be used as a pointer to a memory location or I/O port number

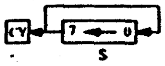
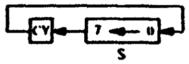
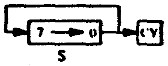
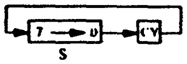
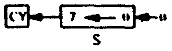
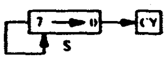
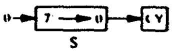
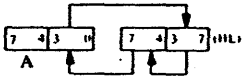
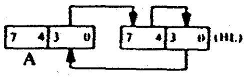
8-bit registers are A, B, C, D, E, H, L, I and R
16-bit register pairs are AF, BC, DE and HL
16-bit registers are SP, PC, IX and IY

Addressing Modes implemented include combinations of the following:

Immediate	Indexed
Immediate extended	Register
Modified Page Zero	Implied
Relative	Register Indirect
Extended	Bit

	Mnemonic	Symbolic Operation	Comments
8-BIT LOADS	LD r, s	r ← s	s ≡ r, n, (HL), (IX+e), (IY+e)
	LD d, r	d ← r	d ≡ (HL), r (IX+e), (IY+e)
	LD d, n	d ← n	d ≡ (HL), (IX+e), (IY+e)
	LD A, s	A ← s	s ≡ (BC), (DE), (nn), I, R
	LD d, A	d ← A	d ≡ (BC), (DE), (nn), I, R
16-BIT LOADS	LD dd, nn	dd ← nn	dd ≡ BC, DE, HL, SP, IX, IY
	LD dd, (nn)	dd ← (nn)	dd ≡ BC, DE, HL, SP, IX, IY
	LD (nn), ss	(nn) ← ss	ss ≡ BC, DE, HL, SP, IX, IY
	LD SP, ss	SP ← ss	ss = HL, IX, IY
	PUSH ss	(SP-1) ← ss_H; (SP-2) ← ss_L	ss = BC, DE, HL, AF, IX, IY
	POP dd	dd_L ← (SP); dd_H ← (SP+1)	dd = BC, DE, HL, AF, IX, IY
EXCHANGES	EX DE, HL	DE ↔ HL	
	EX AF, AF'	AF ↔ AF'	
	EXX	$\begin{pmatrix} BC \\ DE \\ HL \end{pmatrix} \leftrightarrow \begin{pmatrix} BC' \\ DE' \\ HL' \end{pmatrix}$	
	EX (SP), ss	(SP) ↔ ss_L, (SP+1) ↔ ss_H	ss ≡ HL, IX, IY

	Mnemonic	Symbolic Operation	Comments
MEMORY BLOCK MOVES	LDI	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1	
	LDIR	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1 Repeat until BC = 0	
	LDD	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1	
	LDDR	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1 Repeat until BC = 0	
MEMORY BLOCK SEARCHES	CPI	A-(HL), HL ← HL+1 BC ← BC-1	
	CPIR	A-(HL), HL ← HL+1 BC ← BC-1, Repeat until BC = 0 or A = (HL)	A-(HL) sets the flags only. A is not affected
	CPD	A-(HL), HL ← HL-1 BC ← BC-1	
	CPDR	A-(HL), HL ← HL-1 BC ← BC-1, Repeat until BC= 0 or A = (HL)	
8-BIT ALU	ADD s	A ← A + s	
	ADC s	A ← A + s + CY	CY is the carry flag
	SUB s	A ← A - s	
	SBC s	A ← A - s - CY	s ≡ r, n, (HL) (IX+e), (IY+e)
	AND s	A ← A ∧ s	
	OR s	A ← A ∨ s	
	XOR s	A ← A ⊕ s	

	Mnemonic	Symbolic Operation	Comments
8-BIT ALU	CP s	A - s	s = r, n (HL) (IX+e), (IY+e)
	INC d	d ← d + 1	d = r, (HL) (IX+e), (IY+e)
	DEC d	d ← d - 1	
16-BIT ARITHMETIC	ADD HL, ss	HL ← HL + ss	ss ≡ BC, DE HL, SP
	ADC HL, ss	HL ← HL + ss + CY	
	SBC HL, ss	HL ← HL - ss - CY	
	ADD IX, ss	IX ← IX + ss	ss ≡ BC, DE, IX, SP
	ADD IY, ss	IY ← IY + ss	ss ≡ BC, DE, IY, SP
	INC dd	dd ← dd + 1	dd ≡ BC, DE, HL, SP, IX, IY
	DEC dd	dd ← dd - 1	dd ≡ BC, DE, HL, SP, IX, IY
GP ACC. & FLAG	DAA	Converts A contents into packed BCD following add or subtract.	Operands must be in packed BCD format
	CPL	A ← $\overline{A}$	
	NEG	A ← 00 – A	
	CCF	CY ← $\overline{CY}$	
	SCF	CY ← 1	
MISCELLANEOUS	NOP	No operation	
	HALT	Halt CPU	
	DI	Disable Interrupts	
	EI	Enable Interrupts	
	IM 0	Set interrupt mode 0	8080A mode
	IM 1	Set interrupt mode 1	Call to 0038_H
	IM 2	Set interrupt mode 2	Indirect Call
ROTATES AND SHIFTS	RLC s		s ≡ r, (HL) (IX+e), (IY+e)
	RL s		
	RRC s		
	RR s		
	SLA s		
	SRA s		
	SRL s		
	RLD		
	RRD		

	Mnemonic	Symbolic Operation	Comments
BIT S, R, & T	BIT b, s	Z ← $\overline{s_b}$	Z is zero flag
	SET b, s	s_b ← 1	s ≡ r, (HL)
	RES b, s	s_b ← 0	(IX+e), (IY+e)
INPUT AND OUTPUT	IN A, (n)	A ← (n)	
	IN r, (C)	r ← (C)	Set flags
	INI	(HL) ← (C), HL ← HL + 1 B ← B - 1	
	INIR	(HL) ← (C), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	IND	(HL) ← (C), HL ← HL - 1 B ← B - 1	
	INDR	(HL) ← (C), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
	OUT(n), A	(n) ← A	
	OUT(C), r	(C) ← r $ADRS_H$ ← B	
	OUTI	(C) ← (HL), HL ← HL + 1 B ← B - 1	
	OTIR	(C) ← (HL), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	OUTD	(C) ← (HL), HL ← HL - 1 B ← B - 1	
	OTDR	(C) ← (HL), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
JUMPS	JP nn	PC ← nn	cc { NZ PO Z PE NC P C M
	JP cc, nn	If condition cc is true PC ← nn, else continue	
	JR e	PC ← PC + e	
	JR kk, e	If condition kk is true PC ← PC + e, else continue	kk { NZ NC Z C
	JP (ss)	PC ← ss	ss = HL, IX, IY
	DJNZ e	B ← B - 1, if B = 0 continue, else PC ← PC + e	
CALLS	CALL nn	(SP-1) ← PC_H (SP-2) ← PC_L, PC ← nn	cc { NZ PO Z PE NC P C M
	CALL cc, nn	If condition cc is false continue, else same as CALL nn	
RESTARTS	RST L	(SP-1) ← PC_H (SP-2) ← PC_L, PC_H ← 0 PC_L ← L	
RETURNS	RET	PC_L ← (SP), PC_H ← (SP+1)	
	RET cc	If condition cc is false continue, else same as RET	cc { NZ PO Z PE NC P C M
	RETI	Return from interrupt, same as RET	
	RETN	Return from non-maskable interrupt	

Instruction Set Map

	0	1	2	3	4	5	6	7
00	NOP NOP 000-00 4A	LD BC,nn LXI B.nn 001-01 10D	LD (BC),A STAX B 002-02 7A	INC BC INX B 003-03 6A	INC B INR B 004-04 4A--SZV	DEC B DCR B 005-05 4A--SZV	LD B,n MVI B.n 006-06 7B	RLCA RLC 007-07 4A--C
01	EX AF,AF — 010-08 4A--SZVC	ADD *,BC DAD B 011-09 11A/15E--C	LD A,(BC) LDAX B 012-0A 7A	DEC BC DCX B 013-0B 6A	INC C INR C 014-0C 4A--SZV	DEC C DCR C 015-0D 4A--SZV	LD C,n MVI C.n 016-0E 7B	RRCA RRC 017-0F 4A--C
02	DJNZ,e — 020-10 8C/13C	LD DE,nn LXI D.nn 021-11 10D	LD (DE),A STAX D 022-12 7A	INC DE INX D 023-13 6A	INC D INR D 024-14 4A--SZV	DEC D DCR D 025-15 4A--SZV	LD D,n MVI D.n 026-16 7B	RLA RAL 027-17 4A--C
03	JR e — 030-18 12C	ADD *,DE DAD D 031-19 11A/15E--C	LD A,(DE) LDAX D 032-1A 7A	DEC DE DCX D 033-1B 6A	INC E INR E 034-1C 4A--SZV	DEC E DCR E 035-1D 4A--SZV	LD E,n MVI E.n 036-1E 7B	RRA RAR 037-1F 4A--C
04	JP NZ,e — 040-20 7C/12C	LD *,nn LXI H.nn 041-21 10D/14J	LD (nn),* SHLD 042-22 16A/20J	INC * INX H 043-23 6A/10E	INC H INR H 044-24 4A--SZV	DEC H DCR H 045-25 4A--SZV	LD H,n MVI H.n 046-26 7B	DAA DAA 047-27 4A--SZPC
05	JR Z,e — 050-28 7C/12C	ADD *,HL DAD H 051-29 11A/15E--C	LD *,(nn) LHLD 052-2A 16A/20J	DEC * DCX H 053-2B 6A/10E	INC L INR L 054-2C 4A--SZV	DEC L DCR L 055-2D 4A--SZV	LD L,n MVI L.n 056-2E 7R	CPL CMA 057-2F 4A
06	JR NC,e — 060-30 7C/12C	LD SP,nn LXI SP.nn 061-31 10D	LD (nn),A STA 062-32 13D	INC SP INX SP 063-33 6A	INC (*) INR M 064-34 11A/23G--SZV	DEC (*) DCR M 065-35 11A/23G--SZV	LD (*),n MVI M.n 066-36 10B/19H	SCF STC 067-37 4A--C
07	JR C,e — 070-38 7C/12C	ADD *,SP DAD SP 071-39 11A/15E--C	LD A,(nn) LDA 072-3A 13D	DEC SP DCX SP 073-3B 6A	INC A INR A 074-3C 4A--SZV	DEC A DCR A 075-3D 4A--SZV	LD A,n MVI A.n 076-3E 7B	CCF CMC 077-3F 4A--C
10	LD B,B MOV B.B 100-40 4A	LD B,C MOV B.C 101-41 4A	LD B,D MOV B.D 102-42 4A	LD B,E MOV B.E 103-43 4A	LD B,H MOV B.H 104-44 4A	LD B,L MOV B.L 105-45 4A	LD B,(*) MOV B.M 106-46 7A/19G	LD B,A MOV B.A 107-47 4A
11	LD C,B MOV C.B 110-48 4A	LD C,C MOV C.C 111-49 4A	LD C,D MOV C.D 112-4A 4A	LD C,E MOV C.E 113-4B 4A	LD C,H MOV C.H 114-4C 4A	LD C,L MOV C.L 115-4D 4A	LD C,(*) MOV C.M 116-4E 7A/19G	LD C,A MOV C.A 117-4F 4A
12	LD D,B MOV D.B 120-50 4A	LD D,C MOV D.C 121-51 4A	LD D,D MOV D.D 122-52 4A	LD D,E MOV D.E 123-53 4A	LD D,H MOV D.H 124-54 4A	LD D,L MOV D.L 125-55 4A	LD D,(*) MOV D.M 126-56 7A/19G	LD D,A MOV D.A 127-57 4A
13	LD E,B MOV E.B 130-58 4A	LD E,C MOV E.C 140-61 4A	LD E,D MOV E.D 132-5A 4A	LD E,E MOV E.E 133-5B 4A	LD E,H MOV E.H 134-5C 4A	LD E,L MOV E.L 135-5D 4A	LD E,(*) MOV E.M 136-5E 7A/19G	LD E,A MOV E.A 137-5F 4A
14	LD H,B MOV H.B 140-60 4A	LD H,C MOV H.C 140-61 4A	LD H,D MOV H.D 142-62 4A	LD H,E MOV H.E 143-63 4A	LD H,H MOV H.H 144-64 4A	LD H,L MOV H.L 145-65 4A	LD H,(*) MOV H.M 146-66 7A/19G	LD H,A MOV H.A 147-67 4A
15	LD L,B MOV L.B 150-68 4A	LD L,C MOV L.C 151-69 4A	LD L,D MOV L.D 152-6A 4A	LD L,E MOV L.E 153-6B 4A	LD L,H MOV L.H 154-6C 4A	LD L,L MOV L.L 155-6D 4A	LD L,(*) MOV L.M 156-6E 7A/19G	LD L,A MOV L.A 157-6F 4A
16	LD (*),B MOV M.B 160-70 7A/19G	LD (*),C MOV M.C 161-71 7A/19G	LD (*),D MOV M.D 162-72 7A/19G	LD (*),E MOV M.E 163-73 7A/19G	LD (*),H MOV M.H 164-74 7A/19G	LD (*),L MOV M.L 165-75 7A/19G	HALT HALT 166-76	LD (*),A MOV M.A 167-77 7A/19G
17	LD A,B MOV A.B 170-78 4A	LD A,C MOV A.C 171-79 4A	LD A,D MOV A.D 172-7A 4A	LD A,E MOV A.E 173-7B 4A	LD A,H MOV A.H 174-7C 4A	LD A,L MOV A.L 175-7D 4A	LD A,(*) MOV A.M 176-7E 7A/19G	LD A,A MOV A.A 177-7F 4A

	0	1	2	3	4	5	6	7
20	ADD B ADD B 200-80 4A--SZVC	ADD C ADD C 201-81 4A--SZVC	ADD D ADD D 202-82 4A--SZVC	ADD E ADD E 203-83 4A--SZVC	ADD H ADD H 204-84 4A--SZVC	ADD L ADD L 205-85 4A--SZVC	ADD (*) ADD M 206-86 7A/19G--SZVC	ADD A ADD A 207-87 4A--SZVC
21	ADC B ADC B 210-88 4A--SZVC	ADC C ADC C 211-89 4A--SZVC	ADC D ADC D 212-8A 4A--SZVC	ADC E ADC E 213-8B 4A--SZVC	ADC H ADC H 214-8C 4A--SZVC	ADC L ADC L 215-8D 4A--SZVC	ADC (*) ADC M 216-8E 7A/19G--SZVC	ADC A ADC A 217-8F 4A--SZVC
22	SUB B SUB B 220-90 4A--SZVC	SUB C SUB C 221-91 4A--SZVC	SUB D SUB D 222-92 4A--SZVC	SUB E SUB E 223-93 4A--SZVC	SUB H SUB H 224-94 4A--SZVC	SUB L SUB L 225-95 4A--SZVC	SUB (*) SUB M 226-96 7A/19G--SZVC	SUB A SUB A 227-97 4A--SZVC
23	SBC B SBB B 230-98 4A--SZVC	SBC C SBB C 231-99 4A--SZVC	SBC D SBB D 232-9A 4A--SZVC	SBC E SBB E 233-9B 4A--SZVC	SBC H SBB H 234-9C 4A--SZVC	SBC L SBB L 235-9D 4A--SZVC	SBC (*) SBB M 236-9E 7A/19G--SZVC	SBC A SBB A 237-9F 4A--SZVC
24	AND B ANA B 240-A0 4A--SZPC	AND C ANA C 241-A1 4A--SZPC	AND D ANA D 242-A2 4A--SZPC	AND E ANA E 243-A3 4A--SZPC	AND H ANA H 244-A4 4A--SZPC	AND L ANA L 245-A5 4A--SZPC	AND (*) ANA M 246-A6 7A/19G--SZPC	AND A ANA A 247-A7 4A--SZPC
25	XOR B XRA B 250-A8 4A--SZPC	XOR C XRA C 251-A9 4A--SZPC	XOR D XRA D 252-AA 4A--SZPC	XOR E XRA E 253-AB 4A--SZPC	XOR H XRA H 254-AC 4A--SZPC	XOR L XRA L 255-AD 4A--SZPC	XOR (*) XRA M 256-AE 7A/19G--SZPC	XOR A XRA A 257-AF 4A--SZPC
26	OR B ORA B 260-B0 4A--SZPC	OR C ORA C 261-B1 4A--SZPC	OR D ORA D 262-B2 4A--SZPC	OR E ORA E 263-B3 4A--SZPC	OR H ORA H 264-B4 4A--SZPC	OR L ORA L 265-B5 4A--SZPC	OR (*) ORA M 266-B6 7A/19G--SZPC	OR A ORA A 267-B7 4A--SZPC
27	CP B CMLP B 270-B8 4A--SZVC	CP C CMP C 271-B9 4A--SZVC	CP D CMP D 272-BA 4A--SZVC	CP E CMP E 273-BB 4A--SZVC	CP H CMP H 274-BC 4A--SZVC	CP L CMP L 275-BD 4A--SZVC	CP (*) CMP M 276-BE 7A/19G--SZVC	CP A CMP A 277-BF 4A--SZVC
30	RET NZ RNZ 300-C0 5A/11A	POP BC POP B 301-C1 10A	JP NZ,nn JNZ 302-C2 10D	JP nn JMP 303-C3 10D	CALL NZ,nn CNZ 304-C4 10D/17D	PUSH BC PUSH B 305-C5 11A	ADD n ADI 306-C6 7B--SZVC	RST Ø RST Ø 307-C7 11A
31	RET Z RZ 310-C8 5A/11A	RET RET 311-C9 10A	JP Z,nn JZ 312-CA 10D	(SPECIAL) (PREFIX) 313-CB	CALL Z,nn CZ 314-CC 10D/17D	CALL nn CALL 315-CD 17D	ADC n ACI 316-CE 7B--SZVC	RST 1 RST 1 317-CF 11A
32	RET NC RNC 320-D0 5A/11A	POP DE POP D 321-D1 10A	JP NC,nn JNC 322-D2 10D	OUT (n),A OUT 323-D3 11B	CALL NC,nn CNC 324-D4 10D/17D	PUSH DE PUSH D 325-D5 11A	SUB n SUI 326-D6 7B--SZVC	RST 2 RST 2 327-D7 11A
33	RET C RC 330-D8 5A/11A	EXX — 331-D9 4A	JP C,nn JC 332-DA 10D	IN A,(n) IN 333-DB 10B	CALL C,nn CC 334-DC 10D/17D	**IX** (PREFIX) 335-DD	SBC n SBI 336-DE 7B--SZVC	RST 3 RST 3 337-DF 11A
34	RET PO RPO 340-E0 5A/11A	POP * POP H 341-E1 10A/14E	JP PO,nn JPO 342-E2 10D	EX (SP),* XTHL 343-E3 19A/23E	CALL PO,nn CPO 344-E4 10D/17D	PUSH * PUSH H 345-E5 11A/15E	AND n ANI 346-E6 7B--SZPC	RST 4 RST 4 347-E7 11A
35	RET PE RPE 350-E8 5A/11A	JP (*) PCHL 351-E9 4A/8E	JP PE,nn JPE 352-EA 10D	EX DE,HL XCHG 353-EB 4A	CALL PE,nn CPE 354-EC 10D/17D	**** (PREFIX) 355-ED	XOR n XRI 356-EE 7B--SZPC	RST 5 RST 5 357-EF 11A
36	RET P RP 360-F0 5A/11A	POP AF POP PSW 361-F1 10A--SZVC	JP P,nn JP 362-F2 10D	DI — 363-F3 4A	CALL P,nn CP 364-F4 10D/17D	PUSH AF PUSH PSW 365-F5 11A	OR n ORI 366-F6 7B--SZPC	RST 6 RST 6 367-F7 11A
37	RET M RM 370-F8 5A/11A	LD SP,* SPHL 371-F9 6A/10E	JP M,nn JM 372-FA 10D	EI — 373-FB 4A	CALL M,nn CM 374-FC 10D/17D	**IY** (PREFIX) 375-FD	CP n CPI 376-FE 7B--SZVC	RST 7 RST 7 377-FF 11A

A	OPCODE			
B	OPCODE	Operand		
C	OPCODE	Displacement		
D	OPCODE	Operand L	Operand H	
E	OPCODE1	OPCODE		
F	OPCODE1	OPCODE	Operand	
G	OPCODE1	OPCODE	Displacement	
H	OPCODE1	OPCODE	Displacement	Operand
J	OPCODE1	OPCODE	Operand L	Operand H

OPCODE1 = 335-DD For IX Operand
= 375-ED For IY Operand

* MEANS HL, IX, or IY
(*) MEANS (HL), (IX + d), or (IY + d)

	0	1	2	3	4	5	6	7
10	IN B,(C) 100-40 11K--SZP	OUT (C),B 101-41 12K	SBC HL,BC 102-42 15K--SZVC	LD (NN),BC 103-43 20L	NEG 104-44 8K--CZVS	RETN 105-45 14K	IM 0 106-46 8K	LD I,A 107-47 9K
11	IN C,(C) 110-48 11K--SZP	OUT (C),C 111-49 12K	ADC HL,BC 112-4A 15K--SZVC	LD BC,(nn) 113-4B 20L		RETI 115-4D 14K		LD R,A 117-4F 9K
12	IN D,(C) 120-50 11K--SZP	OUT (C),D 121-51 12K	SBC HL,DE 122-52 15K--SZVC	LD (nn),DE 123-53 20L			IM 1 126-56 8K	LD A,I 127-57 9K--SZV
13	IN E,(C) 130-58 11K--SZP	OUT (C),E 131-59 12K	ADC HL,DE 132-5A 15K--SZVC	LD DE,(nn) 133-5B 20L			IM 2 136-5E 8K	LD A,R 137-5F 9K--SZV
14	IN H,(C) 140-60 11K--SZP	OUT (C),H 141-61 12K	SBC HL,HL 142-62 15K--SZVC	LD (nn),HL 143-63 20L				RRD 147-67 18K--SZP
15	IN L,(C) 150-68 11K--SZP	OUT (C),L 151-69 12K	ADC HL,HL 152-6A 15K--SZVC	LD HL,(nn) 153-6B 20L				RLD 157-6F 18K--SZP
16	IN F,(C) 160-70 11K--SZP	OUT (C),F 161-71 12K	SBC HL,SP 162-72 15K--SZVC	LD (nn),SP 163-73 20L				
17	IN A,(C) 170-78 11K--SZP	OUT (C),A 171-79 12K	ADC HL,SP 172-7A 15K--SZVC	LD SP,(nn) 173-7B 20L				
24	LDI 240-A0 16K--P(SZ)	CPI 241-A1 16K--SZP	INI 242-A2 15K--Z(SP)	OUTI 243-A3 15K--Z(SP)				
25	LDD 250-A8 16K--P(SZ)	CPD 251-A9 16K--SZP	IND 252-AA 15K--Z(SP)	OUTD 253-AB 15K--Z(SP)				
26	LDIR 260-B0 21M/16K--P(SZ)	CPIR 261-B1 21M/16K--SZP	INIR 262-B2 20M/15K--Z(SP)	OTIR 263-B3 20M/15K--Z(SP)				
27	LDDR 270-B8 21M/16K--P(SZ)	CPDR 271-B9 21M/16K--SZP	INDR 272-BA 20M/15K--Z(SP)	OTDR 273-B8 20M/15K--Z(SP)				

(HL) = SOURCE (HL) = ADDRESS
(DE) = DESTINATION (C) = DEVICE
BC = LENGTH B = LENGTH

355 PREFIX GROUP

K 355-ED OPCODE
L 355-ED OPCODE Operand L Operand H
M 355-ED OPCODE Timing when BC≠0

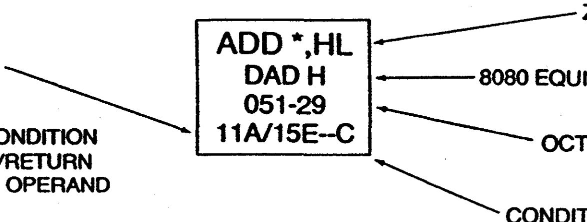
313 PREFIX GROUP

RLC r 313/00r 8B/23H--SZPC	RRC r 313/01r 8B/23H--SZPC	RL r 313/02r 8B/23H--SZPC	RR r 313/03r 8B/23H--SZPC
SLA r 313/04r 8B/23H--SZPC	SRA r 313/05r 8B/23H--SZPC	— 313/0x6 15B	SRL r 313/07r 8B/23H--SZPC
BIT/RES/SET 313/xb6 12B	BIT b,r 313/1br 8B/20H--Z(SP)	RES b,r 313/2br 8B/20H	SET b,r 313/3br 8B/20H

SLASH INDICATES TWO WORD OPCODE
"b" = "bit" 7 = MSB, 0 = LSB
"r" = "register" SEE REGISTER LIST

r	REGISTER
0	B
1	C
2	D
3	E
4	H
5	L
6	(HL)
7	A

Z80 MACHINE CYCLES AND INSTRUCTION FORMAT.
IF TWO SETS OF VALUES:
FIRST ("11A") IS FALSE CONDITION TIMING FOR JUMP/CALL/RETURN INSTRUCTIONS, OR FOR OPERAND "*" = HL.
SECOND ("15E") IS TRUE CONDITION TIMING FOR JUMP/CALL/RETURN INSTRUCTIONS, OR FOR OPERAND "*" = IX or IY.
SEE INSTRUCTION FORMAT LISTING



Z80 MNEMONIC

8080 EQUIVALENT MNEMONIC

OCTAL-HEX OPCODE

CONDITION CODE FLAGS AFFECTED
S = SIGN FLAG
Z = ZERO FLAG
P = PARITY FLAG
V = OVERFLOW FLAG
C = CARRY FLAG
FLAGS ENCLOSED IN PARENTHESIS ARE AFFECTED BUT NOT DETERMINATE

FLAG REGISTER

7	6	5	4	3	2	1	0
S	Z		H		P/V	N	C

* MEANS HL, IX or IY.
(*) MEANS (HL), (IX+d), or (IY+d)

(This page left blank intentionally)

INTRODUCTION TO THE MLZ-91 MAPPING RAMS

The MLZ-91 has three mapping RAMs which are used to dynamically allocate the resources of the MLZ-91.

The mapping RAMs and their functions are:

1. Memory Mapping RAM

   Controls the allocation of all memory and allows the CPU/DMA (which has a 16 bit address bus) to access a full megabyte of memory (which requires a 20 bit address). In addition, the memory mapping RAM controls the memory write protect feature for on-card RAM.

2. I/O Device Mapping RAM

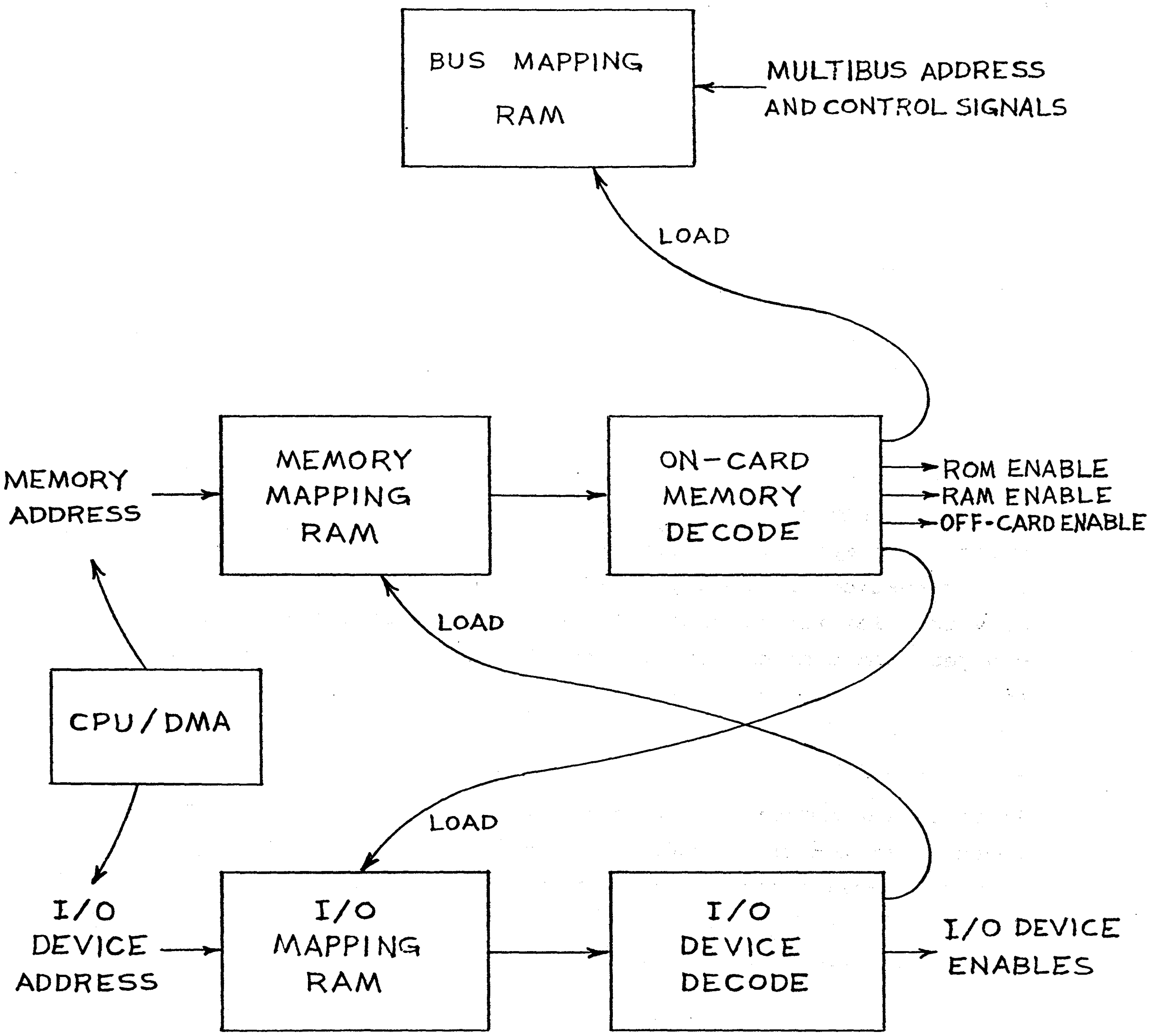
   Specifies the base addresses for the on-card I/O devices. Also specifies the regions of the I/O device address space (8 bits) which are to be used for on-card devices and which are for off-card devices. This feature allows a system to be configured with more than 256 device addresses from being masked by on-card devices. On-card devices may be "shadowed" or moved to different base addresses.

3. Bus Mapping RAM

   This RAM is used to assign the MLZ-91 to a spot on the Multibus. The 20-bit Multibus is divided into 16 regions, specified by the upper four bits. Each MLZ-91 on the Multibus can be assigned to any region or regions. In addition, the mapping RAM specifies what type of operation from the bus is allowed. Memory read, Memory write and I/O device access may each be enabled or disabled in each block.

Since the contents of the mapping RAMs are controllable by software, the program may allocate resources as necessary for the particular application.

Detailed descriptions of each of these mapping RAMs and software examples of their use are presented in the following sections of this manual. Summary information appears on page 47 and software examples start on page 80.



SIMPLIFIED DIAGRAM OF MAPPING RAMS

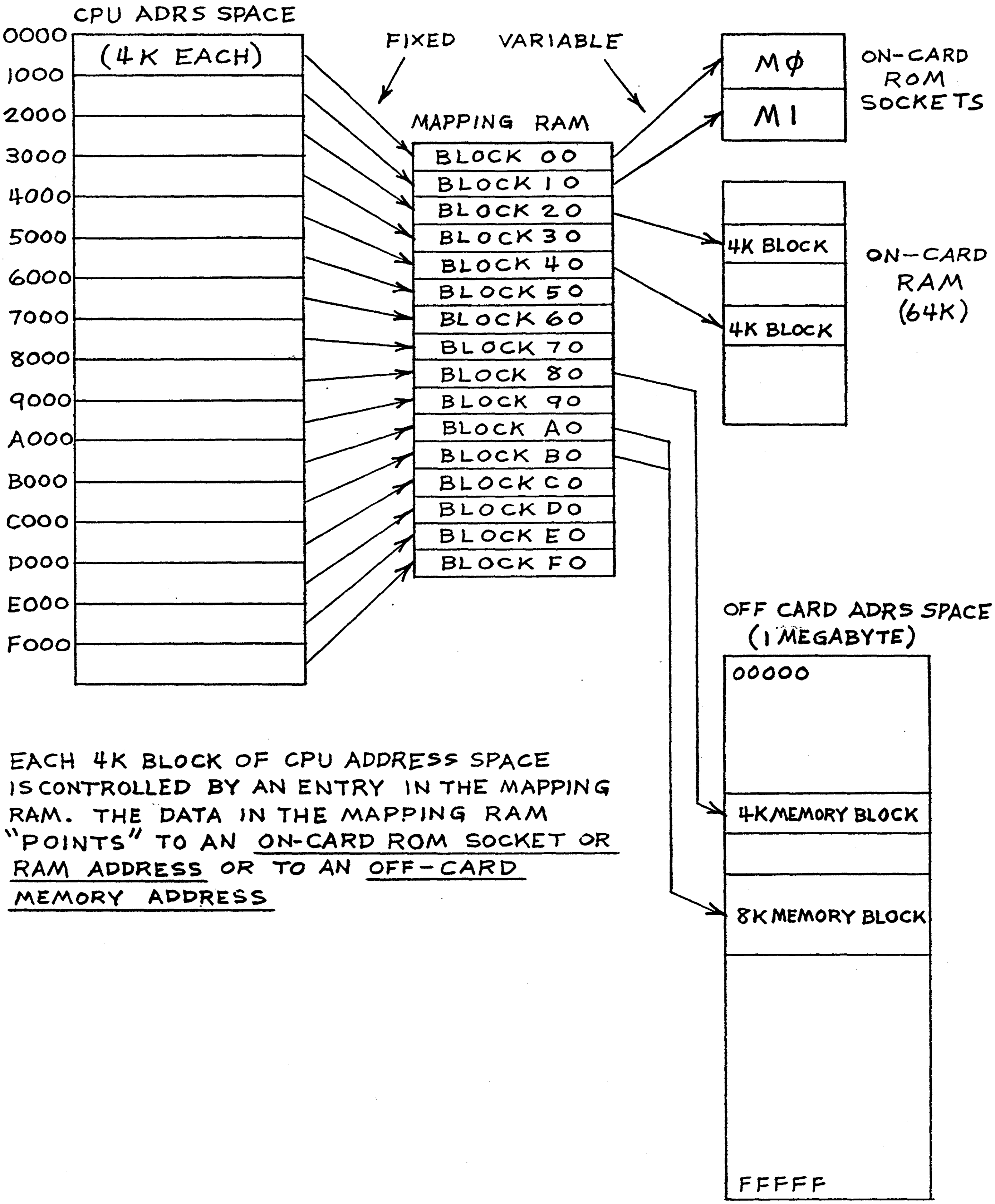
MEMORY MANAGEMENT

Introduction

The MLZ-91 has the capacity to address over one megabyte of memory. This is accomplished by using a system address bus which is 20 bits wide. The CPU and DMA chips, however, are designed using 16 bit internal registers and a 16 bit address bus. The memory mapping logic is the link between these two address buses.

The MLZ-91 memory mapping RAM allows the full 20 bit address bus to be utilized. The mapping RAM output provides the upper four bits plus three of the remaining 16 bits (seven total). The four upper address lines from the CPU, instead of going directly to the address bus, are used to select a location within the mapping RAM. The mapping RAM can be preloaded with various combinations of the upper seven bits. Then, the CPU address specifies the mapping RAM address plus the lower address bus bits. The RAM provides the other upper seven address bits. To switch from one preset block to another all that is required is a variation in the upper CPU address lines, a relatively easy task for the program. The mapping RAM contents may be changed from time to time to keep the most frequently used upper address line combinations always available. (The eighth bit from the mapping RAM is used to designate an on-card or an off-card address.)

Think of the mapping RAM as two, unequal sized funnels attached together at the small ends. The 16 bit CPU address bus feeds into the small funnel. After passing through the mapping RAM (represented by the junction of the funnels) the address bus is expanded into the full 20-bit space.



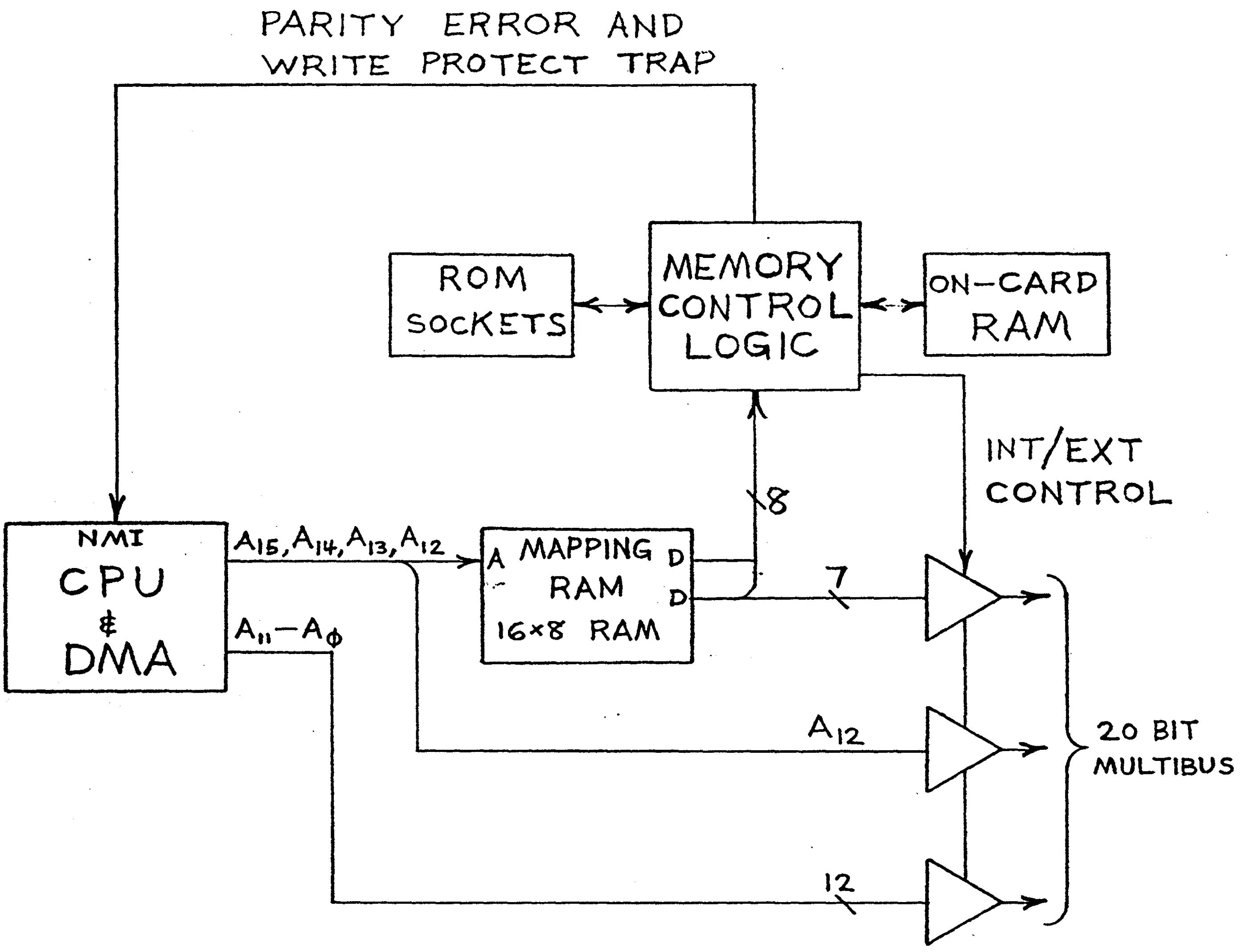
EACH 4K BLOCK OF CPU ADDRESS SPACE IS CONTROLLED BY AN ENTRY IN THE MAPPING RAM. THE DATA IN THE MAPPING RAM "POINTS" TO AN ON-CARD ROM SOCKET OR RAM ADDRESS OR TO AN OFF-CARD MEMORY ADDRESS

MEMORY MAPPING LOGIC

As a direct result of using this method, the MLZ-91 allows easy implementation of multi-tasking systems because each task, or user, can be assigned a section of memory which would only be allocated by the mapping RAM when that task was active. Memory for the inactive tasks would therefore not be wasting any of the CPU's address space.

DMA data transfers could always be made to only one or two blocks as assigned by the mapping RAM. Only potentially active blocks would need to be assigned (inactive memory would be de-allocated) thus leaving more blocks free for use by the CPU.

The user has the option of specifying any combination or mix of on-card and off-card memory. Memory blocks may be turned "on" or "off", overlayed or moved around simply by changing the mapping RAM contents. Program segments can even be "cloned" without physically moving bytes from one location to another.



SIMPLIFIED BLOCK DIAGRAM OF ADDRESS MEMORY LOGIC
(DATA AND CONTROL BUSES NOT SHOWN)

Use of the MLZ-91 Memory Mapping RAM

The following pages describe the mapping RAM in more detail and show the CPU instructions for loading the mapping RAM data. There are also a number of examples showing specific instruction sequences which could be used to setup the mapping logic.

Later sections of this manual describe the Bus mapping RAM (used to designate the position that the MLZ-91 occupies on the Multibus) and the I/O mapping RAM (used to specify the base address of the on-card I/O devices).

Since the mapping RAM initially contains a random bit pattern, some scheme must be employed to load the RAM before any data from it is used. This is automatically accomplished by disabling the mapping RAM and forcing the map logic to address only ROM on the MLZ-91. The mapping RAM is enabled as the first attempt is made to load data into it. The RAMifications (!) of this are explained in the following text.

The basic sequence of instructions which is used to set the memory mapping RAM involves the following:

1. Load register C with the I/O port address assigned to the memory mapping RAM.

```
LD  C,MAPRAM
```

2. Load register B with the high half of the 4K memory address which is to be assigned. Only the upper four bits of the address (i.e. A15, A14, A13 & A12) are significant. This value determines which cell within the mapping RAM will be loaded. The 64K CPU address space is thus divided into 16 4K blocks.

```
LD  B,BLOCK
```

3. Load register A with the data which is to be stored in the mapping RAM. This value is determined from one of the accompanying charts showing the (HEX) data corresponding to each ROM socket (e.g. "ØØ"), on-card RAM block (e.g. "7F") or off-card memory block (e.g. "FB").

```
LD  A,DATA
```

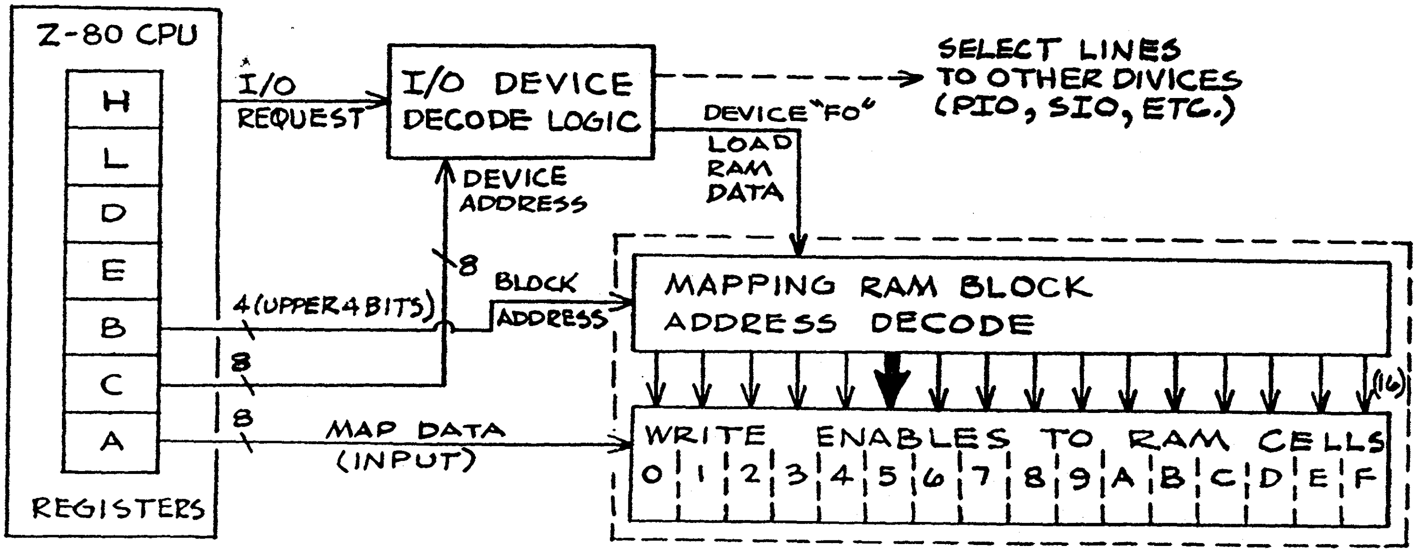
4. Execute an OUTPUT instruction to laod the DATA into the specified BLOCK address of the mapping RAM. (The special z-80 OUTPUT instruction must be used. If your assembler cannot handle anything but pure 8080 mnemonics, use the following sequence of defined bytes (DB): e.g. DB ØEDH, Ø59H.)

For example, the following sequence of instructions will map ROM socket MØ at memory block "5" which starts at CPU address 5ØØØ (HEX)

```
LD   C,MAPRAM        ;LOAD PORT ADDRESS
LD   B,5ØH           ;LOAD BLOCK ADDRESS
LD   A,ØØH           ;LOAD MAP DATA
OUT(C),A             ;SEND TO MAPPING RAM
```

During execution of the OUT instruction, the content of register B appears on the upper 8 CPU address lines, the most significant 4 bits of which specify the memory mapping RAM block address (see diagram). Register C contains the I/O port address which is used when writing to the mapping RAM. The DATA loaded into the mapping RAM specifies the physical location of the memory which is to be assigned to the specified CPU memory BLOCK. In addition, on-card RAM may be write protected by proper specification of a bit in the map DATA.

Summary informations on the memory mapping RAM data format appears at the end of this section, pages 34 and 35. However, if the mapping concept is unfamiliar to you it would be wise to read the text between here and there.



CPU REGISTER USAGE DURING "OUT(C),A" (WITH B = 50 H)

The contents of the memory mapping RAM may be changed as often as desired to reallocate the memory space or to enable or disable the write protection logic on a particular RAM block. After power is applied, after a hardware RESET or following a memory error (parity or write protect) the mapping RAM output is disabled and ROM socket MØ is selected until the first attempt is made to load data into the mapping RAM. Thus, socket MØ must contain a ROM and the initial instructions fetched from the ROM will start at address ØØØØH. Also, the first memory mapping RAM data loaded must be for socket MØ, otherwise execution will continue at an indeterminate address. Once socket MØ has been officially mapped, other memory may be assigned. More on this later.

This discussion has ignored the other MLZ-91 mapping RAMs (the BUS MAP and the I/O MAP) in order not to complicate matters. Be aware, however, that these other mapping RAMs must be properly initialized prior to loading the memory map. Luckily, the BUS and I/O mapping procedures are not difficult, so details on these have been deferred to a later section.

The next pages detail the procedure for mapping:

A) On-card ROM

B) On-card RAM

C) Off-card memory

There are mapping examples (with program listings) of typical map configurations later in this manual. (See page 80).

A. ON-CARD ROM

There are two memory sockets on the MLZ-91, both of which may contain up to 8K bytes of ROM. To setup the memory mapping RAM for the on-card ROM sockets, follow these steps:

1. Select the desired memory socket configuration from the "ROM MAPPING CHART". (See page 28).
2. Locate the two digit hexidecimal number shown under the desired configuation (i.e. "ØØ", "1Ø", "2Ø" or "3Ø". Note: The 8K socket configuration for a 2764 type ROM requires two map entries, one for each 4K half. (See below.)
3. Store the value found above in the memory mapping RAM using the instruction sequence explained earlier.

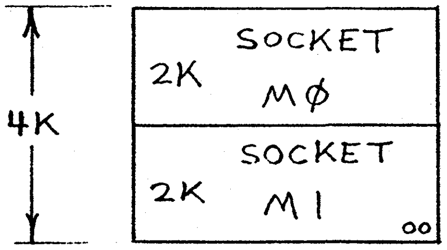
```
     LD   C,MEMMAP          ;LOAD PORT ADDRESS
     LD   B,BLOCK           ;LOAD 4K BLOCK ADDRESS
     LD   A,DATA            ;MAP DATA (found in step 2,
                             above)
     OUT  (C),A             ;SET MAP DATA
```

   If a 2764 is being mapped, the second 4K half may be assigned by using these additional instructions:

```
     LD   B,BLOCK+1ØH       ;NEXT BLOCK
     LD   A,DATA            ;DATA FOR 2ND HALF
     OUT  (C),A             ;SET MAP DATA
```

   Normally, the two halves would be mapped into adjacent blocks, although this is not required.
4. Locate the hardware jumper specification in the chart for the chosen memory type and configuration. Set the jumpers on the MLZ-91 as indicated. (e.g. J12-A, J14-A)

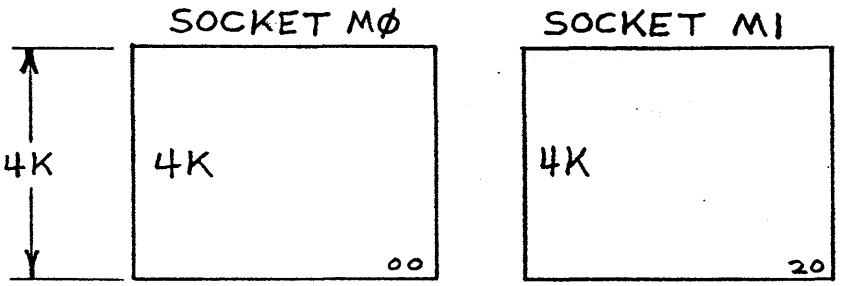
Recall that the memory mapping RAM is disabled following a power-up or manual system RESET and socket MØ is automatically assigned to all 4K memory blocks. When an output is done to load the mapping RAM it is re-enabled. In order to prevent the uninitalized contents of the mapping RAM from turning off the ROM from which we are executing, the first output to the map must be to assign socket MØ to the current memory block being executed. If the CPU is not executing in the proper block (prior to loading the map) simply execute a JUMP instruction to the desired block. The upper 4 CPU address bits have no significance until the mapping RAM is



ROM TYPE: 2716

JUMPERS: J12-B, J14-B

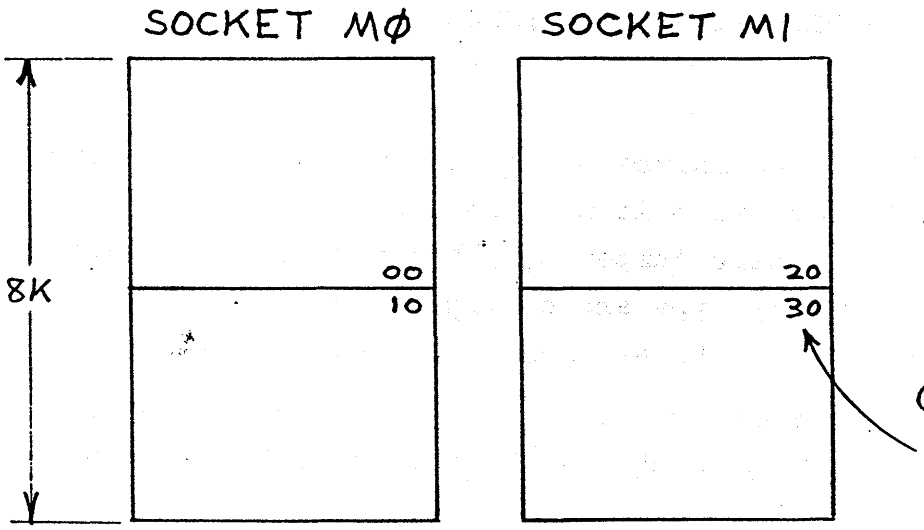
EACH SOCKET OCCUPIES HALF OF A 4K BLOCK. ADDRESS LINE A_{11} SELECTS SOCKET



ROM TYPE: 2732

JUMPERS: J12-A, J14-A

EACH SOCKET OCCUPIES A FULL 4K BLOCK. MEMORY MAP RAM SELECTS SOCKET



ROM TYPE: 2764

JUMPERS: J12-A, J14-A

(NUMBERS INDICATE DATA FOR MEMORY MAPPING RAM)

ROM MAPPING CHART

activated. Socket MØ will always be selected (and will thus "mirror" itself every 4K addresses. (I.e., locations ØØØØH, 1ØØØH, etc, will be identical.) This procedure (of jumping to the desired 4K block prior to setting the memory map) is equivalent to doing a power-on-jump. This procedure is illustrated in the memory mapping program examples. (See page 85.)

After socket MØ is mapped the next step should be to allocate some RAM. Up to this point no user RAM exists!

It is possible to deallocate the ROM and replace it with RAM. The technique for doing this is illustrated in the "SLAVE" software example on page 94 . (Note: If ROM is turned off, the I/O mapping RAM and bus mapping RAM cannot be altered until ROM is reallocated.)

B. ON-CARD RAM

On-card RAM is allocated in a manner similar to assigning the ROM sockets. The only difference is the DATA value used. The data for on-card RAM specifies the physical 4K block of RAM (instead of a physical ROM socket) which is to be assigned to one of the 16 CPU 4K address blocks. Also, on-card RAM may be write protected by specifying the proper data.

Here is the sequence to use to map on-card RAM:

1. Select the desired 4K physical block of RAM to be allocated from the "MEMORY MAP DATA CHART". (Page 35)
2. Locate the two digit hexidecimal number shown for the desired physical block (e.g. "5F").
3. To disable the memory write protect logic for the memory block, add 2ØH to the value found above. If write protection is desired, skip this step and simply use the original value. See page 36 for a description of the write protect logic.
4. Store the result of step 3 in the memory mapping RAM as follows: (This is the same as for the ROM sockets. Only the BLOCK and DATA values are different.)

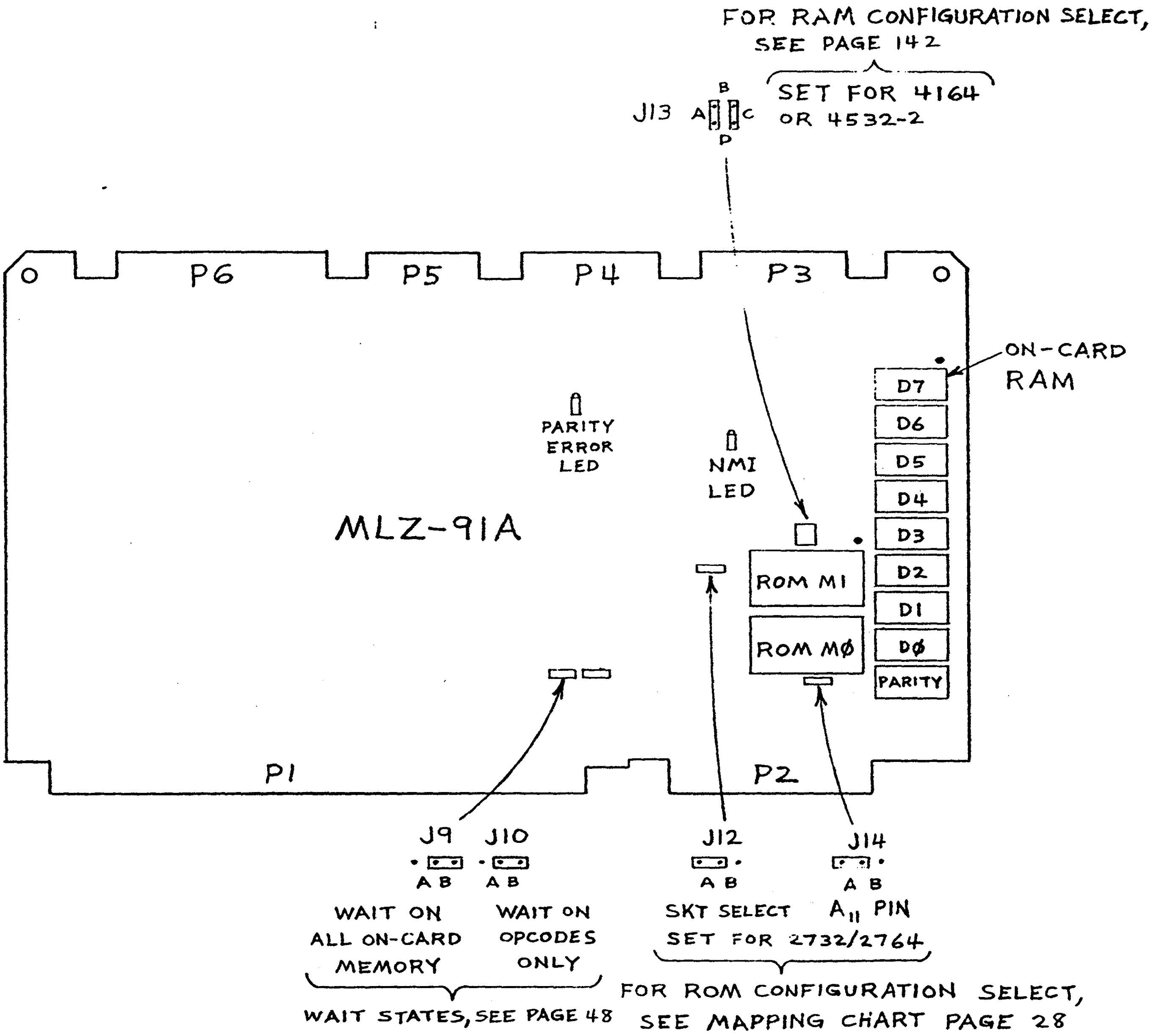
```
        LD   C,MEMMAP           ;LOAD PORT ADDRESS
        LD   B,BLOCK            ;LOAD 4K BLOCK ADDRESS
        LD   A,DATA             ;MAP DATA (from step 3,
                                 above)
        OUT  (C),A              ;SET MAP DATA
```

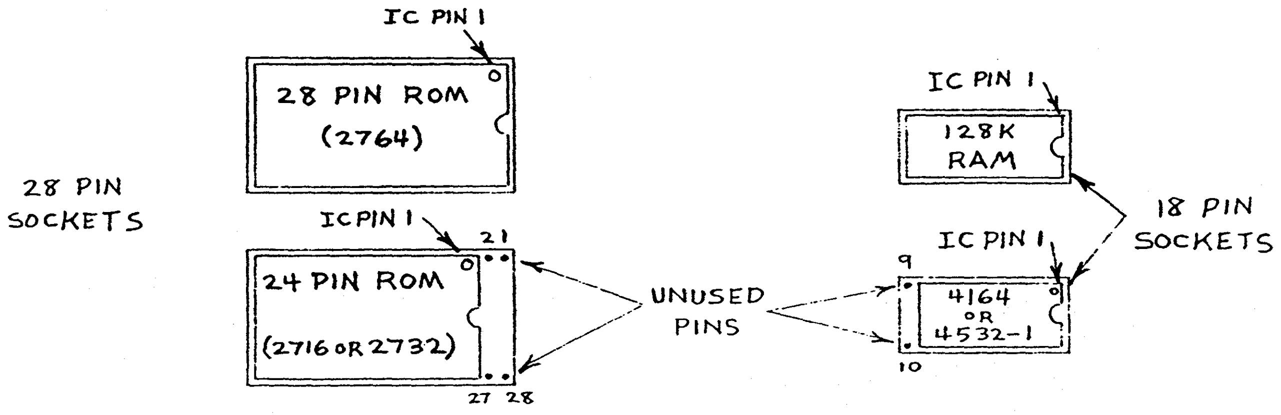
Since the instruction sequence above is similar for both ROM and RAM allocation and since the allocation of numerous blocks of RAM would be repetitive operation with changes only in registers B and A, an instruction loop may be used to simplify the procedure for loading the entire memory map. This technique is illustrated in the software example on page 86.

Note: Jumper J13 must be set according to the type of RAM chip being used (See page 142).

Refer to page 36 for a discussion of the RAM parity and write protect error logic.

MEMORY SOCKET JUMPER LOCATIONS

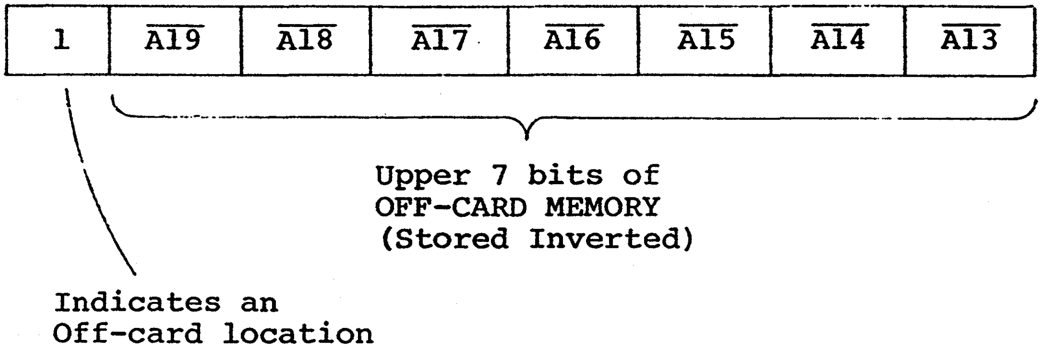




C. OFF-CARD MEMORY

The main difference between on-card and off-card memory is that CPU address line A12 controls both bus address line A12 and the least significant mapping RAM block address. This means that bus address A12 is not independent of the block address. All even numbered 4K groups of physical off-card memory addresses must be mapped by even numbered map blocks (and odd 4K physical off-card memory groups must be mapped by odd numbered map blocks.)

Therefore, the mapping RAM has total control of off-card address space only in 8K blocks if mapping RAM blocks are paired (i.e., same map data loaded into an even-odd pair of map blocks.)



1	$\overline{A19}$	$\overline{A18}$	$\overline{A17}$	$\overline{A16}$	$\overline{A15}$	$\overline{A14}$	$\overline{A13}$

To load the mapping RAM for a particular off-card memory block (8K in length) proceed as follows:

1. Right justify the upper 7 bits of the desired 20 bit address block (shift right one bit) and convert to hexidecimal. These upper 7 bits (A13 through A19) Specify one of the 128 8K memory blocks.

2. Invert all 8 bits (MSB should be ON to indicate an off card address). (See "Memory Map Data Chart", page 35.)

3. Store the result in a pair of map locations as follows:

```
        LD   C,MEMMAP          ;LOAD REG C WITH MAP PORT ADRS
        LD   B,HADRS           ;LOAD THE UPPER 3 BITS OF REG B
                                WITH A MAP BLOCK NUMBER. (EVEN)
                                (Bit 4 should be zero.)
        LD   A,DATA            ;LOAD REG A WITH MAP DATA
                                (Found in previous step)
        OUT  (C),A             ;SET MAP DATA - FIRST PART
        LD   B,HADRS+1ØH       ;SET BIT 4 ON (ODD BLOCK)
        OUT  (C),A             ;SET MAP - SECOND PART
```

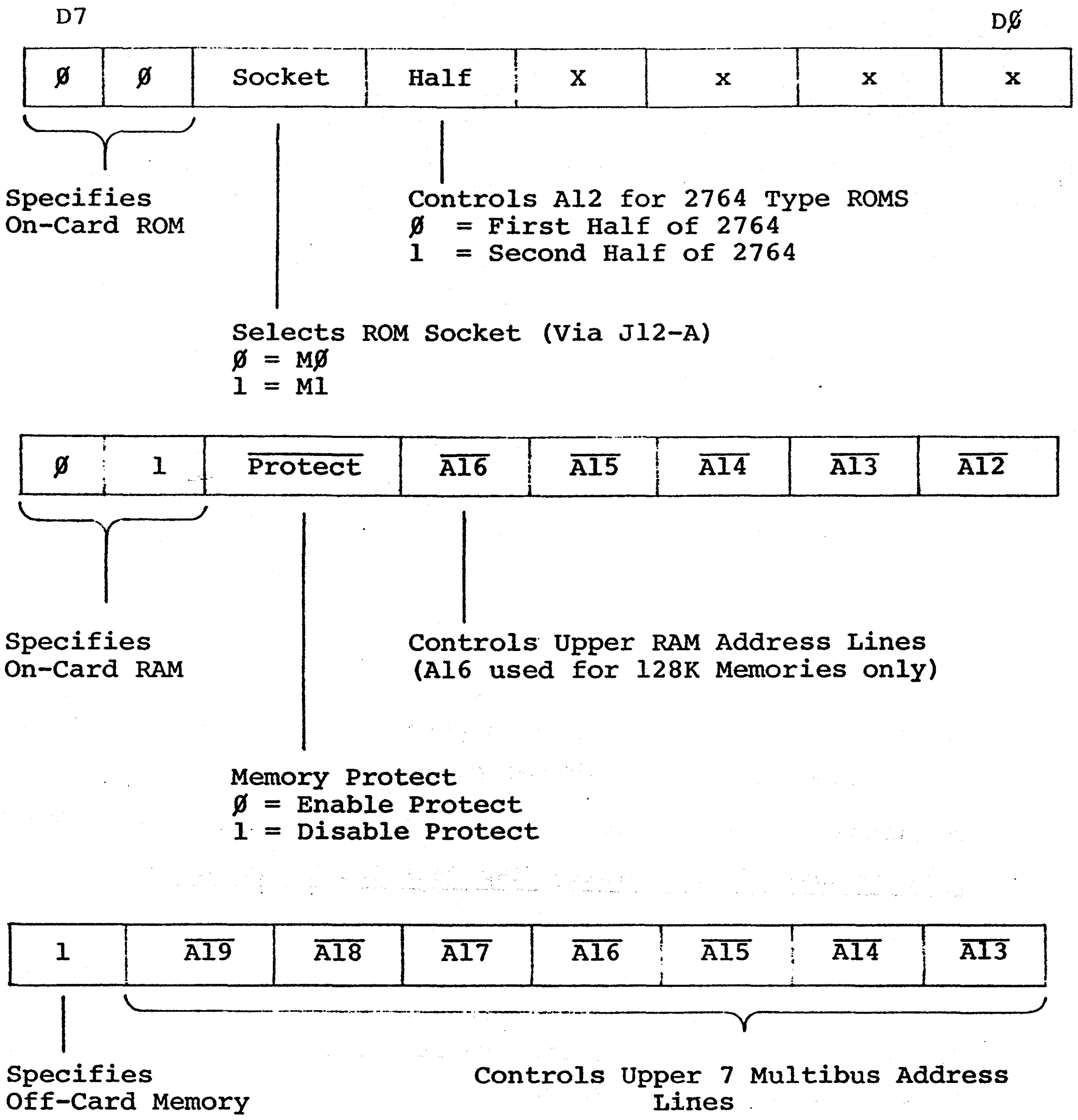
Two OUTPUT instructions are required, each to a different mapping RAM location, in order to prevent address line A12 from affecting the output of the mapping RAM. This is accomplished by setting the data in the two map locations (selected by A12) to the same value. Off-card memory can be allocated in 4K blocks of physical memory must be mapped (controlled) by even numbered map blocks.

The write protect feature is not available for off-card memory.

Note on use of the "Memory Map Data Chart" (page 35)

The data values to use to map on-card or off-card memory are shown on the Memory Map Data Chart, page 35. At any one time, at most 16 of these values may be loaded into the memory mapping RAM in order to specify the physical location of each 4K block of the 64K CPU memory space.

The top line represents the CPU address space from oooo through FFFF (hex address values). The second line represents the memory mapping blocks associated with each 4K of the CPU address space. The following lines show the appropriate data value to load into the mapping RAM in order to assign that particular memory segment. For example, to assign an on-card 2732 ROM in socket MØ to CPU address FØØØ, the data value "ØØ" must be loaded into the 16th (last) cell of the mapping RAM (second line). To assign the off-card physical address 48000 to CPU address ØØØØ, data value "DB" must be loaded into the first cell of the mapping RAM.



D7							DØ
Ø	Ø	Socket	Half	X	x	x	x

Specifies
On-Card ROM

Controls A12 for 2764 Type ROMS
Ø = First Half of 2764
1 = Second Half of 2764

Selects ROM Socket (Via J12-A)
Ø = MØ
1 = M1

Ø	1	$\overline{\text{Protect}}$	$\overline{A16}$	$\overline{A15}$	$\overline{A14}$	$\overline{A13}$	$\overline{A12}$

Specifies
On-Card RAM

Controls Upper RAM Address Lines
(A16 used for 128K Memories only)

Memory Protect
Ø = Enable Protect
1 = Disable Protect

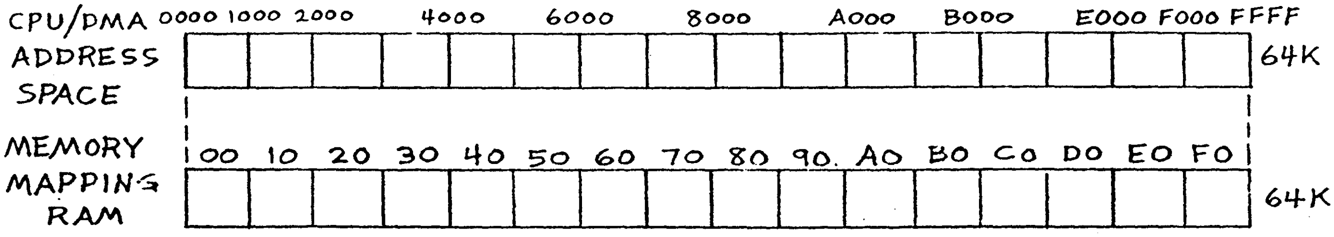
1	$\overline{A19}$	$\overline{A18}$	$\overline{A17}$	$\overline{A16}$	$\overline{A15}$	$\overline{A14}$	$\overline{A13}$

Specifies
Off-Card Memory

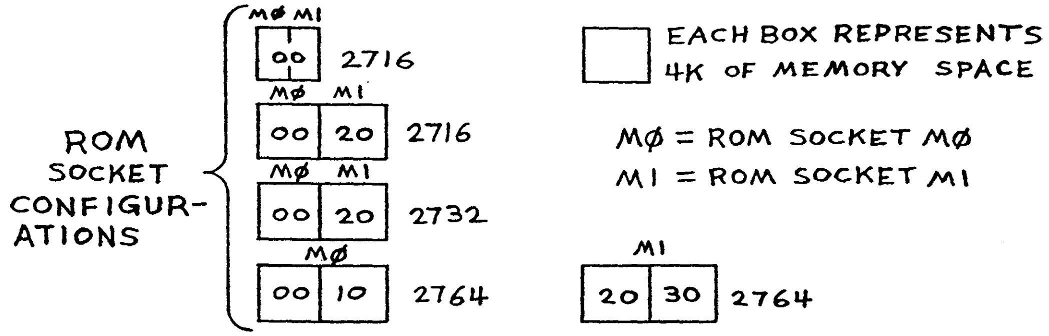
Controls Upper 7 Multibus Address
Lines

Use OUT (C), A to load the memory mapping RAM. Register B, upper 4 bits is memory block address during loading of the mapping RAM. Register C is the device address assigned to the Mapping Ram, "MEMMAP".

Summary of Memory Mapping RAM
Data Format


CPU/DMA ADDRESS SPACE
0000 1000 2000 4000 6000 8000 A000 B000 E000 F000 FFFF
64K
MEMORY MAPPING RAM
00 10 20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0
64K

ON-CARD MEMORY



RAM:

			+2000		+4000		+6000		+8000		+A000		+C000		+E000		
0000	5F	5E	5D	5C	5B	5A	59	58	57	56	55	54	53	52	51	5Ø	64K

ADD 2Ø TO DISABLE RAM WRITE PROTECT (ON CARD RAM ONLY)

OFF-CARD MEMORY

			+2000		+4000		+6000		+8000		+A000		+C000		+E000		
00000	FF	FF	FE	FE	FD	FD	FC	FC	FB	FB	FA	FA	F9	F9	F8	F8	64K
10000	F7	F7	F6	F6	F5	F5	F4	F4	F3	F3	F2	F2	F1	F1	F0	F0	64K
20000	EF	EF	EE	EE	ED	ED	EC	EC	EB	EB	EA	EA	E9	E9	E8	E8	64K
30000	E7	E7	E6	E6	E5	E5	E4	E4	E3	E3	E2	E2	E1	E1	E0	E0	
40000	DF	DF	DE	DE	DD	DD	DC	DC	DB	DB	DA	DA	D9	D9	D8	D8	
50000	D7	D7	D6	D6	D5	D5	D4	D4	D3	D3	D2	D2	D1	D1	D0	D0	
60000	CF	CF	CE	CE	CD	CD	CC	CC	CB	CB	CA	CA	C9	C9	C9	C9	
70000	C7	C7	C6	C6	C5	C5	C4	C4	C3	C3	C2	C2	C1	C1	C0	C0	
80000	BF	BF	BE	BE	BD	BD	BC	BC	BB	BB	BA	BA	B9	B9	B8	B8	
90000	B7	B7	B6	B6	B5	B5	B4	B4	B3	B3	B2	B2	B1	B1	B0	B0	
A0000	AF	AF	AE	AE	AD	AD	AC	AC	AB	AB	AA	AA	A9	A9	A8	A8	
B0000	A7	A7	A6	A6	A5	A5	A4	A4	A3	A3	A2	A2	A1	A1	A0	A0	
C0000	9F	9F	9E	9E	9D	9D	9C	9C	9B	9B	9A	9A	99	99	98	98	
D0000	97	97	96	96	95	95	94	94	93	93	92	92	91	91	90	90	
E0000	8F	8F	8E	8E	8D	8D	8C	8C	8B	8B	8A	8A	89	89	88	88	
F0000	87	87	86	86	85	85	84	84	83	83	82	82	81	81	80	80	64K

FFFFF

(ONE MEGABYTE TOTAL)

EACH BOX REPRESENTS 4096 BYTES. NUMBERS INSIDE BOXES ARE THE MAP DATA (IN HEX) FOR THAT BLOCK OF MEMORY. SEE PAGE 33 FOR NOTE ON USE OF THIS CHART.

MEMORY MAP DATA CHART

PARITY AND WRITE PROTECT LOGIC

A standard feature of the MLZ-91 is the ability to selectively write protect any block or group of blocks of on-card RAM. Bit D5 of the memory mapping RAM data is used to control the write protect logic. If on-card RAM is allocated with this bit in the Ø state (D5 LOW), then any attempt to write to an address in the 4K block so allocated will result in an NMI (Non-maskable Interrupt) to the Z-80 CPU. Generally, this feature is used as follows:

1. Allocate on-card RAM with memory protect disabled. (E.G., map data 7F)
2. Write data to the on-card RAM as desired.
3. Re-allocate the same RAM but with memory protect enabled. (E.G., map data 5F)
4. An NMI will occur if any attempt is made to write to that block of memory. The write operation will not be performed.

See below for a description of the NMI response. On-card RAM may also be protected by de-allocating the block, however this method will prevent any accesses of the memory block.

When the write protect logic is enabled any attempt to write to on-card RAM, whether from the CPU, DMA or from the system bus, will be "trapped". It is also possible to write (or read) protect on-card memory from the system bus only, without inhibiting use of the memory by the CPU or DMA, by properly loading the bus mapping RAM. See page 38.

The parity feature (an MLZ-91 option) is useful to guarantee the integrity of the data stored in the on-card RAM. Whenever data is written to the memory, odd parity is computed and the result is stored as a ninth memory bit. When the data is subsequently read back, the parity computed across all nine bits is checked and, if not odd, an NMI is generated.

Either of the above errors (write protect or parity) produces an NMI. The MLZ-91 responds as follows:

1. The address of the next instruction is pushed into the software stack if a write protect error occurs. The address of the second next instruction is saved in the case of a parity error.
2. The CPU program counter is set to ØØ66 (hex).
3. The memory mapping RAM is disabled which puts ROM at every 4K boundary (mirrors).

A program at address ØØ66 can then service the NMI, generally by printing an error message and re-initializing the program. It is possible for the service routine to determine the cause of the error and the approximate location where it occurred. See page 92 for a software example.

There are two error LED's which indicate the status of the NMI logic. One indicates a parity error, the other indicates a parity error or a write protect error. Thus,

(See page 31 for LED locations)

Both off = error logic is reset
One ON = write protect error
Both ON = parity error

If a HALT instruction is placed at location ØØ66 (in ROM socket MØ), the error indicators will show the error type and the MLZ-91 will halt following an error. However, if the NMI service routine at ØØ66 performs any other task, it must clear the NMI error logic before RAM can be reallocated, and the error indicators will be turned off.

The status of the NMI error logic may be determined by doing an input from I/O port IOSTAT. See page 64 for details. The parity logic may be disabled by shunting jumper J19.

POWER-ON JUMP

Since the entire memory space is controlled by a mapping RAM, any memory socket or external address block may be dynamically allocated by the software. A power-on-jump is easily implemented by properly initializing the mapping RAM.

When program execution begins after a power up or RESET the output of the mapping RAM is forced to select memory socket MØ regardless of the upper four CPU address lines. Execution begins at CPU address ØØØØ from address ØØØ in the ROM. If a jump instruction is executed to a different 4K block without changing the relative location from the base of the block, execution will merely continue at the next sequential ROM address (although the program counter in the CPU will be pointing to the desired 4K block). Then, the mapping RAM can be set so that socket MR is relocated to the desired block.

For an illustration of this method, refer to the MLZ-91 initialization example on page 85.

BUS MAPPING RAM

The conventional method of board assignment in the Multibus address space is to utilize a group of DIP switches or jumpers to specify the base address of the board. The MLZ-91 uses a special mapping RAM instead which is loaded under software control. Bits in the RAM perform the same functions as DIP switches but allow the operating program to modify the board's position and and status on the bus.

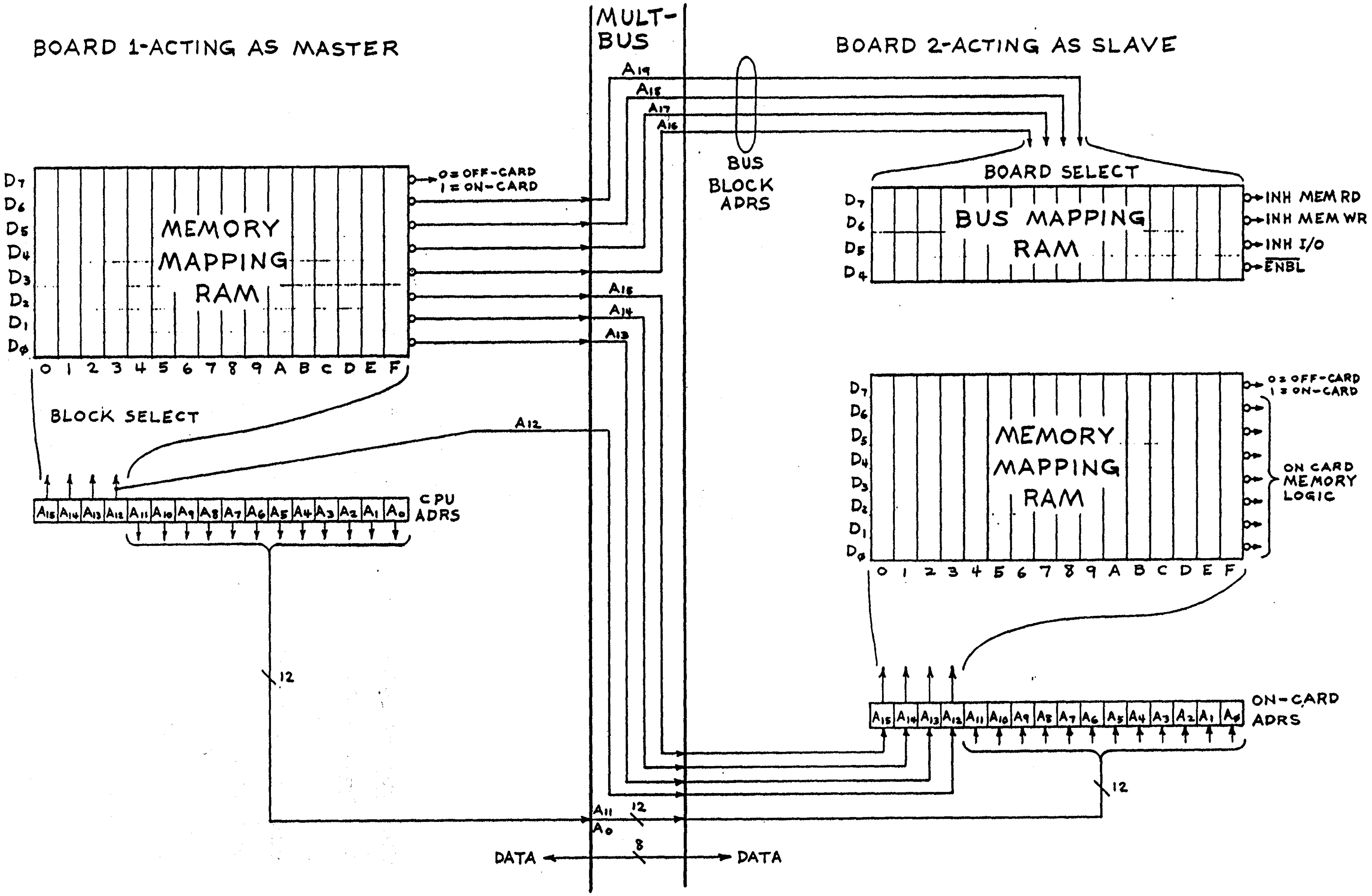
The one megabyte (20 bit) address space on the Multibus is split into 16 blocks. The upper four address lines (i.e. A19, A18, A17 and A16) are used to select an entry in the bus mapping RAM which corresponds to the block being addressed on the bus. The RAM's output specifies:

1. Whether or not the board is assigned to that block.
2. If it is assigned, then other bits determine:
    a. if the board's I/O devices may be accessed,
    b. if the on-card memory may be read from, or
    c. if the on-card memory may be altered.

Thus, a board may be set up to allow memory reads, memory writes, I/O device accesses or any combination of these functions.

Access to the bus mapping RAM by the CPU is done by executing a memory write operation to either on-card ROM socket with A5 set on and A3, A2, A1 and AØ specifying the inverse of the logical bus block number. Note that it would normally be inappropriate to do a write to ROM. This type of operation is intercepted by hardware and the CPU data is stored in the bus mapping RAM. (This technique is also used to load the I/O mapping RAM.)

If on-card ROM has been deallocated (via the memory mapping RAM), one of the ROM sockets must be reassigned, at least temporarily. This is a function of the memory mapping RAM. (See page 24.)



MULTI-PROCESSOR MEMORY ACCESS-SIMPLIFIED DIAGRAM

The following sequence of instructions may be used to load the bus mapping RAM:

```
        LD   A,DATA
        LD   (ROMBASE+2FH-BLOCK),A
```

Where: ROMBASE is the base address of either ROM socket (e. g. FØØØH)

BLOCK is the desired address within the bus mapping RAM. BLOCK may take on the values of ØØH through ØFH corresponding to the values allowable for the upper 4 address bits on the 20 bit Multibus. Note that the expression ROMBASE+2F-BLOCK forms a value which is the compliment of the actual bus block number (plus A5 high.)

DATA is determined as follows:

D7 Set to 1 to allow memory read operations in the specified BLOCK

D6 Set to 1 to allow memory write operations in the specified BLOCK

D5 Set to 1 to allow use of on-card I/O devices in the specified BLOCK

D4 Set to Ø to disable all operations. If D4 = Ø the board will not even respond with $\overline{\text{BACK}}$ (Bus Acknowledge) on the Multibus, thus ignoring all bus requests. If D4 is set to 1 and D7, D6 or D5 are all zero, $\overline{\text{BACK}}$ will still be issued in response to a bus request even though all operations are inhibited.

The bus mapping RAM output is enabled and disabled along with the memory mapping RAM. Thus, the bus mapping RAM output is disabled following power-on, a system reset or an NMI. The bus mapping RAM may still be loaded, however, prior to being enabled. In fact, it should be completely loaded prior to any access of the memory mapping RAM so that the contents are defined when both mapping RAMs are enabled.

We suggest that the bus mapping RAM be loaded with all zeros prior to setting the memory map. The following loop will accomplish this:

```
        LD      HL,ROMBASE+2ØH          ;BUS MAPPING RAM
        LD      B,16                    ;LENGTH
Loop:   LD      (HL),ØØH                ;WRITE TO MAPPING RAM
        INC     HL
        DJNZ    LOOP
```

Then, after the memory map has been initialized, selected entries of the bus map may be changed. For more details, refer to the example MLZ-91 initialization program on page 80.

If the MLZ-91 is used in a system with other master boards which do not support the upper 4 address lines, then the bus mapping RAM must be connected to A15, A14, A13 and A12 on the Multibus. To do this,remove jumper J7 (which will disable the ability of the MLZ-91 to drive the upper 4 lines) and conncect A19 to A15, A18 to A14, A17 to A13 and A16 to A12 on the Multibus. This will allow A12 through A15 to specify the bus block.

D7							DØ
ENBL MEM READ	ENBL MEM WRITE	ENBL I/O	ENBL BLOCK	X	X	X	X

Ø	Ø	Ø	Ø	Disable all functions
Ø	Ø	1	1	Enable I/O only
1	Ø	Ø	1	Enable Memory Read only
1	1	1	1	Enable all functions

(other combinations are also valid)

BUS MAPPING RAM DATA FORMAT

I/O MAP

The on-card I/O devices are divided into two groups. The base address of each group is specified by the I/O Mapping RAM. This allows the I/O devices on the MLZ-91 to be allocated so that off-card I/O addresses are not shadowed by the on-card devices.

The base addresses of each group may be specified to be one of the following:

ØØ	(hex)	for	device	addresses	ØØ	through	3F
4Ø		"	"	"	4Ø	"	7F
8Ø		"	"	"	8Ø	"	8F
CØ		"	"	"	CØ	"	FF

Typically one of the on-card device groups would be assigned base 8Ø and the other would be located at CØ. Then all addresses from ØØ through 7F would be usable as off-card devices. If an off-card device occupied address 9Ø, for example, then it would be possible to put the on-card devices at bases ØØ and CØ to allow access off-card at base 8Ø.

Since the I/O Map is controlled by a RAM, it is possible to alter the map contents at any time and as often as necessary for the particular application.

The I/O mapping RAM contains four locations, one for each of the base addresses listed aboove. The data loaded into the map specifies how the corresponding block of I/O addresses (e.g. 4Ø through 7F) is allocated. Each block may be set to one of the following states:

Assigned to on-card device group A
Assigned to on-card device group B
Assigned to off-card devices
Not assigned.

The following instruction sequence is typical of the method used to load one of the four locations in the mapping RAM.

```
        LD    A, DATA
        LD    (ROMBASE + 1ØH + BASE/4),A
```

Where ROMBASE is the base address of either ROM socket (e.g. FØØØ)

BASE is one of the four I/O base addresses (I.e., ØØH, 4ØH, 8ØH or CØH.

DATA is one of four values determined from the data format chart, below.

To load the entire I/O mapping RAM, four data bytes must be stored, similar to the above sequence, at ROMBASE + 1ØH, ROMBASE + 11H, ROMBASE + 12H and ROMBASE + 13H.

The first operation performed by the software following power on (or a reset) should be to load the I/O Map. Once assigned, the I/O devices may be accessed.

For an example of using the I/O mapping RAM, see page 80.

There are a number of special considerations which must be taken into account when performing interboard I/O. There is a discussion and programming examples starting on page 102.

X	X	X	X	$\overline{\text{IOA}}$	$\overline{\text{IOB}}$	X	$\overline{\text{EXT}}$	Hex
I/O not assigned				1	1	X	1	ØF
Block assigned to I/O Device Group A				Ø	1	X	1	Ø7
Block assigned to I/O Device Group B				1	Ø	X	1	ØB
Block assigned to Off-Card				1	1	X	Ø	ØE

I/O Mapping RAM Data Format

BUS CONTROL

The control logic for the Intel Multibus allows the MLZ-91 to share the bus with other processor cards.

The following signals are used by the bus arbitration and control logic:

BAI- (P1-15) Bus Available In. Low level indicates that no higher priority processor needs the bus.

BAO- (P1-16) Bus Available Out. Low level indicates that neither this board nor a higher level board needs the bus.

> BAI- and BAO- form a daisy chain for priority resolution when BAO- of each board is connected to BAI- of the next lower priority processor. The BAI- of the highest priority processor is forced low by installing jumper J8 on that particular card.

BRQST- (P1-18) Bus Request. Low level indicates that this board has need of the bus. Used to implement a parallel priority structure instead of a daisy chain. BROST- for each slot in the multibus is independent of the other BROST- signals (i.e., not bused).

CBREQ- (P1-29) Common Bus Request. This signal is common for all cards in the system. A low level indicates that there is a bus request from any card not already using the bus, regardless of priority. This signal allows a board to maintain control of the bus, whether actively using the bus facilities or not, until such time as any other board has a request.

BBUSY- (P1-17) Bus Busy. A low level on this line indicates that bus is in use.

BCLK- (P1-13) Bus clock. An 8 MHz clock generated by the highest priority board. Used to synchronize all bus requests and arbitration.

When a processor makes a request for use of the bus, the arbitration logic automatically takes over. If necessary, the requesting board will enter a wait state until the bus is available. When the requested bus operation is completed, the bus will be released according to the state of two control signals which are under soft-ware control as follows:

BC1	BCØ	Bus release status
Ø	Ø	Release bus after every operation.
Ø	1	Release bus if any other board has a request for the bus (Uses CBREQ-)
1	Ø	Release bus only if a higher priority board has a request for the bus. (Uses BAI-)
1	1	Never release bus, once acquired. This state can be used to capture the bus.

The actual status of the bus control logic can be determined by reading the MLZ-91 board status port. (See page 64.)

The two bus control signals are generated by PIO port A. Since there are six other lines on port A, some consideration must be given to properly initializing that port. The table below shows the function of all eight bits:

Bit Number	Name	Type	Function
7 (MSB)	FDIO-INTRQ	Input	Interrupt request from floppy disk
6	WINC-INTRQ	Input	Interrupt request from Winchester
5	APU/GPIB	Input	Interrupt request from APU or GPIB
4	BC1	Output	Bus control bit 1
3	BCØ	Output	Bus control bit Ø
2	S2	Output	DMA ready select 2
1	S1	Output	DMA ready select 1
Ø (LSB)	SØ	Output	DMA ready select Ø

PIO A may be initialized as follows: (This example assumes we want to release the bus if any other board has a request, BC1 LOW.)

```
LD        A,ØCFH              ; PIO "BUT" MODE CONTROL
OUT       (IOPAC),A           ; SEND TO PORT A CONTROL
LD        A,EØH               ; IN/OUT MASK (3 INS, 5 OUTS)
OUT       (IOPAC),A           ; SEND TO PORT A
LD        A,Ø4H               ; BC1 LOW, BØ HIGH
OUT       (IOPAD),A           ; SEND TO PORT A AS DATA
```

Later, the state of the control signals may be changed by doing another output to port IOPAD. However, since that port is also used to select the DMA Ready signal (S2,S1,SØ) it would be advisable to use the following scheme:

```
IN       A,(IOPAD)            ; READ CURRENT DMA SELECT BITS
AND      Ø7H                  ; TURN OTHERS OFF
OR       DATA                 ; TURN BC BITS ON AS DESIRED
OUT      (IOPAD),A            ; SET NEW BUS CONTROL BITS
```

Further details describing the functions of PIO port A can be found in a later section, "PIO (System PIO)" page 57.

MAPPING RAM AND BUS CONTROL SUMMARY INFORMATION

The table below summarizes the data and addresses for the MLZ-91 mapping RAMs and the Sequence in which the mapping RAMs should be loaded. See page 80 for an actual program example.

Sequence	Description	Data (hex)	Address	Ref. pages

1.	Load I/O Mapping RAM	ØØ=not assigned
		Ø7=I/O Device Group A
		Ø8=I/O Device Group B
		ØE=Off-card devices
			ROMBASE +1ØH=base ØØ
			ROMBASE +11H=base 4Ø
			ROMBASE +12H=base 8Ø
			ROMBASE +13H=base CØ
				42

2.	Clear Bus Mapping RAM	ØØ	ROMBASE +2ØH through ROMBASE +2FH	38-41
3.	Assign Socket MØ (ROM)	ØØ	Reg. B=HIGH ROMBASE Reg C=MEMMAP (port) OUT (C),A	27
4.	Assign other memory	See pages 34 & 35	Reg B=HIGH blockadrs Reg c=MEMMAP OUT (C),A	30-35
5.	Assign board to bus	ØØ=disable all oprns 3Ø=enable I/O only 9Ø=enable Memory RD FØ=enable all oprns (others, see page 41)	ROMBASE +2FH-BLOCK	38
6.	Bus Control logic (part of System PIO port A)	BC1 BCØ Release mode Ø Ø every oprn Ø 1 CBREQ- 1 Ø BAI- 1 1 never release	IOPAD (I/O port)	44

MEMORY AND I/O TIMING

A. On-Card Memory of I/O Devices

There are option jumpers on the MLZ-91 that may be set to insert a WAIT state in specific memory access cycles. The following chart details the various jumper configurations:

Wait State Condition	Opcode Fetch Only	All memory Cycles
ROM only	J9-A J10-B	J9-A J10-A
ROM and RAM	J9-B J10-B	J9-B J10-A
No Wait States	Remove both jumpers	Remove both jumpers

After accounting for the gate delays inserted by the memory mapping RAM and chip select logic, the access times for ROM memories should be no longer that specified below:
(Assumes NO WAIT states) (worst case timings)

CPU/DMA clock selected by J1	Max CE to DATA valid ROM	Max ADRS to DATA valid
2MHz	580 nsec	785 nsec
4MHz	205 nsec	320 nsec

If the WAIT state logic has been enabled, add the following times to the worst case values, above:

J10	2 MHz clock	4 MHz clock
J10-A	500 nsec	250 nsec
J10-B	250 nsec	125 nsec

On-card I/O devices operate at full CPU speed regardless of the WAIT state jumpers.

B. Off-card Memory or I/O Devices

When off-card memory or I/O is addressed, the bus interface logic puts the CPU (or DMA) into a wait state until the bus is acquired and the addressed memory or device issues bus acknowledge (BACK-).

External devices of any access time may be used for off-card memory or I/O. The BACK-delay circuit (part of the external control board) must not issue BACK- until the access delay of the addressed memory or device has expired.

On card RAM will be automatically refreshed to prevent loss of data due to a lengthy wait state. This function is completely automatic and transparent to the user.

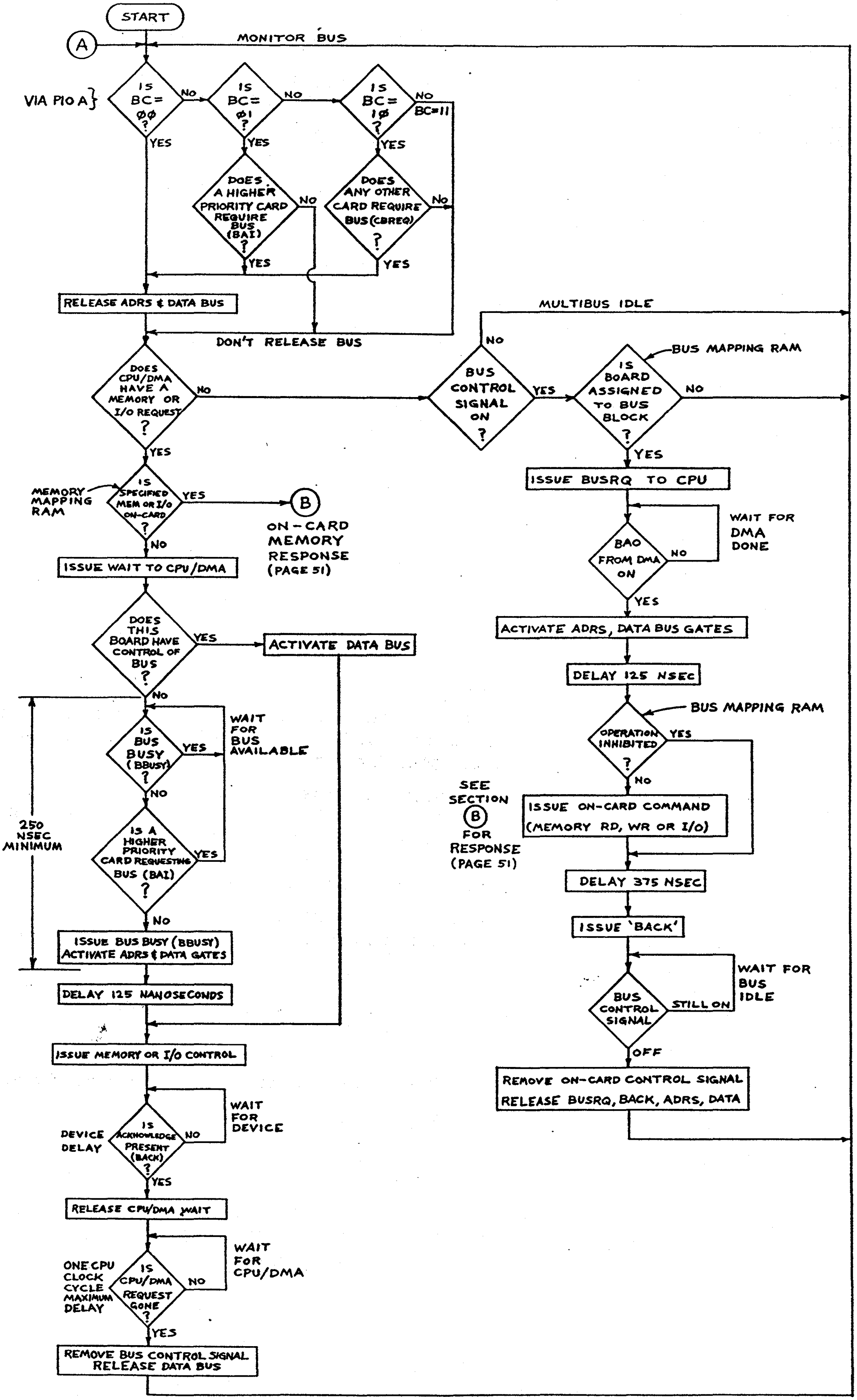
The minimum delay inserted in an off-card access by the bus arbitration logic is 375 nanoseconds (unless the bus has been "captured"). Additional delay time will occur if the bus is unavailable (BBUSY- true).

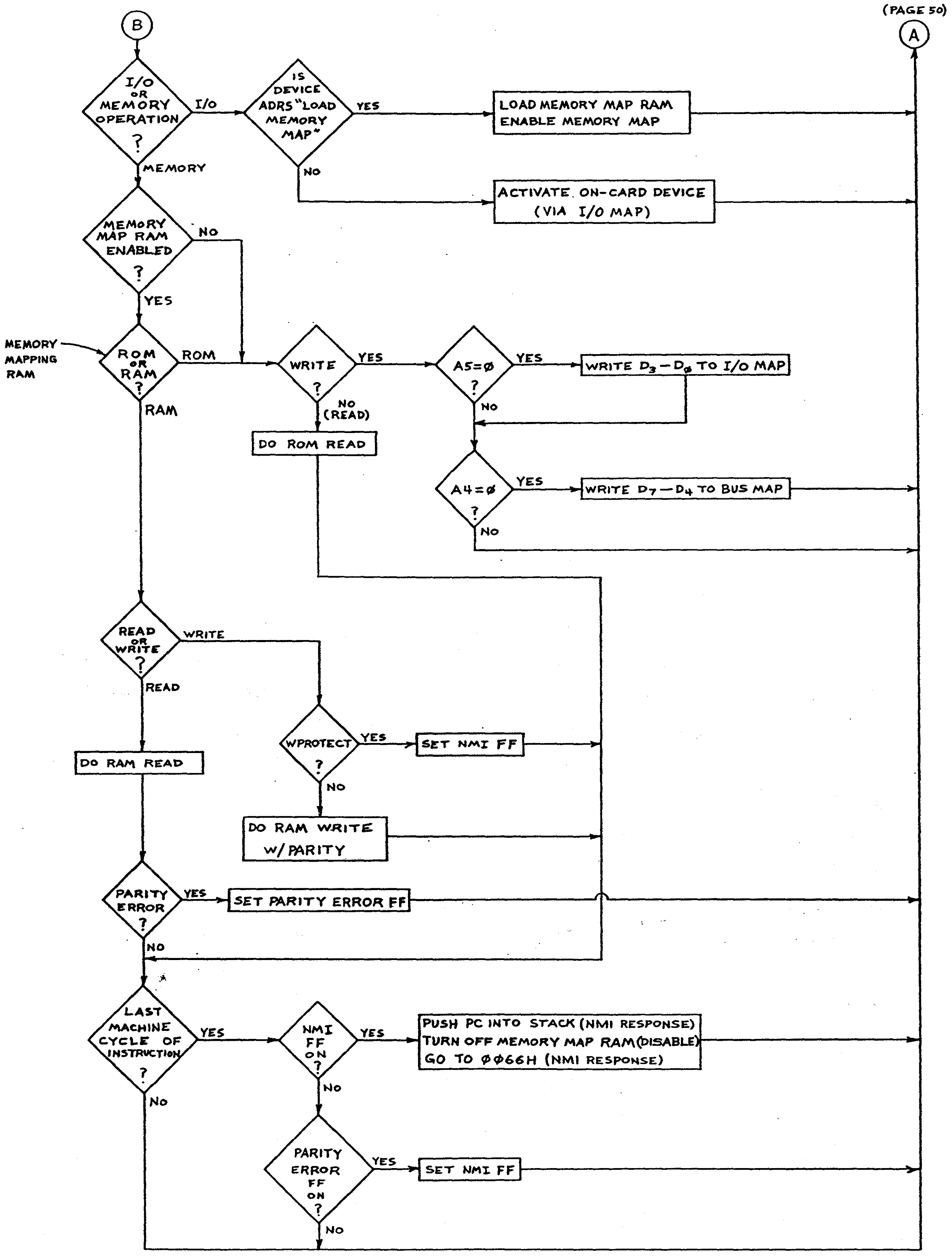
The delay inserted by the bus arbitration logic when an off-card access is attempted depends upon the state of the bus, as follows:

Condition	Minimum Delay inserted by arbitration logic*
Bus Idle	375 nanoseconds Total access time will be a combination of above delay plus BACK- delay from external device.
Bus Busy	375 nanoseconds Total access time will be combination of above delay plus time until bus no longer busy (BBUSY-false) plus BACK-delay from external device.
Bus not released following previous operation	No delay inserted Total access time will be determined only by BACK- delay

*The maximum delay inserted will be the values computed above plus the time for the CPU/DMA to recognize the BAC- signal. This timing depends on the BACK- edge relative to the CPU clock. For a 2 MHz board, add 500 nanoseconds, for 4 MHz boards, add 250 Nanoseconds, maximum.

Refer to the flowchart on page 50 for a graphic description of the memory timing.


START
A
MONITOR BUS
VIA PIO A
IS BC= ØØ ?
NO
IS BC= Ø1 ?
NO
IS BC= 1Ø ?
NO
BC=11
YES
DOES A HIGHER PRIORITY CARD REQUIRE BUS (BAI) ?
NO
YES
DOES ANY OTHER CARD REQUIRE BUS (CBREQ) ?
NO
YES
RELEASE ADRS & DATA BUS
DON'T RELEASE BUS
MULTIBUS IDLE
DOES CPU/DMA HAVE A MEMORY OR I/O REQUEST ?
NO
YES
BUS CONTROL SIGNAL ON ?
NO
YES
BUS MAPPING RAM
IS BOARD ASSIGNED TO BUS BLOCK ?
NO
YES
ISSUE BUSRQ TO CPU
MEMORY MAPPING RAM
IS SPECIFIED MEM OR I/O ON-CARD ?
YES
B
ON-CARD MEMORY RESPONSE (PAGE 51)
NO
ISSUE WAIT TO CPU/DMA
WAIT FOR DMA DONE
BAO FROM DMA ON
NO
YES
ACTIVATE ADRS, DATA BUS GATES
DELAY 125 NSEC
DOES THIS BOARD HAVE CONTROL OF BUS ?
YES
ACTIVATE DATA BUS
NO
IS BUS BUSY (BBUSY) ?
YES
WAIT FOR BUS AVAILABLE
NO
BUS MAPPING RAM
OPERATION INHIBITED ?
YES
NO
SEE SECTION B FOR RESPONSE (PAGE 51)
ISSUE ON-CARD COMMAND (MEMORY RD, WR OR I/O)
250 NSEC MINIMUM
IS A HIGHER PRIORITY CARD REQUESTING BUS (BAI) ?
YES
NO
DELAY 375 NSEC
ISSUE 'BACK'
ISSUE BUS BUSY (BBUSY) ACTIVATE ADRS & DATA GATES
WAIT FOR BUS IDLE
BUS CONTROL SIGNAL
STILL ON
DELAY 125 NANOSECONDS
OFF
ISSUE MEMORY OR I/O CONTROL
REMOVE ON-CARD CONTROL SIGNAL RELEASE BUSRQ, BACK, ADRS, DATA
WAIT FOR DEVICE
DEVICE DELAY
IS ACKNOWLEDGE PRESENT (BACK) ?
NO
YES
RELEASE CPU/DMA WAIT
WAIT FOR CPU/DMA
ONE CPU CLOCK CYCLE MAXIMUM DELAY
IS CPU/DMA REQUEST GONE ?
NO
YES
REMOVE BUS CONTROL SIGNAL RELEASE DATA BUS



"FF" MEANS FLIP-FLOP

MEMORY AND BUS CONTROL LOGIC FLOWCHART

INTERRUPT STRUCTURE

Internally, the MLZ-91 uses either the mode 1 or mode 2 interrupt state of the processor. The mode is selected by the software and has the following characteristics:

Mode 1: Any interrupt from a device causes a CALL to be executed to address 0038 hex (or 070 octal). The interrupt service routine located at that address must then poll each device which was enabled to determine which requires service.

Mode 2: Each device is provided with a vector pointing to the address of an interrupt service routine. The vector is used at the time of the interrupt to cause a CALL to be automatically executed to the specified service routine. If each device is given a different vector, a very efficient means of interrupt service will result since it will not be necessary to poll the devices to determine which generated the interrupt. For some I/O devices (e.g. the SIO) different vectors may be specified for certain conditions (such as transmit buffer empty or receive buffer full).

Externally, the MLZ-91 can support the standard eight interrupts available on the system bus. However, these eight lines do not produce a direct priority interrupt as with a conventional 8080 system. Instead, lines INTO- thru INT7- are connected to a special 8-bit port on P103. This port may be configured to monitor the eight interrupt lines for any specified combination of states and to produce a vectored interrupt when that state occurs. Using this scheme the conventional priority interrupt system or a more complex structure can be achieved.

Since the bus interrupts described above are connected to a PIO chip, some or all lines may be configured as outputs. This will allow the MLZ-91 to activate an interrupt line for multiprocessor communications or for peripheral control. For more discussion on use of this feature, refer to page 57.

In addition to all interrupt modes and signals described above, the processor also has a non-maskable interrupt (NMI) which is always enabled. This interrupt has the highest absolute priority and is internally connected to the memory write protection and parity logic.

When an NMI occurs, the program counter is pushed and control transferred to location ØØ66H in memory socket MØ. Since the NMI cannot be disabled by the CPU, there must be a service routine at location ØØ66H for write protect and parity error recovery.

Each I/O device on the MLZ-91 is connected in a daisy chain which defines the relative device priority for interrupt processing. The table below shows the mode 2 priority structure. Other priorities may be created under software control by selectively enabling or disabling devices during interrupt processing. (See page 56 for daisy chain diagram.)

A higher priority device may cause an interrupt during the servicing of a lower priority device, as long as interrupts have been re-enabled by the lower priority device's service routine. Interrupts from any devices of lower priority than the one currently being serviced will remain pending until the service routine has been completed. This priority structure is implemented via hardware in the microprocessor and the various peripheral chips.

Priority (Mode 2)	Device
1 (highest)	CTC
2	System PIO (FDIO/Winchester/ APU/GPIB/BUS)
3	SIO
4 (lowest)	DMA

VECTORED INTERRUPT OPERATION

At some area in memory, usually ROM, there should be a table of interrupt service routine addresses. All table entries must begin on an even byte address (AØ = zero) and the table must not cross a page boundary (H address constant.) The table might look like this:

```
            ORG   even address
   ITABLE:  DW    CTCØ service routine address
            DW    CTC1 service routine address
            DW    SIOA service routine address
            etc.  etc.
```

During system initialization, the upper half of the table base address must be loaded into the CPU "I" register via a command sequence similar to the following:

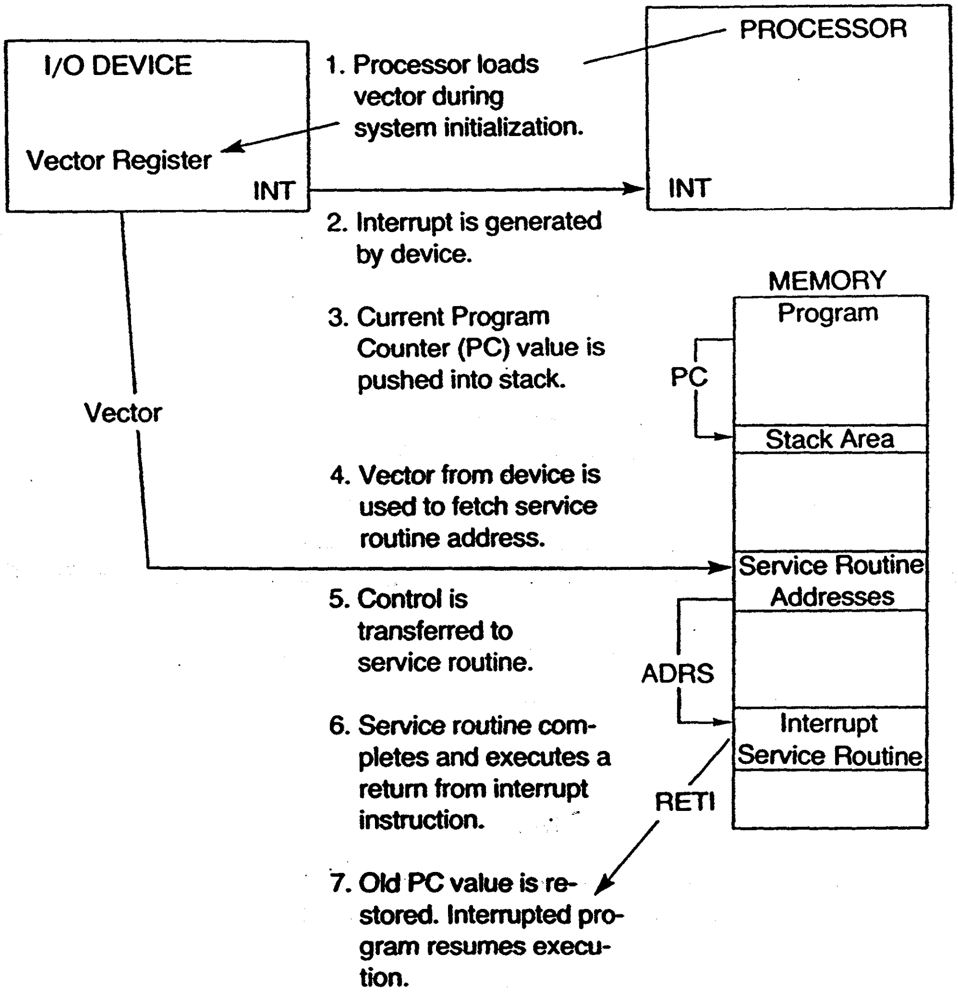
```
            IM    2       ; SET VECTORED INTERRUPT MODE
            LD    A,HADRS; H HALF OF TABLE ADRS  (UPPER 8-BITS)
            LD    I,A     ; LOAD I REGISTER
```

Then the individual I/O devices should be initialized and the low half of the particular table entry (for that device's service routine address) loaded into the device's interrupt vector register. This value is the low half of the table entry address, not the low half of the actual service routine address.

When the CPU's I register is coupled with the device's vector register, as is done during an interrupt acknowledge cycle, the CPU can locate the appropriate service routine address from the list of addresses in the table and use that address to transfer control to the service routine. This method allows an 8-bit vector register in each device to point (indirectly) to a 16-bit memory address.

During the execution of the service routine, CPU interrupts may be re-enabled (EI instruction) and any higher priority devices (as defined by the daisy chain) will be allowed to interrupt. The device being serviced, as well as lower priority devices, will be inhibited. When the RETI instruction is executed at the completion of a service routine, the device being serviced will automatically be reset and its interrupts enabled. (The Z-80 I/O devices monitor the data bus during instruction execution looking for RETI instructions and can determine which service routine is executing by the state of the daisy chain signals.) See page 91 for an example of an interrupt service routine.

Mode 2 Interrupt Diagram

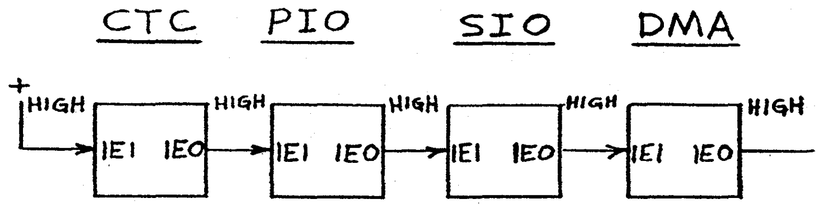


I/O DEVICE
Vector Register
INT
PROCESSOR
INT
1. Processor loads vector during system initialization.
2. Interrupt is generated by device.
MEMORY
Program
PC
Stack Area
3. Current Program Counter (PC) value is pushed into stack.
Vector
4. Vector from device is used to fetch service routine address.
Service Routine Addresses
5. Control is transferred to service routine.
ADRS
Interrupt Service Routine
6. Service routine completes and executes a return from interrupt instruction.
RETI
7. Old PC value is restored. Interrupted program resumes execution.

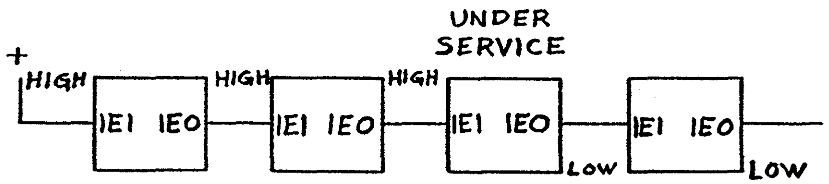
In order to guarantee proper device initialization, we recommend that the following code be inserted prior to enabling interrupts:

```
         LD      B,13             ;LOOP COUNT, 13 DEVICES*
         LD      HL,RETADRS       ;ADRS FOR RETI
LOOP:    PUSH    HL               ;STACK ADRS FOR RETI
         RETI                     ;RESET DAISY CHAIN
RETADRS: DJNZ    LOOP             ;CONTINUE, LOOP
```

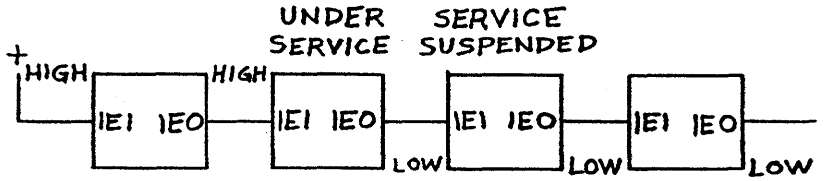
* includes the internal devices within each chip; e.g., the CTC is really 4 devices.



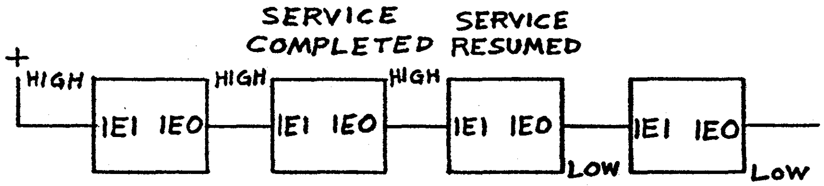
1. PRIORITY INTERRUPT DAISY CHAIN BEFORE INTERRUPT OCCURS.



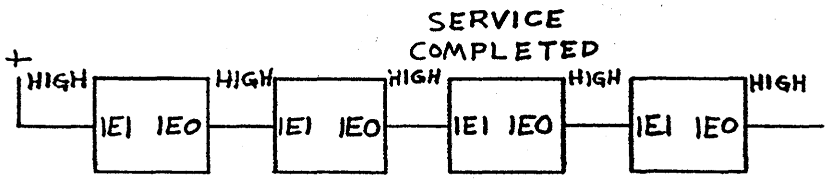
2. THE SIO CHIP INTERRUPTS AND ITS SERVICE ROUTINE STARTS.



3. PIO INTERRUPTS SUSPENDING SERVICE OF THE SIO.

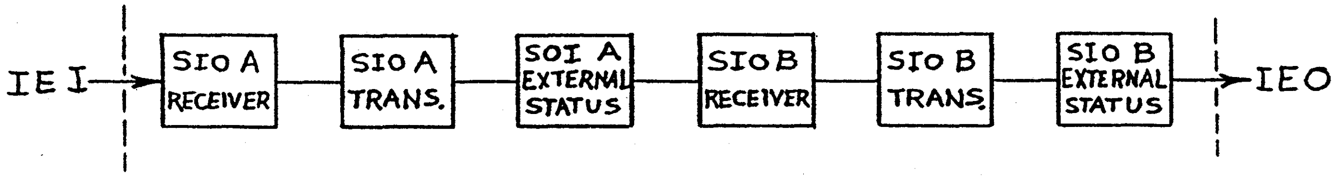


4. PIO SERVICE ROUTINE COMPLETE. RETI ISSUED, SIO SERVICE RESUMED.



5. SIO SERVICE ROUTINE COMPLETE, SECOND RETI ISSUED.

THE DAISY CHAIN ABOVE IS SHOWN ON THE CHIP LEVEL. EACH CHIP HAS AN INTERNAL DAISY CHAIN, SIMILAR TO THE ABOVE, TO PRIORITIZE INDIVIDUAL PORTS. FOR EXAMPLE, THE SIO INTERNAL DAISY CHAIN IS SHOWN BELOW. THIS IS AN EXPANSION OF THE SIO BOX IN THE ABOVE DRAWING:



"IEI" = INTERRUPT ENABLE IN

"IEO" = INTERRUPT ENABLE OUT

INTERRUPT DAISY CHAIN CONFIGURATION

SYSTEM PIO

In addition to the device I/0 ports, there are two additional ports which control certain on-card functions and the eight System Bus interrupt lines.

The functions provided by port A are: (Refer to PIO Block Diagram, Page 59)

1. Interrupt service request from floppy Disk, Winchester, APU or GPIB logic
2. System Bus Release control
3. Interrupt from certain on-card I/0 devices.

Port A

Bit #	Name	Type	Function (True State)
7	FDIO-INTRQ	Input	Interrupt request from FDIO(HIGH)
6	WINC-INTRQ	Input	Interrupt request from WINCHESTER(LOW)
5	APU/GPIB-INT	Input	Interrupt request from APU or GPIB(LOW)
4	BC1	Output	Bus Control bit 1
3	BCØ	Output	Bus Control bit Ø
2	S2	Output	DMA Ready Select 2
1	S1	Output	DMA Ready Select 1
Ø (LSB)	SØ	Output	DMA Ready Select Ø

The two bus control lines(BC1, BCØ)determine what conditions will cause the system bus to be released following a bus operation. This feature is described in detail in another section of this manual (see "Bus Control" page 44). The default value should be BC1 and BCØ LOW which will release the bus between every operation.

The two DMA Ready select lines determine which I/O port READY signal will be used by the DMA to synchronize DMA data transfers. These two bits control the selector as follows:

Select Bit			
S2	S1	SØ	DMA Ready Signal (True state)
Ø	Ø	Ø	Not used
Ø	Ø	1	Not used
Ø	1	Ø	Not used
Ø	1	1	SIO Ready (HIGH)
1	Ø	Ø	Streamer TAPE Ready (LOW)
1	Ø	1	GPIB Ready (LOW)
1	1	Ø	WINCHESTER Ready (LOW)
1	1	1	FDIO Ready (HIGH)

The select bits should be initialized to whichever I/O port will be used with the DMA. It is allowable to change the select bits between DMA operations.

At system initialization, port A should be setup in the BIT control mode and the eight I/O lines should be properly specified as inputs or outputs. This sequence of instructions could be used:

```
     LD         A,ØCFH          ; PIO "BIT" MODE CONTROL
     OUT        (IOPAC),A       ; SEND TO PORT A3 CONTROL
     LD         A,ØEØH          ; IN/IN/IN/OUT/OUT/OUT/OUT/OUT
     OUT        (IOPAC),A       ; SET I/O MASK
```

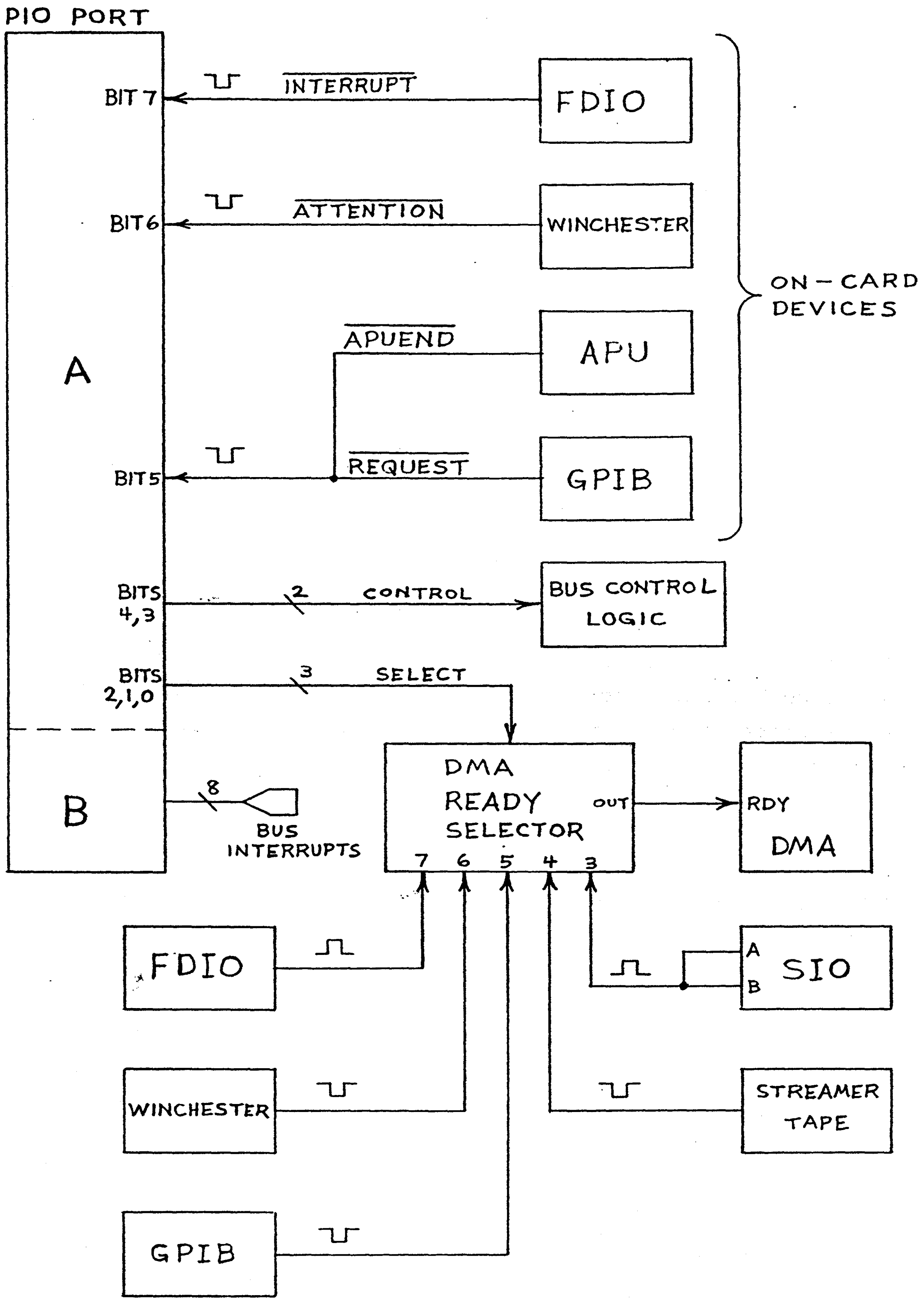
Next, the state of the Bus Control and DMA Ready select lines should be specified:

```
     LD         A,17H           ; BCL LOW, FDIO RDY SELECT
     OUT        (IOPAD),A       ; SET OUTPUT BITS
```

Later, the state of the Bus Control or Ready select bits may be changed by using an instruction sequence similar to the following:

```
     IN         A,(IOPAD)       ; READ CURRENT STATE
     AND        mask            ; TURN OFF DESIRED BITS
     OR         data            ; SET DESIRED BITS
     OUT        (IOPAD),A       ; RESTORE
```

For a description of the use of port A to generate an interrupt, refer to the section on the APU (page 76).



SYSTEM PIO BLOCK DIAGRAM

PIO port B is used to monitor or control the eight System Bus interrupts.. This port should be initialized in the "BIT" mode and, normally, all lines will be inputs, as follows:

```
        LD         A,ØCFH          ; PIO "BIT" MODE CONTROL
        OUT        (IOPBC),A       ; SEND TO PORT B CONTROL
        LD         A,ØFFH          ; I/O MASK (ALL INPUTS)
        OUT        (IOPBC),A       ; SETUP ALL LINES AS INPUTS

        LD         A,vector        ; INTERRUPT MODE 2 VECTOR
                                     (LOW HALF)
        OUT        (IOPBC),A       ; SET VECTOR

        LD         A,97H           ; INTERRUPT ENBL, "OR",
                                     "LOW" STATE
        OUT        (IOPBC),A       ; SEND TO B CONTROL

        LD         A,mask          ; ENBL/DSBL MASK (Ø=ENABLE)
        OUT        (IOPBC),A       ; SELECT LINES TO BE MONITORED

        LD         A,vector        ; HIGH HALF OF INTERRUPT VECTORS
        LD         I,A             ; SET INTERRUPT REGISTER
        IM         2               ; SELECT INTERRUPT MODE 2
        EI                         ; ENABLE INTERRUPTS
```

A mode 2 Z8Ø interrupt will be generated by port B when any selected bus interrupt lines go LOW. By using specific masks while processing an interrupt, a bus interrupt priority structure can be implemented.

The states of the bus interrupt lines are still available even if Z8Ø or port A interrupts are disabled. Executing an INPUT instruction from port A (address IOPBD) will return the current state of the eight interrupt lines.

A bus interrupt can be <u>generated</u> (for another processor board) by port B if the "I/O MASK", above, specifies an interrupt bit as an <u>output</u> line instead of an input. Then, by sending data to port address IOPBD (Port B DATA) these lines may be selectively turned on or off.

<u>Note</u>: The actual values assigned to the port addresses (e.g., IOPAD, IOPBC, etc.) will depend on the data stored in the I/O mapping RAM. See pages 42 and 110.

PIO Programming

LOAD INTERRUPT VECTOR

The Z80-CPU requires an 8-bit interrupt vector be supplied by the interrupting device. The CPU forms the address for the interrupt service routine of the port using this vector. During an interrupt acknowledge cycle the vector is placed on the Z-80 data bus by the highest priority device requesting service at that time. The desired interrupt vector is loaded into the PIO by writing a control word to the desired port of the PIO with the following format.

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	0

signifies this control word is an interrupt vector

SELECTING AN OPERATING MODE

When selecting an operating mode, the 2-bit mode control register is set to one of four values. These two bits are the most significant bits of the register, bits 7 and 6; bits 5 and 4 are not used while bits 3 through 0 are all set to 1111 to indicate "set mode."

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

mode word

signifies mode word to be set

X=unused bit

Mode	M_1	M_0
Output	0	0
Input	0	1
Bidirectional	1	0
Bit	1	1

MODE 0 active indicates that data is to be written from the CPU to the peripheral.

MODE 1 active indicates that data is to be read from the peripheral to the CPU.

MODE 2 allows data to be written to or read from the peripheral device.

MODE 3 is intended for status and control applications. When selected, the next control word must set the I/O Register to indicate which lines are to be input and which lines are to be output.

I/O = 1 sets bit to input.
I/O = 0 sets bit to output.

D7	D6	D5	D4	D3	D2	D1	D0
I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

INTERRUPT CONTROL

Bit 7 = 1 interrupt enable is set–allowing interrupt to be generated.

Bit 7 = 0 indicates the enable flag is reset and interrupts may not be generated.

Bits 6,5,4 are used in the bit mode interrupt operations; otherwise they are disregarded.

Bits 3,2,1,0 signify that this command word is an interrupt control word.

D7	D6	D5	D4	D3	D2	D1	D0
Enable Interrupt	AND/OR	High/Low	Mask follows	0	1	1	1

used in Mode 3 only

signifies interrupt control word

If the "mask follows" bit is high (D4 = 1), the next control word written to the port must be the mask.

D7	D6	D5	D4	D3	D2	D1	D0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

Only those port lines whose mask bit is a 0 will be monitored for generating an interrupt.

The interrupt enable flip-flop of a port may be set or reset without modifying the rest of the interrupt control word by the following command.

D7	D6	D5	D4	D3	D2	D1	D0
Int Enable	X	X	X	0	0	1	1

```
Use Mode 3 for both ports
   i.e., use CF hex
```

DIP SWITCHES AND LEDs

The DIP Switch/LED feature of the MLZ-91 allows software controlled options to be selected and status valves to be displayed.

DIPs

There are 16 dip switches, located near the I/O edge of the board, which are accessible without removing the MLZ-91 board from the Multibus card rack. The functions of each switch are defined by the software (except that four switches are used by the streamer tape interface, if the tape option is used). Examples of switch use are:

1. SIO baud rate selection
2. Software option selection
3. Board position on bus or board priority

In each case, the program is responsible for reading the switch position and taking the action necessary to implement the function (e.g. loading the baud rate generator with the data read from the switches.) See example, page 93.

The 16 switches are arranged as two 8-bit groups with each group accessible as an input device.

Switch Group	Switch Numbers	I/O Port Address
Ø	1-8 (D7-DØ)	IODIPØ
1	9-16(D7-DØ)	IODIP1

Data bit	Switch State
Ø	ON (down)
1	OFF(up)

See diagram, next page for switch locations.

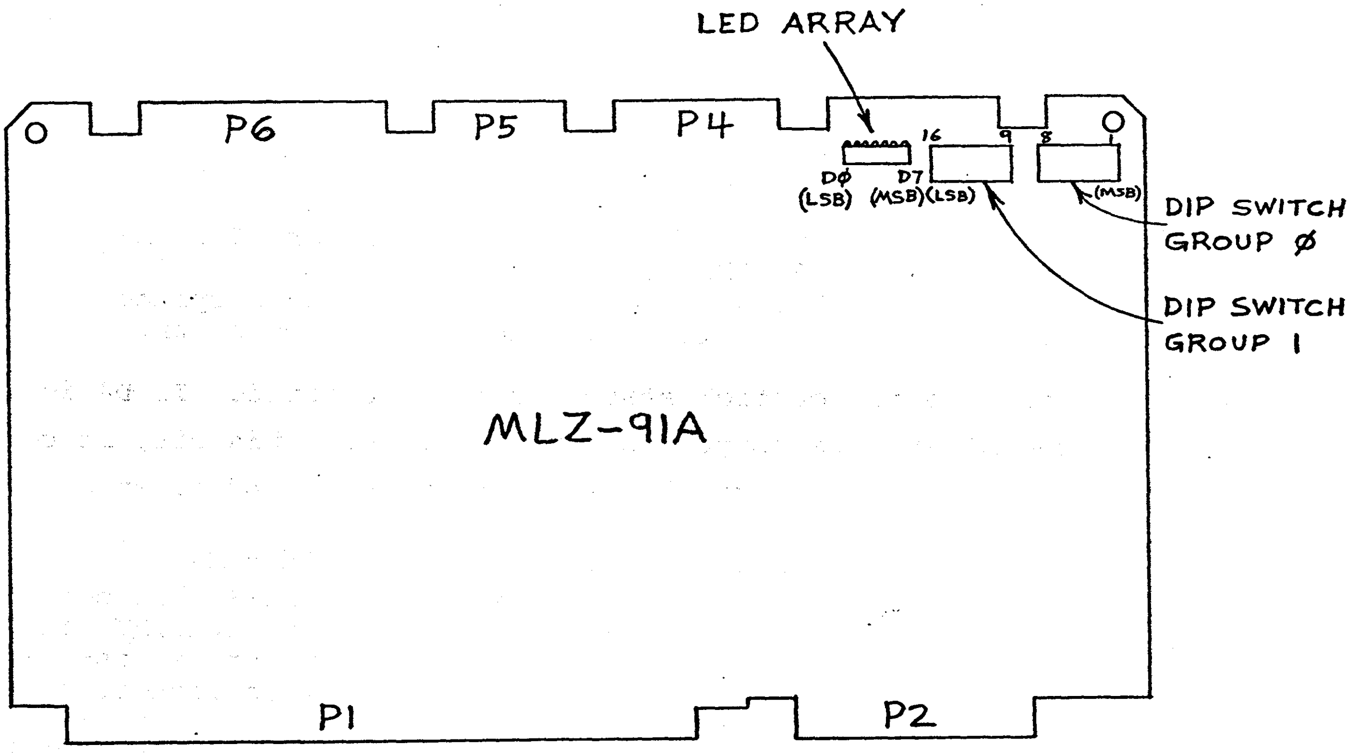
LED ARRAY

There are eight LEDs arranged as one 8-bit output port. The functions of each LED is specified by the software. (Two of the LEDs are required by the streamer tape interface, if the tape option is used.)

LED PORT ADDRESS: IOLED (Write only)

Data bit	LED State
Ø	ON
1	OFF

The LEDs are not initialized to any particular state at power-on and are not changed by RESET.



For restrictions on use of the switches and LEDs with the streamer tape interface, refer to page 73.

BOARD STATUS

It is sometimes useful to know the state of a few on-card signals for memory error processing and bus control functions. There is an input port assigned to allow four signals to be read as follows:

STATUS PORT ADDRESS: IOSTAT

Data bit	Function (Negative true, Ø="on")
D7	NMI flip flop (See page 36)
D6	PARITY ERROR flip flop (See page 36)
D5	(for Winchester I/F, see page 70)
D4	BUS ADRS ENABLE state

The state of D7 & D6 can be used following an NMI to determine the reason for the interrupt. There are also two status LEDs on the MLZ-91 which visually indicate an error condition.

D7	D6	Reason for NMI	Status LEDs
Ø	Ø	On-card RAM parity error	Both on
Ø	1	(invalid)	
1	Ø	On-card RAM write protect error	One on
1	1	(No error, reset state)	Both off

D7 and D6 may be reset (to one) by executing any I/O instruction to I/O PORT ADDRESS: IOCLRN
NOTE: If D6 is not reset back to one by either a system RESET or an access to IOCLRN another NMI cannot occur.

Bit D4 allows the bus control status to be determined. If D4 is low (Ø) then the MLZ-91 has control of the Multibus. This bit, in conjunction with the bus control lines can be used as indicated below:

Bus Control lines (BC1 & BCØ)	ADRS-ENBL (D4)	Meaning
Both ON (11)	D4=Ø	MLZ-91 has control of the MULTIBUS and will never release control (bus lockout state)
	D4=1	MLZ-91 does not have control of the MULTIBUS but will capture control on the next attempt to use the bus
Either OFF (10, 01, 00)	D4=Ø	MLZ-91 has control of the bus. No other board has used the bus since the last bus access by the MLZ-91
	D4=1	MLZ-91 does not have control of the bus. The bus has been used since the last bus access by the MLZ-91

GPIB (IEEE-488) I/F

The General Purpose Interface Bus was originally designed by Hewlett-Packard to provide a universal method of device interconnection. The basic bus definition gained acceptance by the IEEE and has since become known as IEEE standard 488.

The MLZ-91 utilizes a Texas Instruments GPIB controller chip (the TMS-9914) which handles the bus protocol and allows the Z-80 to operate the bus with a minimum amount of concern for the actual bus control signals. For example, the data valid (DAV), not data accepted (NDAC) and not ready for data (NRFD) handshaking during data transfer operations is transparent to the CPU side of the TMS-9914. The CPU tests bits in a status register in order to synchronize data transfers.

The GPIB devices are defined as controllers, talkers or listeners. In any system there is one active controller, one talker and any number of listeners. The controller specifies which devices are talkers and listeners. A device can be both controller and talker or controller and listener. For example, the MLZ-91 could designate itself as controller and listener and receive data from a data acquisition device (talker).

For more details on the TMS-9914, refer to the TI TMS-9914 GPIB ADAPTER DATA MANUAL. For information concerning licensing of the GPIB by Hewlett-Packard, contact Hewlett-Packard, Legal Department.

See page 160 for MLZ-91 connection information.

The program on the following pages illustrates one method of using the GPIB feature of the MLZ-91. (This example has been adapted from the example in the TI Data Manual.)

```
        ASEG
        .Z80
        TITLE   GPIB (IEEE-488) I/F EXAMPLE
;       HEURIKON CORPORATION
RAM     EQU     0E000H          ;RAM BASE
ROM     EQU     0F000H          ;ROM BASE (PGM)
;
IOA     EQU     080H            ;IOA (DEVICE GROUP A) BASE ADRS
IOB     EQU     0C0H            ;IOB (DEVICE GROUP B) BASE ADRS
;
IODIP1  EQU     IOA+39H         ;DIP SWITCH GROUP 1
IOPAD   EQU     IOB+38H         ;I/O PIO PORT A DATA ADRS
IOGPIB  EQU     IOB+30H         ;BASE OF GPIB CHIP REGISTERS
;*************************************************************************
;THIS EXAMPLE HAS BEEN ADAPTED FROM THE SOFTWARE EXAMPLE SHOWN IN THE
;TI MANUAL FOR THE TMS-9914 GPIB CONTROLLER CHIP.  THE PORT ADDRESSES
;AND INSTRUCTIONS HAVE BEEN CHANGED TO CONFORM WITH THE MLZ-91.
;*************************************************************************
;A TYPICAL SYSTEM USING THE GPIB/IEEE-488 BUS CONSISTS OF THREE ITEMS:
;       DEVICE 1:  SYSTEM CONTROLLER (CPU, MEMORY, ETC.)
;       DEVICE 2:  INSTRUMENT PRODUCING DATA (TALKER)
;       DEVICE 3:  INSTRUMENT ACCEPTING DATA (LISTENER - E.G., PRINTER)
;
;THIS PROGRAM EXAMPLE RUNS THE CONTROLLER IN ORDER TO SET UP DEVICE 2.
;THE ASCII CONTROL CHARACTERS REQUIRED TO PROGRAM DEVICE 2 FOR RANGE,
;FUNCTION, ETC., ARE ASSUMED TO HAVE BEEN PREVIOUSLY LOADED INTO MEMORY
;AT LOCATION 'DEVDAT'.  DEVICE 2 IS FIRST ADDRESSED TO LISTEN AND THE CONTROL
;CHARACTERS ARE SENT TO DEVICE 2. THEN DEVICE 2 IS ADDRESSED TO TALK AND
;DEVICE 3 IS ADDRESSED TO LISTEN.  WHEN THE CONTROLLER PUTS ITSELF IN THE
;STANDBY MODE, THE MEASUREMENT DATA FROM DEVICE 2 IS SENT TO DEVICE 3 OVER
;THE GPIB.  IN PRACTICE, THE FORMAT OF THE DATA SENT BETWEEN DEVICES
;WOULD PROBABLY HAVE TO BE CHANGED BUT THIS STEP IS OMITTED HERE FOR
;CLARITY.
;*************************************************************************
;GPIB REGISTER ADDRESSES:
;READ REGISTERS:
INTST0  EQU     IOGPIB+0        ;INTERNAL STATUS REGISTER 0
INTST1  EQU     IOGPIB+1        ;INTERNAL STATUS REGISTER 1
ADDSTS  EQU     IOGPIB+2        ;ADDRESS STATUS
BUSSTS  EQU     IOGPIB+3        ;BUS STATUS
CMDPAS  EQU     IOGPIB+6        ;COMMAND PASS THROUGH
DATIN   EQU     IOGPIB+7        ;DATA IN
;WRITE REGISTERS:
INTMK0  EQU     IOGPIB+0        ;INTERRUPT MASK 1
INTMK1  EQU     IOGPIB+1        ;INTERRUPT MASK 2
AUXCMD  EQU     IOGPIB+3        ;AUXILIARY COMMAND REG
ADDRSS  EQU     IOGPIB+4        ;ADDRESS REG
SERPOL  EQU     IOGPIB+5        ;SERIAL POLL
PARPOL  EQU     IOGPIB+6        ;PARALLEL POLL
DATOUT  EQU     IOGPIB+7        ;DATA OUT
;*************************************************************************
```

```
;***************************************************************************
;TMS-9914 AUXILIARY COMMANDS:
TCA     EQU     0DH             ;TAKE CONTROL ASYNCHRONOUSLY
TON     EQU     8AH             ;TALK ONLY
TONCLR  EQU     0AH             ;CLEAR TALK ONLY
CLRRST  EQU     00H             ;CLEAR CHIP RESET
GTS     EQU     0BH             ;GO TO STANDBY
SDWH    EQU     96H             ;SHADOW HANDSHAKE
SRE     EQU     90H             ;SEND REMOTE ENABLE
SRECLR  EQU     10H             ;CLEAR REMOTE ENABLE
SIC     EQU     8FH             ;SEND INTERFACE CLEAR
SICCLR  EQU     0FH             ;CLEAR INTERFACE CLEAR CMD
;***************************************************************************
;INTERFACE CONTROL COMMANDS:
UNL     EQU     3FH             ;UNLISTEN ALL DEVICES
UNT     EQU     5FH             ;UNTALK ALL DEVICES
;***************************************************************************
;DATA:
        ORG     RAM
COUNT   EQU     7               ;7 BYTES FOR THIS EXAMPLE
DEVDAT: DS      COUNT           ;RESERVE SPACE FOR TALKER CONTROL
;***************************************************************************
;DEVICE ADDRESSES ON GPIB:
LISAD1  EQU     '!'             ;(33) LISTEN ADRS 1
TAKAD1  EQU     'A'             ;(65) TALK ADRS 1
LISAD2  EQU     '"'             ;(34) LISTEN ADRS 2
TAKAD2  EQU     'B'             ;(66) TALK ADRS 2
LISAD3  EQU     '#'             ;(35) LISTEN ADRS 3
TAKAD3  EQU     'C'             ;(67) TALK ADRS 3
;***************************************************************************
```

```
;*********************************************************************
        ORG     ROM
;THIS IS THE MAIN PROGRAM:
;PIO A AND STACK POINTER ARE ASSUMED TO BE INITIALIZED.
;*********************************************************************
;STEP 1:  INITIALIZE THE GPIB CHIP.
;         LOAD THE DEVICE ADRS FROM THE DIP SWITCHES
;         SEND INTERFACE CLEAR
;         CLEAR AUXILIARY RESET
GPIB:   IN      A,(IODIP1)      ;READ DIP SWITCH GROUP 1
        CPL                     ;FIX 'ON' = 1
        OUT     (ADDRSS),A      ;SEND TO GPIB ADDRESS REGISTER
;
        LD      A,SIC           ;I/F CLEAR CMD
        OUT     (AUXCMD),A      ;SEND INTERFACE CLEAR COMMAND
        LD      A,CLRRST        ;CLR S/W RESET CMD
        OUT     (AUXCMD),A      ;START SENDING IFC ON BUS
;
        LD      B,31            ;LOOP COUNT
DLY:    DJNZ    DLY             ;DELAY 100 USEC. (AT 4MHZ CPU CLOCK)
;
        LD      A,SICCLR        ;CLR IFC CMD
        OUT     (AUXCMD),A      ;CLEAR IFC COMMAND
        LD      A,SRE           ;REMOTE ENABLE CMD
        OUT     (AUXCMD),A      ;SEND REMOTE ENABLE CMD
;THE GPIB CHIP IS NOW IN THE CONTROLLER ACTIVE STATE AND REMOTE ENABLE
;HAS BEEN SENT TO ALL DEVICES.
;*********************************************************************
;STEP 2:  ADDRESS DEVICE 2 TO LISTEN
;         SEND DEVICE DEPENDENT CONTROL DATA TO DEVICE 2
        IN      A,(INTSTO)      ;CLEAR BYTE OUT INTERRUPT STATUS
;
        LD      A,LISAD2        ;LOAD DEVICE 2 LISTEN ADRS
        CALL    DATAW           ;SEND TO DATA REG AND WAIT
;
        LD      A,TON           ;TALK ONLY COMMAND
        CALL    AUXW            ;ADDRESS SELF TO TALK
        LD      B,COUNT         ;LOOP COUNT
        LD      HL,DEVDAT       ;BASE OF DEVICE CONTROL DATA
;
LOOP:   LD      A,(HL)          ;GET BYTE
        INC     HL              ;INCR FOR NEXT LOOP
        CALL    DATAW           ;SEND TO DATA REG AND WAIT
        DJNZ    LOOP            ;SEND ALL DEVICE DEPENDENT CONTROLS
;
        LD      A,TCA           ;TAKE CONTROL ASYNCRONOUSLY CMD
        CALL    AUXW            ;SEND AND WAIT FOR BO
        LD      A,TONCLR        ;CLEAR TALK ONLY COMMAND
        OUT     (AUXCMD),A      ;SEND TO AUXCMD REG
        LD      A,UNL           ;UNLISTEN ALL DEVICES CMD
        CALL    DATAW           ;SEND TO DATA REG AND WAIT FOR BO
;DEVICE 2 IS NOW SET UP TO TAKE MEASUREMENTS.
;*********************************************************************
```

```
;***************************************************************************
;STEP 3:  DEVICE 2 IS ADDRESSED TO TALK
;         DEVICE 3 IS ADDRESSED TO LISTEN
        LD      A,LISAD3        ;LISTEN ADRS FOR DEVICE 3
        CALL    DATAW           ;SEND AND WAIT
;
        LD      A,TAKAD2        ;TALK ADRS OF DEVICE 2
        CALL    DATAW           ;SEND AND WAIT
;***************************************************************************
;STEP 4:  THE CONTROLLER NOW RELEASES THE ATTENTION LINE AND MONITORS
;THE 'EOI' LINE UNTIL AN END OCCURS.
        LD      A,SDWH          ;SHADOW HANDSHAKE CMD
        OUT     (AUXCMD),A      ;LOAD INTO AUXCMD REG
        LD      A,GTS           ;RELEASE ATN LINE CMD
        OUT     (AUXCMD),A      ;SEND TO GPIB CHIP
;
;WE ASSUME THAT PIO A HAS BEEN INITIALIZED IN THE BIT CONTROL MODE.
WAIT:   IN      A,(IOPAD)       ;READ GPIB INTERRUPT LINE FROM PIO A
        BIT     5,A             ;TEST GPIB BIT
        JR      NZ,WAIT         ;LOOP HERE UNTIL TRUE
;ALTERNATE: PIO A COULD BE SET TO INTERRUPT IF THE GPIB INT LINE
;GOES TRUE (LOW).
        JP      SOMEWHERE       ;FROM HERE GO TO REST OF PROGRAM
;***************************************************************************
;
;***************************************************************************
;THESE ARE SUBROUTINES WHICH OUTPUT A COMMAND AND THEN WAIT FOR THE
;BYTE OUT ('BO') STATUS BIT.
DATAW:  OUT     (DATOUT),A      ;SEND TO DATA OUT REGISTER
        JR      BOWAIT          ;WAIT FOR BYTE OUT
;
AUXW:   OUT     (AUXCMD),A      ;SEND TO AUXILIARY COMMAND REG
BOWAIT: IN      A,(INTSTO)      ;READ INTERNAL STATUS REGISTER
        BIT     4,A             ;TEST BO BIT
        JR      NZ,BOWAIT       ;LOOP UNTIL READY
        RET
;***************************************************************************
        END
```

WINCHESTER INTERFACE

The MLZ-91 has been designed to allow easy connection to numerous Winchester drives and controllers. The interface provides a byte level, DMA controlled data transfer to achieve maximum speed.

All connections to the drive controllers is via P2 (the "Auxiliary" connector). This 60 pin connector has been logically split into sections and five hardware jumpers are used to account for variations between controllers. The summary below shows the basic controller interfaces which are accommodated by the MLZ-91. Refer to pages 150 through 155 for P2 pinout details and cable specifications.

Controller	P2 Pins	Cable type	Jumpers	Page
Micropolis	1-34	MLZ-P2M	"M"	150
Priam	35-60	MLZ-P2P	n/a	152
Shugart	1-34	MLZ-P2S	"S"	154
Seagate	1-34	MLZ-P2S	"S"	154
DTC	1-34	MLZ-P2S	"S"	154

There are 5 jumpers which must be set either "S" or "M" depending on the controller being used. (See page 144)

J11	"S"		(no "M")
J15	"S"	or	"M"
J16	"S"	or	"M"
J16	"S"	or	"M"
J18	"S"	or	"M"

I/O Port Addresses (See page 111)

The functions of each I/O port, meaning of control signals and commands are highly dependent on the particular controller being used. The following information is summary in nature. Refer to the MLZ-91 schematics and controller manuals or contact Heurikon for assistance.

	FUNCTION		
I/O Port Name	Micropolis	Shugart	Priam
IOWSEL	Select	Select	(none)
IOWCLR	Deselect	Clear SEL	(none)
IOWWRØ	WR DATA	WR DATA/CMD	(Reg B)
IOWWR1	Command	WR DATA/CMD	
IOWRDØ	RD DATA	RD DATA/STATUS	(Reg B)
IOWRD1	Status	RD DATA/STATUS	

IOWRDS	Read general interface status
	D7 = Attention (LOW true)
	D6 = Data Request (LOW true)
	D5 = Busy (LOW true, except HIGH for Micropolis)
	D4 = OUT line (LOW true)		

IOSTAT	Read Board Status D7 (See page 64) D6 (See page 64) D5 = DIRECTION (for Shugart, LOW true) or CBUSY (for Micropolis, HIGH true)		

When I/O instructions are executed with the Priam controller, the contents of register B specifies the Priam control register as follows (use "IN A, (C)" or "OUT (C), A")

Register B	INPUT	OUTPUT
Ø	Result 5	Parameter 5
1	Result 4	Parameter 4
2	Result 3	Parameter 3
3	Result 2	Parameter 2
4	Result 1	Parameter 1
5	Result Ø	Parameter Ø
6	Read Data	Write Data
7	I/F Status	Command

If the MLZ-91 DMA is accessing the Winchester interface, the Priam Data register is automatically selected. (The DMA cannot directly access any of the other Priam registers.)

General

Data signals on the interface lines are inverted with respect to the CPU or DMA. Therefore, commands, status and data passed between the CPU or DMA and the controller will appear inverted from the specifications in the manuals for those controllers which have a positive true data bus, e.g., Priam. Thus, care must be taken to invert command and status bit patterns when using some controllers. The actual sector data need not be inverted in any case since the Winchester media is not removable. If data is written "inverted" by the interface it will be reinverted when read back.

The Data request (or Data Ready) and Attention (or Interrupt Request) are available to the CPU as bits via I/O port IOWRDS. The Data request line may be selected as the DMA Ready signal and the Attention signal can be programmed to cause a CPU interrupt via the System PIO. Refer to page 57 for details on how to configure the System PIO.

Heurikon provides software support for various Winchester controllers. Consult factory for details.

STREAMER TAPE

The streamer tape logic on the MLZ-91 performs two functions:

1. Provide interface to the Archive Corporation Streamer tape for backup of Winchester data and program files.
2. Provide a general purpose 8-bit parallel I/O port for systems not using a streamer tape.

The Archive streamer tape controller has been designed for programmed I/O of commands and status with either programmed I/O or DMA transfer of the actual tape data. The average byte transfer rate for continuous operation at 90 ips is 87,200 bytes/second. Thus, 20 megabytes of Winchester data can be backed up in as little as four minutes.

The hardware interface consists of the following signals:

Signal Name	Source	Function (all are negative true)
HBØ-HB7	(both)	Bi-directional data bus used for commands, status and data transfers.
$\overline{\text{ONLINE}}$ *	MLZ-91	Active during command transfer and execution.
$\overline{\text{REQUEST}}$ *	MLZ-91	Used as a handshake for command and status transfers
$\overline{\text{RESET}}$	MLZ-91	Power-on reset
$\overline{\text{TRANSFER}}$	MLZ-91	Used as a handshake for data transfers
$\overline{\text{ACKNOWLEDGE}}$ *	Archive	Used as a handshake for data transfers
$\overline{\text{READY}}$ *	Archive	Used as a handshake for command and status transfers and as buffer status during data transfers.
$\overline{\text{EXCEPTION}}$*	Archive	Indicates an error condition
$\overline{\text{DIRECTION}}$ *	Archive	Indicates the direction of the data transfer

Consult the Archive product description and specifications for details on these signals and the sequence of operations for using the streamer tape drive.

Those signals marked "*" above share circuits with the DIP switches and LED array.

Signal	Source	Connection	Port Name	State
$\overline{\text{ONLINE}}$	MLZ-91	LED array, bit 1	IOLED	Ø = TRUE
$\overline{\text{REQUEST}}$	MLZ-91	LED array, bit Ø	IOLED	Ø = TRUE
$\overline{\text{READY}}$	Archive	DIPSWØ, bit 7	IODIPØ	Ø = TRUE
$\overline{\text{EXCEPTION}}$	Archive	DIPSWØ, bit 6	IODIPØ	Ø = TRUE
$\overline{\text{DIRECTION}}$	Archive	DIPSWØ, bit 5	IODIPØ	Ø = TRUE
$\overline{\text{ACKNOWLEDGE}}$	Archive	DIPSWØ, bit 4	IODIPØ	Ø = TRUE

When using the streamer tape interface, DIP switches 1,2,3 and 4 must be OFF (OPEN) to allow those bits to be used by the tape controller. Also, LED bits Ø and 1 must be reserved for the tape interface. The other 12 DIP switches and 6 LEDs may be used as desired by the application.

The $\overline{\text{HBØ}}$ - $\overline{\text{HB7}}$ Data bus signals have been defined by Archive as negative true. The MLZ-91 logic, however, does not invert these lines. Thus, commands and status bits must be inverted from the definitions in the Archive documentation. The actual tape data written to the drive need not be inverted since the data will be re-inverted by the read operation.

Heurikon provides software support for the Archive streaming tape drive. Consult factory for details.

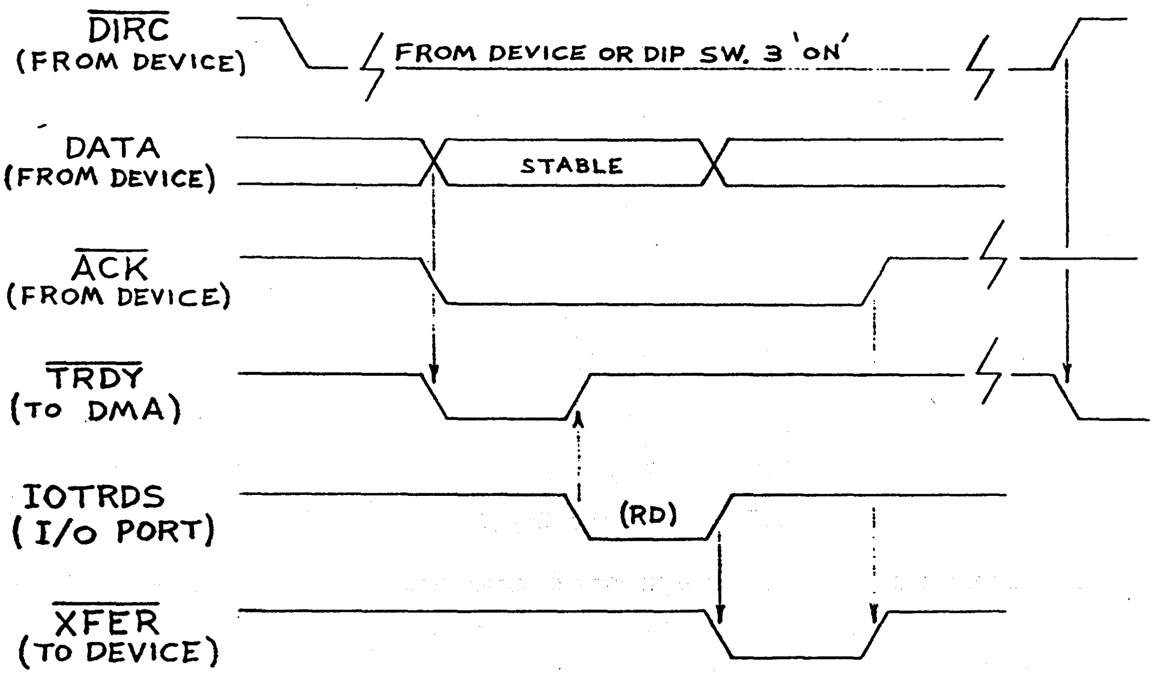
The Archive controller connects to P3. See page 156 for pinout and cable details.

The interface circuits on the MLZ-91 generate a data ready signal for use with DMA data transfers. (Refer to page 57.)

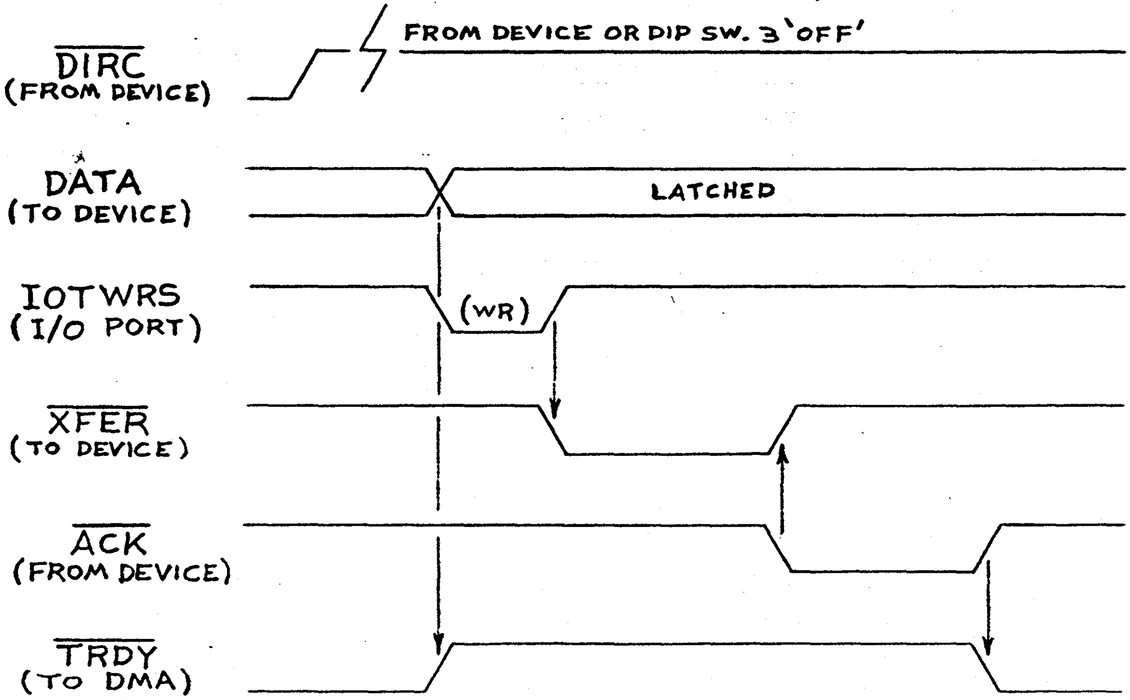
LED "1" will indicate when a data transfer is in progress since ONLINE will true during that time.

Use as a general purpose PIO port

If a streamer tape is not connected to P3, the interface circuits may be used as a general purpose eight bit parallel port. $\overline{\text{ONLINE}}$, $\overline{\text{REQUEST}}$, $\overline{\text{READY}}$ and $\overline{\text{EXCEPTION}}$ may be used as control and handshake signals via the IODIPØ and IOLED ports. The $\overline{\text{DIRECTION}}$ signal may be controlled by DIP switch #3, the device connected to P3 or by the CPU by jumpering $\overline{\text{ONLINE}}$ or $\overline{\text{REQUEST}}$ to $\overline{\text{DIRECTION}}$ via P3. $\overline{\text{DIRECTION}}$ = LOW or DIPSW 3 ON means data transfer is from the device to the MLZ-91.


$\overline{\text{DIRC}}$ (FROM DEVICE)
FROM DEVICE OR DIP SW. 3 'ON'
DATA (FROM DEVICE)
STABLE
$\overline{\text{ACK}}$ (FROM DEVICE)
$\overline{\text{TRDY}}$ (TO DMA)
IOTRDS (I/O PORT)
(RD)
$\overline{\text{XFER}}$ (TO DEVICE)

DATA INPUT (RD) CYCLE


FROM DEVICE OR DIP SW. 3 'OFF'
$\overline{\text{DIRC}}$ (FROM DEVICE)
DATA (TO DEVICE)
LATCHED
IOTWRS (I/O PORT)
(WR)
$\overline{\text{XFER}}$ (TO DEVICE)
$\overline{\text{ACK}}$ (FROM DEVICE)
$\overline{\text{TRDY}}$ (TO DMA)

DATA OUTPUT (WR) CYCLE

APU

The (optional) Am9511 Arithmetic Processing Unit (APU) contains firmware which executes high level math functions and relieves the Z-80 processor lengthy software routines.

The APU performs fixed and floating point arithmetic plus a variety of trigonometric functions. Operands (16, 32 or 64 bits each) are pushed into the APU's internal stack by using the Z80 OUTPUT instruction. A command is then issued, also via an OUTPUT instruction, which initiates a particular math operation. When the function is complete, the result may be popped from the APU's stack by using an INPUT instruction.

Here is a typical example of the command sequence for the AM9511 APU. This example will multiply the contents of register pair HL by the contents of register pair DE.

```
LD      C,IOPUSH          ;APU PUSH DATA PORT ADRS
OUT    (C),L              ;L HALF OPERAND 1
OUT    (C),H              ;H HALF OPERAND 1
OUT    (C),E              ;L HALF OPERAND 2
OUT    (C),D              ;H HALF OPERAND 2
LD     A,ØF6H             ;"SMUL" COMMAND (MULTIPLY,
                                                UPPER)
OUT    (IOAPUW),A         ;INITIATE MULTIPLY
```

At this point, the requested operation is executed. When the operation is complete, the result may be popped from the APU's stack. However, if the operation is not completed when th result is read, the CPU will enter a wait state until the APU is don

```
LD      C,IOPOP           ;APU POP DATA PORT ADRS
IN      H,(C)             ;H HALF RESULT
IN      L,(C)             ;L HALF RESULT
IN      A,(IOAPUR)        ;READ COMPLETION STATUS
```

This example used 16 bit operands. However, some APU commands operate on 32 or 64 bit operands.

In some systems it may be undesirable to have the processor enter an extended wait state prior to reading the result. Two methods may be used to prevent such a condition:

1. Read the APU/FPU status port and wait (in a loop) for the "busy" bit to go false.

```
LOOP:  IN     A,(IOAPUR)        ; READ APU/FPU STATUS
       AND    8ØH               ; TEST BUSY BIT
       JR     NZ, LOOP          ; WAIT FOR NOT BUSY
                                ; NOW READ RESULT
```

2. Enable the APU END interrupt via PIO port A.

   a) Program port A to interrupt when the APU END signal goes true.

   b) Set a flag or read the result in the interrupt routine.

```
LD     A,vector         ;LOW HALF OF A INT VECTOR
OUT    (IOPAC),A        ;SET INTERRUPT VECTOR
LD     A,17H            ;BCL LOW (or as desired)
OUT    (IOPAD),A        ;SET A DATA LINES
LD     A,97H            ;ENABLE INTERRUPT ON END LOW

OUT    (IOPAC),A        ;SET A CONTROL
LD     A,ØDFH           ;MASK (D5 LOW)
OUT    (IOPAC),A        ;ONLY MONITOR DMA READY
```

   c) Load the upper half of the interrupt vector via:

```
LD     A,vector         ;HIGH HALF OF VECTOR
LD     I,A              ;SET I REGISTER
IM     2                ;SELECT INT MODE 2
```

   d) Execute an EI (Enable Interrupt) instruction.

Note: The APU clock rate (2/4 MHZ) is set by jumper J2 and is independent of the processor clock. See page 142.

STATUS REGISTER

BUSY	SIGN	ZERO	ERROR CODE				CARRY
7	6	5	4	3	2	1	0

BUSY: Indicates that AM9511A is currently executing a command (1=Busy).

SIGN: Indicates that the value on the top of stack is negative (1=Negative).

ZERO: Indicates that the value on the top of stack is zero (1=Value is zero).

ERROR CODE: This field contains an indication of the validity of the result of the last operation. (ØØØØ indicates no error).

CARRY: Previous operation resulted in carry or borrow from most significant bit. (1=Carry/Borrow, 0=No Carry/No Borrow)

If the BUSY bit in the status register is a one, the other status bits are not defined; if zero, indicating not busy, the operation is complete and the other status bits are defined as given above.

APU Command Summary (AM9511)

Fixed-Point, 16-bit

SADD	Add TOS to NOS. Result to NOS. Pop Stack.
SSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
SMUL	Multiply NOS by TOS. Lower half of result to NOS. Pop Stack.
SMUU	Multiply NOS by TOS. Upper half of result to NOS. Pop Stack.
SDIV	Divide NOS by TOS. Result to NOS. Pop Stack.

Fixed-Point, 32-bit

DADD	Add TOS to NOS. Result to NOS. Pop Stack.
DSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
DMUL	Multiply NOS by TOS. Lower half of result to NOS. Pop Stack.
DMUU	Multiply NOS by TOS. Upper half of result to NOS. Pop Stack.
DDIV	Divide NOS by TOS. Result to NOS. Pop Stack.

Floating-Point, 32-bit

FADD	Add TOS to NOS. Result to NOS. Pop Stack.
FSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
FMUL	Multiply NOS by TOS. Result to NOS. Pop Stack.
FDIV	Divide NOS by TOS. Result to NOS. Pop Stack.

Derived Floating-Point Functions

SQRT	Square Root of TOS. Result in TOS.
SIN	Sine of TOS. Result in TOS.
COS	Cosine of TOS. Result in TOS.
TAN	Tangent of TOS. Result in TOS.
ASIN	Inverse Sine of TOS. Result in TOS.
ACOS	Inverse Cosine of TOS. Result in TOS.
ATAN	Inverse Tangent of TOS. Result in TOS.
LOG	Common Logarithm (base 10) of TOS. Result in TOS.
LN	Natural Logarithm (base e) of TOS. Result in TOS.
EXP	Exponential (e^x) of TOS. Result in TOS.
PWR	NOS raised to the power in TOS. Result in NOS. Pop Stack.

Data Manipulation Commands

NOP	No Operation
FIXS	Convert TOS from floating point to 16-bit fixed point format.
FIXD	Convert TOS from floating point to 32-bit fixed point format.
FLTS	Convert TOS from 16-bit fixed point to floating point format.
FLTD	Convert TOS from 32-bit fixed point to floating point format.
CHSS	Change sign of 16-bit fixed point operand on TOS.
CHSD	Change sign of 32-bit fixed point operand on TOS.
CHSF	Change sign of floating point operand on TOS.
PTOS	Push 16-bit fixed point operand on TOS to NOS (Copy)
PTOD	Push 32-bit fixed point operand on TOS to NOS. (Copy)
PTOF	Push floating point operand on TOS to NOS. (Copy)
POPS	Pop 16-bit fixed point operand from TOS. NOS becomes TOS.
POPD	Pop 32-bit fixed point operand from TOS. NOS becomes TOS.
POPF	Pop floating point operand from TOS. NOS becomes TOS.
XCHS	Exchange 16-bit fixed point operands TOS and NOS.
XCHD	Exchange 32-bit fixed point operands TOS and NOS.
XCHF	Exchange floating point operands TOS and NOS.
PUPI	Push floating point constant "π" onto TOS. Previous TOS becomes NOS.

Access to the APU chip may be made directly from a program written in a higher level language by using the language's INPUT and OUTPUT commands in a fashion similar to the machine code examples discussed earlier. For example, the following BASIC subroutine will perform a double precision DIVIDE (32 bit operands) according to the formula

$$Q = D/E$$

```
100     FOR J = 1 TO 2   )          Transfer both operands to the APU
110     FOR I = 1 TO 4   |          stack (dividend "D" first, then
120     OUT(170,D)       |          divisor "E").  Operands are
130     LET D = D/256     >         transferred 8-bits at a time,
140     NEXT I           |          LSBs first.
150     LET D = E        |
160     NEXT J           )

170     OUT(171,172)                Issue divide command "DDIV"

180     LET Q = Ø                 ) Read result.  (Automatic WAIT at
190     FOR I = 1 TO 4             > line 200 until APU ready.)
200     Q = Q*256 + INP(168)      |
210     NEXT I                    )

220     IF INP (169) AND 3Ø GOTO 240
230     RETURN
240     PRINT "DIVIDE ERROR"
250     RETURN
```

Device port numbers (168, 169, 170, 171) assume IOA base is 8ØH. This is a function of the I/O mapping RAM.

For single parameter functions (e.g. Square Root), delete lines 100, 150, and 160.

A careful reader may observe that the BASIC routine above uses 8 divides and 4 multiplies within the BASIC code which nullifies any potential savings from the APU. (The "DDIV" command executes in approximately 100 microseconds while most BASIC multiply and divide operators take considerably longer.) The BASIC program was presented as an example to illustrate the software to hardware interface when using a higher level language. To utilize the full power of the APU it would be best to transfer the operands via a machine code subroutine.

SOFTWARE EXAMPLES

There are many methods which may be used to initialize the MLZ-91 mapping RAMs and I/O devices. The program listings which follow show details of one scheme which could be implemented. These programs are written in Z-80 source code and were assembled on Microsoft's Macro-80 assembler. For additional samples plus samples of initialization software written in 8080 source code and assembled with the Digital Research MAC assembler, refer to the ZRAID monitor source code listing, available from Heurikon.

Program	Page	Description
Init & Slave	82	I/O device assignments
	84	Baud Rate generator constants
	85	Power-on initialization-I/O mapping RAM
	86	Memory mapping RAM
	87	Bus mapping RAM
	88	Map utility subroutine
	89	I/O Device initialization
	91	Example vectored interrupt service routine
	92	NMI service routine
	93	Slave logic - converts the MLZ-91 into an intelligent slave I/O board.
Multi-user	95	Example of using memory mapping RAM to switch between tasks in a multi-user environment.
Macros	98	Sample Macro definitions which facilitate loading the MLZ-91 mapping RAMs.

Heurikon offers software support and applications assistance. Contact us if you have questions or need help.

There are also miscellaneous software samples scattered throughout this manual. Some major examples are:

Topic	Page
System PIO	57
GPIB	66
APU	76
SIO	112
DMA	138

```
;**********************************************************************
;**********************************************************************
;**                                                                  **
;**                                                                  **
;**     H   H  EEEEE  U   U  RRRR   IIIII  K   K   OOO   N   N       **
;**     H   H  E      U   U  R   R    I    K  K   O   O  NN  N       **
;**     H   H  E      U   U  R   R    I    K K    O   O  N N N       **
;**     HHHHH  EEEE   U   U  RRRR     I    KK     O   O  N N N       **
;**     H   H  E      U   U  R R      I    K K    O   O  N N N       **
;**     H   H  E      U   U  R  R     I    K  K   O   O  N  NN       **
;**     H   H  EEEEE   UUU   R   R  IIIII  K   K   OOO   N   N       **
;**                                                                  **
;**                                                                  **
;**********************************************************************
;**********************************************************************
;**                                                                  **
;**      COPYRIGHT 1981   HEURIKON CORPORATION   MADISON, WI         **
;**                                                                  **
;**********************************************************************
;**                                                                  **
;**     PROGRAM:        MLZ-91 INIT & SLAVE LOGIC EXAMPLE            **
;**                                                                  **
;**     VERSION:        1                                            **
;**                                                                  **
;**     DATE:           APRIL 17, 1981                               **
;**                                                                  **
;**********************************************************************
;
;
;
        ASEG
        .Z80
        TITLE   MLZ-91 INITIALIZATION EXAMPLE
;       HEURIKON CORPORATION
;
;
;       WRITTEN BY JEFFREY MATTOX
;**************************************************************************
;THESE TABLES AND ROUTINES ARE PROVIDED FOR INSTRUCTIONAL PURPOSES ONLY.
;THEY MAY NOT BE APPROPRIATE FOR ALL APPLICATIONS.  HOWEVER, FEEL FREE
;TO EDIT SECTIONS OF THE FOLLOWING CODE AS NECESSARY FOR YOUR PARTICULAR
;NEEDS.  THIS SOURCE FILE ASSEMBLES ON MICROSOFT'S MACRO-80 Z80/8080
;ASSEMBLER.
;**************************************************************************
```

```
;**************************************************************************
;MEMORY ADDRESS CONSTANTS:
ROMBASE EQU     0F000H          ;BASE OF ROM AFTER MEMORY MAP ACTIVATION
RAMBASE EQU     0E000H          ;BASE OF A 4K RAM BLOCK
CLOCK:  DS      2               ;FOR INT SERVICE EXAMPLE, RT CLOCK
STACK   EQU     RAMBASE+4096    ;STACK AT END OF RAM
IOMAP   EQU     ROMBASE+10H     ;I/O DEVICE MAP (VIA WRITE TO ROM)
BUSMAP  EQU     ROMBASE+20H     ;BUS MAP (ALSO VIA WRITE TO ROM)
;**************************************************************************
;I/O DEVICE CONSTANTS:
IOA     EQU     080H            ;BASE OF I/O DEVICE GROUP "A"
IOB     EQU     0C0H            ;BASE OF I/O DEVICE GROUP "B"
;NOTE:  THE DEVICE GROUP BASE ADDRESSES ARE DETERMINED BY THE I/O MAPPING
;RAM.  THE BASE ADDRESSES MAY BE SET AT 00H, 40H, 80H OR C0H.
IOXXXA  EQU     07H             ;MAP DATA FOR I/O GROUP A (IOA)
IOXXXB  EQU     0BH             ;MAP DATA FOR I/O GROUP B (IOB)
IOXXXO  EQU     0EH             ;MAP DATA FOR OFF-CARD DEVICES
IOXXXD  EQU     0FH             ;MAP DATA FOR NO DEVICE ASSIGNMENT
;********************************
;IOA DEVICE GROUP:
IOBDA   EQU     IOA+00H         ;LOAD BAUD DATA FOR SIO PORT A (D7-D4)
IOBDB   EQU     IOA+08H         ;LOAD BAUD DATA FOR SIO PORT B (D3-D0)
;
IODMA   EQU     IOA+10H         ;DMA CONTROL AND STATUS
IOFSEL  EQU     IOA+18H         ;FDIO DRIVE SELECT AND USER LED
MEMMAP  EQU     IOA+20H         ;MEMORY MAPPING RAM
;
IOPOP   EQU     IOA+28H         ;APU POP DATA
IOAPUR  EQU     IOA+29H         ;APU READ STATUS
IOPUSH  EQU     IOA+30H         ;APU PUSH DATA
IOAPUW  EQU     IOA+31H         ;APU ENTER COMMAND
;
IODIP0  EQU     IOA+38H         ;READ DIP SWITCH GROUP 0 (1-8)
IODIP1  EQU     IOA+39H         ;READ DIP SWITCH GROUP 1 (9-16)
;
IOWCLR  EQU     IOA+3AH         ;CLEAR WINCHESTER MSEL FF
;
IOCNT0  EQU     IOA+3EH         ;CTC CHANNEL 0 COUNT/TRIGGER
;**************************************************************************
```

```
;***********************************************************************
;IOB DEVICE GROUP:
IOSAD   EQU     IOB+00H         ;SIO PORT A DATA
IOSBD   EQU     IOB+01H         ;SIO PORT B DATA
IOSAC   EQU     IOB+02H         ;SIO PORT A CONTROL/STATUS
IOSBC   EQU     IOB+03H         ;SIO PORT B CONTROL/STATUS
;
IOTRDC  EQU     IOB+08H         ;STREAMER TAPE READ DATA & CLR XFER
IOTRDS  EQU     IOB+09H         ;STREAMER TAPE READ DATA & SET XFER
IOTWRC  EQU     IOB+0AH         ;STREAMER TAPE WRITE DATA & CLR XFER
IOTWRS  EQU     IOB+0BH         ;STREAMER TAPE WRITE DATA & SET XFER
IOTRDY  EQU     IOB+0CH         ;STREAMER TAPE SET READY (TRDY)
;
IOLED   EQU     IOB+0EH         ;LOAD LED ARRAY
;
IOFDCS  EQU     IOB+10H         ;FDIO COMMAND/STATUS REGISTER
IOFDTR  EQU     IOB+11H         ;FDIO TRACK REGISTER
IOFDSR  EQU     IOB+12H         ;FDIO SECTOR REGISTER
IOFDAT  EQU     IOB+13H         ;FDIO DATA REGISTER
;
IOCTC0  EQU     IOB+18H         ;CTC 0 DATA & CONTROL
IOCTC1  EQU     IOB+19H         ;CTC 1 DATA & CONTROL
IOCTC2  EQU     IOB+1AH         ;CTC 2 DATA & CONTROL
IOCTC3  EQU     IOB+1BH         ;CTC 3 DATA & CONTROL
;
IOCLRN  EQU     IOB+20H         ;CLEAR NMI FF (PARITY & WRITE PROTECT ERRORS)
;
IOWSEL  EQU     IOB+28H         ;WINCHESTER - SET MSEL FF
IOWWR0  EQU     IOB+2AH         ;WINCHESTER - WRITE DATA/COMMAND (C/D- LOW)
IOWWR1  EQU     IOB+2BH         ;WINCHESTER - WRITE DATA/COMMAND (C/D- HIGH)
IOWRD0  EQU     IOB+2CH         ;WINCHESTER - READ DATA (C/D- LOW)
IOWRD1  EQU     IOB+2DH         ;WINCHESTER - READ DATA/STATUS (C/D- HIGH)
IOWRDS  EQU     IOB+2EH         ;WINCHESTER - READ INTERFACE STATUS
;
IOSTAT  EQU     IOB+2FH         ;READ BOARD STATUS BITS (D7-D4)
;
IOGPIB  EQU     IOB+30H         ;GPIB (IEEE-488) - BASE OF REGISTERS
IOGPDA  EQU     IOGPIB+7        ;GPIB (IEEE-488) - DATA REGISTER
;
IOPAD   EQU     IOB+38H         ;PIO A DATA - SYSTEM INT/BUS/DMA RDY
IOPBD   EQU     IOB+39H         ;PIO B DATA - MULTIBUS INTERRUPTS
IOPAC   EQU     IOB+3AH         ;PIO A CNTRL, SET BIT MODE (CFH) AND E0H MASK
IOPBC   EQU     IOB+3BH         ;PIO B CNTRL, SET BIT MODE (CFH) AND MASK AS REQ
;***********************************************************************
```

```
;**************************************************************************
;OTHER CONSTANTS OF INTEREST:
;BAUD RATE CONSTANTS (D7-D4 ARE FOR PORT A, D3-D0 ARE FOR PORT B)
B50     EQU     000H    ;50 BAUD
B75     EQU     011H    ;75 BAUD
B110    EQU     022H    ;110 BAUD
B134    EQU     033H    ;134.5 BAUD
B150    EQU     044H    ;150 BAUD
B300    EQU     055H    ;300 BAUD
B600    EQU     066H    ;600 BAUD
B1200   EQU     077H    ;1200 BAUD
B1800   EQU     088H    ;1800 BAUD
B2000   EQU     099H    ;2000 BAUD
B2400   EQU     0AAH    ;2400 BAUD
B3600   EQU     0BBH    ;3600 BAUD
B4800   EQU     0CCH    ;4800 BAUD
B7200   EQU     0DDH    ;7200 BAUD
B9600   EQU     0EEH    ;9600 BAUD
B19200  EQU     0FFH    ;19200 BAUD
;**************************************************************************
;THESE CONSTANTS ARE FOR CTC TIMER OPERATION WITH A SYSTEM CLOCK
;OF 4 MHZ.  (THE CTC DIVIDES THE CLOCK BY 256.)
C10MSEC EQU     156     ;APPROX 10 MSEC TIME BASE (156.25 = EXACT)
C16MSEC EQU     250     ;16.0 MSEC TIME BASE
C1MSEC  EQU     15      ;APPROX 1 MSEC TIME BASE (15.625 = EXACT)
C8MSEC  EQU     125     ;8.0 MSEC TIME BASE
;**************************************************************************
```

```
;****************************************************************************
;THE FOLLOWING SEQUENCE IS USED TO INITIALIZE THE MLZ-91 MAPPING RAMS:
;       1.      LOAD THE I/O DEVICE MAP TO ASSIGN I/O DEVICES
;       2.      LOAD THE BUS MAPPING RAM WITH DISABLE CODES TO
;               PREVENT ANY ACCESSES FROM THE BUS UNTIL WE ARE
;               100% READY.
;       3.      ALLOCATE THE ROM VIA THE MEMORY MAPPING RAM
;       4.      ALLOCATE ON-CARD RAM.
;       5.      CHANGE THE BUS MAPPING RAM TO ASSIGN THE MLZ-91 A
;               SPOT ON THE MULTIBUS.
;NOTE THAT THERE ARE THREE SEPARATE MAPPING RAMS:
;       1.  I/O
;       2.  BUS
;       3.  MEMORY
;****************************************************************************
;AT POWER ON ROM EXISTS EVERYWHERE THROUGHOUT THE MEMORY SPACE IN
;4K MIRRORS.  NO RAM OR I/O DEVICES ARE ALLOCATED AND THE MULTIBUS
;INTERFACE LOGIC IS INACTIVE.  THE FIRST STEP IN INITIALIZING THE
;MLZ-91 IS TO SETUP THE I/O MAPPING RAM WHICH WILL ALLOCATE (AND
;THUS ACTIVATE) THE I/O DEVICES.
        ORG     0000H            ;CPU PC AT POWER ON RESET
;THE ROM IS MIRRORED UNTIL WE SET THE MEMORY MAP RAM, LATER ON.
;
        ORG     ROMBASE          ;SET ASSEMBLER PC TO ROM BLOCK
PWRON:  JP      INIT91           ;EXECUTE PWR-ON-JP & SKIP RST, NMI LOCS
;
        ORG     ROMBASE+0100H    ;WE HAVE LEFT SPACE FOR RST & NMI LOCS
INIT91: LD      L,IOXXXO         ;OFF-CARD DEVICE DATA FOR I/O BASE 00H
        LD      H,IOXXXO         ;OFF-CARD DEVICE DATA FOR I/O BASE 40H
        LD      (IOMAP),HL       ;WRITE TO 'ROM' LOADS I/O MAP RAM
;                                ;(THIS LOADS TWO CONSECUTIVE MAP LOACTIONS.)
;
        LD      L,IOXXXA         ;ON-CARD DEVICE DATA FOR I/O BASE 80H (IOA)
        LD      H,IOXXXB         ;ON-CARD DEVICE DATA FOR I/O BASE C0H (IOB)
        LD      (IOMAP+2),HL     ;SET REMAINING TWO LOCATIONS IN I/O MAP
;THE I/O DEVICES ARE NOW ALLOCATED.
;****************************************************************************
;EVENTUALLY, THE BUS MAPPING RAM MUST BE SET TO IDENTIFY WHERE THE MLZ-91
;RESIDES ON THE MULTIBUS AND TO SPECIFY ANY INHIBITED BUS-TO-BOARD
;OPERATIONS.  HERE WE LOAD THE BUS MAP WITH ALL ZEROS WHICH INHIBITS
;ALL ACCESSES FROM THE MULTIBUS.  THIS STEP MUST PRECEDE THE
;INITIALIZATI0N OF THE MEMORY MAPPING RAM BECAUSE THE FISRT WRITE TO
;THE MEMORY MAP WILL ACTIVATE THE BUS MAP AS WELL.  THE BUS MAP WILL
;CONTAIN INDETERMINATE DATA UNLESS IT HAS BEEN FORMALLY INITIALIZED.
;LATER, THE BUS MAP CAN BE MODIFIED TO ENABLE SPECIFIC TYPES OF
;ACCESSES.  THE REASON FOR DELAYING THE ENABLE SEQUENCE IS SO THAT WE
;CAN BE SURE THAT THE MEMORY MAPPING RAM IS FULLY INITIALIZED PRIOR TO
;ENABLING ANY OFF-CARD REQUESTS FOR EITHER MLZ-91 I/O DEVICES OR MEMORY.
        LD      HL,BUSMAP        ;DESTINATION (BUS MAPPING RAM)
        LD      B,16             ;TABLE LENGTH
LOOP:   LD      (HL),00H         ;DISABLE ALL BUS-TO-BOARD OPERATIONS
        INC     HL               ;TO NEXT MAP LOCATION
        DJNZ    LOOP             ;CONTINUE
;****************************************************************************
```

```
;***************************************************************************
;NOW IT IS SAFE TO INITIALIZE THE MEMORY MAPPING RAM.
;THE FIRST LOCATION IN THE MEMORY MAP TO BE SET MUST BE FOR THE ROM
;SOCKET OUT OF WHICH WE ARE EXECUTING (OTHERWISE WE WILL BE SENT TO
;NEVER-NEVER LAND.)  THIS CODE SETS SOCKET MO TO THE BASE ADDRESS
;OF THE (THIS) ROM.
        IN      A,(IOCLRN)      ;BE SURE NMI LOGIC IS RESET BEFORE MAP ON
        LD      C,MEMMAP        ;MEMORY MAPPING RAM PORT ADRS
        LD      B,HIGH ROMBASE  ;MAP RAM BLOCK ADRS (ROM BLOCK)
        LD      A,00H           ;DATA WHICH SPECIFIES SOCKET MO (FOR 2732)
        OUT     (C),A           ;PUT SOCKET MO AT ROMBASE (F000H)
;SOCKET MO NOW EXISTS AND THE MEMORY MAPPING RAM (AS WELL AS THE BUS
;MAPPING RAM) ARE ACTIVATED.  HOWEVER, NO RAM EXISTS YET SO THE NEXT
;STEP SHOULD BE TO FINISH LOADING THE MEMORY MAP.
;***************************************************************************
;THE FOLLOWING CODE ALLOCATES 15 BLOCKS (EACH 4K IN LENGTH) OF ON-CARD
;RAM STARTING AT LOCATION 0000H.  THE 16TH BLOCK (F000H) HAS ALREADY BEEN
;ASSIGNED BY THE PREVIOUS CODE.  WE COULD HAVE PUT THE NECESSARY BYTES
;TO ASSIGN THE ROM SOCKET AT THE BEGINNING OF THE FOLLOWING
;DATA TABLE INSTEAD OF USING A SPECIFIC SET OF INSTRUCTIONS AS WAS
;DONE ABOVE.  (THIS WOULD ONLY WORK FOR ROMBASE = 0F000H BECAUSE REGISTER
;B IN THE MAP ROUTINE CAN ROLLOVER TO BLOCK 00 FOR THE RAM ALLOCATION.)
;NOTE: WRITE PROTECT IS TURNED OFF.
        LD      HL,TABLE        ;DATA FOR MEMORY MAP ROUTINE
        LD      DE,EXIT         ;RETURN ADDRESS (NO STACK EXISTS YET)
        JP      MAP             ;PROCESS THE DATA TABLE
;
TABLE:  DB      00H,7EH         ;*4K OF ON-CARD RAM AT 0000H
        DB      7DH             ;*4K AT 1000H
        DB      7CH             ;*2000H
        DB      7BH             ;3000H
        DB      7AH             ;4000H
        DB      79H             ;5000H
        DB      78H             ;6000H
        DB      77H             ;7000H
        DB      76H             ;8000H
        DB      75H             ;9000H
        DB      74H             ;A000H
        DB      73H             ;B000H
        DB      72H             ;C000H
        DB      71H             ;D000H
        DB      7FH             ;*E000H
        DB      0FH             ;END OF TABLE (MARKER FOR 'MAP' ROUTINE)
EXIT:
;PHYSICAL BLOCK F000 (MAP DATA 70H) WILL BE USED IN THE 'SLAVE I/O' EXAMPLE.
;RAM IS ALLOCATED SO THAT IF TOTAL RAM IS 16K OR 32K (NOT 64K), BLOCKS
;E000, 0000, 1000 & 2000 ALWAYS GET RAM. (SEE '*' LINES IN TABLE)
;
;NOW, I/O DEVICES AND BOTH ROM AND RAM MEMORY HAVE BEEN ALLOCATED.
;***************************************************************************
```

```
;***************************************************************************
;THE CODE BELOW ASSIGNS THE MLZ-91 BOARD TO MULTIBUS LOCATION 0XXXXH.
;ALL I/O DEVICES MAY BE USED FROM THE BUS AND THERE IS NO RESTRICTION
;PLACED ON THE USE OF MEMORY (I.E. NO I/O, MEMORY RD OR MEMORY WR
;INHIBITS ARE ON.)
;FOR EXAMPLE PURPOSES, THIS CODE ALSO ASSIGNS THE MLZ-91 BOARD
;TO MULTIBUS BLOCK 3XXXXH BUT INHIBITS USE OF ON-CARD I/O DEVICES AND
;DOES NOT ALLOW WRITES TO MEMORY.
SETBUS: LD      HL,TABL         ;ADRS OF DATA TABLE FOR BUS MAPPING RAM
        LD      DE,BUSMAP       ;DESTINATION
        LD      BC,16           ;LENGTH
        LDIR                    ;TRANSFER TABLE TO BUS MAPPING RAM
        JP      IOINIT          ;NEXT, INITIALIZE THE ON-CARD I/O DEVICES
;
TABL:   DB      00H             ;BUS BLOCK 15 (INHIBIT ACCESS FROM BUS)
        DB      00H             ;BUS BLOCK 14
        DB      00H             ;BLOCK 13
        DB      00H             ;12
        DB      00H             ;11
        DB      00H             ;10
        DB      00H             ;9
        DB      00H             ;8
        DB      00H             ;7
        DB      00H             ;6
        DB      00H             ;5
        DB      00H             ;4
        DB      90H             ;BLOCK 3 (ALLOW MEMORY READ ONLY)
        DB      00H             ;2 (INHIBIT ALL ACCESS FROM BUS)
        DB      00H             ;1 (INHIBIT ALL)
        DB      0F0H            ;BLOCK 0 (NO INHIBITS)
;...END OF MLZ-91 POWER-ON INITIALIZATION EXAMPLE.
;THERE IS AN EXAMPLE LATER ('SLAVE I/O') OF USING THE DIP SWITCHES TO
;SET THE BUS BLOCK TO WHICH THE BOARD IS ASSIGNED.
;***************************************************************************
;THERE IS AN EASIER WAY TO CREATE ALL OF THE ABOVE CODE BY USING MACROS.
;THE FOLLOWING SEQUENCE OF MACRO CALLS WILL PERFORM THE SAME TASKS AS
;DETAILED ABOVE:  (SEE PAGE 98 FOR MACRO DEFINITIONS)
;       ORG     0               ;CPU PC AT PWR ON
;       ORG     ROMBASE         ;SET ASSEMBLER PC TO ROM BLOCK
;PWRON: JP      INIT91          ;PWR-ON-JMP AND SKIP RST & NMI LOCS
;
;       ORG     ROMBASE+0100H   ;AFTER NMI LOCATION
;INIT91:MAPIO   ROMBASE,0,0,1,2 ;MAP DATA (OFF-CARD,OFF-CARD,IOA,IOB)
;       MAPBUS  ROMBASE         ;CLEAR BUS MAPPING RAM
;       IN      A,(IOCLRN)      ;RESET NMI LOGIC
;       MAPROM  ROMBASE,4       ;MAP SOCKET M0 AT ROMBASE (4K)
;       MAPRAM  0000H,000H,60   ;MAP IN 60K OF ON-CARD RAM AT 0000H
;       MAPBUS  ROMBASE,00H,1,1,1       ;BUS BLOCK 0, ALLOW I/O, RD, WR
;       MAPBUS  ROMBASE,03H,0,1,0       ;BUS BLOCK 3, ALLOW MEMORY RD ONLY
;DONE...
;***************************************************************************
```

```
;************************************************************************
;GENERAL COMMENTS CONCERNING THE INTERACTION OF THE MAPPING RAMS:
;       1.  IF ROM IS NOT ASSIGNED (OR IS TURNED OFF) YOU CANNOT
;           REASSIGN I/O DEVICE ADDRESS GROUPS UNTIL THE ROM IS
;           REALLOCATED.  (BECAUSE A WRITE TO 'ROM' IS REQUIRED TO
;           ACCESS THE I/O MAPPING RAM.)  "ROM" SPACE IS ALSO REQUIRED
;           TO ACCESS THE BUS MAPPING RAM.
;       2.  IF I/O DEVICE GROUP A IS UNALLOCATED YOU CANNOT ALTER
;           THE MEMORY MAPPING RAM UNTIL I/O GROUP A IS REALLOCATED.
;           (BECAUSE I/O PORT 'MEMMAP' IN I/O GROUP A IS USED TO
;           ACCESS THE MEMORY MAPPING RAM.)
;       3.  THIS MEANS THAT THE I/O MAP AND THE MEMORY MAP RAMS ARE
;           'INTERLOCKED' IN THAT:
;               A.  EACH IS USED IN THE ACCESSING OF THE OTHER.
;               B.  IT IS IMPOSSIBLE TO DEALLOCATE BOTH I/O GROUP A
;                   AND ROM, WHICH IS GOOD BECAUSE OTHERWISE BOTH
;                   WOULD BE 'LOST' FOREVER.
;************************************************************************
;THIS IS THE ACTUAL MAP ROUTINE WHICH PROCESSES THE TABLE
;DATA GENERATED BY THE MAPRAM MACRO.  SINCE THE ACT OF
;CHANGING THE MAPPING RAM CONTENTS MAY AFFECT THE LOCATION
;OF THE HARDWARE STACK, THIS ROUTINE CANNOT USE THE STACK
;AND THUS CANNOT BE CALLED AS A SUBROUTINE.  THEREFORE, THE
;RETURN ADDRESS IS PASSED IN REGISTER PAIR DE.
;
MAP:    LD      C,MEMMAP        ;MAPPING RAM PORT ADDRESS
        LD      B,(HL)          ;FETCH B REG ENTRY (MAP BLK)
;
M1:     INC     HL              ;ADVANCE POINTER
        LD      A,(HL)          ;FETCH NEXT REG A VALUE
        CP      0FH             ;TEST FOR END OF TABLE
        JR      Z,MDONE         ;END OF TABLE
        OUT     (C),A           ;SEND TO MAPPING RAM
        LD      A,B             ;PREVIOUS BLOCK
        ADD     A,10H           ;COMPUTE NEXT BLOCK
        LD      B,A             ;FOR NEXT LOOP
        JR      M1              ;PROCESS NEXT ENTRY
;
MDONE:  EX      DE,HL           ;GET RETURN ADRS TO HL
        JP      (HL)            ;RETURN
;************************************************************************
```

```
;*************************************************************************
;HERE WE INITIALIZE THE I/O DEVICES AND SET THE INTERRUPT LOGIC:
IOINIT: LD      SP,STACK        ;INIT STACK POINTER, NOW THAT RAM IS ON
;
        LD      HL,IODATA       ;DATA TABLE FOR IOSUBR
        CALL    IOSUBR          ;INIT ON-CARD I/O DEVICES (VIA TABLE)
;
        LD      A,HIGH VSIO     ;INIT HIGH PART OF INTERRUPT VECTOR
        LD      I,A             ;SET I REGISTER
        IM      2               ;SET INTERRUPT MODE (VECTORED)
;WHEN THE PROGRAM IS READY FOR INTERRUPTS, IT CAN ENABLE EACH DEVICE
;AND DO AN 'EI' INSTRUCTION.  FOR THIS EXAMPLE, IF AN 'EI' IS DONE NOW,
;CTC CHANNEL 1 WILL INTERRUPT EVERY 1.0 SECOND.
;WE ARE NOW DONE WITH THE MLZ-91 MAP AND DEVICE INITIALIZATION EXAMPLES.
        EI                      ;START TIMER (& RT CLOCK)
        JP      SLAVE           ;GO TO REST OF THE PGM...
;*************************************************************************
;THE FOLLOWING ROUTINE AND TABLE MAY BE USED TO INITIALIZE THE ACTUAL
;I/O DEVICES ON THE MLZ-91.  THE I/O SUBROUTINE (IOSUBR) PROCESSES
;A TABLE OF DEVICE PORT ADDRESSES AND INITIALIZATION VALUES.  HERE,
;WE ARE SETTING THE SIO, CTC, PIO, DMA, BAUD RATES AND OTHER MISCELLANEOUS
;PORTS ON THE BOARD.
IOSUBR: LD      C,(HL)          ;GET PORT ADRS
        LD      A,C             ;MOVE FOR TEST
        INC     A               ;TEST FOR OFFH
        RET     Z               ;END OF TABLE
;
        INC     HL              ;TO LENGTH PARAMETER
        LD      B,(HL)          ;LENGTH OF THIS ENTRY
        INC     HL              ;TO FIRST BYTE FOR DEVICE
        OTIR                    ;TRANSFER TABLE DATA TO DEVICE
        JR      IOSUBR          ;CONTINUE UNTIL DEVICE ADRS = OFFH
;*************************************************************************
;HERE IS THE DATA TABLE FOR THE IOSUBR: (CHANGE FOR YOUR APPLICATION)
IODATA: DB      IOSAC           ;SIO PORT A DEVICE ADRS
        DB      9               ;LENGTH
        DB      0               ;(RESET SIO REG COUNTER)
        DB      4,04CH          ;CMD, ASYNC, CLK, STOPS
        DB      5,0EAH          ;CMD 5: RTS, ENBL, 8BITS, DTR
        DB      3,0E1H          ;CMD 3: RTS, ENBL, ENBL, 8BITS
        DB      1,000H          ;CMD 1: NO INT, UPDATE STATUS
;
        DB      IOSBC           ;SIO PORT B ADRS
        DB      11              ;LENGTH
        DB      0               ;(RESET SIO REG COUNTER)
        DB      4,04CH          ;(SEE SIO A INIT FOR DETAILS)
        DB      5,0EAH
        DB      3,0E1H
        DB      1,000H
        DB      2,LOW VSIO      ;INTERRUPT VECTOR (FOR BOTH PORTS)
;*************************************************************************
```

```
;**************************************************************************
        DB      IOBDA           ;DEVICE ADRS (BAUD GEN A)
        DB      1
        DB      B9600           ;BAUD RATE DATA
;
        DB      IOBDB           ;(BAUD GEN B)
        DB      1
        DB      B9600           ;BAUD RATE DATA
;
        DB      IOPAC           ;PIO PORT A DEVICE ADRS
        DB      3               ;ENTRY LENGTH
        DB      0CFH,0E0H       ;BIT MODE, 3 INS & 5 OUTS
        DB      LOW VPIO        ;PIO MODE 2 INTERRUPT VECTOR
;
        DB      IOPAD           ;PIO A DATA
        DB      1
        DB      08H             ;RELEASE BUS IF ANY OTHER BOARD REQUEST
;
        DB      IOPBC           ;PIO PORT B DEVICE ADRS
        DB      3               ;LENGTH
        DB      0CFH,0FFH       ;BIT MODE, ALL INS (THIS DEPENDS ON APPL.)
        DB      LOW VPIO        ;INTERRUPT VECTOR
;
        DB      IOCTC0          ;DEVICE ADRS (CTC 0)
        DB      3               ;LENGTH
        DB      025H            ;CONTROL (DSBL INT, TIMER, /256, TC FOLL.)
        DB      C8MSEC          ;TIME CONSTANT (8.0 MSEC AT 4 MHZ.)
        DB      LOW VCTC        ;INT VECTOR FOR ALL FOUR CHANNELS
;
        DB      IOCTC1          ;DEVICE ADRS (CTC 1)
        DB      2               ;LENGTH
        DB      0C5H            ;CONTROL (ENBL INT, COUNTER, TC FOLLOWS)
        DB      125             ;TIME CONSTANT (8 MSEC/125 = 1.0 SEC)
;
        DB      IOCTC2          ;(CTC 2)
        DB      2               ;LENGTH
        DB      0,0             ;CONTROL AND TIME CONSTANT
;
        DB      IOCTC3          ;(CTC 3)
        DB      2               ;LENGTH
        DB      0,0             ;CONTROL AND TIME CONSTANT
;
        DB      IOFSEL          ;DEVICE ADRS (FDIO DRIVE SELECT, ETC.)
        DB      1
        DB      0FFH            ;ALL OFF
;
        DB      IOLED           ;(LED ARRAY)
        DB      1
        DB      0FFH            ;ALL OFF
;
        DB      0FFH            ;END OF TABLE
;**************************************************************************
```

```
;**************************************************************************
;INTERRUPT (MODE 2) VECTORS FOR I/O DEVICES:
        ORG     $+((-$) AND 7)  ;FORCE CORRECT BYTE BOUNDRY
VSIO:   DW      DUMMY,DUMMY,DUMMY,DUMMY ;SIO PORT A VECTORS
        DW      DUMMY,DUMMY,DUMMY,DUMMY ;SIO PORT B VECTORS
VCTC:   DW      DUMMY,RTCLK,DUMMY,DUMMY ;CTC VECTORS (NOTE 'RTCLK' VECTOR)
VDMA:   DW      DUMMY,DUMMY,DUMMY,DUMMY ;DMA VECTORS
VPIO:   DW      DUMMY,DUMMY             ;PIO VECTORS
;**************************************************************************
;THIS IS AN EXAMPLE OF AN INTERRUPT SERVICE ROUTINE.  HERE WE
;SIMPLY COUNT THE NUMBER OF INTERRUPTS FROM CTC CHANNEL 1 (WHICH
;OCCUR ONCE EVERY SECOND) AND DISPLAY THE LOW 8 BITS OF THE COUNT
;IN THE MLZ-91 LED ARRAY.
RTCLK:  PUSH    HL              ;SAVE USER REGISTERS
        PUSH    PSW
        LD      HL,(CLOCK)      ;GET PREVIOUS COUNTER
        INC     HL              ;ADD (1 SECOND)
        LD      (CLOCK),HL      ;RESTORE
        LD      A,L             ;GET LSB'S
        OUT     (IOLED),A       ;DISPLAY
        POP     PSW             ;RESTORE USER REGS
        POP     HL
;
DUMMY:  EI                      ;RE-ENABLE INTERRUPTS
        RETI                    ;RETURN
;
PC      EQU     $               ;SAVE CURRENT PC VALUE
;**************************************************************************
;
;       EXAMPLE OF OFF-CARD MEMORY ALLOCATION:
;16K OF OFF-CARD MEMORY. PHYSICAL OFF-CARD ADRS OF 14000H.  ALLOCATED
;AT CPU ADRS 8000H.
;       LD      HL,TAB          ;TABLE ADRS
;       LD      DE,EXIT         ;RETURN
;       JP      MAP             ;PROCESS TABLE DATA
;
;TAB:   DB      80H             ;STARTING BLOCK NUMBER
;       DB      0F5H            ;FIRST 4K BLOCK
;       DB      0F5H            ;SECOND 4K BLOCK
;       DB      0F4H
;       DB      0F4H
;       DB      0FH             ;END OF TABLE
;EXIT:
;OR:    MAPMEM  8000H,140H,16   ;WILL DO THE SAME THING
;**************************************************************************
```

```
;**************************************************************************
;WHEN AN NMI OCCURS (AS A RESULT OF A RAM PARITY ERROR OR A WRITE
;PROTECT ERROR), THE PC IS SAVED IN RAM (ASSUMING A STACK HAS BEEN
;ASSIGNED) AND CONTROL IS TRANSFERRED TO LOCATION 0066H IN SOCKET M0.
;THE MEMORY MAPPING RAM IS DISABLED (ALTHOUGH THE MAP RAM CONTENTS ARE
;NOT ALTERED.)  THIS PUTS SOCKET M0 ALL OVER THE MEMORY SPACE (IN 4K
;MIRRORS) AND DEALLOCATES RAM.  IF THE MEMORY MAP IS RE-ACTIVATED (BY
;SETTING ANY MAP LOCATION) THE STACK MAY BE POPPED TO FIND THE
;APPROXIMATE ERROR ADDRESS.  TRIVIAL CASES OF NMI INTERRUPT
;ERROR PROCESSING WOULD BE A 'RETN' INSTRUCTION (WHICH WOULD
;CAUSE THE ERROR TO BE IGNORED) OR A 'HALT' INSTRUCTION (WHICH WOULD
;CAUSE THE SYSTEM TO STOP.)  THIS CODE ASSUMES IOB = 0C0H.
        ORG     ROMBASE+0066H   ;NMI LOCATION (PC WILL BE 0066H)
NMILOC: JP      NMI1            ;EXECUTE A POWER-ON-JMP
NMI1:   IN      A,(IOSTAT)      ;READ ON-CARD STATUS BITS
        LD      L,A             ;SAVE (CAN'T USE STACK, NO RAM IS ON)
;CONSIDER CODE AT THIS POINT TO DO THE FOLLOWING OPERATIONS:
;       1.  RELOAD THE I/O MAPPING RAM (IN CASE SOMEBODY CHANGED IT)
;       2.  CLEAR THE NMI LOGIC (VIA 'IN A,(IOCLRN)') WHICH WILL
;               TURN OFF BOTH ERROR LEDS (THE HUMAN EYE PROBABLY
;               WON'T EVEN SEE THEM FLASH ON) AND ALLOW THE
;               MEMORY MAPPING RAM TO BE TURNED BACK ON.
;       3.  TURN ON THE MEMORY MAP (E.G., RE-ASSIGN ROM)
;               AND RE-ALLOCATE RAM (IN CASE SOMEBODY CHANGED IT)
;SEE ZRAID-91 LISTING FOR A MORE SPECIFIC EXAMPLE OF AN NMI PROCESSING
;ROUTINE (INCLUDES PRINTING ERROR MESSAGES, ETC.)
;CAUTION: DON'T CHANGE REGISTER L (IF YOU WANT TO TEST THE NMI REASON.)
        BIT     6,L             ;TEST ERROR TYPE
        JP      Z,NMIP          ;JUMP IF PARITY ERROR
                                ;ELSE WRITE PROTECT ERROR
;       ...     ...             ;WRITE PROTECT ERROR ROUTINE GOES HERE
        JP      EREXIT          ;END OF ERROR PROC
;
NMIP:                           ;PARITY ERROR RECOVERY ROUTINE GOES HERE
;       ...     ...
;       ...     ...
;
EREXIT:                         ;JUMP SOMEWHERE...
;GENERALLY, THE ORIGINAL PROGRAM SHOULD BE RESTARTED AT THIS POINT.
;(CAUTION: DON'T GO PAST ROMBASE+00FFH WITH THE ABOVE CODE.  THE
;ORG AT 'INIT91' IS THERE.)
;**************************************************************************
;FOR A MORE DETAILED EXAMPLE OF NMI ERROR PROCESSING (INCLUDING ERROR
;ADDRESS CAPTURE,) REFER TO ZRAID SOURCE CODE.
;**************************************************************************
```

```
;***************************************************************************
        ORG     PC                      ;GET BACK TO WHERE WE LEFT OFF BEFORE NMI.
;THE FOLLOWING CODE SEGMENTS SHOW HOW A SLAVE I/O AND MEMORY BOARD COULD
;BE IMPLEMENTED USING A MLZ-91.
;***************************************************************************
SLAVE:  LD      L,IOXXXA        ;ON-CARD DEVICE GROUP A
        LD      H,IOXXXB        ;ON-CARD DEVICE GROUP B
        LD      (IOMAP),HL      ;WRITE TO 'ROM' LOADS I/O MAP RAM
;THE I/O DEVICES HAVE NOW ALLOCATED IN THE LOWER TWO I/O BLOCKS AS WELL
;AS THE UPPER TWO.  THE LOWER TWO BLOCKS WOULD BE USED FROM THE BUS.
;
;THE CODE BELOW ASSIGNS THE MLZ-91 BOARD TO THE MULTIBUS LOCATION
;SPECIFIED BY THE SETTINGS OF DIP SWITCHES 5,6,7 & 8 (DIP SW 0)
;ALL ON-CARD I/O DEVICES MAY BE ACCESSED FROM THE BUS AND THERE IS NO
;RESTRICTION PLACED ON THE USE OF ON-CARD MEMORY (NO INHIBITS).
        IN      A,(IODIP0)      ;READ DIP SWITCH GROUP 0 (INVERTED)
        AND     0FH             ;REMOVE UPPER (GARBAGE) BITS
        ADD     A,LOW BUSMAP    ;COMPUTE LOW ADRS HALF
        LD      L,A             ;MOVE TO L
        LD      H,HIGH BUSMAP   ;FORM COMPLETE ADRS (OF BUS MAP RAM)
        LD      (HL),0F0H       ;ENABLE BUS OPERATIONS IN SELECTED BLOCK
;BE SURE TO DELETE THE CODE AT 'SETBUS' (EARLIER) IF THIS LOGIC IS USED.
;***************************************************************************
;AS LONG AS WE ARE FOOLING AROUND WITH THE DIP SWITCHES, LET'S
;READ IN DIP SWITCH GROUP 1 AND USE THAT VALUE TO SET THE SIO BAUD RATES.
;       DIP SW  SIO PORT
;       ------  --------
;       1,2,3,4    A
;       5,6,7,8    B
;
;THE SWITCH SETTING SPECIFIES WHICH BAUD RATE (FROM 75 TO 19200) SHOULD
;BE GENERATED.  BAUD RATES ARE SET INDEPENDENTLY FOR EACH PORT AS FOLLOWS:
;       SETTING   BAUD                          SETTING   BAUD
;       1 = 'ON'  RATE                          1 = 'ON'  RATE
;       --------  --------                      --------  --------
;        0000    DEFAULT (9600)                  1000      1800
;        0001      75                            1001      2000
;        0010     110                            1010      2400
;        0011     134.5                          1011      3600
;        0100     150                            1100      4800
;        0101     300                            1101      7200
;        0110     600                            1110      9600
;        0111    1200                            1111     19200
;
        IN      A,(IODIP1)      ;READ DIP SWITCHES (GROUP 1)
        CPL                     ;FIX 'ON' = 1
        OR      A               ;TEST FOR SWITCHES 'OFF' (NOT THERE)
        JR      Z,SLAVE1                ;USE DEFAULT BAUD RATES (9600)
        OUT     (IOBDA),A       ;SET UPPER FOUR BITS (SIO PORT A)
        OUT     (IOBDB),A       ;SET LOWER FOUR BITS (SIO PORT B)
        JR      SLAVE1          ;GO TO SLAVE BOARD MAINLINE CODE
;***************************************************************************
```

```
;*************************************************************************
;THIS IS THE SLAVE BOARD MAINLINE LOGIC.  AT THIS POINT ALL OF THE
;I/O DEVICES HAVE BEEN INITIALIZED AND THE BOARD EXISTS ON THE BUS.
;NEXT, WE DO THE FOLLOWING:
;       1.  LOAD THE END OF A RAM BLOCK WITH A SET OF INSTRUCTIONS
;           WHICH WE WILL LATER EXECUTE.  THESE INSTRUCTIONS WILL
;           BE A HALT INSTRUCTION FOLLOWED BY A JUMP TO THE HALT.
;       2.  CHANGE THE MEMORY MAP TO DEALLOCATE THE ROM,
;           ALLOCATE 64K OF RAM AND RESUME EXECUTION AT THE HALT
;           INSTRUCTION IN RAM.
;THIS SLAVE BOARD WILL THEN BECOME 'INACTIVE' UNTIL AN ON-CARD DEVICE
;INTERRUPT OCCURS (E.G., THE CTC - RTCLOCK OR AN NMI DUE TO A RAM ERROR.)
;IF OTHER DEVICE INTERRUPTS ARE ENABLED AND THE APPROPRIATE INTERRUPT
;SERVICE ROUTINES ARE ADDED, THIS BOARD WOULD BE AN INTELLIGENT I/O BOARD.
;
;BEFORE THIS CODE IS EXECUTED, THE FOLLOWING STATE EXISTS:
;       MEMORY BLOCK E000       MAP DATA = 7F   (RAM)
;       MEMORY BLOCK F000       MAP DATA = 00   (ROM)
;       MAP DATA 70H IS NOT ASSIGNED (THE LAST 4K BLOCK OF ON-CARD RAM)
;THIS CODE ASSUMES THERE IS 64K OF RAM (ELSE MAP DATA MUST BE CHANGED).
;
SLAVE1: LD      HL,ROMBASE      ;OBJECT CODE TO BE MOVED (ALL OF ROM)
        LD      DE,RAMBASE      ;DESTINATION (TEMPORARY RAM ADRS)
        LD      BC,4096         ;LENGTH (1 BLOCK)
        LDIR                    ;MOVE CODE FROM ROM TO RAM
;
        LD      C,MEMMAP        ;MEMORY MAP RAM ADRS
        LD      B,HIGH RAMBASE  ;BLOCK WITH OBJECT CODE
        LD      A,70H           ;PUT IN A BLOCK OF 'FRESH' RAM
        OUT     (C),A           ;ALLOCATE THE FRESH BLOCK
;
        LD      B,HIGH ROMBASE  ;(ROM BLOCK)
        LD      A,7FH           ;THE RAM BLOCK THAT HAS THE OBJECT CODE
        OUT     (C),A           ;MAGIC, WE ARE NOW IN RAM
;                               ;THE SYSTEM IS 100% (64K) RAM
;
        JP      HALTI           ;GO TO THE END OF RAM
;
        ORG     ROMBASE+4096-4  ;BACKUP A LITTLE FROM THE END OF MEMORY
HALTI:  HALT                    ;STOP HERE UNTIL NEXT DEVICE INT OR NMI
        JP      HALTI           ;GO BACK TO HALT AFTER INTERRUPT SERVICE
;*************************************************************************
;THIS CODE IS IN CASE THE .HEX FILE IS USED TO PROGRAM A 2716 TYPE EPROM.
;THIS IS A MIRROR OF THE OBJECT CODE AT THE END OF THE 4K BLOCK.
        ORG     ROMBASE+2048-4  ;NEAR END OF A 2716
        HALT
        JP      HALTI
;IF A 2716 TYPE EPROM IS USED, SET THE MLZ-91 BOARD JUMPERS AS FOLLOWS:
;       J12-A, J14-B
;THIS WILL CAUSE THE 2K ROM TO BE MIRRORED IN THE SECOND HALF OF THE
;4K MEMORY BLOCK.
;*************************************************************************
        END
```

MULTI-USER EXAMPLE

The ability to dynamically re-allocate memory allows a number of special programming functions to be implemented without great difficulty.

One such use is in creating a multi-task environment. In its simplest form, the entire memory, except for a small executive routine, can be switched between tasks. For example, two independent programs, both of which execute at the same address can be loaded into different physical memory addresses. Then, the mapping RAM can be set to point to one program or the other depending on which task is to be active. Another method where each task is executing the same program is to leave the program code permanently allocated but switch different blocks of RAM in as each task executes. The trick for either method is to determine when to switch tasks (or when not to) and to save the CPU registers before re-allocating memory.

The "when to switch" decision could be based on a number of conditions. For example, whenever a task is in an idle loop or waiting for a hardware I/O device would be an appropriate opportunity to switch to a different task. Also, a timer interrupt at, say 10 millisecond intervals, could trigger a switch. A "when not to switch" condition could be during a DMA data transfer which is using the current task's RAM.

In order to show the ease of implementing a multi-tasking system using the memory mapping RAM, we converted the ZRAID monitor program from a single user system to a multi-user system.

Single user ZRAID configuration:

4K ROM in socket MØ (at FØØØH)
4K RAM (at EØØØH)
Console device on SIO port B

Multi-user ZRAID configuration:

ROM in socket MØ (at FØØØH)
User 1 RAM at EØØØH when allocated
User 1 console device on SIO port B
User 2 RAM, also at EØØØH when allocated
User 2 console device on SIO port A
Switch to other task when doing character I/O to/from ports A or B.

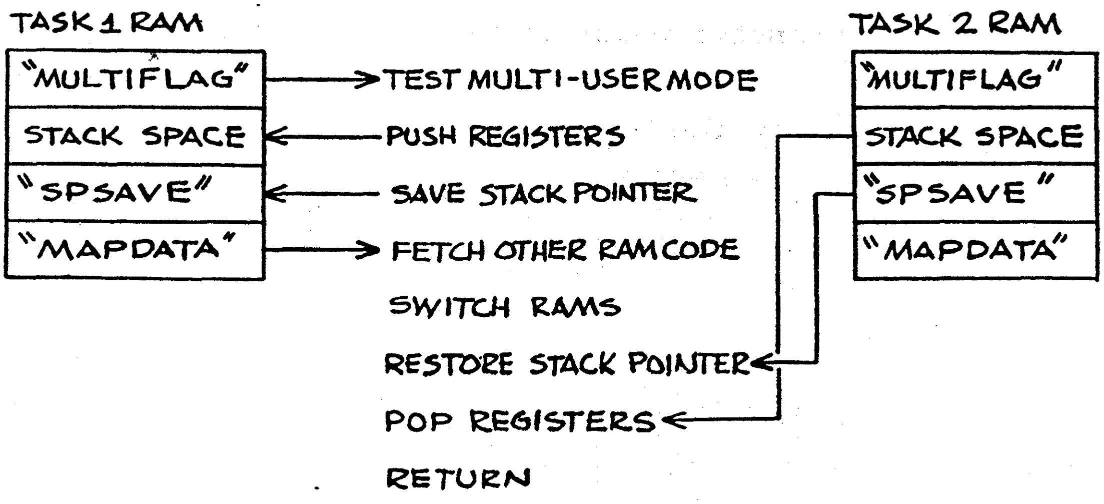
The code used to implement the multi-user ZRAID logic follows. (See next page)

In order to extend these routines to handle three or more users, these steps could be used:

1. At initialization, setup each task's RAM area with a stack, user ID code, flags, etc.

2. The SWITCH routine could maintain a list (or "queue") of tasks which are waiting for service. This task queue could contain information pertaining to the individual task's priority, RAM block code, I/O status and other special conditions.

"SWITCH" ROUTINE LOGIC
SHOWN SWITCHING FROM TASK 1 TO TASK 2



```
; *************************************************************************
;THIS IS THE MULTI-TASKING CODE.  "MULTI" IS THE "!" COMMAND WHICH TURNS
;ON THE MULTIFLAG OF BOTH TASKS AND STARTS TASK 2.  "SWITCH" IS THE ROUTINE
;WHICH ALTERNATES BETWEEN THE TWO TASKS.  SWITCH IS CALLED FROM THE CHARACTER
;I/O ROUTINES AND THE FLOPPY DISK MULTIPLE SECTOR RD/WR LOOP.
;CONSULT ZRAID91 LISTING FOR MORE DETAILS.
;
;EACH TASK HAS THE FOLLOWING VARIABLES:
;MULTIFLAG: DS  1        ;SET NON ZERO TO INDICATE TO CURRENT
;                        ;TASK THAT IT SHOULD SWITCH TO OTHER TASK
;                        ;FROM TIME TO TIME.  IF THIS FLAG IS CLEARED
;                        ;IN EITHER TASK, THE SWITCH LOGIC WILL BE
;                        ;DISABLED, EFFECTIVELY GIVING THE "CURRENT"
;                        ;TASK ABSOLUTE PRIORITY.
;
;MAPDATA:   DS  1        ;CONTAINS THE DATA FOR THE MAPPING RAM WHICH
;                        ;WILL BE USED TO SWITCH TO THE OTHER TASK.
;
;SPSAVE:    DS  1        ;USED TO SAVE THIS TASK'S SP WHILE THE OTHER
;                        ;TASK IS RUNNING.
;
MULTI:  LD      A,1             ;GET SOMETHING NON-ZERO
        LD      (MULTIFLAG),A   ;SET MULTI-MODE FLAG ON
;
        PUSH    HL              ;PUSH SOME REGS (FOR SWITCH POPS)
        PUSH    DE
        PUSH    BC
        LD      (SPSAVE),SP     ;SAVE THIS TASK'S SP

        LD      A,70H           ;MAP DATA FOR OTHER TASK (TASK 2)
        LD      (MAPDATA),A     ;SAVE FOR USE BY SWITCH
        LD      B,0E0H          ;BLOCK ADRS
        LD      C,MEMMAP        ;MAPPING RAM ADRS
        OUT     (C),A           ;SWITCH RAM'S AND SWITCH TASKS
;
        LD      A,7FH           ;ORIGINAL TASK'S RAM (TASK 1)
        LD      (MAPDATA),A     ;SET FOR LATER RETURN TO TASK 1
        LD      (MULTIFLAG),A   ;ALSO KEEP MULTI MODE ON
        LD      SP,STACK        ;INIT TASK 2 STACK
        JP      INITSA          ;INIT TASK 2 ON SIO PORT A
;
; ************************************
;THIS ROUTINE DOES THE ACTUAL SWITCHING BETWEEN TASKS:
SWITCH: LD      A,(MULTIFLAG)   ;TEST FOR MULTI TASK MODE ON
        OR      A
        RET     Z               ;OFF
;
        PUSH    HL              ;SAVE THIS TASK'S REGS
        PUSH    DE
        PUSH    BC
        LD      (SPSAVE),SP     ;AND SAVE STACK POINTER
;
        LD      A,(MAPDATA)     ;OTHER TASK'S RAM MAP DATA
        LD      C,MEMMAP        ;MAP DEVICE ADRS
        LD      B,0E0H          ;RAM BLOCK ADRS
        OUT     (C),A           ;SWITCH TASKS
;
        LD      SP,(SPSAVE)     ;GET OTHER TASK'S SP
        POP     BC              ;AND RESTORE ITS REGS
        POP     DE
        POP     HL
        RET                     ;CONTINUE OTHER TASK
; *************************************************************************
```

```
;*************************************************************************
;THE FOLLOWING MACRO DEFINITIONS ARE TO FACILITATE LOADING OF THE
;THREE MLZ-91 MAPPING RAMS.  THESE MACROS EXECUTE ON THE MICROSOFT
;MACRO-80 ASSEMBLER.
;
;       MACRO   PARAMETERS (FUNCTION)
;       ------  --------------------
;       MAPIO   ROMBASE,B00,B40,B80,BC0
;               (LOAD I/O DEVICE MAPPING RAM)
;                       B00 = CODE FOR I/O BLOCK 00H
;                       B40 = CODE FOR I/O BLOCK 40H
;                       B80 = CODE FOR I/O BLOCK 80H
;                       BC0 = CODE FOR I/O BLOCK C0H
;                               USE '0' FOR OFF-CARD
;                                   '1' FOR IOA (DEVICE GROUP A)
;                                   '2' FOR IOB (DEVICE GROUP B)
;                                   '3' FOR NOT ASSIGNED (DISABLE BLOCK)
;
;       MAPBUS  ROMBASE
;               (CLEAR BUS MAPPING RAM)
;       MAPBUS  ROMBASE,BUSBLOCK,I/OINH,RDINH,WRINH
;               (SET BUS MAPPING RAM FOR BUS BLOCK = 00 TO 0FH)
;                       I/OINH = 0 INHIBIT I/O OPERATIONS
;                       RDINH = 0 MEANS INHIBIT MEMORY RD
;                       WRINH = 0 MEANS INHIBIT MEMORY WR
;                       (ELSE, INHIBITS = 1)
;
;       MAPROM  ROMBASE,KBYTES[,SKT]
;               (ALLOCATE ROM SOCKETS)
;                       KBYTES = 4, 8 OR 16
;                       SKT (OPTIONAL) = 1 TO ALLOCATE ONLY SKT M1
;                               (ELSE, BOTH ALLOCATED IF NECESSARY)
;                       EACH SOCKET = 4K UNLESS KBYTES = 16 (THEN EACH = 8K)
;
;       MAPRAM  BASEADRS,PAGE,KBYTES[,1]
;               (ALLOCATE ON-CARD RAM)
;               BASEADRS = LOGICAL START ADRS (ON A 4K BOUNDARY)
;       PAGE = PHYSICAL STARTING PAGE OF RAM
;               KBYTES = 4, 8, 12, 16, ETC (MULTIPLE OF 4)
;               INCLUDE THE LAST PARAMETER TO ALLOCATED PROTECTED RAM
;
;       MAPMEM  BASEADRS,PAGE,KBYTES
;               (ALLOCATE OFF-CARD MEMORY)
;               SEE MAPRAM MACRO FOR PARAMETERS
;*************************************************************************
```

```
;**************************************************************************
MAPIO   MACRO   ROMB,A,B,C,D
PR0     DEFL    0EH             ;;OFF-CARD
PR1     DEFL    07H             ;;IOA
PR2     DEFL    0BH             ;;IOB
PR3     DEFL    0FH             ;;NOT ASSIGNED
;
        LD      HL,PR&B*256+PR&A;;FORM WORD FOR BLOCKS 0 & 1
        LD      (ROMB+10H),HL   ;;SEND TO DEVICE MAPPING RAM
        LD      HL,PR&D*256+PR&C;;FORM WORD FOR BLOCKS 2 & 3
        LD      (ROMB+12H),HL   ;;SEND TO DEVICE MAPPING RAM
        ENDM
;IT WOULD BE POSSIBLE FOR THIS MACRO TO ALSO REDEFINE THE I/O PORT
;ADDRESSES SO THAT I/O PORT REFERENCES FOLLOWING INVOKATIONS
;OF THIS MACRO WOULD REFERENCE THE PROPER ADDRESS.
;**************************************************************************
MAPBUS  MACRO   ROMB,BLOCK,IO,RD,WR
        LOCAL   LOOP
        IFB     <BLOCK>         ;;TEST FOR OPTIONAL FORM
        LD      HL,ROMB+20H     ;;START OF BUS MAPPING RAM
        LD      B,16            ;;LENGTH
LOOP:   LD      (HL),00H        ;;CLEAR BUS MAP RAM
        INC     HL              ;;TO NEXT MAP ENTRY
        DJNZ    LOOP            ;;CONTINUE
;
        ELSE
;
        LD      A,16*(1+RD*8+WR*4+IO*2) ;;FORM NIBBLE FOR BUS MAP RAM
        LD      (ROMB+2FH-BLOCK),A      ;;SEND TO BUS MAPPING RAM
        ENDIF
        ENDM
;**************************************************************************
```

```
;***************************************************************************
MAPROM  MACRO   BLOCK,KBYTES,SKT1
        LD      C,MEMMAP        ;;MEMORY MAP DEVICE ADRS
        LD      B,BLOCK/256     ;;DESIRED ROM BASE
;
        IFB     <SKT1>          ;;TEST OPTIONAL FORM
        LD      A,00            ;;DATA FOR SOCKET MO
        OUT     (C),A           ;;SEND TO MAPPING RAM
;
        IFF     KBYTES-8
        LD      A,20H           ;;DATA FOR SOCKET M1
        LD      B,BLOCK/256+10H ;;NEXT MAP BLOCK
        OUT     (C),A           ;;SET MAP RAM
        ENDIF
;
        IFF     KBYTES-16
        LD      A,10H           ;;DATA FOR SECOND HALF OF SKT MO
        LD      B,BLOCK/256+10H ;;NEXT MAP BLOCK
        OUT     (C),A           ;;SET MAP RAM
        LD      A,20H           ;;FIRST HALF OF SOCKET M1
        LD      B,BLOCK/256+20H ;;NEXT MAP BLOCK
        OUT     (C),A           ;;TO MAP
        LD      A,30H           ;;SECOND HALF OF M1
        LD      B,BLOCK/256+30H ;;NEXT MAP BLOCK
        OUT     (C),A           ;;TO MAP
        ENDIF
;
        ELSE
;
        LD      A,20H           ;;DATA FOR SOCKET M1
        OUT     (C),A           ;;SEND TO MAPPING RAM
;
        IFF     KBYTES-8
        LD      A,30H           ;;SECOND HALF OF M1
        LD      B,BLOCK/256+10H ;;NEXT MAP BLOCK
        OUT     (C),A
        ENDIF
        ENDIF
        ENDM
;***************************************************************************
```

```
;**************************************************************************
MAPSET  MACRO   TABLE,BLOCK,EXIT        ;;UTILITY MACRO
        LD      HL,TABLE        ;;START OF DATA TABLE
        LD      DE,EXIT         ;;RETURN POINT
        JP      MAP             ;;'CALL' MAP ROUTINE
TABLE:  DB      BLOCK/256       ;;INITIAL BLOCK ADRS IN MAP
        ENDM
;**************************************************************************
MAPRAM  MACRO   BASE,PAGE,KBYTES,PROTCT
        LOCAL   TABLE,EXIT
        MAPSET  TABLE,BASE,EXIT ;;GENERATE INITIAL CODE
;
DATA    DEFL    7FH-PAGE/16     ;;STARTING DATA VALUE
        IFNB    <PROTCT>
DATA    DEFL    DATA-20H        ;;SET PROTECTED MODE BIT
        ENDIF
;
        REPT    KBYTES/4
        DB      DATA
DATA    DEFL    DATA-1          ;;FOR NEXT LOOP
        ENDM
        DB      0FH             ;;END OF TABLE
EXIT:
        ENDM
;**************************************************************************
MAPMEM  MACRO   BASE,PAGE,KBYTES
        LOCAL   TABLE,EXIT
        MAPSET  TABLE,BASE,EXIT ;;GENERATE INITIAL CODE
;
DATA    DEFL    NOT PAGE/32     ;;STARTING DATA VALUE
BLK     DEFL    BASE/4096       ;;STARTING BLOCK ADRS
;
        REPT    KBYTES/4
        DB      DATA
DATA    DEFL    DATA-(BLK AND 1);;COMPUTE NEXT DATA VALUE
BLK     DEFL    BLK+1           ;;KEEP TRACK OF BLOCK VALUE
        ENDM
        DB      0FH             ;;END OF TABLE
EXIT:
        ENDM
;**************************************************************************
```

OFF-CARD I/O

I/O devices on the Multibus may be accessed by the MLZ-91 appropriately setting the I/O mapping RAM to enable off-card accesses. If the MLZ-91 is sharing the bus with other cards, none of which are other MLZ-91's or boards monitoring the upper 4 Multibus address bits,(A19, A18, A17 and A16), then the standard Z-80 INPUT and OUTPUT instructions may be used and the memory mapping RAM contents are not significant.

However, if the MLZ-91 is sharing the bus with another MLZ-91, or similar board, then the upper 4 address lines must be specified during any I/O operation on the bus. To do this, the special Z-80 INPUT and OUTPUT instructions are used so that register B can be used to specify the state of the upper address lines via the memory mapping RAM. The proper method to do bus I/O operations is as follows:

1. Setup the I/O map to specify one (or more) quarters of the I/O space as "EXTERNAL" I/O. (Store the value ØEH in the I/O map.)
2. Load one (BLOCK) entry in the memory map with D6, D5, D4 and D3 specifying A19-, A18-, A17- and A16- (Note that the data stored in the memory mapping RAM is negative true with respect to the bus address lines.) The value of these upper address lines specifies which MLZ-91 board on the Multibus will respond to a bus I/O or memory request.
3. Execute the desired I/O operations using a sequence similar to the following:

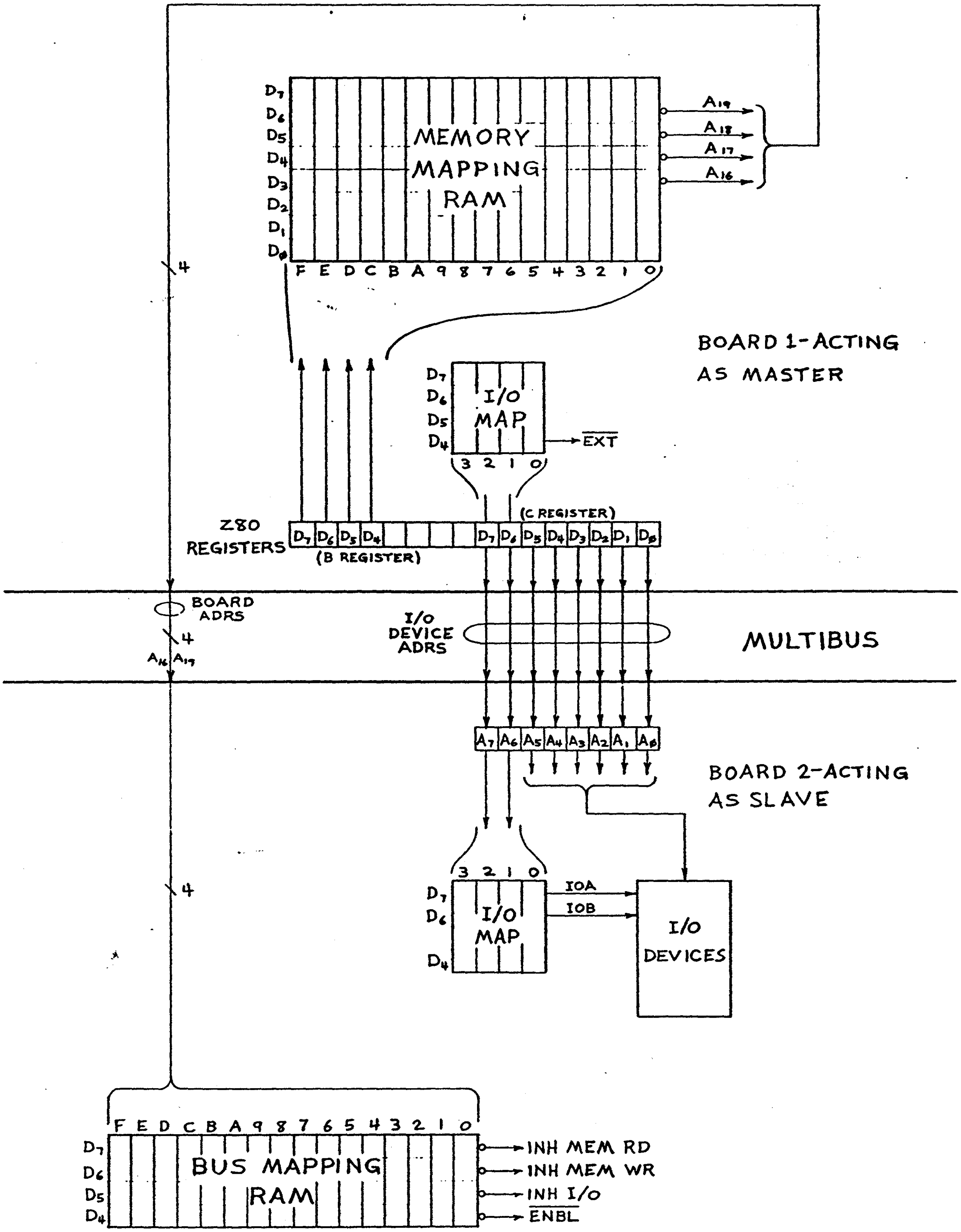
```
LD      C,IOPORT          ;DESIRED PORT ADRS (EXTERNAL)
LD      B,BLOCK           ;MEMORY MAPPING RAM BLOCK
                           same block address used in
                           step 2)
OUT     (C),A             ;EXECUTE EXTERNAL I/O
```

See page 104 for a program listing and more information.

X	$\overline{A19}$	$\overline{A18}$	$\overline{A17}$	$\overline{A16}$	$\overline{A15}$	$\overline{A14}$	$\overline{A13}$

Specifies upper 4
Address lines
(Bus Block)

MEMORY MAP CONTENTS DURING OFF-CARD I/O



MLZ-91 EXTERNAL I/O LOGIC

```
        ASEG
        .Z80
        TITLE   EXAMPLE OF MLZ-91 INTER-BOARD I/O
;**************************************************************************
;CONSTANTS (USUALY THESE WILL BE PART OF A LARGER PROGRAM):
IOXXXO  EQU     OEH             ;I/O MAP DATA FOR OFF-CARD DEVICES
IOXXXA  EQU     07H             ;I/O MAP DATA FOR DEVICE GROUP IOA
IOXXXB  EQU     0BH             ;I/O MAP DATA FOR DEVICE GROUP IOB
IOA     EQU     080H            ;I/O GROUP A BASE ADRS
IOB     EQU     0C0H            ;I/O GROUP B BASE ADRS
MEMMAP  EQU     IOA+20H         ;MEMORY MAPPING RAM DEVICE ADRS
;
ROMBASE EQU     0F000H          ;BASE OF ROM
IOMAP   EQU     ROMBASE+10H     ;I/O DEVICE MAP (VIA WRITE TO ROM)
BUSMAP  EQU     ROMBASE+20H     ;BUS MAP (ALSO VIA WRITE TO ROM)
;**************************************************************************
;THIS PROGRAM ILLUSTRATES HOW TO DO I/O OPERATIONS BETWEEN MLZ-91 BOARDS.
;MANY OF THE TECHNIQUES USED HERE ARE ALSO APPROPRIATE FOR OTHER SYSTEM
;CONFIGURATIONS, PARTICULARLY THOSE WHERE I/O DEVICES ON THE MULTIBUS
;HAVE I/O PORT ADDRESSES WHICH CANNOT BE MOVED AND OCCUPY PORT ADDRESSES
;WHICH ARE USED ON-CARD AS WELL.  SINCE THE ON-CARD DEVICE ADDRESSES MAY
;BE MOVED AROUND, VIA THE I/O DEVICE MAPPING RAM, ALL BUS I/O ADDRESS
;CAN BE USED BY AN MLZ-91.  THERE IS A SIMPLER METHOD FOR SYSTEMS WHICH
;ONLY HAVE TWO MLZ-91'S, DESCRIBED LATER.  THE FOLLOWING EXAMPLES
;ARE FOR 'GENERAL' CASES.
;**************************************************************************
```

```
;***************************************************************************
;FOR A TWO PROCESSOR SYSTEM:
;IT IS POSSIBLE TO USE THE CONVENTIONAL INPUT (IN) AND OUTPUT (OUT)
;INSTRUCTIONS IN A TWO BOARD SYSTEM.  ALSO, THE PROCEDURE FOR DOING
;INTER-BOARD I/O IS SIMPLER.  THE MAPPING RAMS CAN BE PRESET AND NO
;FIDDLING IS REQUIRED WHEN DOING THE I/O OPERATION.
;
;FIRST EACH BOARD ASSIGNS ITS I/O SPACE AS FOLLOWS:
;
;       BOARD   GROUP 0 GROUP 1 GROUP 2 GROUP 3
;       -----   ------- ------- ------- -------
;         1     IOXXXO  IOXXXO  IOXXXA  IOXXXB          IOXXXO = OFF-CARD
;         2     IOXXXA  IOXXXB  IOXXXO  IOXXXO          IOXXXA = IOA GROUP
;       BASE=    000H    040H    080H    0C0H           IOXXXB = IOB GROUP
;
;ON BOARD 1:
        LD      L,IOXXXO        ;OFF-CARD DEVICE DATA
        LD      H,IOXXXO        ;OFF-CARD DEVICE DATA
        LD      (IOMAP),HL      ;SET GROUP 0 & 1
        LD      L,IOXXXA        ;ON-CARD DEVICE GROUP A
        LD      H,IOXXXB        ;ON-CARD DEVICE GROUP B
        LD      (IOMAP+2),HL    ;SET GROUP 2 & 3
;
;AND ON BOARD 2 (ALMOST THE SAME):
        LD      L,IOXXXA        ;ON-CARD DEVICE GROUP A
        LD      H,IOXXXB        ;ON-CARD DEVICE GROUP B
        LD      (IOMAP),HL      ;SET GROUP 0 & 1
        LD      L,IOXXXO        ;OFF-CARD DEVICE DATA
        LD      H,IOXXXO        ;OFF-CARD DEVICE DATA
        LD      (IOMAP+2),HL    ;SET GROUP 2 & 3
;
;THEN DO THIS ON BOTH BOARDS:
        LD      HL,BUSMAP       ;DESTINATION (BUS MAPPING RAM)
        LD      B,16            ;LENGTH
;
BLOOP:  LD      (HL),0F0H       ;ENABLE ALL ACCESSES
        INC     HL              ;TO NEXT MAP LOCATION
        DJNZ    BLOOP           ;DO ALL 16 BUS BLOCKS
;
;NOW, BOARD 1 USES THE FIRST TWO I/O GROUPS (DEVICE ADDRESSES 00H THROUGH
;07FH) TO ACCESS THE OTHER BOARD'S DEVICES AND THE SECOND TWO I/O GROUPS
;(DEVICE ADDRESSES 80H THROUGH 0FFH) TO ACCESS ITS ON-CARD DEVICES.  FOR
;BOARD 2 IT'S THE OPPOSITE: DEVICES 00 THROUGH 7FH ARE ON-CARD AND 80H
;THROUGH 0FFH ARE OFF-CARD.
;
;FOR EXAMPLE, THE LED ARRAY ADDRESSES FOR EACH BOARD ARE: (PORT 'IOLED')
;       BOARD 1:  IOB (FOR BOARD 1) + 0EH = 0C0H + 0EH = 0CEH
;       BOARD 2:  IOB (FOR BOARD 2) + 0EH = 040H + 0EH = 04EH
;***************************************************************************
```

```
;***************************************************************************
;THIS SECTION SHOWS HOW TO DO I/O OPERATIONS WHEN THERE ARE MORE THAN
;TWO MLZ-91'S IN A SYSTEM WHICH MUST SHARE I/O DEVICES.  CAUTION: THIS
;LOOKS RATHER COMPLICATED, BUT IF SOME CONSTRAINTS ARE USED WHEN
;DESIGNING YOUR SOFTWARE, MANY OF THESE STEPS CAN BE DELETED.
;THIS SHOULD BE CONSIDERED AN 'ADVANCED' EXAMPLE.  KNOWLEDGE OF THE
;VARIOUS MAPPING RAM FUNCTIONS IS REQUIRED (SEE THE MLZ-91 USER MANUAL).
;***************************************************************************
;STEP 1:  (REQUIRED ONLY IF THE TARGET I/O DEVICE IS ON ANOTHER MLZ-91.)
;PICK ANY MEMORY BLOCK (4K) WHICH IS NOT BEING USED (THERE WOULD BE AT
;MOST 15 OF THEM) AND LOAD THE BLOCK DATA SO AS TO POINT (VIA THE UPPER
;FOUR MULTIBUS ADDRESS LINES) TO THE TARGET MLZ-91.  THE CHOSEN BLOCK
;WILL BE USED FOR THE I/O OPERATION (NOT FOR A MEMORY OPERATION.)
;IF ALL MEMORY BLOCKS ARE 'IN USE' THEN PICK ONE FOR TEMPORARY
;REALLOCATION.  IF THERE ALREADY IS A BLOCK WHICH POINTS TO THE TARGET
;BOARD (E.G. PART OF THE TARGET'S MEMORY) THEN THIS STEP IS NOT
;NECESSARY. ALSO, THE TARGET BOARD MUST NOT HAVE ITS I/O INHIBIT BIT
;SET IN ITS BUS MAPPING RAM.  HERE WE GO...
        LD      B,BLOCK/256     ;LOAD DESIRED BLOCK
        LD      C,MEMMAP        ;MEMORY MAPPING RAM DEVICE ADRS
        LD      A,NOT (BOARD*8) ;DATA FOR MEMORY MAP (D7 = 1
                                ;D6, D5, D4 & D3 = BOARD NBR (INVERTED)
        OUT     (C),A           ;SETUP MAP RAM
;***************************************************************************
;STEP 2:  SETUP I/O DEVICE MAPPING RAM TO GO OFF-CARD FOR THE PARTICULAR
;DEVICE NUMBER.
        LD      A,IOXXXO        ;OFF-CARD DEVICE ENABLE CODE
        LD      (IOMAP+N),A     ;PUT IN I/O MAP. N = 0, 1, 2 OR 3
                                ;DEPENDING ON THE I/O GROUP OF THE DEVICE
;***************************************************************************
```

```
;***************************************************************************
;STEP 3:  DO THE I/O OPERATION.  THE MEMORY MAPPING RAM BLOCK NUMBER,
;SELECTED IN STEP 1, ABOVE, MUST BE IN REGISTER B DURING THE I/O.  THUS,
;THE 'SPECIAL' Z80 I/O INSTRUCTIONS MUST BE USED.
        LD      B,BLOCK/256     ;MEMORY BLOCK BEING USED FOR I/O
        LD      C,DEVICE        ;THE ACTUAL DEVICE NUMBER
        IN      A,(C)           ;(OR 'OUT') DO THE I/O OPERATION
;***************************************************************************
;STEP 4:  RESTORE THE I/O MAPPING RAM IF NECESSARY (MAY HAVE BEEN ALTERED
;IN STEP 2, ABOVE.)
        LD      A,IOXXXA        ;(EXAMPLE)
        LD      (IOMAP+N),A     ;RESTORE
;***************************************************************************
;STEP 5:  RESTORE THE MEMORY MAP RAM IF NECESSARY (MAY HAVE BEEN ALTERED
;IN STEP 1, ABOVE)
        LD      B,BLOCK/256     ;BLOCK ADRS
        LD      C,MEMMAP        ;MEMORY MAPPING RAM DEVICE ADRS
        LD      A,DATA          ;ORIGINAL VALUE
        OUT     (C),A           ;RESTORE
;***************************************************************************
;IN MANY APPLICATIONS, IT WOULD BE POSSIBLE TO DO STEPS 1 & 2 ONLY ONCE.
;THEN, STEP 1 WOULD BE USED TO CHANGE TARGET BOARDS.  STEPS 4 & 5 MAY
;NEVER BE REQUIRED IF THE MEMORY BLOCK AND DEVICE GROUP CAN REMAIN
;ALLOCATED FOR OFF-CARD USE.
;***************************************************************************
;CONSTANTS USED FOR EXAMPLE PURPOSES:
BOARD   EQU     3               ;MULTIBUS BOARD NBR (A19, A18, A17, A16)
N       EQU     2               ;I/O MAP GROUP OF OFF-CARD DEVICE
BLOCK   EQU     0E000H          ;MEMORY MAPPING RAM BLOCK USED FOR I/O
DEVICE  EQU     78H             ;THE ACTUAL OFF-CARD DEVICE ADRS
DATA    EQU     51              ;THE ORIGINAL MEM MAP DATA FOR BLOCK E
;***************************************************************************
;***************************************************************************
;JUST FOR FUN, LET'S SEE WHAT THE TARGET BOARD WOULD HAVE HAD TO DO TO
;ALLOW THE ACCESS ILLUSTRATED ABOVE.  THIS CODE WOULD HAVE BEEN DONE ON
;THE TARGET BOARD:
;FIRST, THE TARGET BOARD'S BUS MAP WOULD HAVE TO BE SET TO ALLOW I/O
;THROUGH BUS BLOCK 3 ('BOARD')
        LD      A,0F0H          ;NO INHIBITS, I/O IS ALLOWED
        LD      (BUSMAP+0FH-BOARD),A     ;SET BUS MAPPING RAM
;SECOND, THE TARGET BOARD WOULD HAVE HAD TO HAVE DEVICE 78H ('DEVICE')
;ALLOCATED.  THIS PROBABLY WOULD ALREADY HAVE BEEN DONE.  DEVICE 78H, IF
;I/O GROUP IOA IS IN BLOCK 2 ('N'), IS DIP SWITCH GROUP 0.
        LD      A,IOXXXA        ;I/O MAP DATA FOR GROUP A
        LD      (IOMAP+N),A     ;SET MAP DATA
;***************************************************************************
        END
```

****** NOTE: PAGES 108 AND 109 HAVE BEEN INTENTIONALLY OMITTED. ******

INPUT/OUTPUT DEVICE ADDRESSES

The MLZ-91 I/O space (256 devices) is divided into four groups. The MLZ-91 I/O devices are divided into two groups. Each device group may be assigned to one of the four I/O groups via the I/O mapping RAM. (See page 42 for details.) The two device groups are named IOA and IOB. The four I/O space blocks start at ØØH, 4ØH, 8ØH and CØH. Thus, the base of IOA or IOB will be one of those four values. The chart below shows the offset from the base address assigned to each group. Thus, if the base of IOA is 8ØH, then the device address for the FDIO select port is 8ØH + 18H or 98H. (IOFSEL)

```
****************************************************************************
MEMORY ADDRESS CONSTANTS:

NAME            ADDRESS         DESCRIPTION
-------         -------         -----------
ROMBASE         F000 (HEX)      BASE OF ROM (TYPICAL)
RAMBASE         E000 (HEX)      BASE OF A 4K RAM BLOCK (TYPICAL)
IOMAP           ROMBASE+10H     I/O DEVICE MAP (VIA WRITE TO ROM)
BUSMAP          ROMBASE+20H     BUS MAP (ALSO VIA WRITE TO ROM)
****************************************************************************
I/O DEVICE CONSTANTS:
GROUP NAME      BASE ADDRESS    DESCRIPTION
----------      ------------    -----------
IOA             080H            BASE OF I/O DEVICE GROUP "A" (TYPICAL)
IOB             0C0H            BASE OF I/O DEVICE GROUP "B" (TYPICAL)
NOTE:  THE DEVICE GROUP BASE ADDRESSES ARE DETERMINED BY THE I/O MAPPING
RAM.  THE BASE ADDRESSES MAY BE SET AT 00H, 40H, 80H OR C0H.
****************************************************************************
IOA DEVICE GROUP:

DEVICE NAME     DEVICE ADDRESS  FUNCTION
-----------     --------------  --------
IOBDA           IOA+00H         LOAD BAUD DATA FOR SIO PORT A (D7-D4)
IOBDB           IOA+08H         LOAD BAUD DATA FOR SIO PORT B (D3-D0)

IODMA           IOA+10H         DMA CONTROL AND STATUS
IOFSEL          IOA+18H         FDIO DRIVE SELECT AND USER LED
MEMMAP          IOA+20H         MEMORY MAPPING RAM

IOPOP           IOA+28H         APU POP DATA
IOAPUR          IOA+29H         APU READ STATUS
IOPUSH          IOA+30H         APU PUSH DATA
IOAPUW          IOA+31H         APU ENTER COMMAND

IODIP0          IOA+38H         READ DIP SWITCH GROUP 0 (1-8)
IODIP1          IOA+39H         READ DIP SWITCH GROUP 1 (9-16)

IOWCLR          IOA+3AH         CLEAR WINCHESTER MSEL FF

IOCNT0          IOA+3EH         CTC CHANNEL 0 COUNT/TRIGGER
****************************************************************************
```

****
IOB DEVICE GROUP:

DEVICE NAME	DEVICE ADDRESS	FUNCTION
IOSAD	IOB+00H	SIO PORT A DATA
IOSBD	IOB+01H	SIO PORT B DATA
IOSAC	IOB+02H	SIO PORT A CONTROL/STATUS
IOSBC	IOB+03H	SIO PORT B CONTROL/STATUS
IOTRDC	IOB+08H	STREAMER TAPE READ DATA & CLR XFER
IOTRDS	IOB+09H	STREAMER TAPE READ DATA & SET XFER
IOTWRC	IOB+0AH	STREAMER TAPE WRITE DATA & CLR XFER
IOTWRS	IOB+0BH	STREAMER TAPE WRITE DATA & SET XFER
IOTRDY	IOB+0CH	STREAMER TAPE SET READY (TRDY)
IOLED	IOB+0EH	LOAD LED ARRAY
IOFDCS	IOB+10H	FDIO COMMAND/STATUS REGISTER
IOFDTR	IOB+11H	FDIO TRACK REGISTER
IOFDSR	IOB+12H	FDIO SECTOR REGISTER
IOFDAT	IOB+13H	FDIO DATA REGISTER
IOCTC0	IOB+18H	CTC 0 DATA & CONTROL
IOCTC1	IOB+19H	CTC 1 DATA & CONTROL
IOCTC2	IOB+1AH	CTC 2 DATA & CONTROL
IOCTC3	IOB+1BH	CTC 3 DATA & CONTROL
IOCLRN	IOB+20H	CLEAR NMI FF (PARITY & WRITE PROTECT ERRORS)
IOWSEL	IOB+28H	WINCHESTER - SET MSEL FF
IOWWR0	IOB+2AH	WINCHESTER - WRITE DATA/COMMAND (C/D- LOW)
IOWWR1	IOB+2BH	WINCHESTER - WRITE DATA/COMMAND (C/D- HIGH)
IOWRD0	IOB+2CH	WINCHESTER - READ DATA (C/D- LOW)
IOWRD1	IOB+2DH	WINCHESTER - READ DATA/STATUS (C/D- HIGH)
IOWRDS	IOB+2EH	WINCHESTER - READ INTERFACE STATUS
IOSTAT	IOB+2FH	READ BOARD STATUS BITS (D7-D4)
IOGPIB	IOB+30H	GPIB (IEEE-488) - BASE OF REGISTERS
IOGPDA	IOGPIB+7	GPIB (IEEE-488) - DATA REGISTER
IOPAD	IOB+38H	PIO A DATA - SYSTEM INT/BUS/DMA RDY
IOPBD	IOB+39H	PIO B DATA - MULTIBUS INTERRUPTS
IOPAC	IOB+3AH	PIO A CNTRL, SET BIT MODE (CFH) AND E0H MASK
IOPBC	IOB+3BH	PIO B CNTRL, SET BIT MODE (CFH) & REQ'D MASK

****

SIO

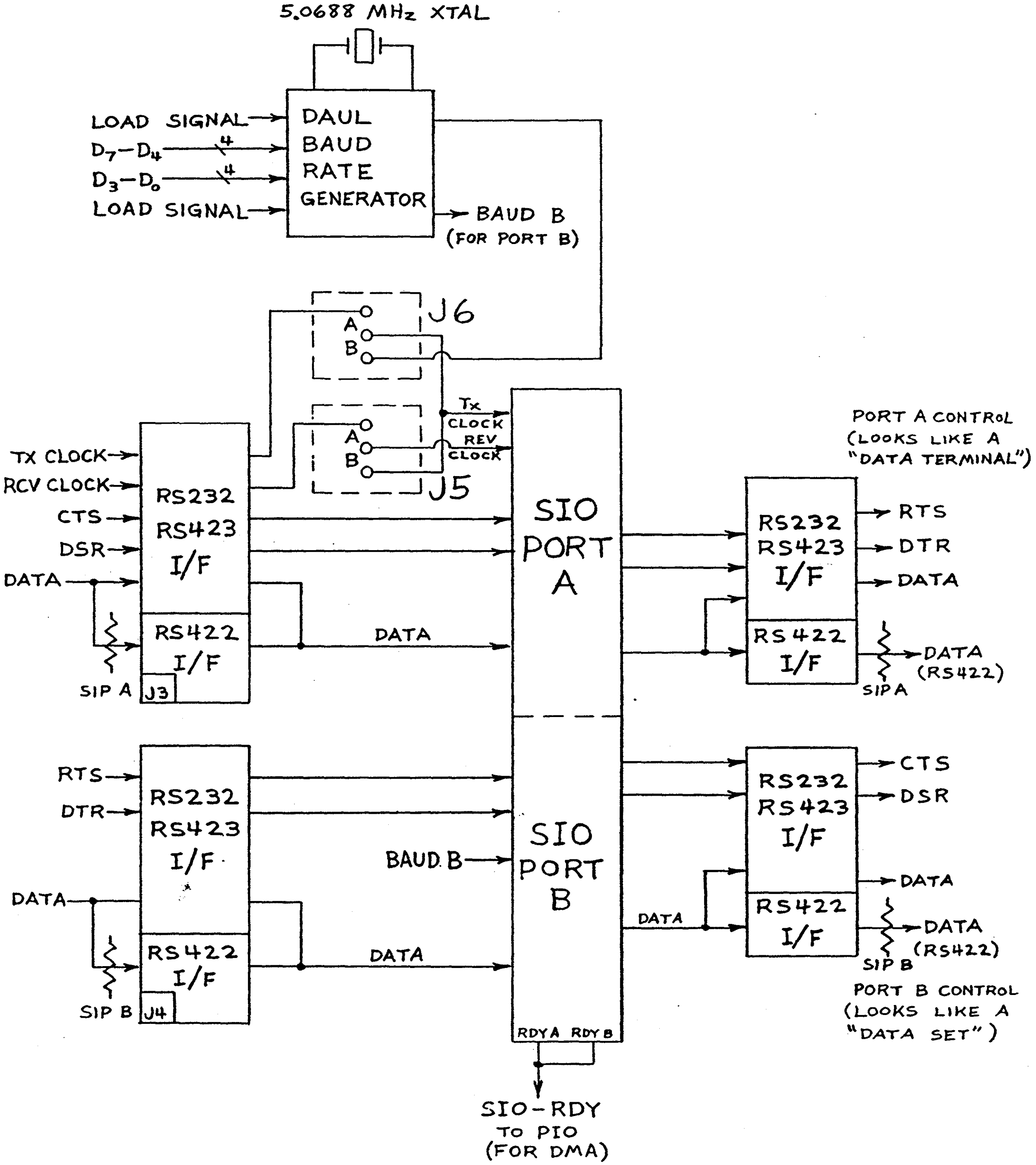
The dual SIO is a complex chip which has several commands and status registers. As an example of a simple setup procedure, the command sequence used in ZRAID is shown below. This sequence defines SIO port B as an asynchronous port without interrupts. The commands are transferred to the SIO port B control register (I/O port IOSBC) in the order listed. (An OTIR instruction sequence could easily be used.)

COMMAND	CMD TYPE	FUNCTION
ØØ	Ø	Reset register select logic
Ø4	Ø	Select write register 4
4C	4	Set X16 clock, transmit 2 stop bits/character
Ø5	Ø	Select write register 5
EA	5	Set DTR active, transmit 8 bits/character, enable transmitter, set RTS active.
Ø3	Ø	Select write register 3
Cl	3	Set receive 8 bits/character, enable receiver
Ø1	Ø	Select write register.
ØØ	1	Disable interrupts

The following subroutines can then be used to test the receive and transmit register status bits and to transfer data (Port B):

```
Receiver:  RWAIT:   XOR    A            Clear Accumulator
                    OUT    (IOSBC),A    Select read register Ø
                    IN     A,(IOSBC)    Read Receiver status
                    AND    1            Mask Data Ready bit
                    JP     Z,RWAIT      Wait for ready
                    IN     A,(IOSBD)    Get Character from Data port
                    RET                 Done
Transmitter:  TX:   PUSH   PSW          Save character
           TWAIT:   XOR    A            Clear Accum
                    OUT    (IOSBC),A    Select read register Ø
                    IN     A,(IOSBC)A   Read transmitter status
                    AND    4            Mask TX buffer empty
                    JP     Z,TWAIT      Wait for empty
                    POP    PSW          Get character
                    OUT    (IOSBD),A    Transmit data
                    RET                 Done
```

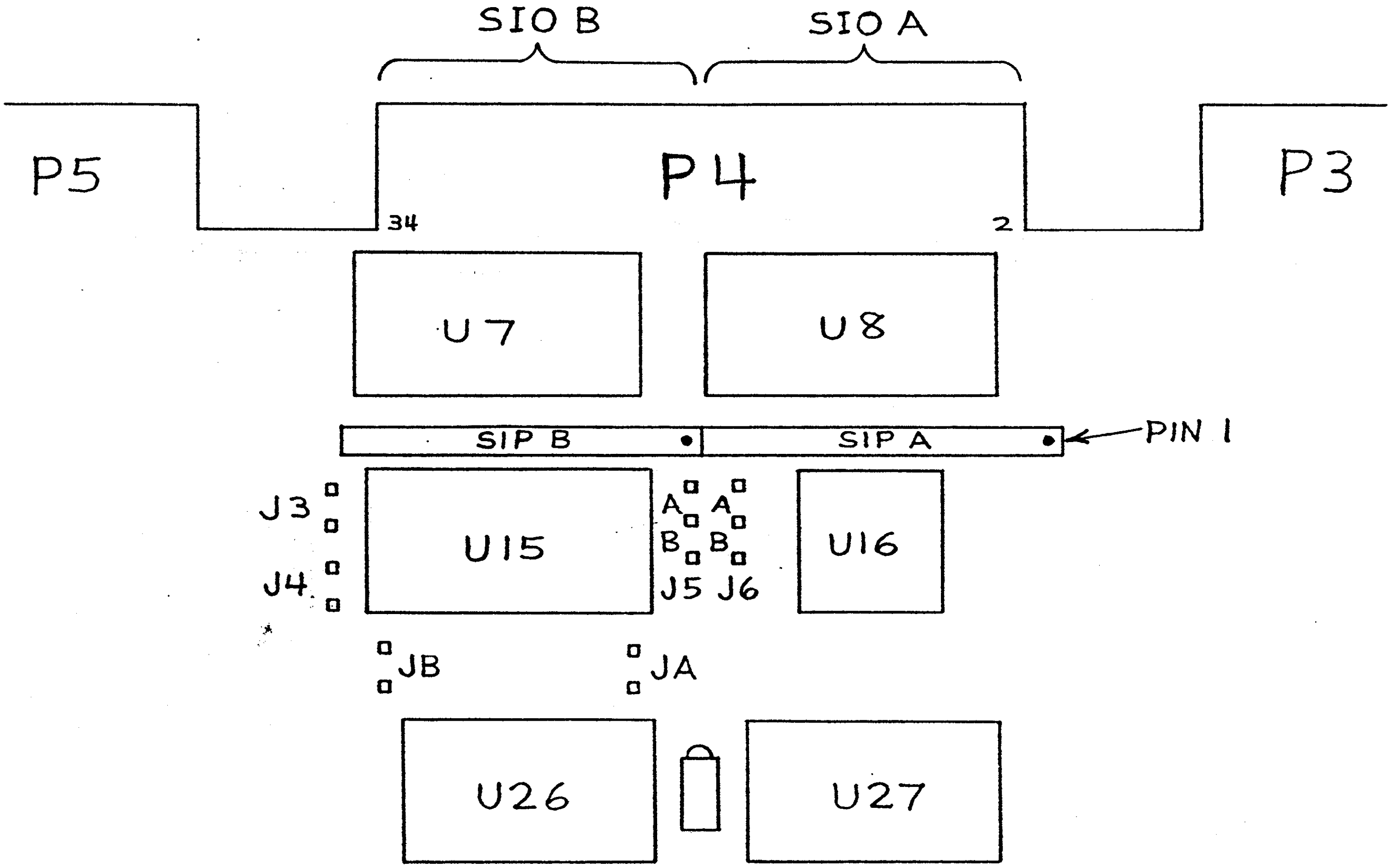
For complete information in programming the SIO chip, refer to the SIO manual. Summary information follows. See page 158 for connector information.



SIO BLOCK DIAGRAM

SIO PORT	RS232/423	RS422 Terminated	RS422 Unterminated
A	Remove SIP A* Install J3	Install U15 Install SIP A Remove J3	Install U15 Remove SIP A Remove J3
B	Remove SIP B** Install J4	Install SIP B Install U15 Remove J4	Remove SIP B Install U15 Remove J4

*or cut trace at Ja
**or cut trace at Jb



Z-80 SIO Architecture

Data Path

The transmit and receive data path

is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in Asynchronous modes—the character length.

The transmitter has an 8-bit transmit data register that is loaded from the internal data bus, and a 20-bit transmit shift register that can be loaded from the sync character buffers (WR6 and WR7) or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data Output (TxD).

Functional Description

The functional capabilities of the Z80-SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data, and meets the requirements of various data communications protocols; as a Z80 family peripheral, it interacts with the Z80-CPU and other Z80 peripheral circuits, and shares the data, address and control busses, as well as being a part of the Z80 interrupt structure. As a peripheral to other microprocessors, the Z80-SIO offers valuable features such as non-vectored interrupts, polling and simple handshake capability.

The first part of the following functional description describes the interaction between the CPU and Z80-SIO; the second part introduces its data communications capabilities.

I/O Interface Capabilities

The Z80-SIO offers the choice of Polling, Interrupt (vectored or non-vectored) and Block Transfer modes to transfer data, status and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. There are no interrupts in the Polled mode. Status registers RR0 and RR1 are updated at appropriate times for each function being performed (for example, CRC Error status valid at the end of the message). All the interrupt modes of the Z80-SIO must be disabled to operate the device in a polled environment.

While in its Polling sequence, the CPU examines the status contained in RR0 for each channel; the RR0 status bits serve as an acknowledge to the Poll inquiry. The two RR0 status bits D_0 and D_2 indicate that a data transfer is needed. The status also indicates Error or other special status conditions (see "Z80-SIO Programming"). The Special Receive Condition status contained in RR1 does not have to be read in a Polling sequence because the status bits in RR1 must be accompanied by a Receive Character Available status in RR0.

Interrupts. The Z80-SIO offers an elaborate interrupt scheme to provide fast interrupt response in real-time applications. Channel B registers WR2 and RR2 contain the interrupt vector that points to an interrupt service routine in the memory. To service operations in both channels and to eliminate the necessity of writing a status analysis routine, the Z80-SIO can modify the interrupt vector in RR2 so it points directly to one of eight interrupt service routines. This is done under program control by setting a program bit (WR1, D_2) in Channel B called "Status Affects Vector." When this bit is set, the interrupt vector in WR2 is modified according to the assigned priority of the various interrupting conditions. The table in the Write Register 1 description (Z80-SIO Programming section) shows the modification details.

Transmit interrupts, Receive interrupts and External/Status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control with Channel A having a higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted by the transmit buffer *becoming* empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on the first received character
- Interrupt on all received characters
- Interrupt on a Special Receive condition

Interrupt On First Character is typically used with the Block Transfer mode. Interrupt On All Receive Characters has the option of modifying the interrupt vector in the event of a parity error. The Special Receive Condition interrupt can occur on a character or message basis (End Of Frame interrupt in SDLC, for example). The Special Receive condition can cause an interrupt only if the Interrupt On First Receive Character or Interrupt On All Receive Characters mode is selected. In Interrupt On First Receive Character, an interrupt can occur from Special Receive conditions (except Parity Error) after the first receive character interrupt (example: Receive Overrun interrupt).

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$ and $\overline{SYNC}$ pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition or by the detection of a Break (Asynchronous mode) or Abort (SDLC mode) sequence in the data stream. The interrupt caused by the Break/Abort sequence has a special feature that allows the Z80-SIO to interrupt when the Break/Abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Break/Abort condition in external logic.

CPU/DMA Block Transfer. The Z80-SIO provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers (Z80-DMA or other designs). The Block Transfer mode uses the $\overline{\text{WAIT}}/\overline{\text{READY}}$ output in conjunction with the Wait/Ready bits of Write Register 1. The $\overline{\text{WAIT/READY}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{READY}}$ line in the DMA Block Transfer mode.

To a DMA controller, the Z80-SIO READY output indicates that the Z80-SIO is ready to transfer data to or from memory. To the CPU, the WAIT output indicates that the Z80-SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle. The programming of bits 5, 6 and 7 of Write Register 1 and the logic states of the $\overline{\text{WAIT/READY}}$ line are defined in the Write Register 1 description (Z80-SIO Programming section).

Data Communications Capabilities

In addition to the I/O capabilities previously discussed, the Z80-SIO provides two independent full-duplex channels that can be programmed for use in Asynchronous, Synchronous and SDLC (HDLC) modes. These different modes are provided to facilitate the implementation of commonly used data communications protocols. The following is a short description of the data communications protocols supported by the Z80-SIO. A more detailed explanation of these modes can be found in the *Z80-SIO Technical Manual.*

Asynchronous Modes. The Z80-SIO offers transmission and reception of five to eight bits per character, plus optional even or odd parity. The transmitter can supply one, one and a half or two stop bits per character and can provide a break output at any time. The receiver break detection logic interrupts the CPU only at the start and end of a received break. Reception is protected from spikes by a transient spike rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the Receive Data input. If the Low does not persist—as in the case of a transient—the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The Z80-SIO does not require symmetric Transmit and Receive Clock signals—a feature that allows it to be used with a Z80-CTC or any other clock source. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the Receive and Transmit Clock inputs.

In Asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed for an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The Z80-SIO supports both byte-oriented and bit-oriented synchronous communication. Synchronous byte-oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16-bit sync pattern (Bisync), or with an external sync signal. Leading sync characters can be removed without interrupting the CPU. CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16}+X^{15}+X^2+1$) and CCITT ($X^{16}+X^{12}+X^5+1$) error checking polynomials are supported. In all non-SDLC modes, the CRC generator is initialized to 0's; in SDLC modes, it is initialized to 1's. (This means that the Z80-SIO cannot generate or check CRC for IBM-compatible soft-sectored disks.) The Z80-SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high-speed transmissions under DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in Synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length. Since the CPU can read status information from the Z80-SIO, it can determine the type of transmission (data, CRC or sync characters) that is taking place at any time.

The Z80-SIO supports synchronous bit-oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion and CRC generation. A special command can be used to abort a frame in transmission. The Z80-SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. An interrupt warns the CPU of this status change so an abort may be issued if a transmitter underrun has occurred. One to eight bits per character can be sent, which allows transmission of a message exactly as received with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame and provides a synchronization signal that can be programmed to interrupt. In addition, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers. The receiver can be programmed to search for frames addressed to only a specified user-selectable address or to a global broadcast address. In this mode, frames that do not match the user-selected or broadcast address are ignored. The Address Search mode provides for a single-byte address recognizable by the hardware. The number of address bytes can be extended under software control.

The Z80-SIO can be conveniently used under DMA control to provide high-speed reception. The Z80-SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The Z80-SIO then issues an End Of Frame interrupt and the CPU checks the status of the received message. Thus, the CPU is freed for other service while the message is being received. A similar scheme allows message transmission under DMA control.

Z80-SIO Programming

To program the Z80-SIO, the system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity are first set, then the interrupt mode and, finally, receiver or transmitter enable. The WR4 parameters must be issued before any other parameters are issued in the initialization routine.

Both channels contain command registers that must be programmed via the system program prior to operation. The Channel Select input (B/$\overline{A}$) and the Control/Data input (C/$\overline{D}$) are the command structure addressing controls, and are normally controlled by the CPU address bus.

Write Registers

The Z80-SIO contains eight registers (WR0-WR7) in each channel that are programmed separately by the system program to configure the functional personality of the channels. With the exception of WR0, programming the write registers requires two bytes. The first byte contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the Z80-SIO.

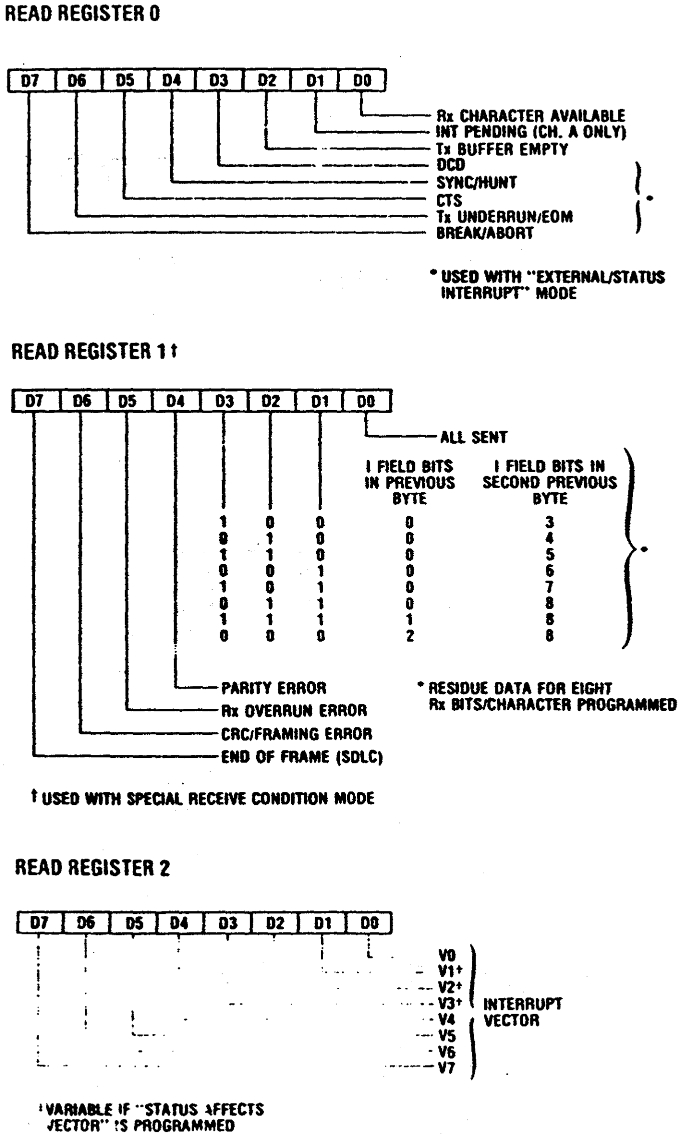
WR0 is a special case in that all the basic commands (CMD_0-CMD_2) can be accessed with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0.

Read Registers

The Z80-SIO contains three registers, RR0-RR2 (Figure 6), that can be read to obtain the status information for each channel (except for RR2 — Channel B only). The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing an input instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).



D3	D2	D1	I FIELD BITS IN PREVIOUS BYTE	I FIELD BITS IN SECOND PREVIOUS BYTE
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

Figure 6. Read Register Bit Functions

WRITE REGISTER 0

D7	D6	D5	D4	D3	D2	D1	D0

D2	D1	D0	
0	0	0	REGISTER 0
0	0	1	REGISTER 1
0	1	0	REGISTER 2
0	1	1	REGISTER 3
1	0	0	REGISTER 4
1	0	1	REGISTER 5
1	1	0	REGISTER 6
1	1	1	REGISTER 7

D5	D4	D3	
0	0	0	NULL CODE
0	0	1	SEND ABORT (SDLC)
0	1	0	RESET EXT/STATUS INTERRUPTS
0	1	1	CHANNEL RESET
1	0	0	ENABLE INT ON NEXT Rx CHARACTER
1	0	1	RESET TxINT PENDING
1	1	0	ERROR RESET
1	1	1	RETURN FROM INT (CH-A ONLY)

D7	D6	
0	0	NULL CODE
0	1	RESET Rx CRC CHECKER
1	0	RESET Tx CRC GENERATOR
1	1	RESET Tx UNDERRUN/EOM LATCH

WRITE REGISTER 1

D7	D6	D5	D4	D3	D2	D1	D0

- D0 — EXT INT ENABLE
- D1 — Tx INT ENABLE
- D2 — STATUS AFFECTS VECTOR (CH. B ONLY)

D4	D3	
0	0	Rx INT DISABLE
0	1	Rx INT ON FIRST CHARACTER
1	0	INT ON ALL Rx CHARACTERS (PARITY AFFECTS VECTOR) *
1	1	INT ON ALL Rx CHARACTERS (PARITY DOES NOT AFFECT VECTOR) *

- D5 — WAIT/READY ON R/T
- D6 — $\overline{\text{WAIT}}$/READY FUNCTION
- D7 — WAIT/READY ENABLE

* OR ON SPECIAL CONDITION

WRITE REGISTER 2 (CHANNEL B ONLY)

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	V0

V0–V7: INTERRUPT VECTOR

WRITER REGISTER 3

D7	D6	D5	D4	D3	D2	D1	D0

- D0 — Rx ENABLE
- D1 — SYNC CHARACTER LOAD INHIBIT
- D2 — ADDRESS SEARCH MODE (SDLC)
- D3 — Rx CRC ENABLE
- D4 — ENTER HUNT PHASE
- D5 — AUTO ENABLES

D7	D6	
0	0	Rx 5 BITS/CHARACTER
0	1	Rx 7 BITS/CHARACTER
1	0	Rx 6 BITS/CHARACTER
1	1	Rx 8 BITS/CHARACTER

WRITE REGISTER 4

D7	D6	D5	D4	D3	D2	D1	D0

- D0 — PARITY ENABLE
- D1 — PARITY EVEN/$\overline{\text{ODD}}$

D3	D2	
0	0	SYNC MODES ENABLE
0	1	1 STOP BIT/CHARACTER
1	0	1½ STOP BITS/CHARACTER
1	1	2 STOP BITS/CHARACTER

D5	D4	
0	0	8 BIT SYNC CHARACTER
0	1	16 BIT SYNC CHARACTER
1	0	SDLC MODE (01111110 FLAG)
1	1	EXTERNAL SYNC MODE

D7	D6	
0	0	X1 CLOCK MODE
0	1	X16 CLOCK MODE
1	0	X32 CLOCK MODE
1	1	X64 CLOCK MODE

WRITE REGISTER 5

D7	D6	D5	D4	D3	D2	D1	D0

- D0 — Tx CRC ENABLE
- D1 — RTS
- D2 — $\overline{\text{SDLC}}$/CRC-16
- D3 — Tx ENABLE
- D4 — SEND BREAK

D6	D5	
0	0	Tx 5 BITS (OR LESS)/CHARACTER
0	1	Tx 7 BITS/CHARACTER
1	0	Tx 6 BITS/CHARACTER
1	1	Tx 8 BITS/CHARACTER

- D7 — DTR

WRITE REGISTER 6

D7	D6	D5	D4	D3	D2	D1	D0
SYNC BIT 7	SYNC BIT 6	SYNC BIT 5	SYNC BIT 4	SYNC BIT 3	SYNC BIT 2	SYNC BIT 1	SYNC BIT 0

*ALSO SDLC ADDRESS FIELD

WRITE REGISTER 7

D7	D6	D5	D4	D3	D2	D1	D0
SYNC BIT 15	SYNC BIT 14	SYNC BIT 13	SYNC BIT 12	SYNC BIT 11	SYNC BIT 10	SYNC BIT 9	SYNC BIT 8

*FOR SDLC IT MUST BE PROGRAMMED TO "01111110" FOR FLAG RECOGNITION

Write Register Bit Functions

FUNCTION	TYPICAL PROGRAM STEPS		COMMENTS
INITIALIZE	*REGISTER:*	*INFORMATION LOADED:*	
	WR0	CHANNEL RESET	Reset SIO
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 4, RESET EXTERNAL/STATUS INTERRUPT	
	WR4	ASYNCHRONOUS MODE, PARITY INFORMATION, STOP BITS INFORMATION, CLOCK RATE INFORMATION	Issue parameters
	WR0	POINTER 3	
	WR3	RECEIVE ENABLE, AUTO ENABLES, RECEIVE CHARACTER LENGTH	
	WR0	POINTER 5	
	WR5	REQUEST TO SEND, TRANSMIT ENABLE, TRANSMIT CHARACTER LENGTH, DATA TERMINAL READY	Receive and Transmit both fully initialized. Auto Enables will enable Transmitter if $\overline{CTS}$ is active and Receiver if $\overline{DCD}$ is active.
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPT	
	WR1	TRANSMIT INTERRUPT ENABLE, STATUS AFFECTS VECTOR, INTERRUPT ON ALL RECEIVE CHARACTERS, DISABLE WAIT/READY FUNCTION, EXTERNAL INTERRUPT ENABLE	Transmit/Receive interrupt mode selected. External Interrupt monitors the status of the $\overline{CTS}$, $\overline{DCD}$ and $\overline{SYNC}$ inputs and detects the Break sequence. Status Affects Vector in Channel B only.
	TRANSFER FIRST DATA BYTE TO SIO		This data byte must be transferred or no transmit interrupts will occur.
IDLE MODE	EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM		Program is waiting for an interrupt from the SIO.
DATA TRANSFER AND ERROR MONITORING	Z80 INTERRUPT ACKNOWLEDGE CYCLE TRANSFERS RR2 TO CPU *IF A CHARACTER IS RECEIVED:* • TRANSFER DATA CHARACTER TO CPU • UPDATE POINTERS AND PARAMETERS • RETURN FROM INTERRUPT		When the interrupt occurs, the interrupt vector is modified by: 1. Receive Character Available; 2. Transmit Buffer Empty; 3. External/Status change; and 4. Special Receive condition.
	IF TRANSMITTER BUFFER IS EMPTY: • TRANSFER DATA CHARACTER TO SIO • UPDATE POINTERS AND PARAMETERS • RETURN FROM INTERRUPT		Program control is transferred to one of the eight interrupt service routines.
	IF EXTERNAL STATUS CHANGES: • TRANSFER RR0 TO CPU • PERFORM ERROR ROUTINES (INCLUDE BREAK DETECTION) • RETURN FROM INTERRUPT		If used with processors other than the Z80, the modified interrupt vector (RR2) should be returned to the CPU in the Interrupt Acknowledge sequence.
	IF SPECIAL RECEIVE CONDITION OCCURS: • TRANSFER RR1 TO CPU • DO SPECIAL ERROR (E.G. FRAMING ERROR) ROUTINE • RETURN FROM INTERRUPT		
TERMINATION	REDEFINE RECEIVE/TRANSMIT INTERRUPT MODES		When transmit or receive data transfer is complete.
	DISABLE TRANSMIT/RECEIVE MODES		
	UPDATE MODEM CONTROL OUTPUTS (E.G. RTS OFF)		In Transmit, the All Sent status bit indicates transmission is complete.

Asynchronous Mode

Dual Baud Rate Generator

To obtain the desired baud rate for each SIO port, output data to the baud rate generator port according to the following chart.

The output signal produced by the Baud Rate Generator is actually 16 times the value for the asynchronous rates shown below.

Baud Rate		PORT A		PORT B	
(x16)	(xl)				
ASYNC	SYNC	Hex	Octal	Hex	Octal
50		00	000	00	000
75	1200	10	020	01	001
110		20	040	02	002
134.5		30	060	03	003
150	2400	40	100	04	004
300	4800	50	120	05	005
600	9600	60	140	06	006
1200	19200	70	160	07	007
1800		80	200	08	010
2000		90	220	09	011
2400		A0	240	0A	012
3600		B0	260	0B	013
4800		C0	300	0C	014
7200		D0	320	0D	015
9600		E0	340	OE	016
19200		F0	360	OF	017

The baud clock for SIO port A may be driven by an external source, such as a modem. Jumpers J5 and J6 select the source as follows:

J5	Clock Source For Receive Data
J5-A	P4-8 (25 pin "D" pin 17, EIA signal "DD")
J5-B	Output of J6, below

J6	Clock Source for Transmit Data
J6-A	P4-4 (25 pin "D" pin 15, EIA signal "DB")
J6-B	Baud generator for port A as listed above.

For example, to run both receive and transmit clocks from the baud rate generator for port A, set J5-B and J6-B.

When using the ZRAID monitor, the baud rates may be individually set via the DIP switches on the MLZ-91. (Refer to the ZRAID manual for details. Also page 93.)

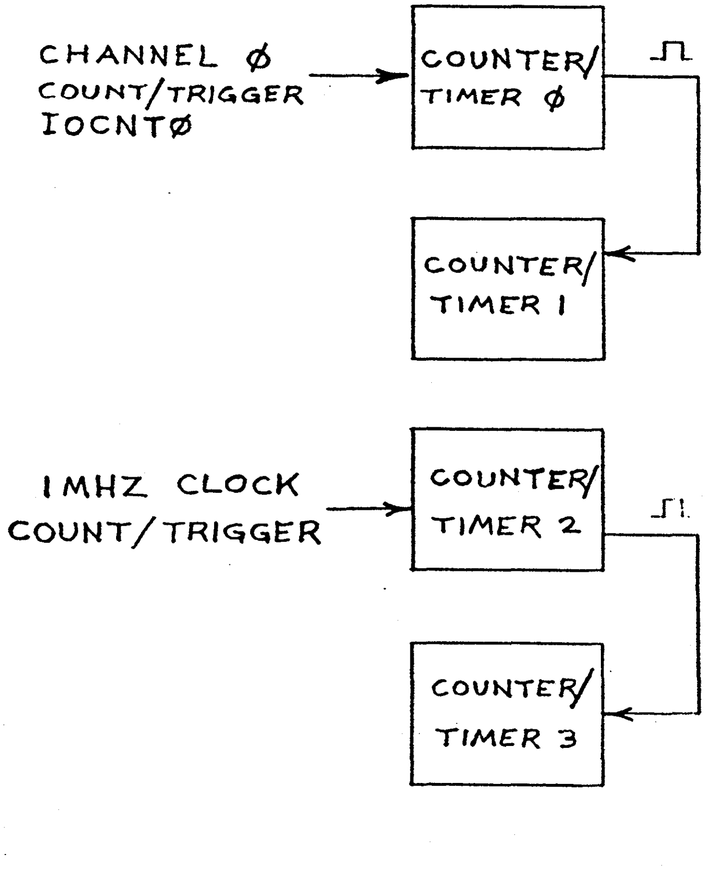
(This page left blank intentionally)

CTC

The Z80 Counter/Timer Circuit contains four channels which may be used to count external events or generate time interrupts to the CPU.

The count limit or time interval may be programmed by the CPU. Each of the four channels can independently interrupt when the programmed count or time has been reached.

The count/trigger input of channel one is assigned an IO device port number. Access of that port can be used to trigger the timer action or as a simple means of counting events without using a software counter. The other three channels are inter-connected to enable a multiple precision count or time interval to be programmed.



CTC CONFIGURATION

CTC Programming

SELECTING AN OPERATING MODE

When selecting a channel's operating mode, bit Ø is set to 1 to indicate this word is to be stored in the channel control register.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT ENABLE	MODE	RANGE	STOP	TRIGGER	LOAD TIME CONSTANT	RESET	1
		USED IN TIMER MODE ONLY		USED IN TIMER MODE ONLY			

Bit 7 = Ø Channel interrupts disabled.

Bit 7 = 1 Channel interrupts enabled to occur every time Down Counter reaches a count of zero. Setting Bit 7 does not let a preceding count of zero cause an interrupt.

Bit 6 = Ø Timer Mode – Down counter is clocked by the prescaler. The period of the counter is:
$t_c \bullet P \bullet TC$
t_c = system clock period
P = prescale of 16 or 256
TC = 8 bit binary programmable time constant (256 max)

Bit 6 = 1 Counter Mode – Down Counter is clocked by external clock. The prescaler is not used.

Bit 5 = Ø Timer Mode Only–System clock Φ is divided by 16 in prescaler.

Bit 5 = 1 Timer Mode Only–System clock Φ is divided by 256 in prescaler.

Bit 4 = Ø Timer Mode – negative edge trigger starts timer operation.
Counter Mode – negative edge decrements the down counter.

Bit 4 = 1 Timer Mode – positive edge trigger starts timer operation.
Counter Mode – positive edge decrements the down counter.

Bit 3 = Ø Timer Mode Only – Timer begins operation on the rising edge of T_2 of the machine cycle following the one that loads the time constant.

Bit 3 = 1 Timer Mode Only – External trigger is valid for starting timer operation after rising edge of T_2 of the machine cycle following the one that loads the time constant. The Prescaler is decremented 2 clock cycles later if the setup time is met, otherwise 3 clock cycles.

Bit 2 = Ø No time constant will follow the channel control word. One time constant must be written to the channel to initiate operation.

Bit 2 = 1 The time constant for the Down Counter will be the next word written to the selected channel. If a time constant is loaded while a channel is counting, the present count will be completed before the new time constant is loaded into the Down Counter.

Bit 1 = Ø Channel continues counting.

Bit 1 = 1 Stop operation. If Bit 2 = 1 channel will resume operation after loading a time constant, otherwise a new control word must be loaded.

LOADING A TIME CONSTANT

An 8-bit time constant is loaded into the Time Constant register following a channel control word with bit 2 set. All zeros indicate a time constant of 256.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
TC_7	TC_6	TC_5	TC_4	TC_3	TC_2	TC_1	TC_0

LOADING AN INTERRUPT VECTOR

The Z80-CPU requires that an 8-bit interrupt vector be supplied by the interrupting channel. The CPU forms the address for the interrupt service routine of the channel using this vector. During an interrupt acknowledge cycle the vector is placed on the Z80 Data Bus by the highest priority channel requesting service at that time. The desired interrupt vector is loaded into the CTC by writing into channel Ø with a zero in DØ. D_7-D_3 contain the stored interrupt vector, D_2 and D_1 are not used in loading the vector. When the CTC responds to an interrupt acknowledge, these two bits contain the binary code of the highest priority channel which requested the interrupt and $D_Ø$ contains a zero since the address of the interrupt service routine starts at an even byte. Channel Ø is the highest priority channel.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	X	X	0

FLOPPY DISK CONTROLIER

Heurikon has floppy disk control software which provides an easy interface to the floppy disk logic on the MLZ-91. The information below is intended for the user who wishes to interface directly with the floppy hardware. (Since the nitty-gritty of the floppy software is fairly complex and involves numerous sections on the MLZ-91, e.g. CPU, DMA, FDIO, PIO, etc., we suggest that you examine the methods used in Heurikon's ZRAID monitor program in order to become familiar with the problems involved if you plan to write your own software interface.)

A. Drive, Side and Single/Double Density Selection

When power is first applied, the drive, side and density select lines come up in a random state. Software must set these lines to the desired condition. The ZRAID-91 monitor initializes these signals as follows:

Drive select lines: all off (HIGH)
Side select line: side 1 (HIGH)
Density select line: single density (HIGH)

Later, when the floppy disk I/O routines are executed, these lines are set as necessary for the specified operation.

The control port for these lines is IOFSEL (typically 98H if I/O group A base address is set at 8ØH by the I/O mapping RAM.) and the function of each bit is as specified below: (This port is write only.)

D7							DØ
$\overline{DS3}$	$\overline{DS2}$	$\overline{DS1}$	$\overline{DSØ}$	$\overline{SS}$	$\overline{DDEN}$	X	$\overline{LED}$

Four Drive Select Lines
Ø = On (Select)
1 = Off (Deselect)

Ø = User Led On
1 = User Led Off

Ø = Double Density
1 = Single Density

Ø = Side Ø
1 = Side 1

Normally, the four drive select lines ($\overline{DSØ}$ thru $\overline{DS3}$) would select one of four drives. However, additional drives may be connected and selected via a binary combination of these four signals.

B. FDIO Controller Chip

There are four register addresses in the FD1793 FDIO chip. The table below indicates the port addresses assigned to the various registers. For details on the function of the registers, refer to the Western Digital manuals.

Port Address	Read(IN)	Write (OUT)
IOFDCS	Status register	Command register
IOFDTR	Track register	Tractk register
IOFDSR	Sector register	Sector register
IOFDAT	Data register	Data register

The FDIO chip can execute the following commands:

1. RESTORE to track Ø
2. SEEK track
3. STEP IN or OUT
4. READ TRACK ID
5. READ or WRITE SECTOR
6. READ or WRITE TRACK
7. FORCE INTERRUPT

Data separation for both single and double density is provided via a PLL (phase locked loop) circuit in order to achieve a high data recovery reliability. In addition, write pre-compensation is used for double density formats.

C. FDIO Data Request

There are two methods which can be used to synchronize data transfers to and from the FDIO chip.

1. Software can monitor the Data Request bit in the FDIO status register while in a wait loop and then transfer each byte via programmed I/O. This method generally proves unsatisfactory for most applications due to the high data transfer rate of the FDIO chip, especially for the double density mode. In addition, the CPU cannot be allowed to service interrupts without suffering some loss of disk data.

2. The FDIO-DRQ (Data Request) signal can be routed to the DMA chips READY input by selecting the FDIO-DRQ line via PIO port A. (See page 57.) The DMA is then programmed to handle the data transfer to or from the FDIO logic. Meanwhile, the CPU can wait for the completion of the data transfer and is free to perform other tasks as required by the particular application. This is the method used by the floppy disk routines in the ZRAID monitor program.

D. FDIO Interrupt

When the FDIO chip completes a command, the INTRQ line (Interrupt Request) goes active, (LOW). This signal is connected to bit 7 of PIO Port A (which may be programmed to generate an interrupt if INTRQ goes true.)

To enable the interrupt logic of port A and to select bit 7 for monitoring, the following instruction sequence could be used: (Assumes PIO A has previously been initialized to BIT mode.)

```
LD       A,vector               ; LOW HALF OF INT VECTOR
OUT     (IOPAC),A               ; SET INTERRUPT VECTOR
LD       A,                     ; ENABLE INTERRUPT
OUT     (IOPAC),A               ; SEND TO PIO A CONTROL
LD       A,7FH                  ; MASK FOR D7 (FDIO-INTRQ)
OUT     (IOPAC),A               ; SEND TO PIO A CONTROL
```

Note: ZRAID does not use this method. Instead, the FDIO status register is polled in a software wait loop until the BUSY bit indicates that the command has been completed. (Refer to the discussion of PIO on page 57)

E. Electrical adjustments

There are three pots which control data separation and write precompensation near the floppy disk connector, P6. These adjustments are factory set. We do not recommend that they be adjusted in the field unless it is certain that such adjustments are required. (Most disk errors can be traced to incorrect supply voltages, bad media, dirty heads, noise on power supply, drive mechanical failures or intermittent connections.

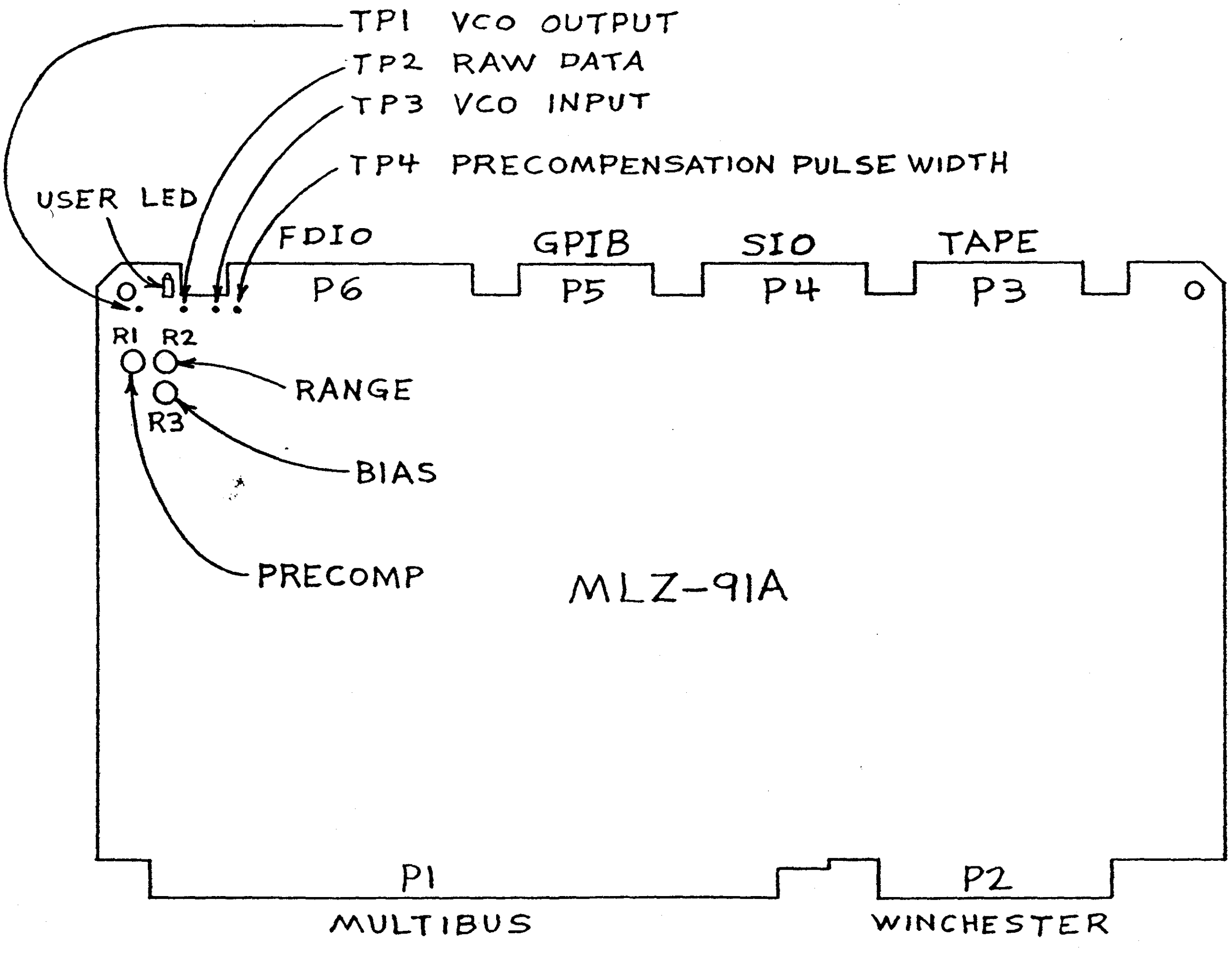
The adjustment procedure is as follows:

1. Verify that the Vcc power supplied to the MLZ-91 is 5.0 volts ± 0.1 volt.
2. Disconnect the drives from P6. Connect P6-46 (TP-2) to Vcc. (Usually, it is sufficient to simply leave P6-46 open as the terminating resistor netword will put P6-46 to the HIGH state.)

a) Adjust R3 (BIAS) for 1.4 volts at ± 5% at TP-3/

b) Adjust R2 (RANGE) for 4.0 MHZ ± 5% at TP-1.
(For 5-¼" drive configuration, adjust for 2.0 MHZ.)

3. Reconnect the drives to P6. (Disconnect any jumper to TP-2). Via a software routine, continuously write sectors. ZRAID has a command to perform this function. Enter control-W 12 (hex). ZRAID will write continuously until the drive door is opened or the next ZRAID command is entered. While writing, adjust R3 (PRECOMPENSATION) for 200 nsec. pulses (± 5%) at TP-4. These pulses will occur in busts approximately 175 msec. apart. Use a scratch disk formatted for double density, 1024 bytes per sector. (For writing single density use control-W 13 as a command to ZRAID. Be sure to use a scratch, single density diskette). If the diskette is defective or not properly formatted, no writes will occur.



FDIO ADJUSTMENTS

F. FDIO with 5¼" drives

The MLZ-91 is configured and adjusted at the factory for the standard 8" drives. To configure the board for use with 5¼" drives, set jumpers as follows:

Jumper	Position for 8"	Position for 5¼"
FC	8	5
FN	8	5
FI	8	5
FS	(remove)	5
FR	8	5
FP	(remove)	5

It may be necessary to perform the adjustments described in the previous section following any drive type reconfiguration.

G. Drives

Numerous drives may be used with the MLZ-91. The head step rate when using ZRAID FDIO routines is 10 msec, although this may be changed for fast seeking drives. It is usually necessary to change option jumpers on the drives from those shipped with the drives. As an example of proper drive set-up, the listing on the next page shows the correct jumper settings (on the drives) when using the popular Shugart drives with the MLZ-91.

H. Connector Pinout

See page 162.

I. Drive Power Supply

Be sure to select an adequate power supply for the drives. One note on Shugart drives. The 24V Return and +5 Return must be connected together at the supply. See Shugart manuals for details.

DRIVE	Jumpers to be installed on all drives	Install one each on one drive only (Drive Select)	Install on "LAST" drive only	Remove on all drives
SA-801 (8")	A, B, C, Z, DS, WP, 800 also RR, RI, R, I, S	DS1, DS2, DS3, DS4	T1, T3, T4, T5, T6	X, Y, T2, HL, D NP, DDS, DC, 801
SA-850/851 (8")	A, B, C, Z, DS, WP, 850 S2, AF, RS, FS, IT also RR, RI, R, I, M, S	DS1, DS2, DS3, DS4	RPACK 5E	NF, HI, HLL IW, RM D, 2S, DC X, Y, HL, NP, DL, 851 1B, 2B, 3B, 4B S1, S3, TS, F
SA-400 (5¼")	T2, HS MX (if only one drive)	DS1, DS2, DS3	T1, T3, T4, T5, F	HM MX (if more than one drive)
SA-410/460 (5¼")	MX (if only one drive) MM	DS1, DS2, DS3	RPACK	MX (if more than one drive) MS
Siemans FDD-200-8	RAD STEP 2 RR,RI, 18, 22, L, E, G	RAD SEL Ø, 1, 2, or 3	Terminator 8D	RAD STEP 1 K, J, V, H, F
Qume DT-8	S2, A, B, C, DS RR, R, RI, I, Y, DC 2S, WP	DS1, DS2, DS3, DS4	TM1, TM2	B1, B2, B3, B4 S1, S3, DDS X, W, HL, DL, D Z, HA, NP, T4Ø

FLOPPY DISK JUMPERS

User LED

This single LED is part of the FDIO option on the MLZ-91 and is separate from the 8-bit LED array. The function of this LED is left completely up to the user. It is located near P6, the floppy disk I/O connector.

The state of the LED is controlled by bit DØ of the floppy disk selection port, IOFSEL. Care must be taken when using this port to turn the LED on or off that the floppy disk select, density and side bits are not altered. (See page 124 for a full description of the control byte.)

DØ = Ø	LED ON
DØ = 1	LED OFF

The following subroutines may be used to control the LED without affecting the other bits. A ram value, LSAVE, is used to save the state of the other 7 bits of the byte. (LSAVE must be updated by the FDIO select logic, too.)

```
LEDON:   LD      B,Ø          ;CLEAR DØ (REG B)
         JR      USER
LEDOFF:  LD      B, 1         ;SET DØ ON (REG B)
         LD      A,(LDATA)    ;GET CURRENT IMAGE
         AND     ØFEH         ;MASK OTHER BITS
         OR      B            ;SET/CLEAR DØ
         LD       (LDATA),A   ;SAVE FOR NEXT PASS
         OUT      IOFSEL      ;SEND TO H/W
         RET
```

The MLZ-91 ZRAID monitor turns the LED on whenever the floppy disk routines are in use. The LED will flash on during disk I/O.

DMA

The DMA is a rather complicated chip and it helps to have some "hands on" experience with the chip in order to feel comfortable programming it.

The DMA chip is used primarily to do data transfers between memory and the FDIO logic. However, the DMA chip will allow any I/O device or memory address to be used both as source or destination ports. Thus, memory to memory or I/O device to I/O device transfers are possible.

When the DMA is active, all bus control lines are controlled by the DMA chip in the same fashion as if the CPU were conducting a memory or I/O operation. The "WAIT" signal, produced during external memory or I/O accesses, will synchronize the DMA to the external device data transfer rate. This means that the DMA is able to operate with external facilities (memory or I/O) without any special considerations.

Refer to the DMA manual for programming details. Some highlights appear on the following pages. If you anticipate doing anything fancier than shown here, we highly recommend that you get a DMA Technical Manual and call Zilog to discuss your particular application.

DMA Architecture

A block diagram of the Z80 DMA is shown in Figure 1. The internal structure consists of the following circuitry:

- *Bus Interface*: provides driver and receiver circuitry to interface to the Z80-CPU Bus.
- *Control Logic and Registers*: set the class, mode and other basic control parameters of the DMA.
- *Address, Byte Count and Pulse Circuitry*: generates the proper port addresses for the read and write operations, with provisions for incrementing or decrementing the address. When zero bytes remain to be handled, the byte count circuitry sets a flag in the status register. Pulse circuitry generates a pulse each time the byte counter lower 8-bits equal the pulse register.
- *Timing Circuitry:* allows the user to completely specify the read/write timing for each port.
- *Match Circuitry*: holds the match byte and a mask byte which allows for the comparison of only certain bits within the byte. If a match is encountered during a Search or Transfer, this circuitry sets a flag in the status register.
- *$\overline{\text{INT}}$ and $\overline{\text{BUSRQ}}$ Circuitry:* includes a control register which specifies the conditions under which the DMA can generate an interrupt; priority encoding logic to select between the generation of an $\overline{\text{INT}}$ or $\overline{\text{BUSRQ}}$ output under these conditions; and an interrupt vector register for automatic vectoring to the interrupt service routine.
- *Status Register*: holds current status of DMA.

Register Description

The following DMA-internal registers are available to the programmer:

Control Registers: Write only; 8 bits. Hold DMA control information: such as, when to initiate an interrupt or pulse, what mode or class of operation to perform, etc.

Timing Registers: Write only; 8 bits. Hold read/write timing parameters for the two ports.

Interrupt Vector Register: Read/write; 8 bits. Holds the 8-bit vector that the DMA will put onto the data bus after receiving an $\overline{\text{IORQ}}$ during an interrupt acknowledge sequence if it is the highest priority device requesting an interrupt. (This register is readable only during interrupt acknowledge cycles.)

Block Length Register: Write only; 16 bits. Contains total block length of data to be searched and/or transferred.

Byte Counter: Read only; 16 bits. Counts number of bytes transferred (or searched). On a Load or Continue the Byte Counter is reset to zero. Thereafter, each byte transfer operation increments it until it matches the contents of the Block Length Register, at which time End of Block is set in the status register and operation is suspended if programmed. Also if so programmed the DMA will generate an interrupt.

Match Register: Write only; 8 bits. Holds the byte for which a match is being sought in Search operations.

Mask Register: Write only; 8 bits. Holds the 8-bit mask to determine which bits in the match register are to be examined for a match.

Starting Address Registers (Port A and Port B): Write only; 16 bits each. Hold the starting addresses (upper and lower 8 bits) for the two ports involved in Transfer operations. In Search Only operations, only one port address would have to be specified. Only memory starting addresses require both upper and lower 8 bits; I/O ports are generally addressed with only the lower 8 bits, and in this case the address contained in the register is a generally fixed address.

Address Counters (Port A and Port B): Read only; 16 bits each. These counters are loaded with the contents of the corresponding Starting Address Registers whenever Searches or Transfers are initiated with a Load or Continue. They are incremented, decremented or remain fixed, as programmed.

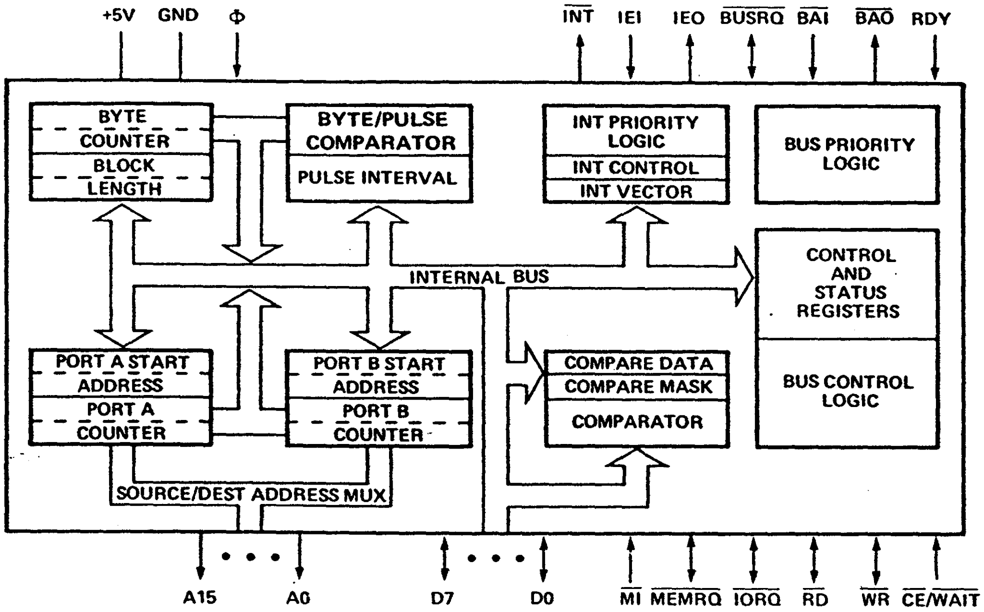
Pulse Control Register: Write only; 8 bits. The content of this register is continuously compared with the lower eight bits of the byte counter. When they become equal, the $\overline{\text{INT}}$ output is activated. Since this occurs while $\overline{\text{BUSRQ}}$ and $\overline{\text{BUSAK}}$ are both active, the CPU does not interpret this as an interrupt request. Instead, the signal is used to communicate with a peripheral I/O device. When the Pulse Control Register contains a value n, the first pulse is generated after n + 1 bytes of search or transfer. The next and all subsequent pulses occur at 256-byte intervals.

Status Register: Kead only; 8 bits. Match, End of Block, Ready Active, Interrupt Pending, and DMA Cycle Occurred bits indicate these functions when set.

Modes of Operation

The DMA may be programmed for one of four modes of operation. (See Command Register 2B.)

- *Byte at a time*: control is returned to the CPU after each one-byte cycle.
- *Burst*: operation continues as long as the DMA's RDY input is active, indicating that the relevant port is ready. Control returns to the CPU when RDY is inactive or at end of block or a match if so programmed.
- *Continuous*: the entire Search and/or Transfer of a block of data is completed before control is returned to CPU.



DMA Internal Block Diagram

Fig.1

Reading the DMA Internal Registers

The CPU can read seven internal DMA registers, always in the following order: Status, lower byte of the Block Length register, upper byte of the Block Length register, lower byte of the Port A Address, upper byte of the Port A Address, lower byte of the Port B Address and the upper byte of the Port B Address.

The Read Mask register must be programmed to either include or exclude any of these seven registers by programming a 1 (include) or 0 (exclude) in the appropriate positions of the Read Mask register. After a Reset or Load, the read sequence must be initiated through an Initiate Read Sequence command (Command Byte 2D). The sequence of reading all registers that are not excluded by the Read Mask register must be completed before a new Initiate Read Sequence or RD Status command.

Programming the DMA

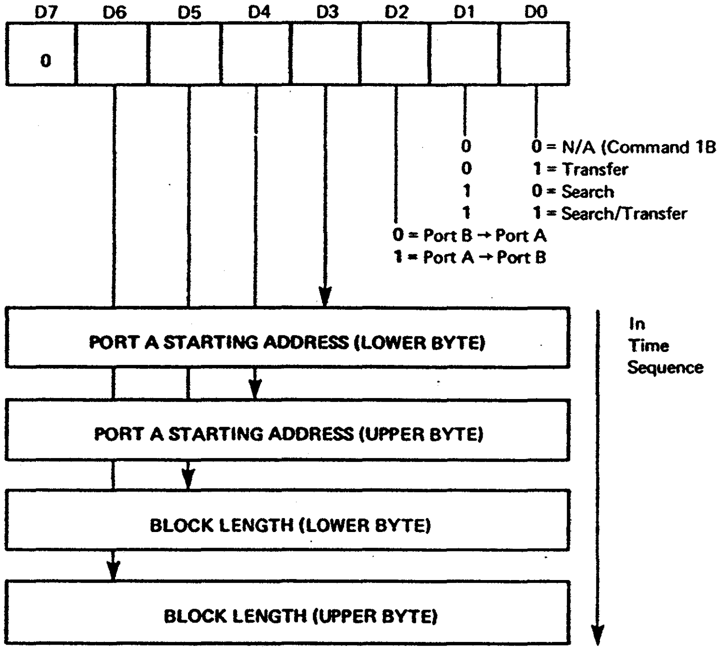
Previous sections of this specification have indicated the various functions and modes of the DMA. The diagrams and charts below show how the DMA is programmed to select among these functions and modes and to adapt itself to the requirements of the user system.

The Z80-DMA chip may be in an "enable" state, in which it can gain control of the system buses and direct the transfer of data between its ports, or in a "disable" state, when it cannot gain control of the bus. Program commands can be written to it in either state, but writing a command to it automatically puts it in the disable state, which is maintained until an enable command is issued to the DMA. The CPU must program it in advance of any data search or transfer by addressing it as an I/O port and sending it a sequence of command bytes via the system data bus using Output instructions. When the DMA is powered up or reset by any means, the DMA is automatically placed into a disable state, in which it can initiate neither bus requests nor data transfers nor interrupts.

The command bytes contain information to be loaded into the DMA's control and other registers and/or information to alter the state of the chip, such as an Enable Interrupt command. The command structure is designed so that certain bits in some commands can be set to alert the DMA to expect the next byte written to it to be for a particular internal register.

The following diagrams and charts give the function of each bit in the six different command bytes. Two of these are defined as being from Group 1, and are termed command bytes 1A and 1B. These Group 1 commands contain the most basic DMA set-up information. The other four are categorized as Group 2, and are termed commands 2A, 2B, 2C and 2D. Group 2 words specify more detailed set-up information.

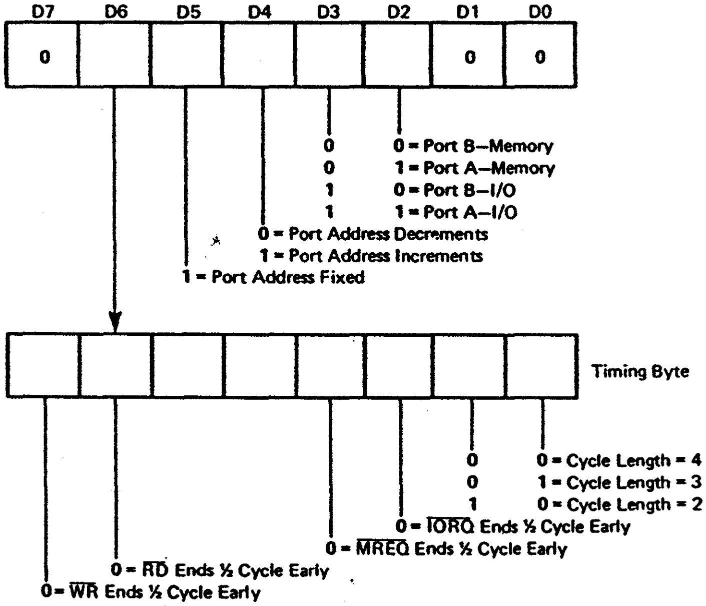
Command Register 1A



A "1" in positions D3 through D6 means that the indicated byte will follow. Note that the sequence of bytes is absolutely rigid.

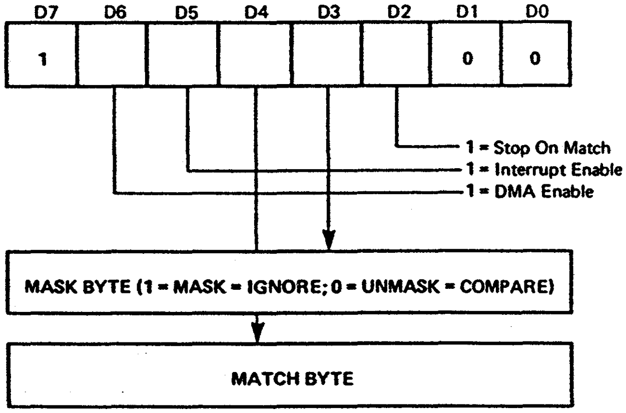
The DMA always transfers or searches one byte more than the number written into the Block Length registers. A "0" in the block length register results in the transfer or search of 2^{16} + 1 bytes. The shortest programmable block length is therefore two bytes long, programmed by writing a 1 into the Block Length register.

Command Register 1B

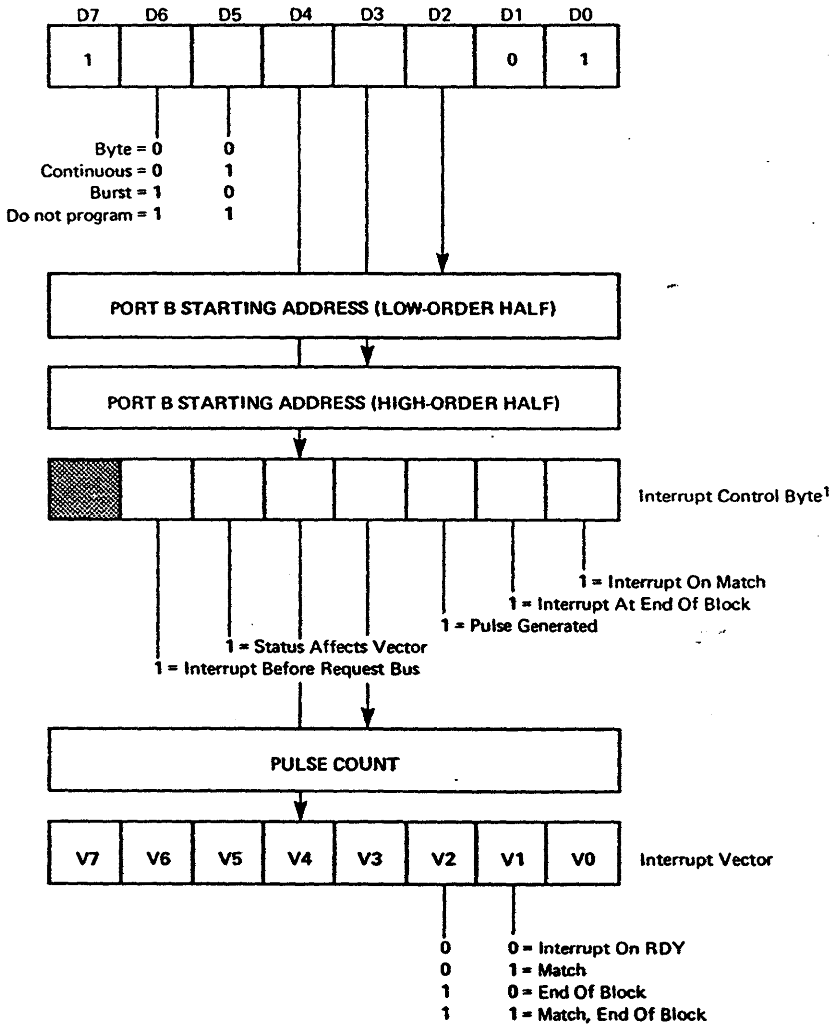

D7 D6 D5 D4 D3 D2 D1 D0
0 0 0
0 0 = Port B—Memory
0 1 = Port A—Memory
1 0 = Port B—I/O
1 1 = Port A—I/O
0 = Port Address Decrements
1 = Port Address Increments
1 = Port Address Fixed
Timing Byte
0 0 = Cycle Length = 4
0 1 = Cycle Length = 3
1 0 = Cycle Length = 2
0 = $\overline{IORQ}$ Ends ½ Cycle Early
0 = $\overline{MREQ}$ Ends ½ Cycle Early
0 = $\overline{RD}$ Ends ½ Cycle Early
0 = $\overline{WR}$ Ends ½ Cycle Early

For transfers, this byte is normally written twice, once for Port A and again for Port B.

Command Register 2A



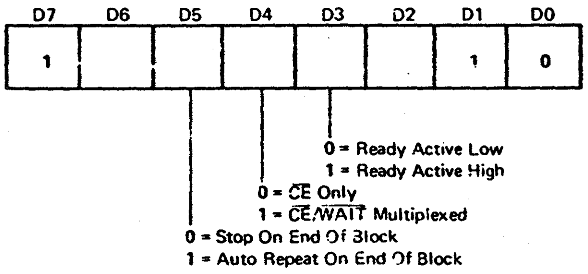
Command Register 2B


D7
D6
D5
D4
D3
D2
D1
D0
1
0
1
Byte = 0 0
Continuous = 0 1
Burst = 1 0
Do not program = 1 1
PORT B STARTING ADDRESS (LOW-ORDER HALF)
PORT B STARTING ADDRESS (HIGH-ORDER HALF)
Interrupt Control Byte[1]
1 = Interrupt On Match
1 = Interrupt At End Of Block
1 = Pulse Generated
1 = Status Affects Vector
1 = Interrupt Before Request Bus
PULSE COUNT
V7
V6
V5
V4
V3
V2
V1
V0
Interrupt Vector
0 0 = Interrupt On RDY
0 1 = Match
1 0 = End Of Block
1 1 = Match, End Of Block

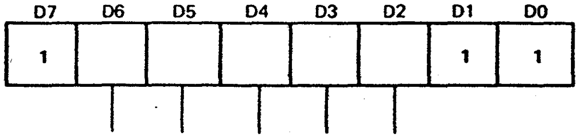
[1] If "Interrupt Before Requesting Bus" is selected (by a 1 in bit 6 of the Interrupt Control byte), the Z-80 DMA does not request the bus until the following set of instructions has been received by the Z-80 DMA:

- Enable after RETI command (B7 in Command byte 2D)
- Enable DMA command (87 in Command byte 2D)
- A RETI instruction that resets the IUS (Interrupt Under Service latch) in the Z-80 DMA

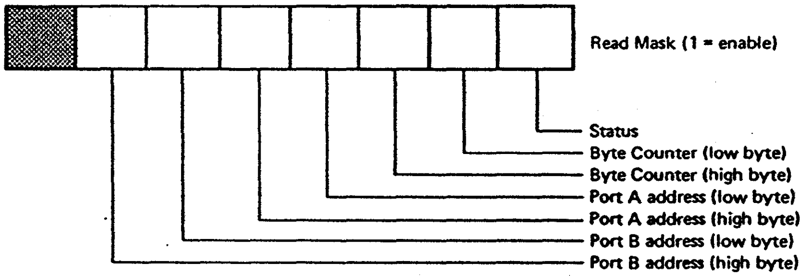
Command Register 2C



Command Register 2D



HEX	D6	D5	D4	D3	D2
C3	1	0	0	0	0 = Reset Interrupt circuitry, disable interrupt and bus request logic, unforce internal ready condition, disable "MUXCE" and stop auto repeat.
C7	1	0	0	0	1 = Reset Port A Timing to standard Z-80 CPU timing.
CB	1	0	0	1	0 = Reset Port B Timing to standard Z-80 CPU timing.
CF	1	0	0	1	1 = Load starting address for both ports, clear byte counter.*
D3	1	0	1	0	0 = Addresses continue from present locations, clear byte counter.
AB	0	1	0	1	0 = Enable interrupts
AF	0	1	0	1	1 = Disable interrupts
A3	0	1	0	0	0 = Reset and disable interrupt circuits (like RETI) and unforce the internal ready condition
87	0	0	0	0	1 = Enable DMA ⎤ Both affect all operations except interrupts, but do not
83	0	0	0	0	0 = Disable DMA ⎦ reset any functions.
A7	0	1	0	0	1 = Initiate read sequence to the first register designated as readable by the Read Mask register.
BF	0	1	1	1	1 = Set read status so next read is from status register.
B3	0	1	1	0	0 = Force an internal ready condition independent of the RDY input. Used for memory-to-memory operations where no RDY signal is needed. This command does not function in the "byte-at-a-time" mode.
8B	0	0	0	1	0 = Clear Match and End of Block status bits.
B7	0	1	1	0	1 = Enable after RETI so DMA will request bus only after receiving a RETI. Must be followed by an Enable DMA command.
BB	0	1	1	1	0 = Read mask is the following byte.


Read Mask (1 = enable)
Status
Byte Counter (low byte)
Byte Counter (high byte)
Port A address (low byte)
Port A address (high byte)
Port B address (low byte)
Port B address (high byte)

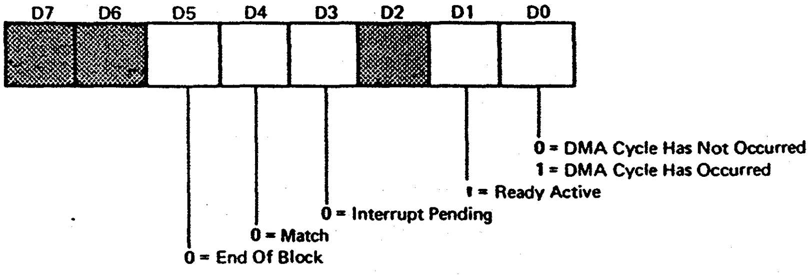
* Loading Port Addresses. The "Load" command (CF in Command Register 2D) loads a fixed address only into a port selected as the source, not into a port selected as the destination. Therefore, the destination address must be loaded by temporarily mislabeling the destination as the source.

The following example is a set-up procedure for a transfer from Port A to Port B:

1. Command byte 1A with B as source port
2. Command byte 2D with CF = load
3. Command byte 1A with A as source port
4. Command byte 2D with CF = load
5. Command byte 2D with 87 = Enable DMA

This manipulation is required only when the destination has a fixed address.

Status Register



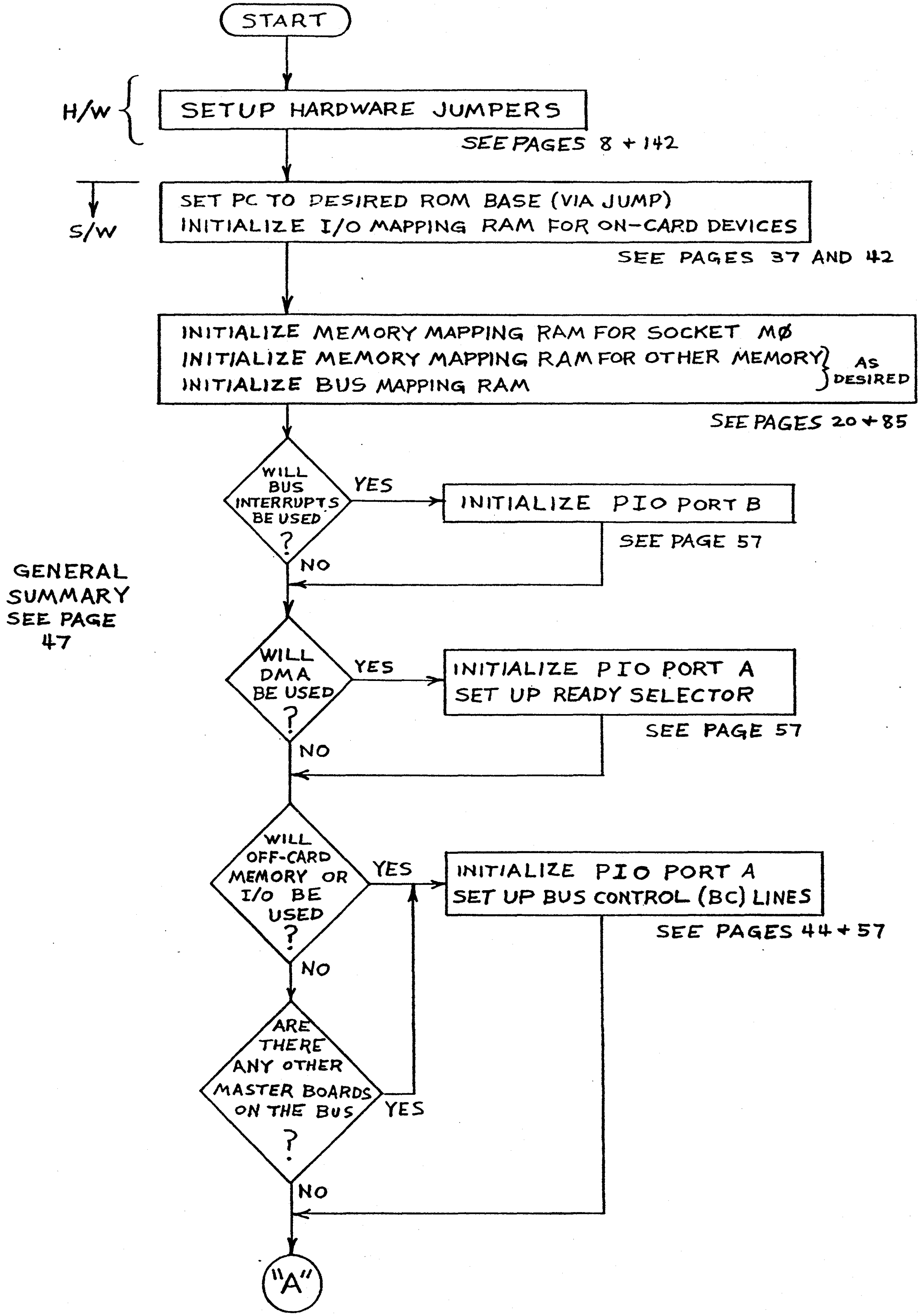
The Sample DMA Program shows how the DMA may be programmed to transfer data from memory (Port A) to a peripheral device (Port B). In this example, the Port A memory starting address is 1050H and the Port B peripheral fixed address is 05H. Note that the data flow is 1001H bytes—one more than specified by the block length. The table of DMA commands may be stored in consecutive memory locations and transferred to the DMA with an output instruction such as OTIR.

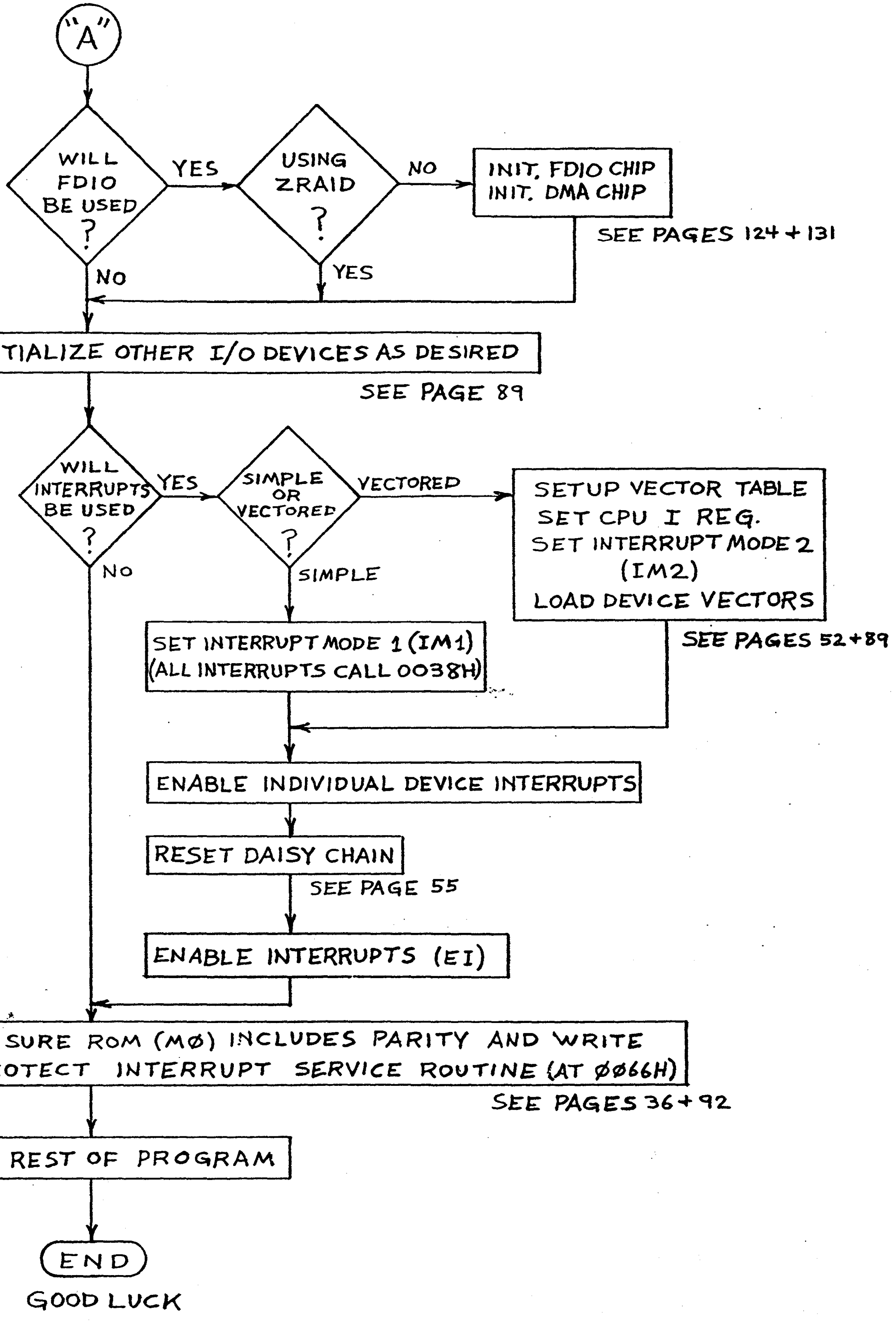
Sample DMA Program

	D7	D6	D5	D4	D3	D2	D1	D0	HEX
1) Command Register 1A sets DMA to receive block length, Port A starting address and temporarily sets Port B as source.	0 Group One	1 Block Length Upper Follows	1 Block Length Lower Follows	1 Port A Upper Addr Follows	1 Port A Lower Addr Follows	0 B→A Temporary For Loading B Address	0 Command Byte 1A Transfer, No Search	1	79
2) Port A address (lower)	0	1	0	1	0	0	0	0	50
3) Port A address (upper)	0	0	0	1	0	0	0	0	10
4) Block length (lower)	0	0	0	0	0	0	0	0	00
5) Block length (upper)	0	0	0	1	0	0	0	0	10
6) Command Register 1B defines Port A as memory with incrementing address.	0 Group One	0 No Timing Follows	0 Address Changes	1 Address Increments	0 Port Is Memory	1 This Is Port A	0 Byte 1B	0	14
7) Command Register 1B defines Port B as peripheral with fixed address.	0 Group One	0 No Timing Follows	1 Fixed Address	0 Not Used	1 Port Is I/O	0 This Is Port B	0 Byte 1B	0	28
8) Command Register 2B sets mode to Burst, sets DMA to expect Port B address.	1 Group Two	1 Burst Mode	0	0 No Interrupt Control Byte Follows	0 No Upper Address	1 Port B Lower Addr Follows	0 Byte 2B	1	C5
9) Port B address (lower)	0	0	0	0	0	1	0	1	05
10) Command Register 2C sets Ready active High.	1 Group Two	0 Not Used	0 No Auto Restart	0 No Wait States	1 RDY Active HIGH	0 Not Used	1 Byte 2C	0	8A
11) Command Register 2D loads Port B address and resets block counter.	1 Group Two	1	0	0 Load	1	1	1 Byte 2D	1	CF
12) Command Register 1A sets Port A as source. *	0 Group One	0	0	0 No Addr Or Block Length Bytes	0	1 A→B	0 Byte 1A, Transfer No Search	1	05
13) Command Register 2D loads Port A address and resets block counter. *	1 Group Two	1	0	0 Load	1	1	1 Byte 2D	1	CF
14) Command byte 2D enables DMA to start operation.	1 Group Two	0	0	0 Enable DMA		1	1 Byte 2D	1	87

NOTE: The actual number of bytes transferred is one more than specified by the block length.
* These commands are necessary only in the case of a fixed destination address.

```
; *********************************************************
;THIS IS AN EXAMPLE OF THE METHOD USED TO INITIALIZE THE
;ZILOG DMA CHIP. THIS CODE IS USED BY ZRAID FOR FLOPPY DISK
;DATA TRANSFERS.  NOTE:  CONSULT COMPLETE ZRAID LISTINGS FOR
;THE ACTUAL ADDRESSES.
; *********************************************************
FDDMA:  LD      HL,TABLE        ;TABLE ADRS
        LD      DE,RAM          ;TEMP RAM LOCATION
        LD      BC,LENGTH       ;LENGTH
        LDIR                    ;TRANSFER TABLE TO RAM
;                               ;(SO IT CAN BE CHANGED)
;
        LD      (RAM+12),A      ;SET RD/WR COMMAND
        LD      HL,(DMAADRS)    ;DATA ADRS
        LD      (RAM+9),HL      ;SET ADRS OF DATA
;
        LD      A,(DENSITY)     ;GET SD/DD FLAG
        CP      SINGLE          ;TEST FOR SINGLE
        JR      Z,FDDMA1        ;LEAVE LENGTH AS IS
        LD      HL,1024-1       ;NEW LENGTH - 1
        LD      (RAM+3),HL      ;SET NEW LENGTH FOR DD
;
FDDMA1: LD      HL,RAM          ;SOURCE
        LD      BC,IODMA+LENGTH*256     ;PORT & LENGTH
        OTIR                    ;SEND TABLE TO DMA
        RET
;
; *********************************************************
TABLE:  DB      0303Q           ;RESET DMA
        DB      0155Q           ;RD/WR CONTROL (SET FOR RD)
        DB      IOFDAT          ;FDIO DATA PORT ADRS
        DB      0177Q,0         ;LENGTH (DEFAULT FOR S-DENS)
        DB      054Q,020Q,0332Q,0215Q   ;DEFINE PORTS & RDY
        DB      0,0             ;FOR DMA DATA ADRS
        DB      0317Q           ;LOAD DATA
        DB      0               ;FOR FINAL RD/WR CONTROL
        DB      0317Q,0207Q     ;LOAD DATA (AGAIN), ENABLE
LENGTH EQU      $-TABLE
;
;THE DOUBLE LOAD DATA COMMANDS ABOVE ARE REQUIRED
;IF THE DESTINATION ADDRESS OF THE DMA IS "FIXED", THE LOAD
;COMMAND DOESN'T ALWAYS LOAD ALL THE PARAMETERS.  THEREFORE,
;THE DMA IS FIRST SETUP WITH THE DESTINATION AS "VARIABLE"
;(IN THIS CASE A DISK READ HAS A DESTINATION WHICH IS RAM
;AND IS A VARIABLE SOURCE 'PORT' ADRS WITH RESPECT TO THE
;DMA).  THEN, THE ACTUAL DMA DIRECTION IS SET (AT TABLE +12)
;AND ANOTHER LOAD COMMAND IS EXECUTED.
;(SEE NEXT PAGE FOR EXPL OF DMA DATA.)
; *********************************************************
; ***********************************************************************
;THE DMA INITIALIZATION TABLE IS EXPANDED AND DETAILED BELOW:
;
;HEX    BINARY/BIT DEFINITION
;
;C3     RESET DMA
;
;6D     0110 1101
;       -      --TRANSFER
;             PORT A -> B
;            A ADRS FOLLOWS (L)
;        --B LENGTH (L,H) FOLLOWS
;
;XX     PORT A ADRS (FDIO DATA REGISTER)
;
;LL     LENGTH (-1) LOW HALF
;HH     LENGTH HIGH HALF
;
;2C     0010 1100
;       -     ---PORT A DEFINITION
;            PORT IS I/O DEVICE
;         ADRS IS FIXED
;
;10     0001 0000
;       -     ---PORT B DEFINITION
;            PORT IS MEMORY
;         --ADRS INCREMENTS
;
;DA     1101 1010
;       -      --RDY/WAIT SPECIFICATIONS
;            READY IS ACTIVE HIGH
;          CE/WAIT MULTIPLEXED
;         STOP AT END OF TRANSFER
;
;8D     1000 1101
;       -      --MODE CONTROL
;            --PORT B STARTING ADRS (L,H) FOLLOWS
;        --BYTE-AT-A-TIME MODE
;
;LL     PORT B ADRS L (DMA MEMORY ADRS)
;HH     PORT B ADRS H
;
;CF     LOAD DMA WORKING REGISTERS.  THIS COMMAND ALWAYS WORKS BECAUSE
;               THE DESTINATION PORT (B, MEMORY) IS NOT A "FIXED" ADRS.
;               THE DMA LOAD COMMAND DOES NOT REALLY LOAD ALL THE WORKING
;               REGISTERS IF THE DESTINATION PORT IS A FIXED I/O PORT ADRS.
;
;XX     WILL BE EITHER 01H OR 05H TO CONTROL RD/WR DIRECTION
;       01H:    0000 0001
;               -      --TRANSFER
;                     PORT B IS SOURCE (FDIO WR)
;       05H:    0000 0101
;               -      --TRANSFER
;                     PORT A IS SOURCE (FDIO RD)
;
;CF     ANOTHER LOAD COMMAND IN CASE FINAL DIRECTION IS PORT B -> PORT A.
;
;87     ENABLE DMA
; ***********************************************************************
```


START
H/W
SETUP HARDWARE JUMPERS
SEE PAGES 8 + 142
S/W
SET PC TO DESIRED ROM BASE (VIA JUMP)
INITIALIZE I/O MAPPING RAM FOR ON-CARD DEVICES
SEE PAGES 37 AND 42
INITIALIZE MEMORY MAPPING RAM FOR SOCKET MØ
INITIALIZE MEMORY MAPPING RAM FOR OTHER MEMORY
INITIALIZE BUS MAPPING RAM
AS DESIRED
SEE PAGES 20 + 85
WILL BUS INTERRUPTS BE USED ?
YES
NO
INITIALIZE PIO PORT B
SEE PAGE 57
GENERAL SUMMARY SEE PAGE 47
WILL DMA BE USED ?
YES
NO
INITIALIZE PIO PORT A
SET UP READY SELECTOR
SEE PAGE 57
WILL OFF-CARD MEMORY OR I/O BE USED ?
YES
NO
INITIALIZE PIO PORT A
SET UP BUS CONTROL (BC) LINES
SEE PAGES 44 + 57
ARE THERE ANY OTHER MASTER BOARDS ON THE BUS ?
YES
NO
"A"



USER CHECKLIST

SEE INDIVIDUAL SECTIONS OF THIS MANUAL FOR DETAILS ON EACH ITEM.

HARDWARE JUMPERS

A minimum of hardware jumpers are used on the MLZ-91 (most options are under software control). These jumpers will usually be installed once and not changed unless the MLZ-91 is used in a different system.

Each jumper group is assigned a name. A letter designation is used within each group to indicate the location of a shorting pin. For example, "J6-A" means install a shorting pin between the two posts on either side of "A" at location "J6".

Name	Function	Options	
J1	CPU Clock	J1-A	2MHz
	(one required)	J1-B	4MHz
J2	APU Clock	J2-A	2MHz
	(one required for APU option)	J2-B	4MHz
J3	SIO Port A Interface		
	Install for RS-232 or RS423 I/F on SIO Port A		
J4	SIO Port B Interface		
	Install for RS-232 or RS423 I/F on SIO Port B		
J5	SIO Port A Receive Data Clock	J5-A	P4-8 (D pin 17)
	(one required)	J5-B	J6 output
J6	SIO Port B Transmit Data Clock	J6-A	P4-4 (D pin 15)
	(one required)	J6-B	Baud Gen. A
J7	Upper Address Enable		
	Install to enable use of Multibus lines A19, A18, A17 & A16		
J8	Processor Priority		
	Install on highest priority board in Multibus		
J9	Wait State Logic - Memory Type	J9-A	ROM sockets only
	(one or none required)	J9-B	All on-card memory
		none	No WAIT states
J10	Wait State Logic - Cycles	J10-A	All cycles
	(one required)	J10-B	Opcode fetch only
J11	Winchester configuration		
	Install for Shugart controller		
J12	ROM type select	[✓ J12-A]	2732/2764
	(one required) - See also J14	J12-B	2716
J13	RAM type select	J13-A	4532-1
	(one, two or none required)	J13-A,C	4532-2/4164
		J13-B	4532-3
		J13-B,D	4532-4
		nohe	4116

J14	ROM type select (one required) - See also J12	[✓ J14-A] J14-B	2732/2764 2716
J15	Winchester configuration	J15-M J15-S J15-W None	Micropolis Shugart WD1000 Priam
J16	Winchester configuration	J16-M J16-S J16-W None	Micropolis Shugart WD1000 Priam
J17	Winchester configuration	J17-M J17-S None	Micropolis Shugart Others
J18	Winchester configuration	J18-M J18-S None	Micropolis Shugart Others
J19	Disable Parity Logic Install to disable on-card RAM parity logic		
J20	Winchester configuration	J20-W J20-P	WD1000 Others
J21,22,23,24	Reserved for MLZ-92		
J25	Winchester configuration	J25-S J25-W J25-X	Shugart WD1000 Others
J26	XACK response time	J26-A J26-B	Fast Slow
J27,28,29	J27-B, J28-B, J29	J27-B, J28-B, J29 J27-A, J28-A, J28-C No J29	Normal Port A Tx Clock

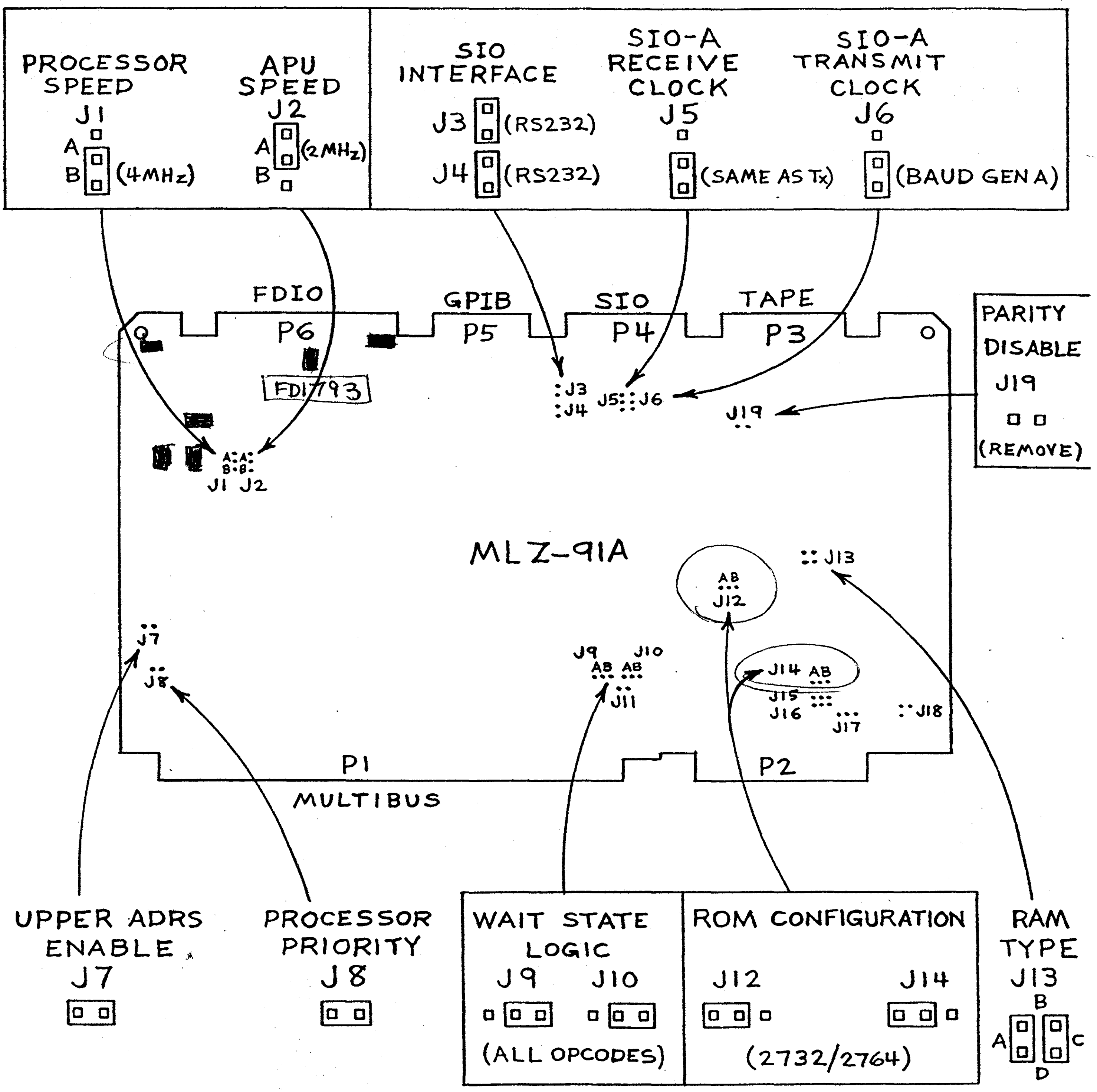
Other Jumpers

Ja Open if RS232 required on SIO port A and SIP A soldered in.

Jb Open if SIP B is soldered in place and RS232/423 I/F required on SIO port B.

FC, FN, FI, FS, FR Floppy disk drive jumpers (8" or 5¼").
FP See page 128.

See page 144 for jumper locations.



NOTE: J11, J15, J16, J17, & J18 SET FOR "S" OR "M" (SEE PAGE 70)
JUMPERS FC, FN, FI, FS (FOR FLOPPY DISK DRIVE TYPE) NOT SHOWN
FR, FP

HARDWARE JUMPER LOCATIONS

Signal Definitions (P1)

All signals are active HIGH or active LOW as specified in the table below. A minus sign (-) following a signal name also indicates an active LOW signal. The listing is alphabetical by name and covers those signals appearing on connectors P1 (Main System Bus).

Signal Name	Pin #	Active State	In/Out Bidirectional	Description
A0-	P1-57	LOW	OUT	Address Bus bit 0 (LSB)
A1-	P1-58	LOW	OUT	Address Bus bit 1
A2-	P1-55	LOW	OUT	Address Bus bit 2
A3-	P1-56	LOW	OUT	Address Bus bit 3
A4-	P1-53	LOW	OUT	Address Bus bit 4
A5-	P1-54	LOW	OUT	Address Bus bit 5
A6-	P1-51	LOW	OUT	Address Bus bit 6
A7-	P1-52	LOW	OUT	Address Bus bit 7
A8-	P1-49	LOW	OUT	Address Bus bit 8
A9-	P1-50	LOW	OUT	Address Bus bit 9
a10-	P1-47	LOW	OUT	Address Bus bit 10
A11-	P1-48	LOW	OUT	Address Bus bit 11
A12-	P1-45	LOW	OUT	Address Bus bit 12
A13-	P1-46	LOW	OUT	Address Bus bit 13
A14-	P1-43	LOW	OUT	Address Bus bit 14
A15-	P1-44	LOW	OUT	Address Bus bit 15
A16-	P1-28	LOW	OUT	Address Bus bit 16
A17-	P1-30	LOW	OUT	Address Bus bit 17
A18-	P1-32	LOW	OUT	Address Bus bit 18
A19-	P1-34	LOW	OUT	Address Bus bit 19 (MSB)
BACK-	P1-23	LOW	IN	Bus Acknowledge. Used by external memory or I/O devices to acknowledge a read or write request.
BAI-	P1-15	LOW	IN	Bus Available In. Indicates that the System Bus is idle and there are no higher priority processors requesting use of the Bus. This signal forms a daisy chain when connected to the next higher priority BAO-.
BAO-	P1-16	LOW	OUT	Bus Available Out. Indicates that the System Bus is idle, there are no higher priority processors requesting the Bus, and that the MLZ-91 does not require use of the System Bus. BAO-goes low when the MLZ-91 desires use of the Bus and the Bus is idle. This signal forms a daisy chain with another processor's BAI-.

Signal Name	Pin #	Active State	In/Out Bidirectional	Description
BBUSY-	P1-17	LOW	BI	Indicates that the System Bus is in use. Inhibits any other processor from requesting use of the Bus.
BCLK-	P1-13	--	IN or OUT	Bus Clock. Used to synchronize BBUSY- and BAO-. Generated by the highest priority board. 8 MHZ.
BRQST-	P1-18	LOW	OUT	Bus Request. Goes LOW whenever the MLZ-91 requires the System Bus for an external memory or I/O operation. Used to implement an external bus priority network.
CBREQ-	P1-29	LOW	BI	Common Bus Request. Pulled low whenever any processor requires use of the system bus.
CC-	P1-31	--	OUT	Constant Clock. Generates an 8 MHz clock signal to allow external device synchronization where necessary. Active only on the highest priority card.
D0-	P1-73	LOW	BI	Data Bus bit 0 (LSB)
D1-	P1-74	LOW	BI	Data Bus bit 1
D2-	P1-71	LOW	BI	Data Bus bit 2
D3-	P1-72	LOW	BI	Data Bus bit 3
D4-	P1-69	LOW	BI	Data Bus bit 4
D5-	P1-70	LOW	BI	Data Bus bit 5
D6-	P1-67	LOW	BI	Data Bus bit 6
D7-	P1-68	LOW	BI	Data Bus bit 7 (MSB)
INT0-	P1-41	LOW	BI*	Bus Interrupt 0, Port B3, bit 0
INT1-	P1-42	LOW	BI*	Bus Interrupt 1, Port B3, bit 1
INT2-	P1-39	LOW	BI*	Bus Interrupt 2, Port B3, bit 2
INT3-	P1-40	LOW	BI*	Bus Interrupt 3, Port B3, bit 3
INT4-	P1-37	LOW	BI*	Bus Interrupt 4, Port B3, bit 4
INT5-	P1-38	LOW	BI*	Bus Interrupt 5, Port B3, bit 5
INT6-	P1-35	LOW	BI*	Bus Interrupt 6, Port B3, bit 6
INT7-	P1-36	LOW	BI*	Bus Interrupt 7, Port B3, bit 7

* The eight System Bus interrupt lines (INT0- through INT7-) may be treated as inputs or as outputs by using the proper port specification to PIO port B .

Signal Name	Pin #	Active State	In/Out Bidirectional	Description
IORD-	P1-21	LOW	OUT	I/O Read Request. Indicates that the address of an I/O device is on the System Address Bus (A0 through A7) and that the device should place data on the System Data Bus.
IOWR-	P1-22	LOW	OUT	I/O Write Request. Indicates that the address of an I/O device is on the System Address Bus and the data on the System Data Bus is valid for an I/O write.
MEMRD-	P1-19	LOW	OUT	Memory Read Request. Indicates that the System Address Bus has a stable memory address and that the data should be placed on the System Data Bus.
MEMWR-	P1-20	LOW	OUT	Memory Write Request. Indicates that the System Address Bus has a stable memory address and that the data on the System Data Bus should be written into the addressed memory.
RESET-	P1-14	LOW	BI	System Reset. May be used as an input or output.

WINCHESTER I/O CONNECTOR (P2)

The pinout of P2 is arranged for easy connection of the MLZ-91 to any of the following Winchester controllers: (Only one may be connected at a time)

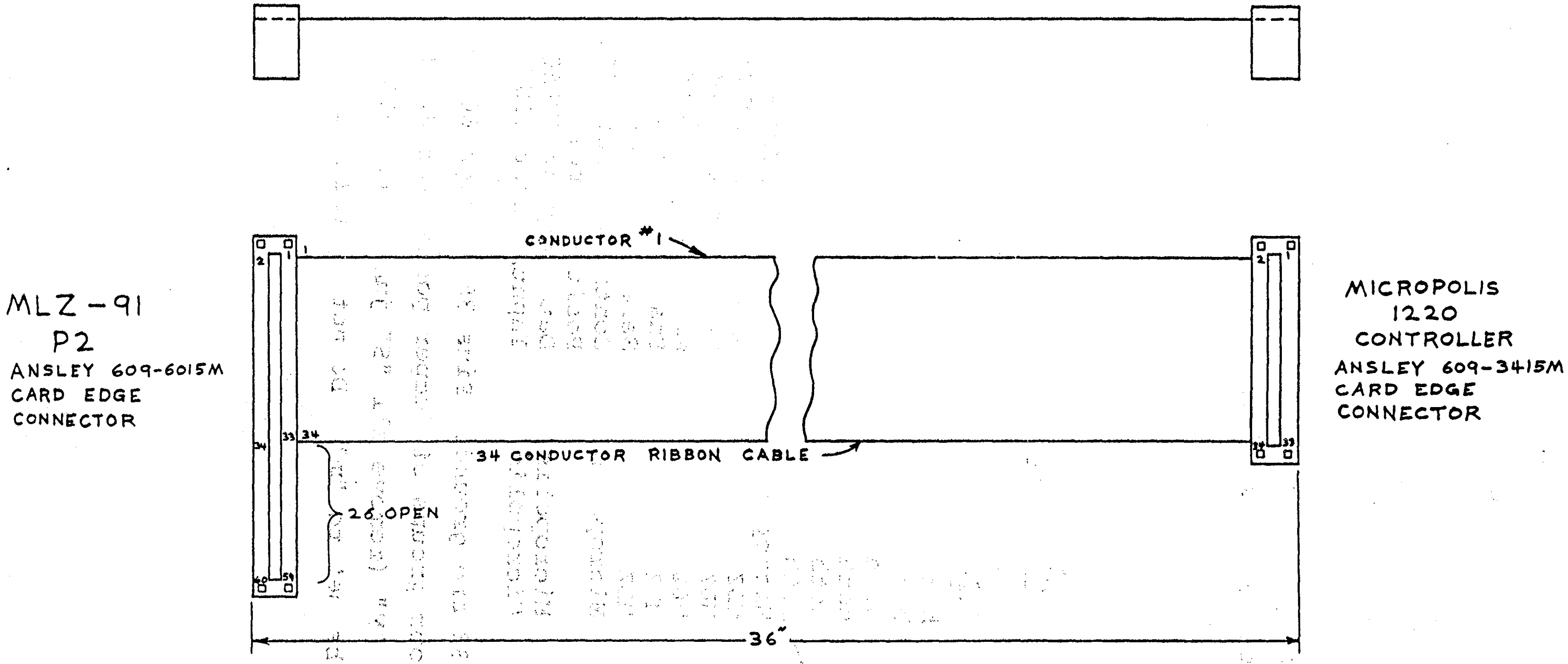
1. Priam "SMART" Interface
2. Micropolis 1220 controller
3. Shugart 1403D series
4. Seagate Technology
5. DTC

P2 is a 60 pin connector on the Multibus edge of the MLZ-91. Pins 1 through 34 are used for Micropolis and Shugart while pins 35 through 60 are used for Priam.

MICROPOLIS

P2 Pin #	Micropolis Pin #	Source MLZ/Micropolis	Function (Negative True)
2	2	both	Data bit 7 (MSB)
4	4	both	Data bit 6
6	6	both	Data bit 5
8	8	both	Data bit 4
10	10	both	Data bit 3
12	12	both	Data bit 2
14	14	both	Data bit 1
16	16	both	Data bit Ø (LSB)
18	18	Micropolis	Attention (ATTN-)
20	20	MLZ	Data/Control (DATA-)
22	22	MLZ	Read Strobe (RSTB-)
24	24	MLZ	Write Strobe (WSTB-)
26	26	MLZ	Enable (ENABLE-)
28	28	MLZ	Select (MSEL-)
30	30	Micropolis	Controller Busy (CBUSY) Positive True
32	32	Micropolis	Data Request (DREQ-)
34	34	Micropolis	Input/Output (OUT-)

P2 odd pins 1 through 33 are ground. Pins 35 through 60 are not used. There are four groups of jumper posts near P2 which must be configured for "M" (remove all "S" jumpers). J15, J16, J17 and J18 must be set to "m". Do not install J11.



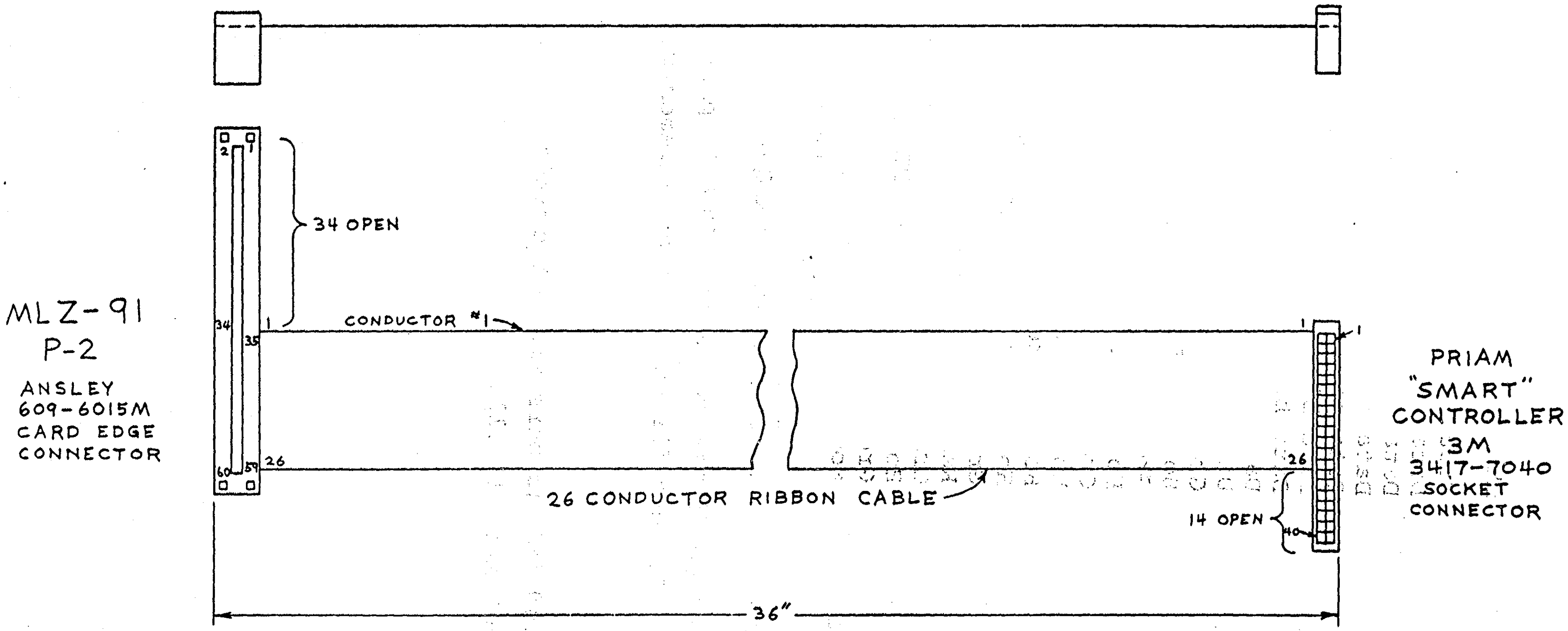
HEURIKON CORP. MADISON, WISCONSIN COPYRIGHT 1981	CABLE MLZ-P2M FOR MLZ-91	CHECKED: JM	DATE 2-81
		DRAWN: NGK	

PRIAM

P2 Pin #	Priam Pin #	Source MLZ/Priam	Function
35	1		Ground
36	2	both	Data bit Ø (Positive True)
37	3	both	Data bit 1
38	4	both	Data bit 2
39	5	both	Data bit 3
40	6	both	Data bit 4
41	7	both	Data bit 5
42	8	both	Data bit 6
43	9	both	Data bit 7 (MSB)
44	10		Ground
45	11	MLZ	Host Read (RSTB-)
46	12		Ground
47	13	MLZ	Host Write (WSTB-)
48	14		Ground
49	15	MLZ	Host Address 2 (HAD2)
50	16	MLZ	Host Address 1 (HAD1)
51	17	MLZ	Host Address Ø (HADØ)
52	18		Ground
53	19	MLZ	Reset (RESET-)
54	20		Ground
55	21	Priam	Host Interrupt (ATTN-)
56	22		Ground
57	23	Priam	Host Read/Write (OUT-)
58	24	Priam	Data bus enable (BUSY-)
59	25		Ground
60	26	Priam	Data Request (DREQ-)

Priam connector pins 27 through 40 are not used. MLZ-P2 pins 1 through 34 are not used. For details on cable arrangements, see MLZ-P2P diagram.

The positions of the "M" and "S" jumpers are not critical when using the Priam interface. Do not install J11.



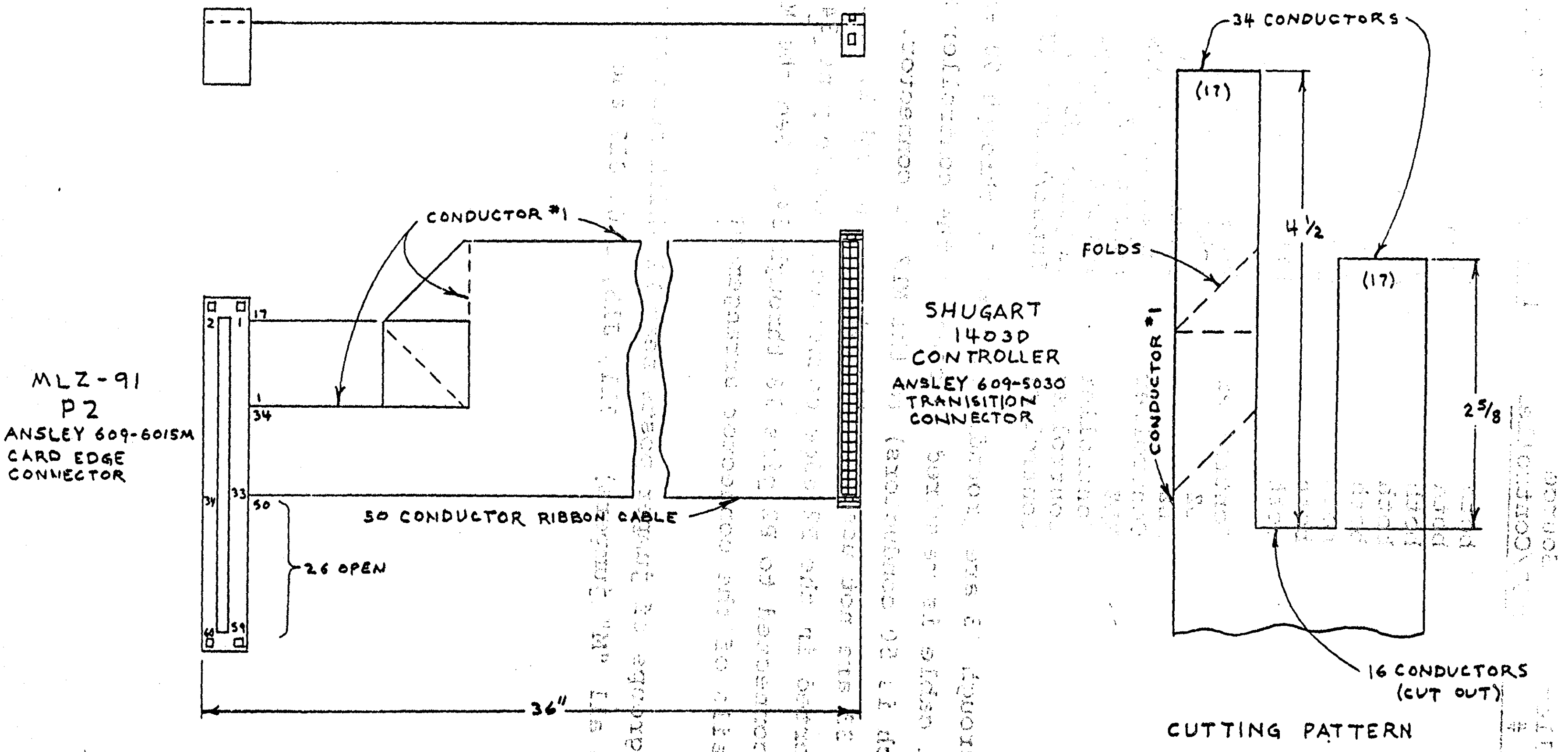
HEURIKON CORP. MADISON, WISCONSIN COPYRIGHT 1981	CABLE MLZ-P2P FOR MLZ-91	CHECKED: JM	DATE 2-81
		DRAWN: NGK	

SHUGART/SEAGATE/DTC

P2 Pin #	Controller Pin #	Source MLZ/Controller	Function (Negative True)
2	16	both	Data bit 7 (MSB)
4	14	both	Data bit 6
6	12	both	Data bit 5
8	10	both	Data bit 4
10	8	both	Data bit 3
12	6	both	Data bit 2
14	4	both	Data bit 1
16	2	both	Data bit Ø (LSB)
18	34		
20	36	Controller	Busy (BUSY-)
22	38	MLZ	Acknowledge (ACK-)
24	40	MLZ	Reset (RESET-)
26	42	Controller	Attention (ATTN-)
28	44	MLZ	Select (SEL-)
30	46	Controller	Direction (DIR-)
32	48	Controller	Request (REQ-)
34	50	Controller	Input/Output (I/O-)

P2 odd pins 1 through 33 are ground. P2 pins 35 through 60 are not used. A special cable is required to connect the controller interface cable (which is 50 conductors) to the MLZ P2 connector. Cable pins 18 through 33 are not used. Cable pins 1 through 17 are reversed when inserted in the P2 edge connector. Cable pins 34 through 50 are connected to P2 pins 18 through 34. See the MLZ-P2S diagram for details of the connector arrangement.

There are five groups of jumper posts near P2 which must be configured for "S" (remove all "M" jumpers). J11, J15, J16, J17 and J18 must be set to "S".



NOTE: CONDUCTORS 1-17 ARE FOLDED TO REVERSE ORDER WHEN CRIMPING IN MLZ-91 P2

HEURIKON CORP.
MADISON, WISCONSIN
COPYRIGHT 1981

CABLE MLZ-P2S FOR MLZ-91

CHECKED:

DATE 10-81

DRAWN: DHH.

STREAMER TAPE I/O CONNECTOR (P3)

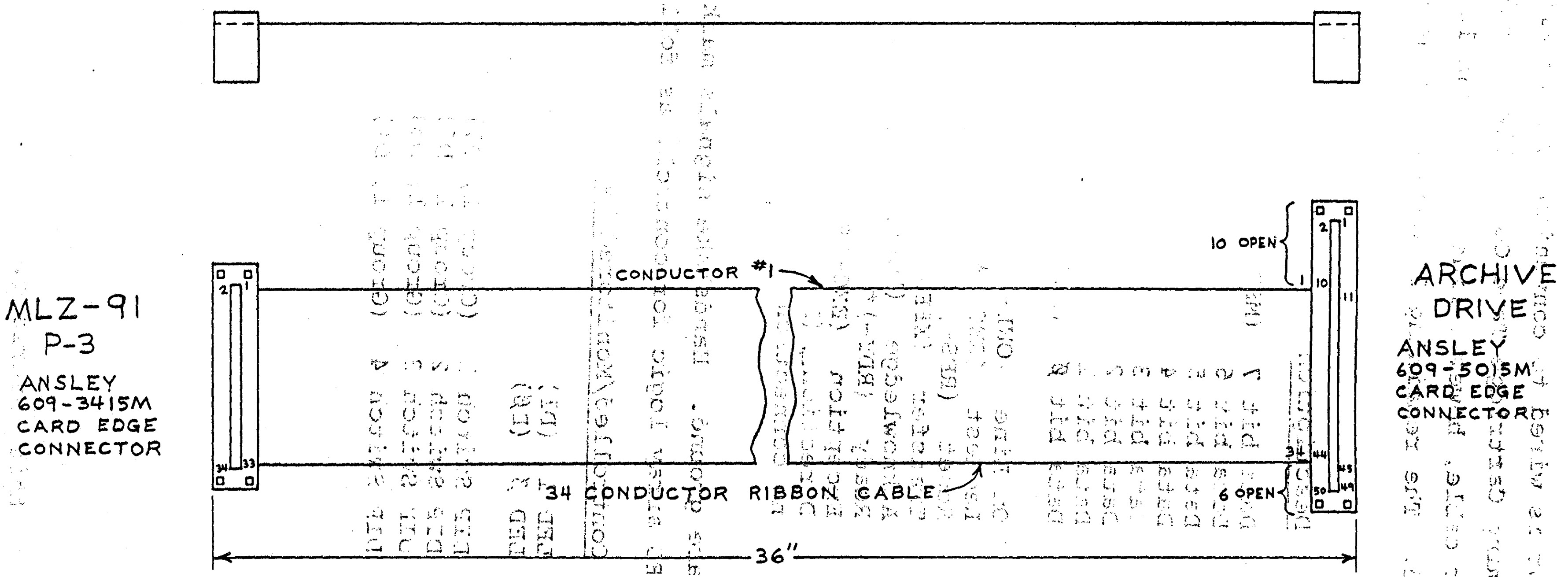
The streamer tape I/O connector is wired to conveniently mate with the Archive Corporation Streaming Cartridge Tape Controller. That controller uses a 40 conductor cable, however lines 1 through 10 and 45 through 50 are not used. The remaining 34 conductors mate directly with P3, as follows:

P3 Pin #	Archive Pin #	Source MLZ/Archive	Description
2	12	both	Data bit 7 (MSB)
4	14	both	Data bit 6
6	16	both	Data bit 5
8	18	both	Data bit 4
10	20	both	Data bit 3
12	22	both	Data bit 2
14	24	both	Data bit 1
16	26	both	Data bit Ø (LSB)
18	28	MLZ	On Line (ONL-)*
20	30	MLZ	Request (REQ-)*
22	32	MLZ	Reset (RES-)
24	34	MLZ	Transfer (XFER-)
26	36	Archive	Acknowledge (ACK-)*
28	38	Archive	Ready (RDY-)*
30	40	Archive	Exception (EXC-)*
32	42	Archive	Direction (DIR-)*
34	44	-	no connection

All odd numbered pins on P3 are ground. Handshake signals marked (*) utilize the DIP switch and LED array logic for control, as follows:

Signal	Controlled/Monitored by
ONL-	LED 1 (D1)
REQ-	LED Ø (DØ)
RDY-	DIP Switch 1 (Group 1, D7)
EXC-	DIP Switch 2 (Group 1, D6)
DIR-	DIP Switch 3 (Group 1, D5)
ACK-	DIP Switch 4 (Group 1, D4)

See page 73 for discussion of this interface.



HEURIKON CORP.
MADISON, WISCONSIN
COPYRIGHT 1981

CABLE MLZ-P3N FOR MLZ-91
STREAMER TAPE I/O

CHECKED: JM
DATE 2-81
DRAWN: NGK

SERIAL I/O CONNECTOR (P4)

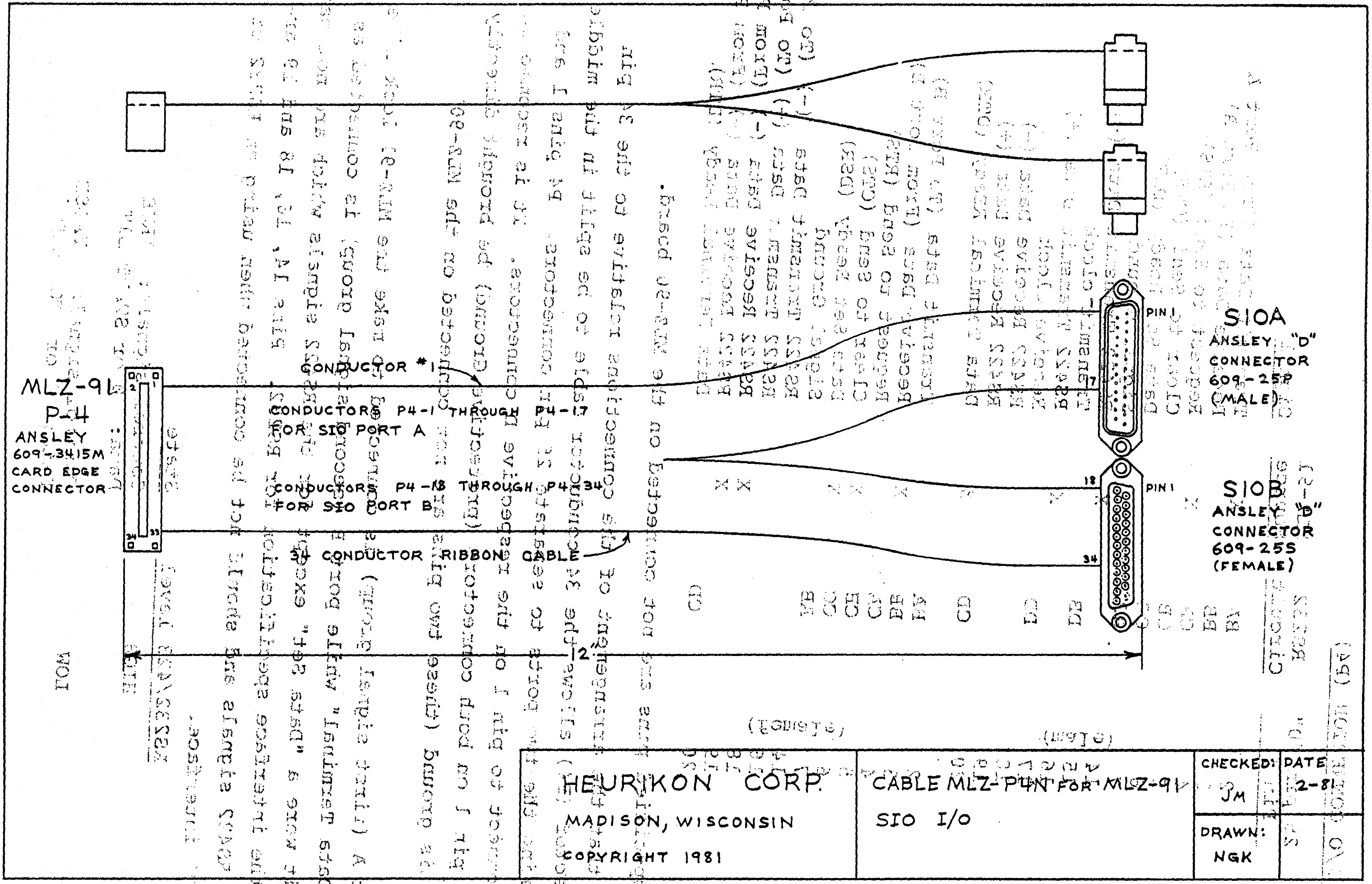
P4 Pin #	25 pin "D" Pin #	RS232 Circuit	MLZ-91 Source	Signal
3	2	BA	X	Transmit Data (From Port A)
5	3	BB		Receive Data (To Port A)
7	4	CA	X	Request to Send (RTS)
9	5	CB		Clear to Send (CTS)
11	6	~~CC~~		Data Set Ready (DSR)
13	7	AB		Signal Ground
2	14 (male)		X	RS422 Transmit Data (-)
4	15	DB		Transmit clock
6	16		X	RS422 Transmit Data (+)
8	17	DD		Receive clock
10	18			RS422 Receive Data (-)
12	19			RS422 Receive Data (+)
14	20	CD	X	Data Terminal REady (DTR)
20	2	BA		Transmit Data (To Port B)
22	3	BB	X	Receive Data (From Port B)
24	4	CA		Request to Send (RTS)
26	5	CB	X	Clear to Send (CTS)
28	6 (female)	CC	X	Data Set Ready (DSR)
30	7	AB		Signal Ground
19	14			RS422 Transmit Data (-) (To Port B)
23	16			RS422 Transmit Data (+) (To Port B)
27	18		X	RS422 Receive Data (-) (From Port B)
29	19		X	RS422 Receive Data (+) (From Port B)
31	20	CD		Data Terminal Ready (DTR)

Unspecified pins are not connected on the MLZ-90 board.

Note that the arrangement of the connections relative to the 34 pin connector (P4) allows the 34 conductor cable to be split in the middle to bring the two ports to separate 25 pin connectors. P4 pins 1 and 18 connect to pin 1 on the respective D connectors. It is recommended that pin 1 on both connectors (protective Ground) be brought directly to chassis ground (these two pins are not connected on the MLZ-90).

Port A (first signal group) is connected to make the MLZ-91 look like a "Data Terminal" while port B (second signal group) is connected as if it were a "Data Set" except for the RS422 signals which are not part of the interface specification for RS232. Pins 14, 16, 18 and 19 are the RS422 signals and should not be connected when using an RS232 or RS423 interface.

RS232/423 level	State
HIGH	Control signals: TRUE Data: Ø or START BIT
LOW	Control signals: FALSE Data: 1 or STOP BIT



HEURIKON CORP.
MADISON, WISCONSIN
COPYRIGHT 1981

CABLE MLZ-P4N FOR MLZ-91
SIO I/O

CHECKED: JM

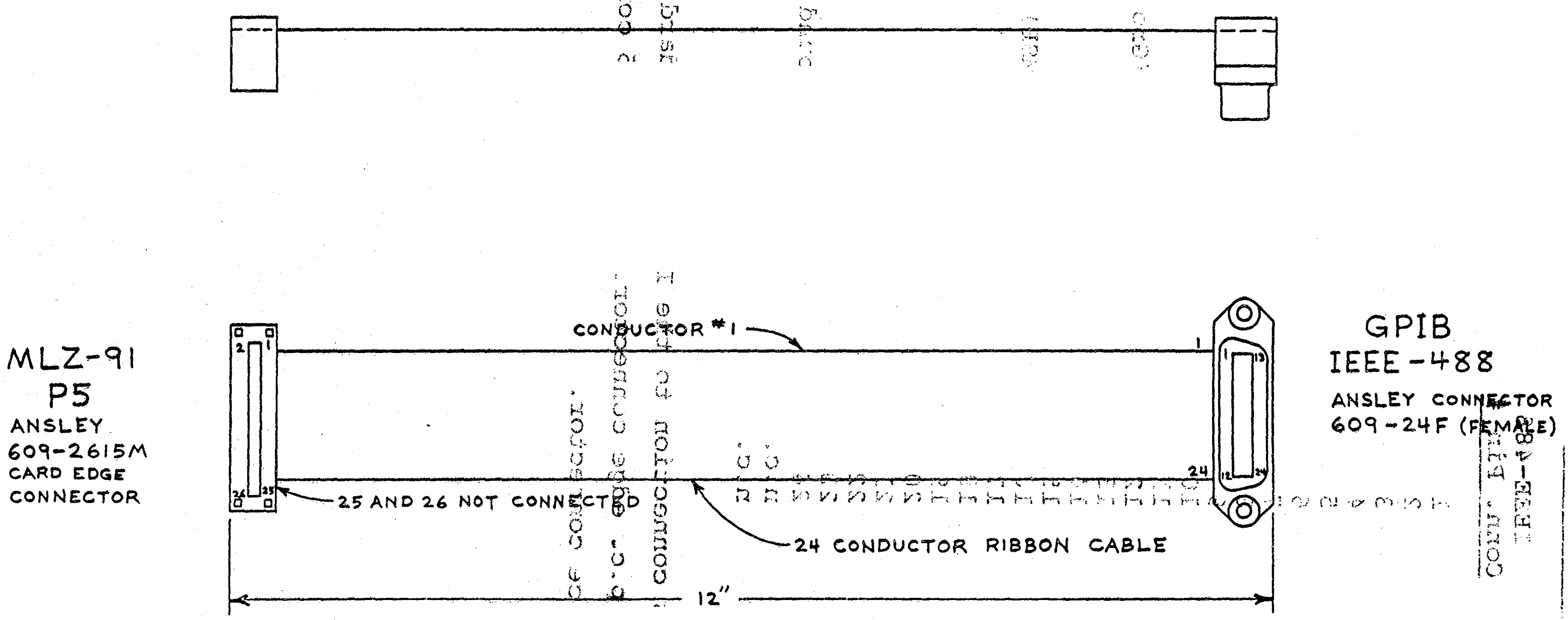
DATE 2-81

DRAWN: NGK

GPIB (IEEE-488) I/F CONNECTOR (P5)

P5 Pin #	IEEE-488 Conn. Pin #	Description
1	1	DIO 1 (LSB)
3	2	DIO 2
5	3	DIO 3
7	4	DIO 4
9	5	EOI
11	6	DAV
13	7	NRFD
15	8	NDAC
17	9	IFC
19	10	SRQ
21	11	ATN
23	12	Shield (Ground)
2	13	DIO 5
4	14	DIO 6
6	15	DIO 7
8	16	DIO 8 (MSB)
10	17	REN
12	18	Ground
14	19	Ground
16	20	Ground
18	21	Ground
20	22	Ground
22	23	Ground
24	24	Logic Ground
25	n.c.	-
26	n.c.	-

The pinout of P5 allows connection to the IEEE-488 standard connector via a ribbon cable and p.c. edge connector. Pin 1 of P5 connects to pin 1 of the interface connector.



HEURIKON CORP. MADISON, WISCONSIN COPYRIGHT 1981	CABLE MLZ-P5N FOR MLZ-91	CHECKED: JM	DATE -2-81
		DRAWN: NGK	

FLOPPY DISK I/O CONNECTOR (P6)

The MLZ-91 supports standard 8" drives or the smaller 5¼" drives. The connector pinout is similar for the two drive types.

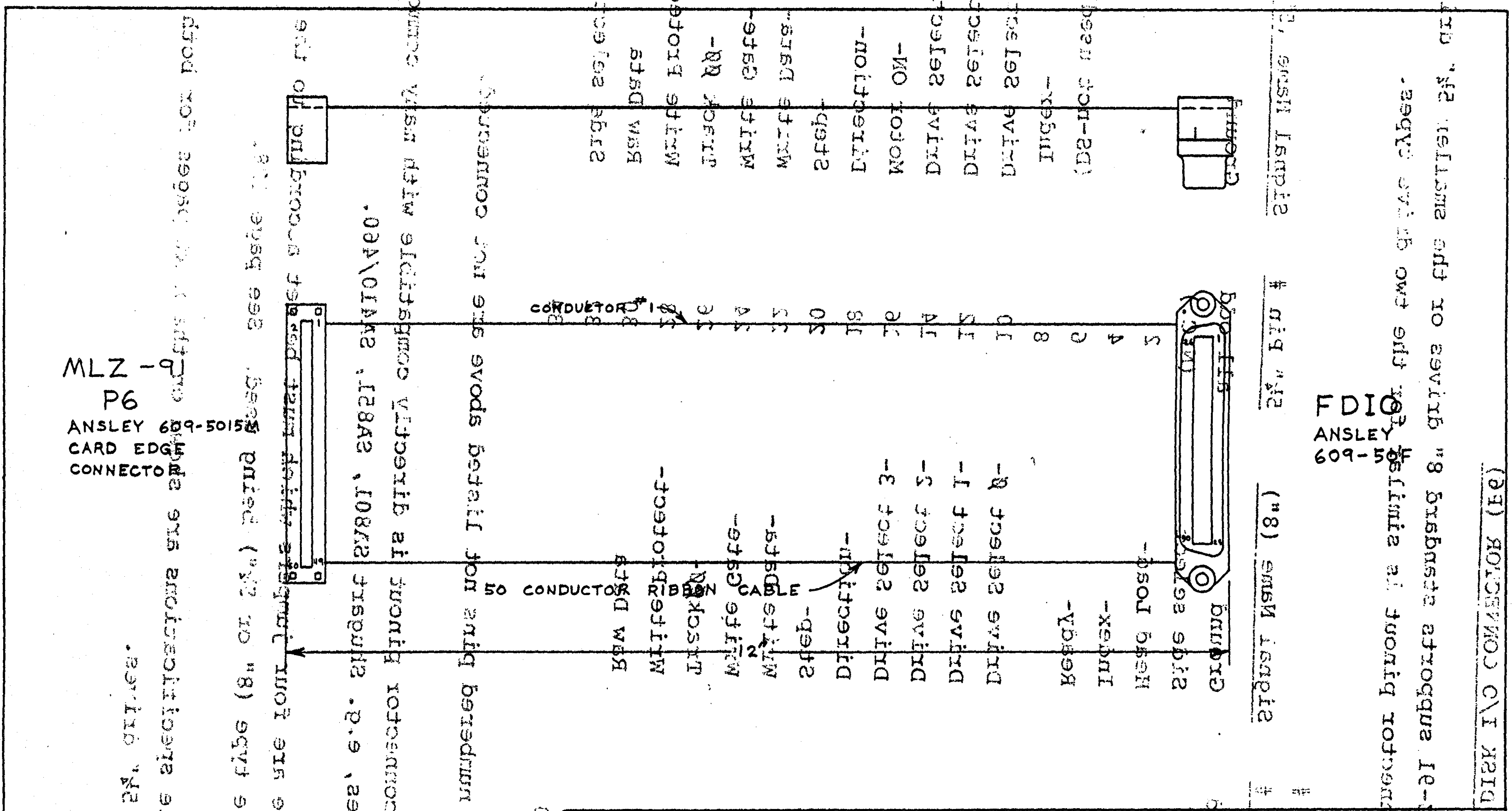
PG Pin # 8" Pin #	Signal Name (8")	5¼" Pin #	Signal Name (5¼")
all odd	Ground	all odd	Ground
14	Side select	(N.C.)	
18	Head Load-	2	
20	Index-	4	
22	Ready-	6	(DS-not used)
24		8	Index-
26	Drive Select Ø-	10	Drive Select Ø-
28	Drive Select 1-	12	Drive Select 1-
30	Drive Select 2-	14	Drive Select 2-
32	Drive Select 3-	16	Motor ON-
34	Direction-	18	Direction-
36	Step-	20	Step-
38	Write Data-	22	Write Data-
40	Write Gate-	24	Write Gate-
42	TrackØØ-	26	Track ØØ-
44	Write Protect-	28	Write Protect-
46	Raw Data	30	Raw Data
48		32	Side select
50		34	

Even numbered pins not listed above are not connected.

The connector pinout is directly compatible with many common drives, e.g. Shugart SA801, SA851, SA410/460.

There are four jumpers which must be set according to the drive type (8" or 5¼") being used. See page 128.

Cable specifications are shown on the next pages for both 8" and 5¼" drives.



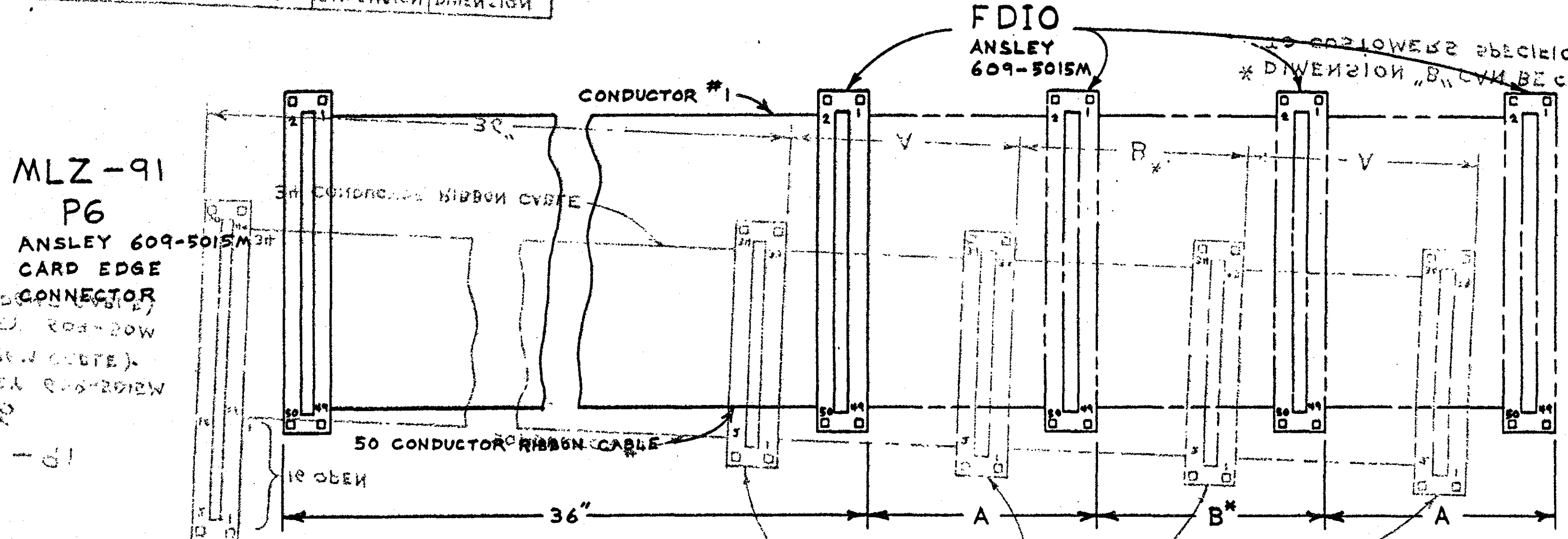
HEURIKON CORP.
MADISON, WISCONSIN
COPYRIGHT 1981

CABLE MLZ-P6H FOR MLZ-91

CHECKED: JM

DATE 2-81

DRAWN: NGK



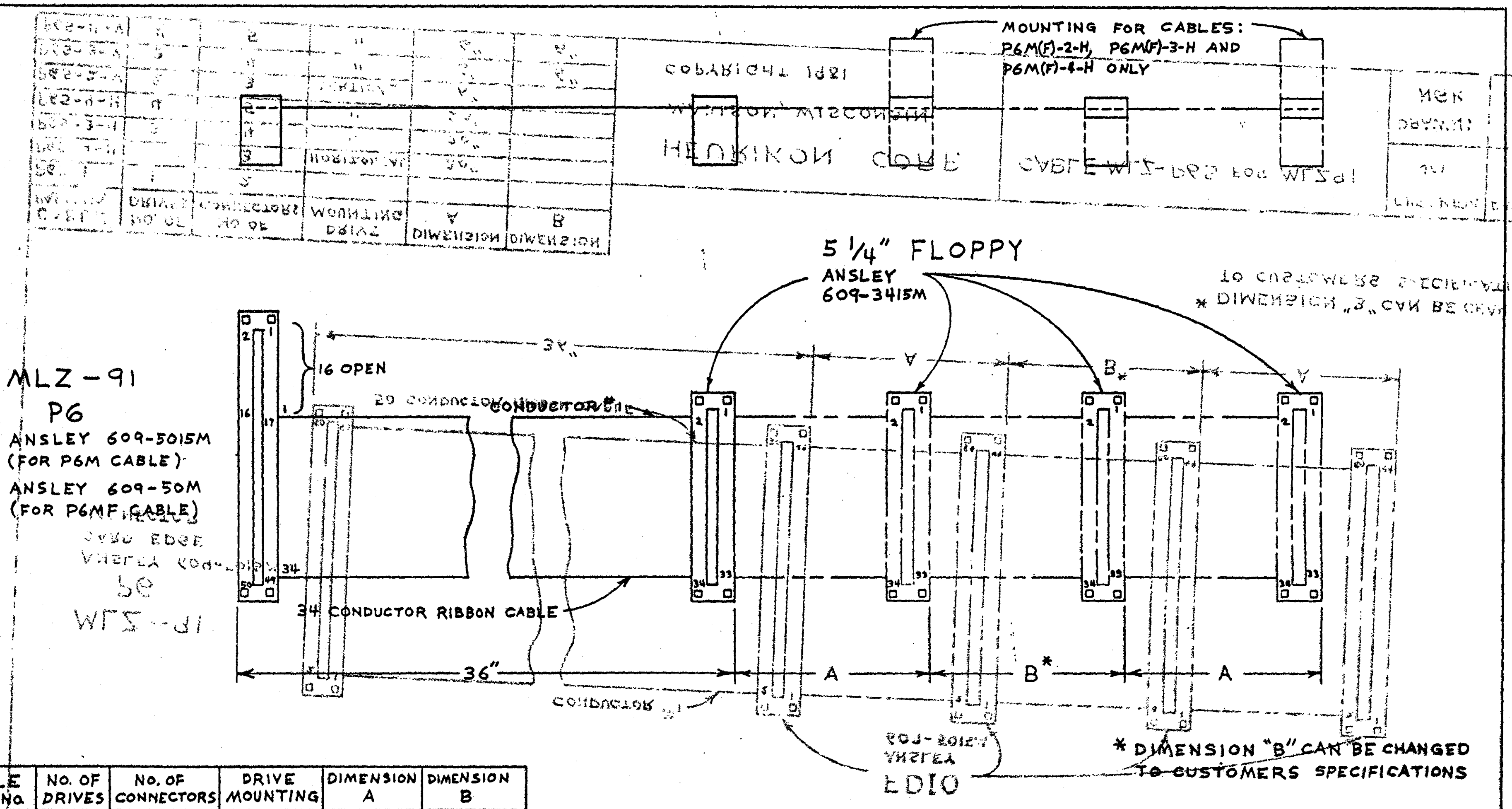
* DIMENSION "B" CAN BE CHANGED TO CUSTOMERS SPECIFICATIONS

CABLE PART NO.	NO. OF DRIVES	NO. OF CONNECTORS	DRIVE MOUNTING	DIMENSION A	DIMENSION B
P6S-1	1	2			
P6S-2-H	2	3	HORIZONTAL	20"	
P6S-3-H	3	4	"	20"	
P6S-4-H	4	5	"	20"	
P6S-2-V	2	3	VERTICAL	6"	
P6S-3-V	3	4	"	6"	6"
P6S-4-V	4	5	"	6"	6"

HEURIKON CORP.
MADISON, WISCONSIN
COPYRIGHT 1981

CABLE MLZ-P6S FOR MLZ91

CHECKED: JM	DATE 2-81
DRAWN: NGK	



* DIMENSION "B" CAN BE CHANGED TO CUSTOMERS SPECIFICATIONS

CABLE PART No.	No. OF DRIVES	No. OF CONNECTORS	DRIVE MOUNTING	DIMENSION A	DIMENSION B
P6M(F)-1	1	2			
P6M(F)-2-H	2	3	HORIZONTAL	17"	
P6M(F)-3-H	3	4	"	17"	
P6M(F)-4-H	4	5	"	17"	
P6M(F)-2-V	2	3	VERTICAL	6"	
P6M(F)-3-V	3	4	"	6"	6"
P6M(F)-4-V	4	5	"	6"	6"

HEURIKON CORP.
MADISON, WISCONSIN
COPYRIGHT 1981

CABLE MLZ-P6M AND P6MF
FOR MLZ91

CHECKED: JM

DATE 2-81

DRAWN: NGK

POWER REQUIREMENTS

	+5	+12	-12
Base logic including CPU, DMA, CTC ROM, RAM, bus I/F, GPIB Winchester I/F, Tape I/F, LEDs, DIP switches	X		
FDIO	X	X	
SIO with RS422 I/F	X		
SIO with RS232/RS423 I/F	X	X	X

In the maximum configuration (Including FDIO, RS232, ROM) but without APU, the power requirements are:

+5 volts:	2.5 amps (typ.)	3.0 amps (max.)
+12 volts:	125 ma. (typ.)	200 ma. (max.)
-12 volts:	25 ma. (typ.)	50 ma. (max.)

ENVIRONMENTAL

Temperature: 0 to 65 degrees C

Pursuant to FCC Technical Standards for Computing Devices, section 15.805 (Interim Labeling), the following notice is included:

> "Warning: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to redio communications. As temporarily permitted by regulation it has not been tested for compliance with the limits for Class A computing devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference."

ACCESSORIES

The following items, available from Heurikon, can be used with the MLZ-91A:

A. **HARDWARE**

91-P2M	I/F cable for Micropolis-type Winchester controllers. See page 151.
91-P2P	I/F cable for Priam "Smart" Winchester controller. See page 153.
91-P2S	I/F cable for Shugart/DTC/Seagate Winchester-type controllers. See page 155.
91-P3A	I/F cable for Archive Streamer Tape drive. See page 157.
91-P4N	I/F cable for GPIB (IEEE-488). See page 159.
91-P5N	I/F cable for two serial ports. See page 161.
91-P6H	I/F cable for floppy disk (8" drives). See page 164.
91-P6S-5	I/F cable for floppy disk (5¼" drives). See page 165.

Other hardware items available include:

Expansion I/O boards
Miscellaneous cable assemblies
Custom interface cards
CRT terminals
Printers

B. **SOFTWARE**

ZRAID-91 CP/M Monitor with FDIO routines and CP/M bootstrap (in ROM).

CP/M 2.2 Digital Research disk operating system with editor, assembler (8080) and utility programs. (On diskette)

Other BASIC interpreters and compiler
FORTRAN compiler
Z-80 assembler
Custom programs

Complete development systems are also available which consist of combinations of the above hardware and software items. For a complete list of hardware and software, refer to the Heurikon Product List or consult Heurikon direct.

heurikon corporation

"ZRAID-91" MANUAL

with support for the

CP/M Operating System

4/81 Version 91-1
Revision B

HEURIKON CORPORATION, 3001 LATHAM DRIVE, MADISON, WISCONSIN 53713 U.S.A. (608) 271-8700

ZRAID-91 MANUAL

CONTENTS

ZRAID Command Summary

Command	Function	Example	Reference Page
Hnn	Set upper eight bits of POINTER	H1Ø	9
Lnn	Set lower eight bits of POINTER	L24	9
Snnnn	Set POINTER (both H and L halves)	S1E34	9
A	Print POINTER value in H, L, format	A	9
Z	Print POINTER value in 16-bit octal format	Z	9
W	Print contents of addressed location	W	9
.nn	Set addressed location	.13	9
I	Increment POINTER, print location	I	9
D	Decrement POINTER, print location	D	9
G	Set/reset auto-verify option	G	10
P	Print - 32 lines (8 values per line)	P	10
Pn	Print - nn lines (nn in hex)	P2	10
N	Read the AUXPOINT in 16-bit octal	N	10
*	Transfer control to POINTER address via a JUMP	*	10
C	Transfer control to POINTER Address via a CALL (User stack)	C	11
Cn	CALL using ZRAID's stack	C2	11
J	Indirect CALL	J	11
—	Transfer control to AUXPOINT via a JUMP		11
&	Return to user's PROC CALL	&	11
'	(Apostrophe) CP/M Bootstrap	'	12
Y	Input from I/O device L	Y	12
Fnn	Output to I/O device L	F12	12

O	Set disk track (H) and Sector (L) value	O	13
(	Read from disk (Load)	(	13
)	Write to disk	)	14
CNTRL-V	Print floppy disk error counter (also exit from CP/M to ZRAID)	H152/	14
Rn	Insert a RST instruction	R2	15
$n	Force another RST instruction	$1	15
U	Remove the last RST instruction	U	15
=	Set memory mapping RAM	=FF	16
@	Set I/O mapping RAM	@7	16
B	Set Bus mapping RAM	BFØ	16
T	Transfer I/O between TTY and CRT	T	17
K	Set/Reset Echo flag	K	17
X	Set/Reset octal/hex I/O mode	X	17
"	Set bus control mode	"3	17
E	Blink LEDs	E	18
!	Initiate Multi-user mode	!	18
-	(Minus Sign) Enter Slave mode	-	18
/	Cancel previous input	H152/	18

Line feed and rubout characters are ignored. Underscored characters are operator inputs.

All commands except "/" must be followed by a space or a carriage return to cause execution to begin.

Note: The examples above and those which follow in the text use HEX numeric values which is the power-up default mode. These commands also work in octal. Use the X command to flip-flop between modes.

GETTING GOING

This section is an outline of the minimum work necessary to get the MLZ-91 "on the air":

Items required: (See diagram)

MLZ-91 Microcomputer Board
ZRAID-91 Software Monitor program (ROM)
RS232 Interactive Terminal and cable with male "D" connector
MLZ-P4N Serial Interface cable and connectors
Power supply (+5, +12, -12, volts)
Card Rack

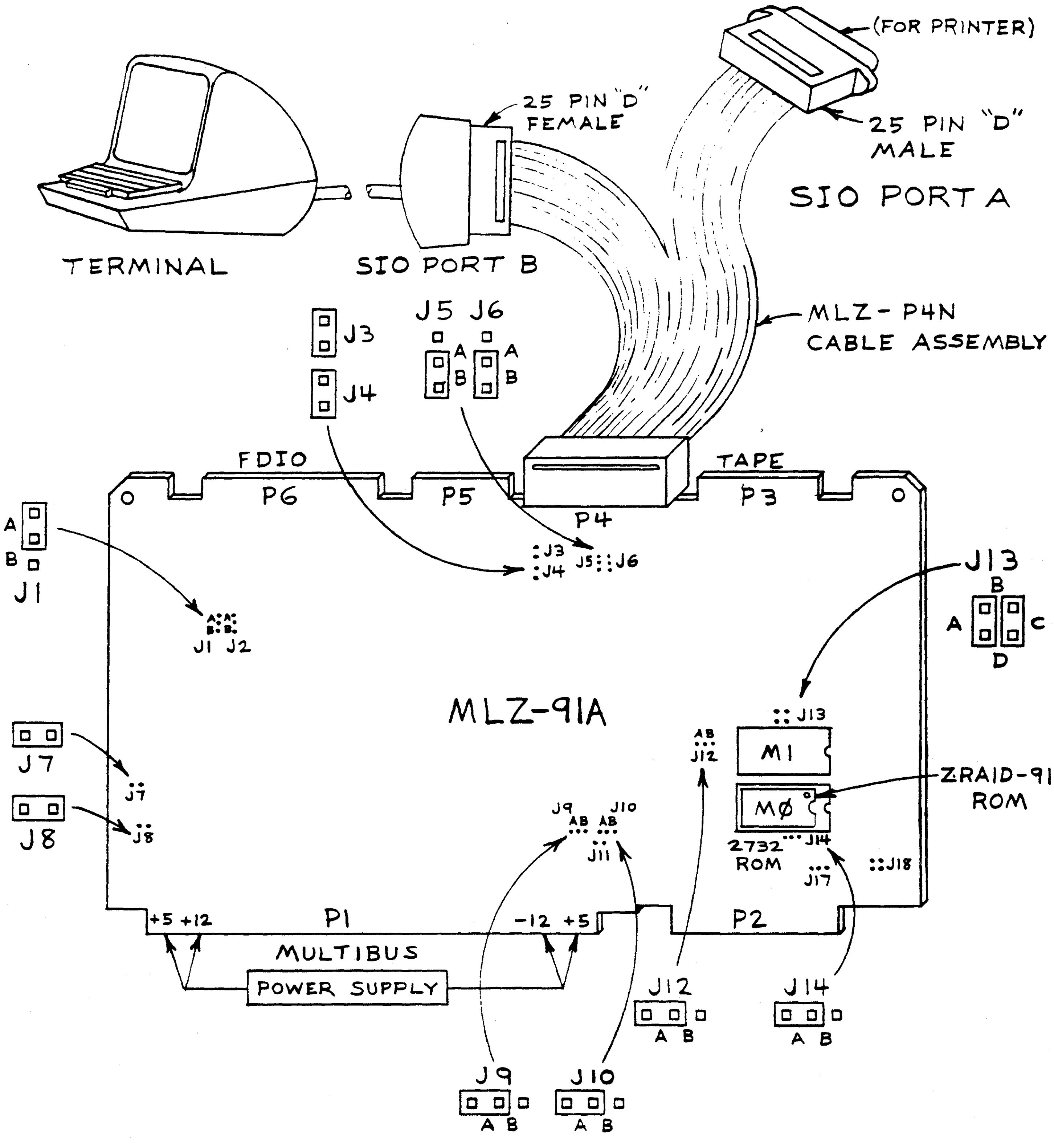
1. Insert ZRAID ROM in socket MØ (See diagram for position detail.)
2. Install Jumpers as follows:

J1-A	(2MHz clock)
J3,J4	(SIO I/F)
J5-B, J6-B	(SIO Port A Receive clock)
J7	(Upper address enable)
J8	(Processor priority)
J9-A, J10-A	(Wait states for ROM)
J12-A, J14-A	(ROM type 2732)
J13-A,C	(RAM type) (Assumes 4164 or 4532-1)

3. Connect console terminal to SIO port B via MLZ-P4N cable and P4 connector. Use the female D connector on the cable.

"D" Pin #	
2	Data from terminal
3	Data to terminal
4-5	Jumper (RTS-CTS)
6-20	Jumper (DSR-DTR)
7	Ground

4. Set baud rate on terminal for 9600 baud.
5. Set terminal options, if available, to 8 bits, no parity, two stop bits.
6. Apply power to MLZ-91 and terminal.
7. Activate RESET (momentarily ground P1-14)
8. ZRAID sign-on message should appear on terminal.
9. Consult ZRAID manual for further details. ZRAID automatically sets up the MLZ-91 mapping RAMs and allows access to all memory and I/O devices from the terminal.



MLZ - 91A WITH ZRAID - SETUP DIAGRAM

To load the CP/M operating system, follow the above steps but also connect a floppy disk drive to P6. Set the floppy disk configuration jumpers for "8" or "5" as appropriate. After turning on power and resetting the system, insert the CP/M system diskette in drive "A" and enter apostrophe space on the terminal.

ZRAID-91 Initialization State

The ZRAID-91 monitor initializes the mapping RAMs and on-card I/O derives as follows:

A. Memory Mapping RAM (See diagram, next page)

ROM socket MØ at CPU address FØØØ (hex)

On-card RAM allocated from address ØØØØ through address EFFF.

B. I/O Mapping RAM

I/O Addresses	Assignment
ØØ thru 3F	Off-card
4Ø thru 7F	Off-card
8Ø thru BF	On-card I/O Group A (e.g. Baud Gen)
CØ thru FF	On-card I/O Group B (e.g. CTC)

C. Bus Mapping RAM

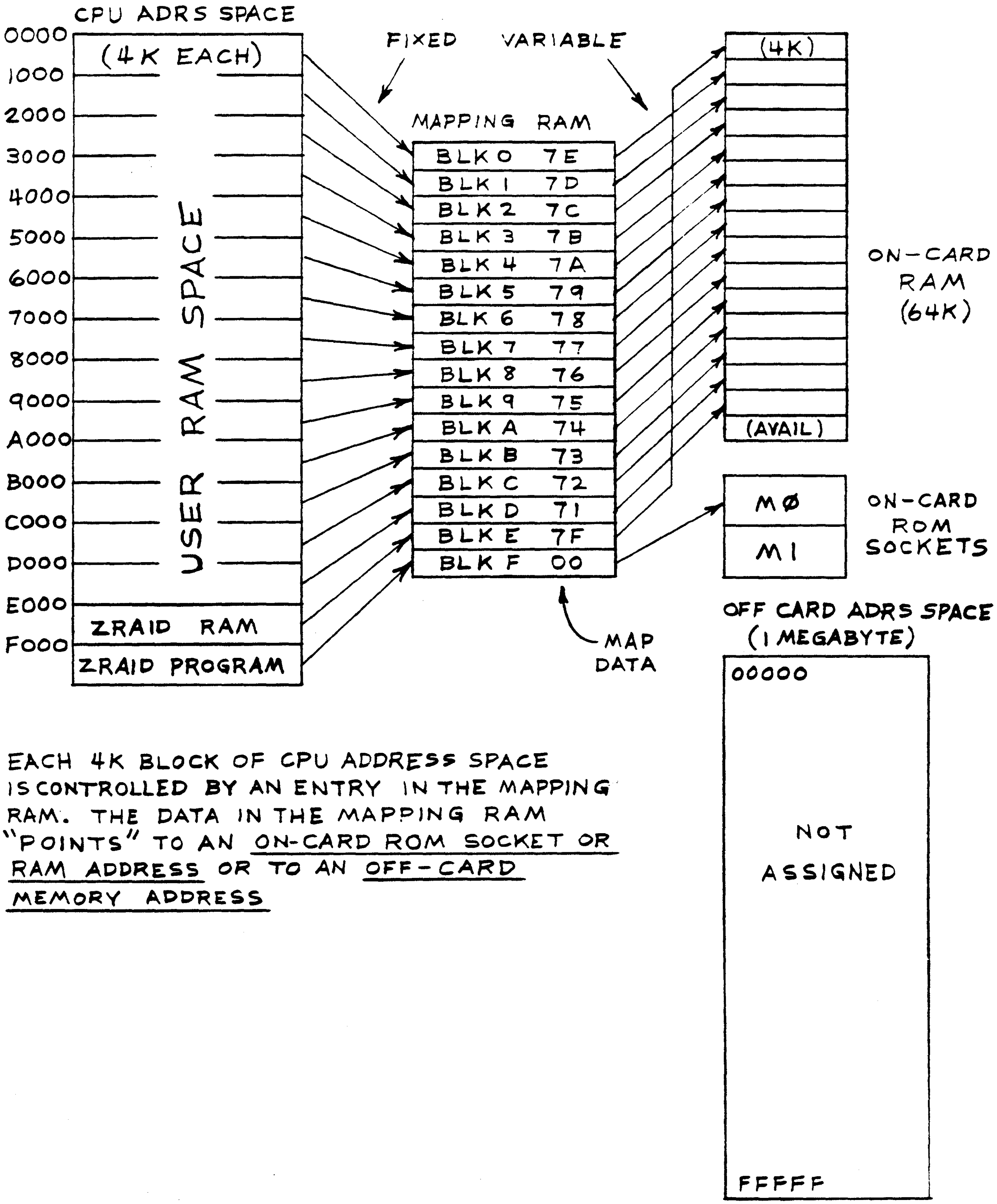
If DIP switches installed:

Board is assigned to the bus block (Ø - F) as specified by switches 5, 6, 7 and 8 of DIP switch group Ø. Otherwise, the board is assigned to block Ø (default). In either case, all board operations are enabled (i.e., Memory RD, WR and I/O operations are valid.)

D. SIO Baud rates:

If DIP switches installed, the baud rates are set according to DIP switch group 1 as shown on page 24. Otherwise, both SIO port baud rates are set at 9600 baud (default).

Note: These values may be modified by special ZRAID commands or the initial values may be changed in the ZRAID ROM.



EACH 4K BLOCK OF CPU ADDRESS SPACE IS CONTROLLED BY AN ENTRY IN THE MAPPING RAM. THE DATA IN THE MAPPING RAM "POINTS" TO AN ON-CARD ROM SOCKET OR RAM ADDRESS OR TO AN OFF-CARD MEMORY ADDRESS

ZRAID-91 INITIAL MEMORY MAP

ZRAID

(It will kill bugs dead)

CP/M Version

Command Descriptions

All ZRAID (Z-80 Realtime Analysis and Interactive Debugger) commands use the following format:

Xnnn

followed by a space. "X" represents a valid command character and "nnn" represents an optional parameter. The commands are used to set, modify, and examine contents of the system memory. The parameter specifies address values, memory contents, and word counts.

All commands must be terminated by a space or a carriage return. After each command is executed, ZRAID types a prompt character, ">", to indicate that it is ready to receive another operator input. The radix defaults to hexidecimal (but may be changed to octal, see "X" command).

ZRAID maintains an entry in its data area called the "POINTER". This is a 16-bit address value that parallels the functions of the processor's H and L register pair. Most commands in ZRAID use the POINTER in some way. The POINTER is composed of two parts, called "H" and "L", each representing eight bits. Some commands require two addresses. For those commands another pointer, called the AUXILIARY POINTER, or "AUXPOINT" is provided.

Memory Commands

Command   Description

Hnnn   Set the "H" half (upper 8 bits) of the POINTER to "nnn". The other half of the pointer can be set by using the "L" command.

Lnnn   Set the "L" half (lower 8 bits) of the POINTER to "nnn". The "H" and "L" commands may be used together to set the POINTER to any memory address, or the "S" command, below, can be used to set the POINTER in one operation.

Snnnn   Set the value of the POINTER to "nnnn" (all 16 bits). This command combines the functions of the "H" and "L" functions.

A   Display the value of the POINTER in "Hnn Lnn" format.

Z   Display the value of the POINTER in 16-bit octal format.

W   Display the contents of the memory location specified by the POINTER ("What's there?").

.nnn   Set the contents of the memory location specified by the POINTER to "nnn". The POINTER is automatically incremented to the next location (POINTER + 1). Successive store commands may be used to set contiguous locations or enter a complete program.

After this command is executed, the contents of addressed location are automatically verified. If the location did not store properly an error warning is generated (bell or beep).

I   Increment the POINTER to the next memory location and display the contents of that location.

D   Decrement the POINTER to the previous memory location and display the contents of that location.

Command Description

G Turn the auto-verify feature ON and OFF. In certain cases the auto-verify feature may not be desired. Examples of this are "memory locations" that are actually ports for I/O devices. The auto-verify feature will result in the location to be accessed twice for every write operation. Auto-verify is initially set ON.

The "G" command will also turn verify back on after a previous "G" command. To test if the verification logic is ON or OFF, execute a store command to a ROM location (but be sure the location chosen has address bits A5 and A4 ON to prevent altering of the I/O or bus mapping RAMs).

Pnn Print "nn" lines of consecutive memory locations, eight locations per line. If "nn" is zero (or not entered) 32 (decimal) lines will be output. The output may be cut off early by hitting any key on the keyboard.

Each print line will contain the starting POINTER value plus eight values for eight locations, each preceded by a ".".

M Set the AUXPOINT contents equal to the POINTER value. This is the only command which alters the value of AUXPOINT.

N Print the current AUXPOINT value.

Register Display and Alteration

See Page 21.

Parity and Write Protect Errors

See Page 23.

Transfer of Control Commands

All commands which transfer control to a location outside of ZRAID restore the Z-80 internal registers, prior to the transfer, from the "register save area" in ZRAID's memory area. To set the Z-80 registers to specific values, modify the appropriate memory locations corresponding to the registers prior to issuing the transfer of control command. See page 21 for a description of the register save area.

Prior to transfering control, these commands issue a "!" to the display to indicate command execution.

Command — Description

* — Transfer control to the location specified by the POINTER via a JUMP instruction.

C — Transfer control to the location specified by the POINTER via a CALL instruction. This allows ZRAID to test a subroutine which terminates with a RETURN instruction.

If the "C" command is issued without a parameter (or with a zero parameter) the stack pointer register is restored according to the value in the register save area which must, therefore, specify a valid RAM address. If the "C" command is issued with a non zero parameter (e.g. C4) the stack pointer is automatically set to ZRAID's stack area prior to transferring control.

J — Transfer control, via a CALL (using ZRAID's stack), to the location specified by the contents of the memory address specified by the POINTER. This is an indirect CALL. The POINTER specifies the address of the low byte half of a 16-bit value which is taken as the desired address.

← — (Left arrow or underscore) Transfer control, via a JUMP, to the location specified by the auxiliary pointer, AUXPOINT.

& — Transfer control, after restoring the Z-80 registers, via a RETURN instruction. This command is useful to return to a user program following ZRAID entry via location RPROC. (See page 30.)

Command Description

' CP/M bootstrap. The CP/M system is loaded from drive A. See CP/M System Considerations, page 25, for details.

I/O Commands

Command Description

Y Input from the device address specified by the low half of the POINTER. When this command is executed, the H half of the pointer appears on the upper CPU address lines to specify a memory mapping block address, as may be required for off-card I/O operations.

Fnnn Output data "nnn" to the device address specified by the low half of the POINTER. This command also places the high half of the pointer on the upper CPU address lines. (This is done to facilitate I/O to the GPIB, PRIAM Winchester or the system bus.)

Floppy Disk Control Commands

These commands allow access to specific tracks and sectors on a floppy diskette. Drive "A" is used for all disk operations.

Command — Description

O — Set the starting track and sector values for the next floppy disk access.

TRACK = H half of the POINTER
SECTOR = L half of the POINTER

M — Set the AUXPOINTER from the POINTER. This address represents the starting location of memory data for the disk data transfer.

(n — (Left paren.) Initiate a floppy disk READ (load) operation. The "O" and "M" commands must have previously been issued and the POINTER must specify the ending memory data address. The correct sequence for using this command is as follows:

1. Use "Hnnn" to specify the starting TRACK.
2. Use "Lnnn" to specify the starting SECTOR.
3. Use "O" to set the TRACK and SECTOR values.
4. Use "Hnnn" and "Lnnn" or "Snnn" to specify the starting data address.
5. Use "M" to set the starting memory data address.
6. Use "Hnnn" and "Lnnn" or "Snnn" to set the ending memory data address.
7. Use "(n" to initiate the read operation, where "n" is a hexidecimal value as follows:

none or Ø	Single density, 128 bytes/sector
6Ø	Double density 1024
7Ø	Double density 128

The "C" command will cause a data verification message to be printed with the following format:

R Hnnn Lnnn - Hnnn Lnnn Tnn Snn ?

where "R" means disk READ (will be "W" for disk WRITE)
The first H,L pair is the starting memory address
The second H,L pair is the ending address
"T" and "S" are the beginning TRACK and SECTOR

Command Description

If the data is correct, enter ":" to execute the disk command. To abort the command enter any other character (e.g., space).

An exclamation point ("!") will be displayed on the successful completion of the disk operation. Otherwise, an error message will be displayed.
(See "DISK I/O ERROR TYPES ", page 34.

)n (Right paren.) Disk WRITE command. Same as above except data is written to the floppy diskette. The same parameter setup procedures apply.

Control-V Print the running total of the floppy disk error reentry counter. This count value is reset at the beginning of each "(", ")" or "'" command.

Program Debugging Commands

These commands are used to control breakpoints for use in debugging programs.

Command — Description

Rn — Insert a restart instruction (RSTØ to RST7) at the location specified by the POINTER. The instruction which is replaced is saved for subsequent reinsertion later. (Verification is performed unless shut off by the "G" command.) "n" may take on the values of Ø through 7 corresponding to one of the desired 8 restart instructions.

An error "beep" will be generated if an attempt is made to insert a second restart before removing the first, or if the restart failed to store correctly.

NOTE: A JUMP to DEBUG must be placed at the chosen restart location, for example RST 1 at ØØØ8H, in the first page of memory. When the restart is encountered during program execution the registers and flags will be saved and ZRAID will become active.

$n — Force a restart at the location specified by the POINTER even if one has already been inserted by an "R" instruction. Caution: It is easy to loose track of excess restarts.

U — Remove the restart inserted by the last "R" or "$" command. The value of the POINTER is not used or affected by this command.

Mapping RAM Control Commands

These commands facilitate the loading of the various mapping RAMs on the MLZ-91. Whenever ZRAID is restarted, the mapping RAMs are setup with specific initial values as listed on page 5. These commands may be used to alter the mapping RAM contents. Refer to the MLZ-91 User Manual for details on the mapping RAM logic.

Command — Description

=nn Set the contents of the memory mapping RAM block specified by the H half of the pointer (upper 4 bits) to the value "nnn". For example, to specify 4K of off-card memory at location 1ØØØ (hex) do the following:

```
    H1Ø     (block address)
    =FF     (map data for some off-card memory)
```

@nn Set the contents of the I/O Device mapping RAM block specified by the L half of the pointer (lower 2 bits) to the value "nnn". For example, to specify I/O Device Group A to be located at I/O block 2 (base 8ØH) do the following:

```
    L2      (block address)
    @7      (map data for I/O group A)
```

Bnn Set the contents of the BUS mapping RAM block specified by the L half of the pointer (lower 4 bits) to the value "nnn". For example, to enable all on-card functions and assign the board to bus block "Ø", do the following:

```
    LØ      (bus block Ø)
    BFØ     (enable all operations)
```

Note: ZRAID expects a certain configuration of the I/O map and memory mapping RAMs. If the I/O mapping RAM contents are changed, ZRAID may not function correctly since the I/O device addresses are assumed to be constant.

Consult the MLZ-91 User's Manual for the correct data values to use with these commands.

Other Commands

Command — Description

T — Transfer control of ZRAID from SIO port A to SIO port B, or vice versa.

K — Echo all ZRIAD commands and responses on the other SIO port. If the echo feature is already ON, this command will turn echo OFF.

If a printer is connected to the alternate SIO port, this command may be used to print any or all ZRAID responses. For example, to print a memory dump, issue a "K" command and a group of "P" commands.

X — Switch the numeric I/O base (radix) to/from hex or octal modes. ZRAID defaults to hex mode. (Note: The "Z" command always outputs in octal, regardless of the selected base.)

"n — Set the system Multibus control mode as follows:

Command	Function
"	Release bus between each operation
"1	Release on any other board request
"2	Release only on higher priority board request
"3	Never release the bus

Special Commands

These commands perform special functions and are provided mainly to illustrate certain features of the MLZ-91.

Command — Description

E — Count in binary, display count value on the LED array and blink the floppy disk USER LED. This command will remain active until the next command is entered.

!1 — Initiate the MULTI-USER mode. This command demonstrates the memory mapping features. To use, connect a console device to both SIO ports. Issue this command from the console connected to port B. Both consoles will become
- active. To return to the single user mode, enter "!" (with no parameter). (This command is not supported in some versions of ZRAID.)

(Minus sign) Enter SLAVE mode. This command converts the MLZ-91 to a slave I/O board for use in a multiprocessor system. The following state will exist:

    I/O Blocks Ø & 2 = On-card device group A (IOA)
    I/O Blocks 1 & 3 = On-card device group B (IOB)
    Bus Block - Value specified by DlP SW group Ø
    (low 4 bits)

The on-card memory is also re-allocated to 100% RAM. (Note: 64K of RAM is required for this command.) If this command is issued with a non-zero parameter,
CNTRL-W 12 the memory is converted to 100% RAM but ZRAID remains active. This allows temporary RAM changes to be made to ZRAID.

This is a special command which will execute continuous double density writes to the floppy disk for use when adjusting the write precompensation logic. Note: This command requires a parameter value of 12 (hex) to be entered with the command. Caution should be taken when using this command since disk writes are performed which could destroy valid disk data. Use a "scratch"
CENTRL-W 13 disk only. Refer to the MLZ-91 User's Manual for details on the adjustment procedures.

Same as above except single density.

Error Correction

If an error is made while making an entry, type "/" (slash). The previous input characters will be ignored. For example,

L15 (no space entered as yet)

may be cancelled and the correct entry reentered, as follows:

```
L15/>H15
   | |
   | | corrected command entered
   | |
   | prompt
   |
  slash cancels previous erroneous entry
```

Numerical errors may be corrected simply by typing the corrected value. For example, suppose "H15" is desired, but instead we type:

H16 (no space entered as yet)

the correct numerals may be entered as follows:

H1615

In the hex mode, only the last two digits will be used as the parameter for commands expecting an eight bit value (or the last four digits for the 16-bit parameters). If the I/O mode is octal, only the last 3 (or 6) characters will be used for the parameter.

TROUBLESHOOTING

1. System locks up following a command: The addressed memory location or I/O device is not assigned or the addressed board is not issuing BACK (bus acknowledge).

2. Memory mapped I/O devices are not responding to the store command properly: The auto-verify feature is following each store with a read. Turn auto-verify OFF by issuing a "G" command.

3. Program does not run following a transfer of control command: The POINTER value or register save area was not set correctly prior to issuing the command, or the user program at the POINTER location does not function properly.

4. ZRAID does not issue sign-on message and does not respond to commands: Terminal connected incorrectly or to wrong port. Possible power supply or jumper problem.

Register Display and Alteration

When ZRAID is entered through "DEBUG" or "RPROC" the flags and the contents of the primary user registers are saved. The POINTER is initialized to location EEØØ, (hex.) The flags and registers are saved at the following locations:

Location	Register
EEØØ	F (PSW flags)
EEØ1	A
EEØ2	C
EEØ3	B
EEØ4	E
EEØ5	D
EEØ6	L
EEØ7	H
EEØ8	SP-L (User stack pointer, L half)
EEØ9	SP-H (User stack pointer, H half)
EEØA	IX-L
EEØB	IX-H
EEØC	IY-L
EEØD	IY-H

Since the POINTER is initialized to EEØØ, the content of the registers can be output with a "P2" command as follows:

```
        HEE  LØØ  .ff  .aa  .cc  .bb  .ee  .dd  .ll  .hh
        HEE  LØ8  .sl  .sh  .xl  .xh  .yl  .yh  .xx  .xx
```

Where "ff" and "aa" represent the flags and the contents of register A, etc. respectively.

Where "ff" through "hh" represent the content of the flags and the internal user registers, and "sl" through "yh" represent the content of the stack pointer (low half, high half), index register IX and index register IY.

To modify the user registers prior to a return to the user program (via the "*, C, or ←" commands), set the desired values in the appropriate address of the register save area. When ZRAID exits to the user program the registers and flags will be restored from the save area.

The flag word is arranged as follows:

Z-80 FLAG DATA BIT

7	6	5	4	3	2	1	0
S	Z	-	H	-	p/v	N	C

where:

S = Sign flag

Z = Zero flag

H = Half carry flag (for DAA)

p/v = Parity/Overflow flag

N = Add/Subtract flag

C = Carry flag

- = indeterminate value

Parity and Write Protect Logic

Whenever an on-card RAM parity error or write protect error occurs, the MLZ-91 executes a RESTART to location ØØ66. This is the NMI (Non-Maskable Interrupt) location.

ZRAID-91 contains an interrupt processing routine which determines the error type and the approximate program execution address when the error occurred. For example, the message:

```
WRITE PROTECT ERROR, DURING INSTRUCTION PRECEDING H2Ø LDA
```

means that the instruction just prior to hex address 2ØDA attempted to write to protected memory. Note that the address printed is related to the CPU execution address, not the memory address of the protected memory (or the address having the parity error.) If the error occurred during an off-card access of on-card memory or during a DMA memory cycle, the address value displayed will not have any relation to the error address. The address of the next instruction is pushed into the software stack if a write protect error occurs. The address of the second next instruction is saved in the case of a parity error.

To test the NMI error logic, enter these commands:

1. For WRITE PROTECT error:
   a) Write protect a block of memory:
      i) H (Set H=ØØ, block Ø)
      ii) =5E (Load memory mapping RAM with data value to protect block Ø)
   b) Write to protected memory:
      . (Try to write ØØ to block Ø)
      (H half of the POINTER is still ØØ)
   c) An immediate write protect error should result
   d) Unprotect memory
      i) H (Optional, H Still = ØØ)
      ii) =7E (Turn write protect off)

2. For PARITY error:
   a) Set POINTER to any unused RAM address
   b) Enter "P" to display memory. A parity error should occur after printing a few locations. If not, enter another "P" command. If all of memory has been initialized (written to), then momentarily turn power-off.

BAUD RATES and BOARD BUS POSITION

The DIP switches, if present, are used to select the SIO baud rates and the bus block occupied by the MLZ-91.

(If the DIP switch option is not installed, the baud rates for both SIO ports defaults to 9600 and the board position defaults to bus block Ø.)

DIP switch group Ø is used to select the bus block as follows:

SWITCH	BUS BLOCK
5,6,7,8	Binary value formed by switches e.g. OFF, OFF, OFF, OFF = block Ø OFF, OFF, OFF, ON = block 1 etc.

DIP switch group 1 is used to select the SIO baud rates as follows:

SWITCH	PORT
9,10,11,12	SIO PORT A
13,14,15,16	SIO PORT B

SWITCH SETTING (Ø = OFF, 1=ON)	BAUD RATE
ØØØØ	96ØØ (Default)
ØØØ1	75
ØØ1Ø	110
ØØ11	134.5
Ø1ØØ	150
Ø1Ø1	300
Ø11Ø	600
Ø111	1200
1ØØØ	1800
1ØØ1	2000
1Ø1Ø	2400
1Ø11	3600
11ØØ	4800
11Ø1	7200
111Ø	9600
1111	19200

Note: If you have DIP switches installed, set them for the desired baud rate. Do not use the "default" position as that will cause ZRAID to ignore switches 1-8.

The MLZ-91 must be RESET in order for a new baud rate selection to take effect.

CP/M System Considerations

One feature of ZRAID is the inclusion of CP/M* system bootstrap and I/O routines. The "'" (apostrophe) command will transfer control to the bootstrap procedure and then automatically transfer control to CP/M.

Input and output routines for the console, list, punch and reader devices, as well as the disk I/O routines, are contained in ROM. After CP/M has been loaded, ZRAID automatically overlays a Jump Table in the CP/M "BIOS" area which effectively substitutes the I/O procedures available through ZRAID for those provided by CP/M.

The actual location of the BIOS Jump Table is determined by examining the original CP/M BIOS Jump Table on the diskette. This method allows the size of the CP/M system to be modified (via the CP/M "MOVCPM" and "SYSGEN" command) without changing ZRAID. The load location for CP/M is also computed from the BIOS Jump Table. The largest system size allowed is 56K.

Specifically, the following procedure is used to load CP/M:

1. Enter ' to ZRAID (or 'n where "n" is a parameter as described below.)
2. System RAM (address ØØØØ through E3FF is cleared to zeroes. This prevents parity errors from occurring with some CP/M programs which read from uninitialized memory locations. (This step will be skipped if parameter bit D7 is non-zero.)
3. The winchester I/F is tested to see if a controller is connected. If so, the system is loaded from the winchester (see below). Otherwise, or if the parameter bit D7 is non-zero, the system is loaded from floppy. See page 28 for physical drive address assignments.
4. The beginning sector of CP/M's BIOS is read into ZRAID's memory.
5. The first byte of this sector is checked to see if it is the proper code for a JUMP instruction. A "wrong disk" error occurs and the load is aborted if the JUMP is not found.
6. The third word is fetched (the H half of the BIOS JUMP to CP/M's CBOOT) and masked to determine the base address of the BIOS.

* CP/M is a product of Digital Research.

7. The load location for CP/M's CCP and BDOS is computed and the whole CP/M system is loaded into memory starting from the computed CCP location.
8. The BIOS Jump Table is overlayed with a new Jump Table which points to the special CP/M I/O procedures that are part of ZRAID ("CBIOS").
9. The message
   ```
        xxK CP/M VERSION a.b
        HEURIKON CORP  Zc.d
   ```
   is output to the console indicating the system size ("xxK" bytes), the CP/M version ("a.b") and the ZRAID version ("Zc.d").
10. The first 8 words of memory are initialized according to CP/M specifications. In addition a JUMP to DEBUG is stored at location HØ LØ8 to allow a RST1 to return to ZRAID.
11. Control is transferred to the CP/M CCP.

ZRAID supports the "IOBYTE" control system referred to in the CP/M documentation. "Logical" and "physical" I/O devices may be associated with each other by using the CP/M "STAT" command. The physical I/O devices are defined as follows:

Logical Device		Physical Device	
CON	(Console)	TTY	SIO Port A
		CRT	SIO Port B (Default)
		BAT	SIO Port A
		UC1	Dummy
LST	(List)	TTY	SIO Port A (Default Value)
		CRT	SIO Port B
		LPT	SIO Port A
		UC1	Dummy
PUN	(Punch)	TTY	SIO Port A (Default Value)
		PTP	SIO Port A
		UP1	SIO Port B
		UP2	Dummy
RDR	(Reader)	TTY	SIO Port A (Default Value)
		PTR	SIO Port A
		UR1	SIO Port B
		UR2	Dummy

A "dummy" device appears as an I/O device which is always ready. On input, a value of all zeros is returned as the input "data". Physical device addresses may be changed (e.g. to an external I/O port). Consult factory for details.

The default device for the CONSOLE (TTY or CRT) is automatically set to the same device being used to input ZRAID commands. (This can be changed via the CP/M "STAT" command.)

Some CP/M programs terminate with a system warmstart which reloads the operating system from diskette. ZRAID outputs a left and right bracket and a beep to the console to indicate a warmstart operation is in progress.

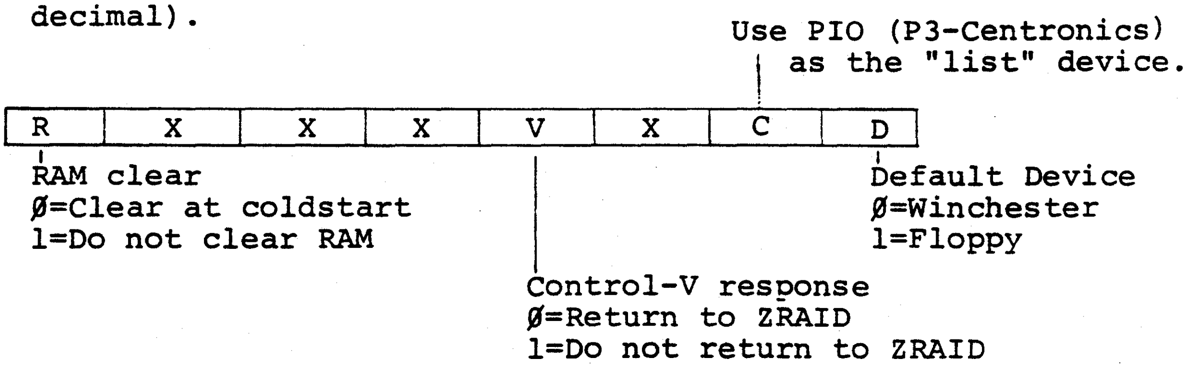
Control can be returned to ZRAID from CP/M by one of two methods:

1. Push the system RESET button.
2. Enter CNTRL-V on the console. This will cause the console input routine to CALL DEBUG in ZRAID. A return to CP/M can be made without affecting CP/M's status by entering the & command to ZRAID. (or a coldstart can be done via "'".)

If the CNTRL-V exit from CP/M is not desirable (some CP/M programs use CNTRL-V as a command) use a parameter value of Ø8 (hex) when loading CP/M, as follows:

'8

During CP/M disk I/O, if an error occurs, ZRAID will output a message indicating the track and sector location and the error type. (This is in addition to the normal error message generated by CP/M.) The location of the error will be output as well. (The error types are described on page 34.) For example, the message "DISK ERR $Ø6-ØØ14.Ø5" means a CRC ERROR (error type 6 decimal) occurred on track 14 sector 5 (all numbers are decimal).



CP/M PARAMETER FORMAT

CP/M Diskette Configuration

Floppy diskette based system:

1. The system diskette is double density, 1024 bytes per sector, 8 sectors per track.
2. The system resides on track 1, starting at sector 1.
3. Track Ø is not used.
4. The system diskette must be obtained from Heurikon in order to assure proper operation of CP/M, SYSGEN and MOVCPM.
5. Drives are assigned as follows:

Drive Name	Physical Unit #	Configuration
A	Ø	DD-1024
B	1	DD-1024
C	2	DD-1024
D	3	DD-1024
E	Ø	SD-128
F	1	SD-128
G	2	SD-128
H	3	DD-128
I	WINCHESTER	-

Winchester based system:

1. The system must be on the Winchester drive (via SYSGEN).
2. The Winchester has space reserved for 2048 directory entries.
3. Drives are assigned as follows:

Drive Name	Physical Unit #	Configuration
A	WINCHESTER	-
B	1	DD-1024
C	2	DD-1024
D	3	DD-1024
E	0	SD-128
F	1	SD-128
G	2	SD-128
H	3	DD-128
I	0	DD-1024

The drive parameters may be changed by making modifications to ZRAID. Consult the factory if a special configuration is needed.

DD-1024 means double density floppy, 1024 bytes per sector.

DD-128 means double density floppy, 128 bytes per sector for MDS compatibility.

SD-128 means single density floppy, 128 bytes per sector for 1.4 compatibility.

See page 40 for a drive configuration chart.

Typical ZRAID Usage (without CP/M)

1. Push the system RESET button to enter ZRAID and initialize to the desired device. Note: If RTS or DTR is not TRUE on SIO Port B, ZRAID will automatically switch to Port A even if location RCRT was entered. See page 37.

2. Set location ØØØ8 (hex) to a JUMP to DEBUG (as a breakpoint) while debugging a program. Note: For use with CP/M, this step is not required; it is automatic.

3. Use ZRAID to load and edit the desired program or to boot strap CP/M. (See CP/M System Considerations)

4. Use the "*", C, or "←" commands to execute the program.

5. If the program "crumps" push the RESET button to re-enter ZRAID.

6. Make any necessary modifications to the program and repeat steps 4 through 7 as necessary.

7. Use the "R1" and "U" commands to insert and "march" a breakpoint (RST 1) through the program, if necessary, to examine the operation of the program. When the RST 1 is encountered a "CALL" to "DEBUG" will occur due to the JUMP to DEBUG at location ØØØ8.

Note: CP/M has a debugging program, "DDT", which has more power than ZRAID. ZRAID's main purpose is to allow easy access to all board facilities for testing of small programs.

Entry Points and Links

ZRAID entry points and addresses are maintained by a JUMP table at the beginning of ZRAID. This method allows future program versions to be compatible with older versions.

The entry points are as follows: (NOTE: Only the "L" half of the address is listed. The "H" half is FØ hex or 360 octal.)

Address "L"			Registers	
Hex	octal	Name	Affected	Function
ØØ	ØØØ	COLDSTART	all	ZRAID Power-On-Jump address. Performs I/O and memory init.
Ø3	ØØ3	DEBUG	none	Entry to save user registers. The previously enabled console device is used for I/O
Ø6		RPROC	none	This entry assumes the user program executed a CALL RPROC. If the console device is not ready (no key hit) a RET is executed. If the console has a character ready, ZRAID is given control at DEBUG and the user program is suspended. The user program status can be examined using ZRAID commands and a return to the user program can be effected with the "&" command.
Ø9	Ø11	RSIOAS	A,F	Test S1O A data ready status and set Register A and the condition code as follows: Character ready: non-zero Character not ready: zero
ØC	Ø14	RSIOAD	A,F	Test S1O A data ready status and wait for data ready. Return with the input character in Register A.

Address "L" Hex	 Octal	 Name	Registers Affected	 Function
ØF	Ø17	RSIOA	none	Output character in Register A to SIO port A
12	Ø22	RSIOBS	A,F	Same as RSIOAS but test SIO port B status.
15	Ø25	RSIOBD	A,F	Same as RSIOAD but input from SIO port B.
18	Ø3Ø	RSIOB	none	Same as RSIOA but output to SIO port B
1B	Ø33	RSTAT	A,F	Same as RSIOAS but test ZRAID console device.
1E	Ø36	RREAD	A,F	Input one character from ZRAID console device (SIO port A or port B) to register A.
21	Ø41	RWRITE	none	Output one character from register A to ZRAID console device.
24	Ø44	RPRINT	A,F	Output one character from register A to the non-ZRAID console device. (e.g. Printer)
27	Ø47	ROCT	A,F, D,E	Print contents of register A in octal to the ZRAID console device. Three digits plus a space are output.
2A	Ø52	RHEX	A,F, D,E	Print contents of register A in hexidecimal to the ZRAID console device. Two digits plus a space are output.
2D	Ø55	RWRITEHL	A,F, D,E	Print contents of HL in "Hxxx Lxxx" format to the ZRAID console device. Octal or hex output is determined by the current mode switch ("X" command) in ZRAID.
3Ø	Ø6Ø	RPOS	A,F	Output a CR,LF sequence to ZRAID console device.

Address "L"			Registers	
Hex	Octal	Name	Affected	Function
33	Ø63	RFDIO	A,F	Perform FLOPPY DISK I/O according to the parameter list address passed in H,L. See page 33.
36	Ø66	RIOSUBR	(all)	Output the I/O device initialization values (or character string) from the table whose address passed in H, L. Refer to the MLZ-91 User's Manual or the ZRAID-91 source code listing for details.
39	Ø71	RMAP	(all)	Set the memory mapping RAM from the table whose address is passed in H,L. Return to the address specified in register pair DE. Refer to the MLZ-91 User's Manual or the ZRAID-91 source code listing for details
3C	Ø74	(reserved)		
3F		(reserved)		
FFFE		---		Contains ZRAID91 Version number
FFFF		---		Contains 91 (hex)

Disk I/O Via ZRAID Routines

The disk I/O routines of ZRAID are accessible for general purpose disk I/O operations. The routines contain automatic track seek logic plus error recovery and retry logic. The calling sequence is as follows:

```
LZI  H,PARAMLIST
CALL RFDIO
```

All of the calling register values are saved and restored except A and F. The Parameter List ("PARAMLIST") must be preset to the following format:

PARAMLIST+Ø

D7	D6	D5	D4	D3	D2	D1	DØ
V	DD	DØØ	DSIZE	X	RI	C1	CØ

where: V = 1 to enable verification (i.e., read data following a write and compare or read again following a read and compare.)

DD= Ø Single Density
= 1 Double Density

DØØ= Ø Double Density only on tracks above ØØ
= 1 Double Density on all tracks

DSIZE= Ø Double Density sector size = 1024 bytes
= 1 Double Density sector size = 128 bytes
(Single density sector size is always 128 bytes)

RI= Ø Enable error recover retry operations
= 1 Inhibit error retry procedure

C1,CØ= COMMAND
ØØ = Initialize drive (RESTORE)
Ø1 = READ DATA
1Ø = WRITE DATA
11 = (invalid)

PARAMLIST+1	Starting data address-L half
PARAMLIST+2	Starting data address-H half
PARAMLIST+3	Desired drive (Ø-3)
PARAMLIST+4	Desired sector (1-52)
PARAMLIST+5	Desired track (Ø-76)
PARAMLIST+6	Reserved (future)
PARAMLIST+7	Reserved (future)

A successful I/O operation is indicated by a return with NO CARRY. An error is indicated by the CARRY flag set in which case register A will contain the error type. See the next section for error type meanings.

Refer to the ZRAID Source Code Listing for more details.

Disk I/O Error Types

Error Type		
hex		Error Description
Ø1	A	DISK DRIVE NOT READY The diskette is either not inserted, not inserted correctly, or there is a drive malfunction.
Ø2	B	SEEK ERROR, TRACK NOT FOUND The specified track cannot be found. The diskette is probably defective.
Ø3	C	WRITE PROTECTED DISK An attempt has been made to write on a write protected diskette.
Ø4	D	LOST DATA * One or more bytes of data have been lost during the transfer of data to or from the diskette. This generally indicates a timing problem due to unavailable facilities, such as a busy memory or DMA controller. Retry the original request.
Ø5	E	SECTOR NOT FOUND * The specified sector cannot be found or there is a CRC error in the ID field. The diskette is probably defective or the disk I/O hardware is malfunctioning.
Ø6	F	CRC ERROR * A CRC (Cyclic Redundancy Check) error has occurred in the transfer of data from the diskette. This generally indicates a defective diskette, dirty head, or a worn pressure pad.

See note page 36

Disk I/O Error Types (continued)

Error Type		
hex		Error Description
Ø7	G	RECORD TYPE NOT ØØ * The record type for the requested sector was not 'data'. This may not be an error if the sector has been marked as "deleted" by changing the sector's ID mark.
Ø8	H	ILLEGAL COMMAND The command type (PARAMLIST+Ø) was not recognized as a valid command.
Ø9	I	ILLEGAL TRACK OR SECTOR The track and/or sector value specified in the command is out of range. This is a calling software error.
ØA	J	(Not assigned)
ØB	K	VERIFICATION ERROR * After a write with verify command this error indicates that the data read back from the disk did not match the data block specified for writing. (Only the last byte is tested although the entire CRC is checked.)
ØC	L	LOST DATA ON VERIFY See error ØØ4 "D".
ØD	M	SECTOR NOT FOUND ON VERIFY * See error ØØ5 "E".
ØE	N	CRC ON VERIFY * See error ØØ6 "F".

Disk I/O Error Types (Continued)

Error Type hex		Error Description
ØF	O	(Not assigned)
1Ø	P	DRIVE SELECT ERROR The specified drive number is out of range. (Must be Ø,1,2,or 3.)
11	Q	WRONG DISK The disk in drive "A" (Drive Ø) is not a CP/M system disk. This error could occur during a CP/M bootstrap or warmstart.
12	R	MEMORY DMA WRAPAROUND The upper limit of address space (64K) has been reached in a disk RD/WR operation.
EØ		OPERATOR ABORT

Note: The decimal equivalent of the error number is displayed when using CP/M.

* The command has automatically been retried (unless inhibited by bit D2 of the command) and the disk head position has been checked prior to the error return. These errors may not be recoverable since more than one attempt has already been tried.

SIO Port Connections

The recommended connections are as follows: (Consult the MLZ-91 User Manual for details.)

	25 pin "D"		
	Connector	Name	Connections
Port A:	2	Tx Data	Data to device A
	3	Rcv Data	Data from device A
	4	RTS	CTS or open
	5	CTS	RTS, TRUE, or from device receiver control timing logic
	6	DSR	DTR or TRUE
	7	Ground	Ground
	20	DTR	DSR or open

CTS (Clear to Send) must be TRUE in order for ZRAID to transmit to the device. DSR (Data Set Ready) must be TRUE in order for ZRAID to receive data.

Port B:	2	Tx Data	Data from device B
	3	Rcv Data	Data to device B
	4	RTS	CTS or TRUE
	5	CTS	RTS or open
	6	DSR	DTR or open
	7	Ground	
	20	DTR	DSR or TRUE

RTS must be TRUE in order for data to be sent to the device and DTR must be TRUE for data from the device to be recognized. At least one (RTS or DTR) must be TRUE in order to initialize ZRAID to port B.

Note that Port A is configured as a data terminal device while Port B is setup as a data set.

Octal - Hex Conversion Functions

OUTPUT → INPUT ↓	H, L octal	16-bit octal	H, L hex
H, L octal	- -	Enter values via "H" and "L" commands Output 16-bit octal via "Z" command	Enter values via "H" and "L" commands Switch to hex mode via "X" command Output hex values via "A" command
16-bit octal	Enter value via "S" command Output values via "A" command	- -	Enter value via "S" command Switch to hex mode via "X" command Output hex values via "A" command
H, L hex	Enter values via "H" and "L" commands Switch to octal mode via "X" command Output values via "A" command	Enter values via "H" and "L" commands Output value via "Z" command	- -

Note: For all cases above, use the "X" command prior to entering the data, if necessary, to switch to the correct data base mode. Hex is the normal (at initialization) mode.

Enhancements - AUTOBOOT/AUTOSLAVE

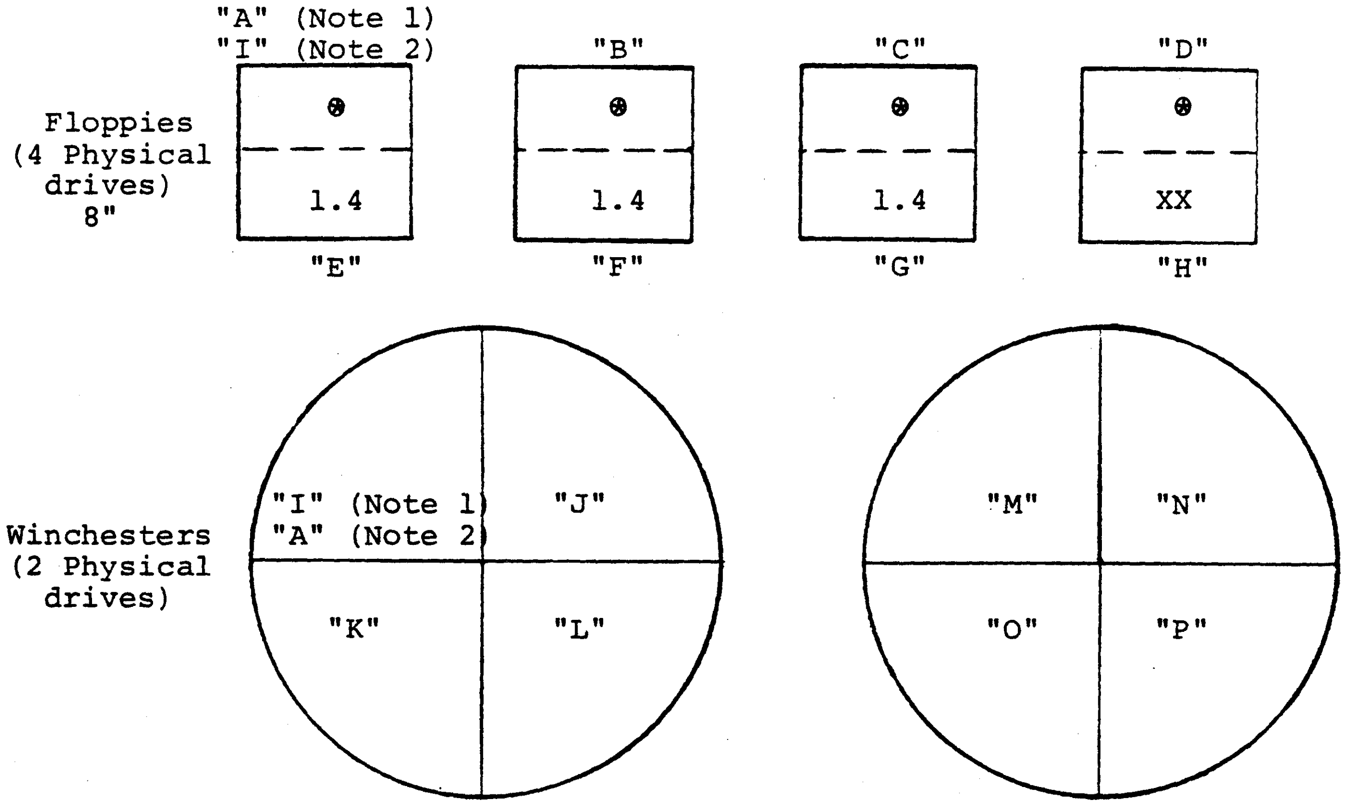
1. Normally, the apostrophe command is used to boot the CP/M operating system. However, ZRAID will automatically load CP/M following a hardware reset if DIP switch group zero, switches 5,6,7 and 8 are all "ON". AUTOBOOT

2. By setting DIP switch group zero, switches 1,2,3 and 4 all "ON", the monitor will automatically convert to "slave" mode and 100% RAM. This is useful for a multiprocessor system (e.g., when using MP/M). AUTOSLAVE

3. For systems using P3 to connect to a Centronics-type printer, use '2 command to boot CP/M. This will direct listings to P3 instead of SIO port A.

ZRAID Version 1.xS
(Single-sided, 8" Floppy Versions)

CP/M Drive Name Assignments



Key	Media Type
"⊛"	CP/M 2.2 (Heurikon), single-sided, double-density, 1024 b/s
"1.4"	CP/M 1.4 compatible, single-sided, single-density, 128 b/s
"XX"	Single-sided, double-density, 128 b/s (MFM)

Letters (e.g., "A", "B") refer to the CP/M drive name.

Note 1: When the system is loaded from floppy, "A" is the first floppy, "I" is part of the Winchester.

Note 2: When the system is loaded from Winchester, "A" is assigned to the Winchester and "I" is assigned to the floppy.