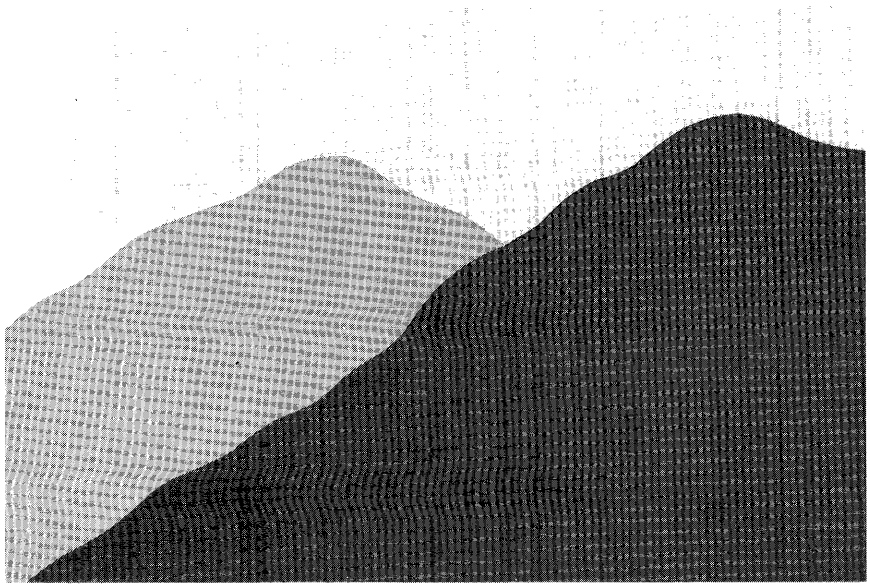


Hardware Reference Manual

RIDGE

9007

Hardware Reference Manual

Third Edition: 9007-C (SEP 85)

PUBLICATION HISTORY

Manual Title: Hardware Reference Manual

First Edition: 9007 (AUG 82)
Second Edition: 9007-B (APR 84)
Third Edition: 9007-C (SEP 85)

NOTICE

PREFACE

For each Ridge I/O board, this technical manual describes the control switches, indicator lights, pin assignments, internal cabling, theory of operation, and programming details. Each I/O board is optional equipment for the base system.

The base Ridge 32 system contains a CPU fetch board, CPU execute board, clock board, and a memory board, all of which are proprietary devices not explained in detail in this manual.

This manual may be reworked in the future into one or more manuals.

The Floppy Disc / Line Printer (FDLP) board is also a part of the base system, but is explained in more detail than the CPU and clock boards.

CONTENTS

Chapter 3: HARD DISC CONTROLLER BOARD

Chapter 8: SYSTEM DIAGNOSTICS

APPENDIX A: Z80 REGISTERS

ILLUSTRATIONS

Chapter 1
RIDGE 32 SYSTEM

STANDARD BOARDS

RIDGE 32 CARD CAGE

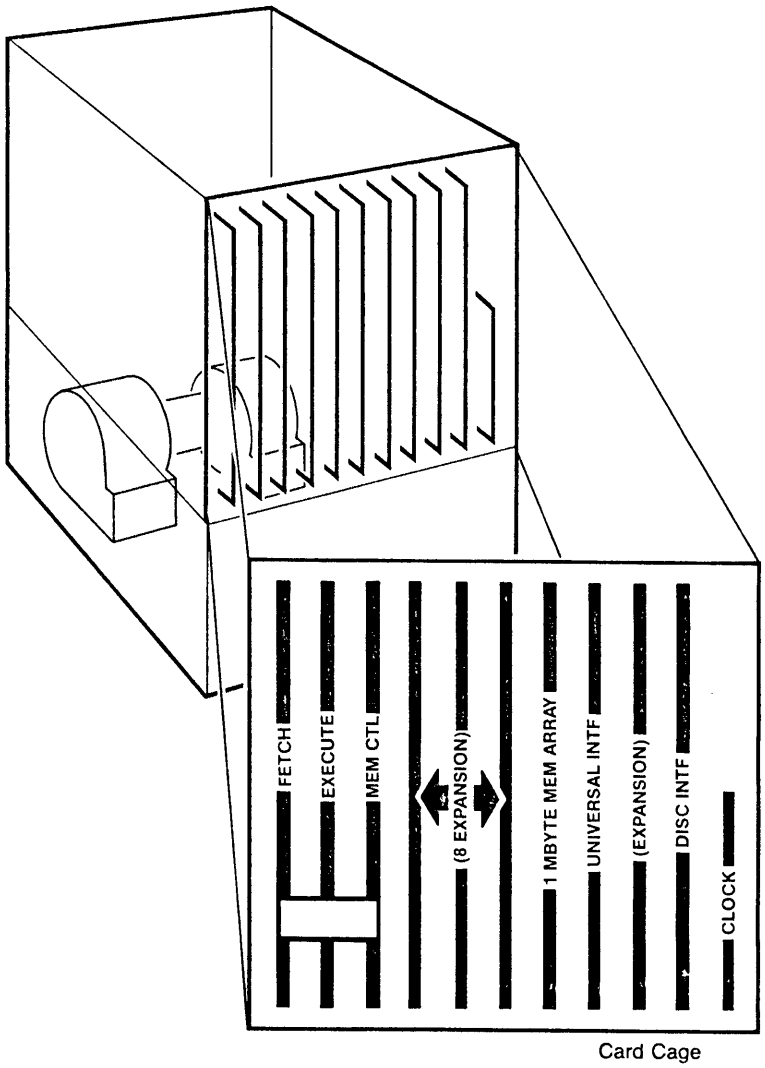
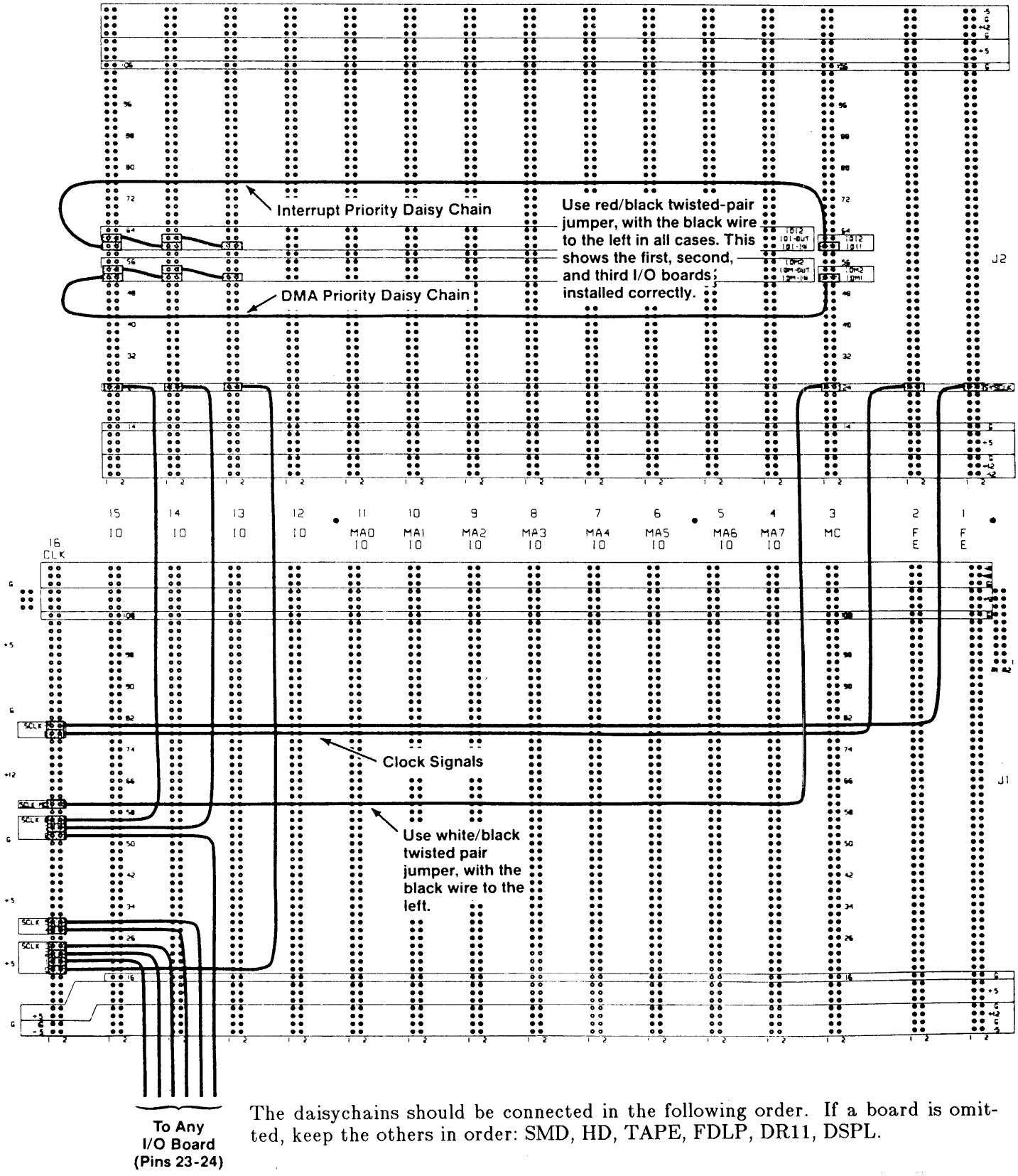


Figure 1-1. Placement of Boards in Ridge 32 Card Cage



The daisychains should be connected in the following order. If a board is omitted, keep the others in order: SMD, HD, TAPE, FDLP, DR11, DSPL.

Figure 1-2. Ridge 32 Backplane Jumpers

FRONT PANEL SWITCHES

Switch	Function
Power On/Off	Controls AC power.
Device 1/2	When in the rest position (position 1), system software will be loaded from the hard disc at load time. If pressed and held in position 2 at load time, the floppy disc drive is selected as the source of the system software.
Load	This is a momentary switch and, when depressed, resets the CPU and loads memory from device "1" or "2." In order to load from the floppy drive, depress and hold the device switch in the "2" position. Depress and release the load switch. After two seconds the floppy disc drive head loads, lighting the activity light on the floppy disc drive. Once the floppy disc drive begins loading, the device switch may be released.

CLOCK BOARD CONTROL SWITCHES

The Ridge clock board is a proprietary device which is not explained in detail in this book. The clock board is inserted in the Ridge card cage at the far right.

The six switches on the clock board, however, are described in the order that they are mounted on the board, from high to low position:

Switch	Function
Single-Clock	This momentary switch supplies the system with one clock pulse when depressed, if either the main clock switch or never-frozen clock switch is set to the frozen position.
Main-Clock	During normal operation this switch is set to the left position. When set to the right, the main system clock is frozen. Single clocks may be supplied by depressing the single-clock switch.
Never-Frozen Clock	During normal operation this switch is set to the left position. When set to the right, clock pulses supplied to refresh main memory are frozen. Single never-frozen clocks may be supplied by depressing the single clock switch.
Reset	This momentary switch resets the system hardware and loads memory from the device selected on the front panel. This switch differs from the front panel load switch in that never-frozen clocks are still provided, preserving main memory while still resetting system hardware.

Switch	Function
Load Enable	During normal operation, this switch is set to the left. The position of this switch can be tested by the software ELOGR instruction. This switch is used by microcode when recovering from a power glitch. When set to the left, "load is enabled" and the CPU resumes executing. When set to the right, "load is disabled" and the CPU begins executing at the switch 0 interrupt location in the CCB.
Switch 0	This momentary switch interrupts the CPU, causing execution to begin at the location of the switch 0 interrupt in the CCB.

The system is reset whenever AC power is applied, or the reset switch or load switch is depressed.

CLOCK BOARD INDICATOR LIGHTS

The clock board contains six light-emitting diodes (LEDs) that indicate system status. The LEDs are located below the set of switches. The LEDs and their functions are listed below.

LED	Function
Sync	Indicates that the clock board is generating clock signals. When the system is reset, this LED goes off for one-half second, then is relighted.
Lost DC	Indicates that DC power was interrupted some time previously. When the system is reset, this LED lights until the boot command from CPU microcode is issued.
bottom four	When all four are lighted, DC power supply of four is working properly.

In addition to the above switches and indicators, there is a jumper which disables the timekeeping facilities. Timer 1 and timer 2 interrupts are inhibited by placing a jumper across pins 37 and 38 of the edge connector on the clock board. The jumper is placed horizontally on the seventh from the top pair of pins.

SYSTEM BOOT

The system is booted whenever it is reset by turning on the power, or by pressing the front panel load button or the reset button on the clock board. When the system is booted, the CPU microcode sends a boot command to the selected I/O device. If the hard disc drive is selected, 4096 bytes are read from page 1 of the disc (0-origin) and placed in memory at location 3E000H. When the floppy

disc drive is selected, the entire double density track of 8192 bytes is read from head 0, track 2 and placed in memory at location 3E000H. After loading memory, the booting device interrupts the CPU, and the CPU begins executing in kernel mode at location 3E000H. SR11 (the CCB pointer) is set to 1, disabling timer 1 and timer 2 interrupts.

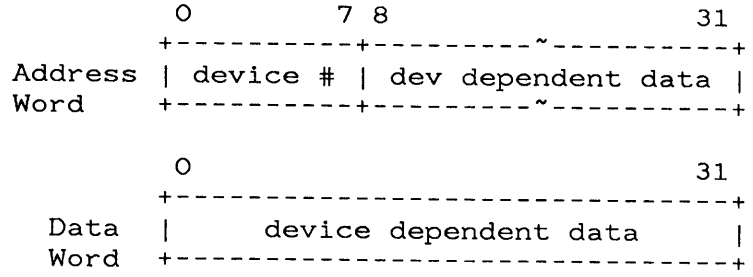
When loading from the floppy disc, turn on the power with the floppy door open. If a floppy error occurs during loading, the floppy disc controller will retry up to three times.

SELF-TEST

Any time the system is reset, the CPU self-test microcode is executed. When the system is working properly, the eight LEDs mounted on the edge of the execute board flash in sequence, from bottom to top. (The execute board is customarily inserted at the far left of the Ridge card cage.) All LEDs then go out, and the self-test continues. If there are no errors, the boot command is issued to the disc drive. If there is an error, the top LED goes on and stays on.

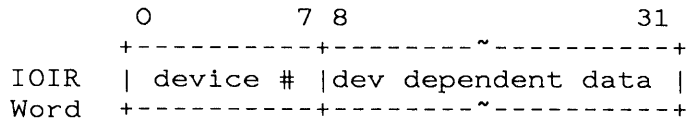
CONTROLLER BOARDS

The Ridge processor communicates to I/O interfaces using the WRITE and READ instructions, and the I/O Interrupt Read (IOIR) word on interrupts. The WRITE instruction sends an address word and a data word to a device in the following format:



The most significant byte of the address word is the device number, from 0-255. The remainder of the address word and data word contains device-dependent data. The READ instruction sends an address word to the device and receives a data word from the device on reply. If the device specified in the READ or WRITE instruction does not respond in two microseconds, an I/O timeout occurs and a 1 is placed as a return indication in Rx.

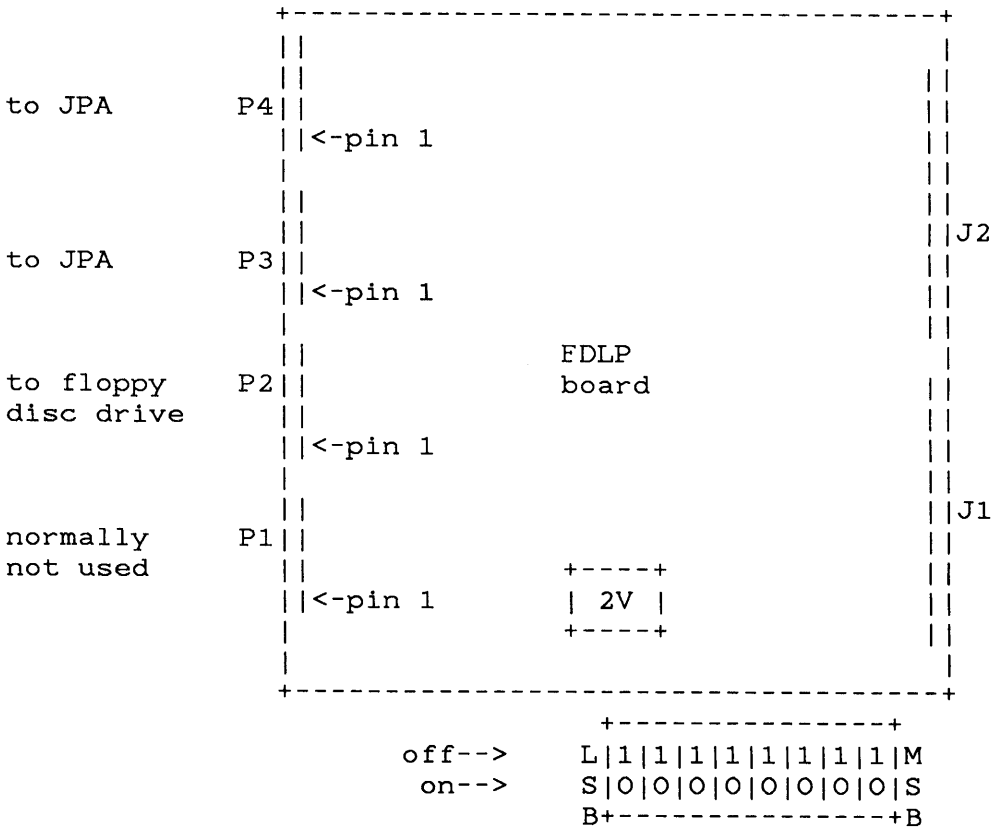
When an I/O device interrupts, the CPU microcode issues an I/O Interrupt Read, and receives an IOIR word. This word is placed in Special Register 0 (SR0) upon entry to the kernel. The format of the IOIR is below:


0
7 8
31
IOIR
Word
device #
dev dependent data

Chapter 2
FLOPPY DISC / LINE PRINTER CONTROLLER

FDLP CONTROL SWITCHES, INDICATOR LIGHTS, AND PIN ASSIGNMENTS

FDLP BOARD LAYOUT



NOTE: JPA is the Junction Panel Assembly on the Ridge back panel.

Figure 2-1. Position of Connectors on FDLP Board

FDLP BOARD CABLING

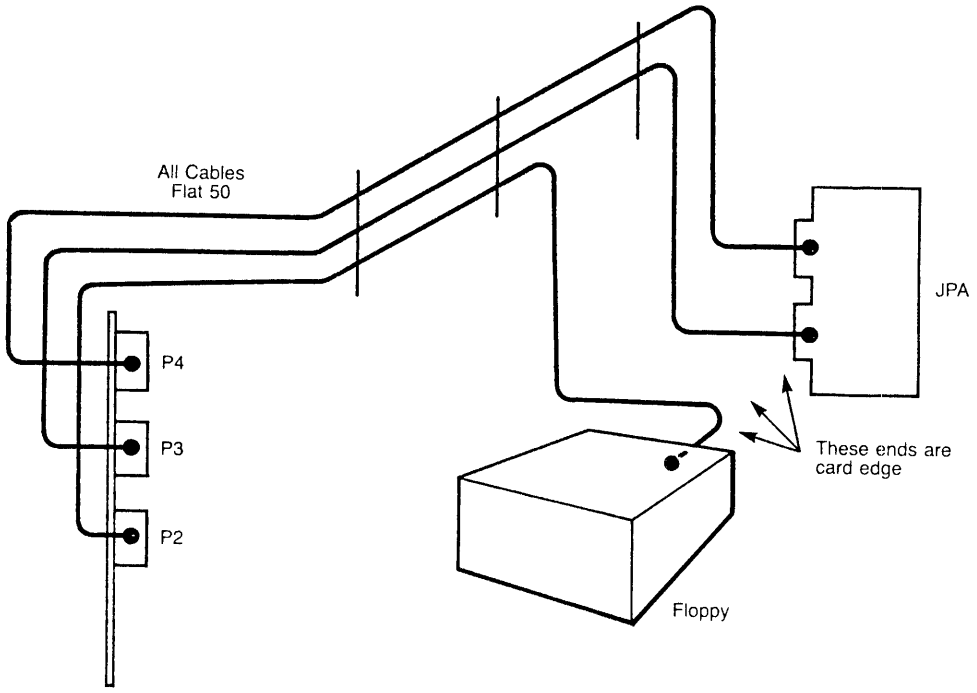


Figure 2-2. FDLP Board Cabling

FDLP BOARD P4 CONNECTOR

Pin	Signal Name	Pin	Signal Name
1	ground	26	ground
2	+ DP online	27	received data port1
3	+ DP ready	28	ground
4	+ DP top of form	29	transmit data
5	- Centronics fault	30	ground
6	- Centronics select	31	ring port2
7	+ Centronics paper out	32	clear to send port2
8	- Versatec online	33	data carrier detect port2
9	+ Versatec paper out	34	request to send port2
10	ground	35	data terminal ready port2
11	ring port0	36	ground
12	clear to send port0	37	received data port2
13	data carrier detect port0	38	ground
14	request to send port0	39	transmit data port2
15	data terminal ready port0	40	ground
16	ground	41	ring port3
17	received data port0	42	clear to send port3
18	ground	43	data carrier detect port3
19	transmit data port0	44	request to send port3
20	ground	45	data terminal ready port3
21	ring port1	46	ground
22	clear to send port1	47	received data port3
23	data carrier detect port1	48	ground
24	request to send port1	49	transmit data port3
25	data terminal ready port1	50	ground

FDLP BOARD P3 CONNECTOR

Pin	Signal Name	Pin	Signal Name
1	ground	26	Versatec Remote Form Feed
2	- Centronics acknowledge	27	ground
3	ground	28	Versatec Remote EOT
4	+ Centronics busy	29	ground
5	ground	30	Vers Remote Line Terminate
6	+ DP demand	31	ground
7	ground	32	Versatec Bit 1 (lsb)
8	+ Versatec busy	33	Versatec bit 2
9	ground	34	Versatec bit 3
10	+ Versatec P.I. clock	35	Versatec bit 4
11	ground	36	Versatec bit 5
12	+ DP strobe	37	Versatec bit 6
13	ground	38	Versatec bit 7
14	- DP/C strobe	39	Versatec bit 8 (msb)
15	ground	40	ground
16	- DP/C clear	41	ground
17	ground	42	DP/C bit 1 (lsb)
18	- Versatec print	43	DP/C bit 2
19	ground	44	DP/C bit 3
20	Vers simultan'us plot/prnt	45	DP/C bit 4
21	ground	46	DP/C bit 5
22	Versatec clear	47	DP/C bit 6
23	ground	48	DP/C bit 7
24	Versatec reset	49	DP/C bit 8 (msb)
25	ground	50	ground

FDLP BOARD P2 CONNECTOR FLOPPY DRIVE INTERFACE

Pin	Signal Name	Pin	Signal Name
1	ground	26	unit select 0
2	low current	27	ground
3	ground	28	unit select 1
4	fault reset	29	ground
5	ground	30	unit select 2
6	fault	31	ground
7	ground	32	unit select 3
8	not used	33	ground
9	ground	34	direction
10	two-sided	35	ground
11	ground	36	step
12	not used	37	ground
13	ground	38	write data
14	side select	39	ground
15	ground	40	write enable
16	head load 1	41	ground
17	ground	42	track 0
18	head load 0	43	ground
19	ground	44	write protected
20	index	45	ground
21	ground	46	read data
22	ready	47	ground
23	ground	48	head load 2
24	not used	49	ground
25	ground	50	head load 3

FDLP BOARD P1 CONNECTOR (TEST PORT)

Pin	Signal Name
1	ground
2	ground
3	switch 3
4	ground
5	switch 2
6	ground
7	switch 1
8	ground
9	switch 0
0	ground
11	external reset
12	ground
13	
14	ground

FDLP RS-232 PORTS ON JUNCTION PANEL ASSEMBLY (JPA)

The active pins on the four Ridge RS-232 ports are:

RS-232 Pin	Signal Name
1	ground
2	transmit data
3	receive data
4	request to send
5	clear to send
7	signal ground
8	carrier detect
20	data terminal ready
22	ring indicator

For instructions on modifying the baud rates, parity, etc. of the RS-232 ports, see the Ridge Operating System Reference Manual (part 9010), section 1, stty(1) utility program.

CENTRONICS PRINTER CABLE

Signal Name	Ridge End (on JPA)	Wire Color	Printer End
bit1	1	red	2
bit1 return	20	black	20
bit2	2	white	3
bit2 return	21	black	21
bit3	3	green	4
bit3 return	22	black	22
bit4	4	blue	5
bit4 return	23	black	23
bit5	5	yellow	6
bit5 return	24	black	24
bit6	6	brown	7
bit6 return	25	black	25
bit7	7	orange	8
bit7 return	26	black	26
bit8	8	white	9
bit8 return	27	red	27
clear	9	green	31
clear return	28	red	(30)
cstrobe	10	blue	1
cstrobe return	29	red	19
cbusy	11	yellow	11
cbusy return	30	red	29
cpaper	12	brown	12
cpaper return	31	red	(30)
cselect	14	orange	13
cselect return	33	red	(33)
cack	19	white	10
cack return	37	green	28
cfault	32	blue	32
cfault return	34	green	(33)

~~Connect 2 grounds together to pins 30 and 33.~~ Should not be connected together.

Ridge end connector is 37 pin D-sub male.

Printer end is Amphenol 57-30360.

DATA PRODUCTS CABLE (OLD STYLE WINCHESTER)

Signal Name	Ridge End (on JPA)	Wire Color	Printer End
bit1	1	red	B
bit1 return	20	black	D
bit2	2	white	F
bit2 return	21	black	J
bit3	3	green	L
bit3 return	22	black	N
bit4	4	blue	R
bit4 return	23	black	T
bit5	5	yellow	V
bit5 return	24	black	X
bit6	6	brown	Z
bit6 return	25	black	b
bit7	7	orange	n
bit7 return	26	black	k
bit8	8	white	h
bit8 return	27	red	e
clear	9	green	A
clear return	28	red	H
dptof	12	blue	S
dptof return	29	red	U
dpbof	13	yellow	M
dpbof return	30	red	P
dpready	15	brown	CC
dpready return	34	red	EE
dponline	16	orange	y
dponline return	35	red	AA
dpdmd	17	white	E
dpdmd return	36	green	C
dpstrobe	18	blue	j
dpstrobe return	37	green	m

On Winchester connector, tie p to s and v to x.

Ridge end connector is 37-pin D-sub male

Printer end connector is 50-pin Winchester MRAC50P-JTDH 8

DATA PRODUCTS CABLE (NEW STYLE 50 PIN D-SUB)

Signal Name	Ridge End (on JPA)	Wire Color	Printer End
bit1	1	red	19
bit1 return	20	black	3
bit2	2	white	20
bit2 return	21	black	4
bit3	3	green	1
bit3 return	22	black	2
bit4	4	blue	41
bit4 return	23	black	40
bit5	5	yellow	34
bit5 return	24	black	18
bit6	6	brown	43
bit6 return	25	black	42
bit7	7	orange	36
bit7 return	26	black	35
bit8	8	white	28
bit8 return	27	red	48
clear	9	green	31
clear return	28	red	15
dptof	12	blue	24
dptof return	29	red	8
dpbof	13	yellow	25
cbusy return	30	red	9
dpready	15	brown	22
dpready return	34	red	6
dponline	16	orange	21
dpdmd return	35	red	5
dpdmd	17	white	23
dpdmd return	36	green	7
dpstrobe	18	blue	38
dpstrobe return	37	green	37

Ridge end connector is 37-pin D-sub male.

Printer end connector is 50-pin D-sub male.

VERSATEC CABLE

Signal Name	Signal Pin	Common Pin	Pair	Mnemonics
input bit 1(lsb)	1	20	1	in01
input bit 2	2	21	2	in02
input bit 3	3	22	3	in03
input bit 4	4	23	4	in04
input bit 5	5	24	5	in05
input bit 6	6	25	6	in06
input bit 7	7	26	7	in07
input bit 8(msb)	8	27	8	in08
clear	9	28	9	-(clear)
parallel input clock	10	29	10	piclk
ready	11	30	11	-(ready)
printer	12	31*	12	print
parallel	13	31*	13	parin
simultaneous print/plot	14	33	14	-(spp)
remote reset	15	34	15	-(reset)
remote form feed	16	35	16	-(rffed)
remote end of transmission	17	36	17	-(reotr)
remote line terminate	18	37	18	-(rlter)
no paper sense	19**		19	nopap
on-line	32**		-(onlin)	

* Two commons tied to pin 31.

** No common for these two signals.

NOTE: There is no ground return provided for long line drivers and receivers.

FDLP THEORY OF OPERATION

FDLP INTRODUCTION

The FDLP board controls four RS-232 ports, two line printer ports, and is capable of controlling two 8-inch floppy disc drives. It has a local Z80 processor with its own local memory and various circuits and chips to interface to the various devices. All data passes through the local memory before going to or from the devices. The devices are serviced by the Z80 in a variety of ways: the serial terminal ports are interrupt-driven; the floppy disc has a Z80 DMA chip associated with it, so that the Z80 may service other devices while those transfers are in progress; the line printers are interfaced with Z80 OUT instructions supported with special logic to check the printer handshaking.

The software organization of the Z80 code is task oriented. Each unit has a task control state. Although interrupts are enabled at most times (for instance the SIO chips handling the RS-232 ports interrupt and are serviced), tasks are scheduled and run to a certain state, the state is recorded and a mechanism is enabled to reschedule the task, the task relinquishes control of the Z80, and a Z80 dispatcher chooses the next task to run. Normally, an interrupt occurs and the interrupt procedure inserts a request in a queue to restart a task at the recorded state.

The communication between the Ridge CPU and the board is:

1. I/O Write instructions start an operation.
2. The Z80 usually requests further information about the request, which it obtains from a dedicated area in Ridge memory called a Device Control Block (DCB).
3. If the operation is a device write, the Z80 will copy the data to its local memory.
4. The operation is performed.
5. If the operation is a device read, the data is moved from the local memory to Ridge memory.
6. A Ridge interrupt is generated and the I/O Interrupt Read that the Ridge CPU will perform will return the board's device number, the unit on the board, and an indication of the success of the requested operation.

FDLP BOARD LAYOUT

The primary logical parts of the board are :

1. I/O Read, I/O Write, and I/O Interrupt logic
2. a DMA sequencer to copy data in both directions between Ridge memory and Z80 memory. It has one word count register and two address registers: one for the Ridge address and one for the Z80 address

3. the Z80 CPU and its I/O decoders
4. 16 K bytes of dynamic ram
5. 8 K bytes of EPROM
6. a DMA chip to service the floppy disc chip
7. an NEC uPD765 floppy disc chip with associated buffer and an analog phase-locked loop (PLL) to generate a read clock for the floppy disc chip to decode the read data from the floppy drive
8. two Z80 SIO chips for the RS-232 ports and Z80 CTC chips for generating the baud rate clocks for the SIO's. Also RS-232 drivers and receivers to translate to/from TTL levels
9. two sets of line printer logic that check the line printer handshaking protocol as the Z80 writes to the printers. If the Z80 writes before the last data operation is complete, the operation is aborted before the point where the printers would see it, and the Z80 is interrupted. The Z80 interrupt procedure alters the path of instruction flow of the interrupted procedure. The printer logic also has the capability of monitoring the state of printer "ready" lines and generating interrupts on changes.

I/O READ, I/O WRITE, AND I/O INTERRUPT LOGIC

The I/O Read, Write, and Interrupt sections are small-state machines with four states.

Interrupt State Machine

The Interrupt state machine is normally in the idle state. The Z80 sets a bit in a register it can write to; this advances the state. When the Z80 resets that bit, the state machine advances the state again and asserts IOIREQ1 or IOIREQ2. It also blocks ACKIOIout so that, when multiple devices request an interrupt at the same time (or before the first device is acknowledged), any lower priority devices (further down the daisy-chain) will not see the ACKIOI signal, and will not gate their device numbers, but will continue to assert IOIREQ1 or IOIREQ2. When the board finally sees ACKIOIin, it advances the state once more and asserts its device number and the status register on the I/O bus. The state advances to the null state on the next clock unconditionally.

I/O Read State Machine

The I/O Read state machine is normally in the 0 (null) state. It enters state 1 on an MCIOREQ signal from the memory controller. From state 1, it advances to state 3 if three conditions are true at the clock edge:

1. MCIOREQ has remained true
2. MCIOWT is false (an I/O Read in progress) and

3. the high order byte on the Ridge I/O bus matches the value set in this board's device number switch.

If MCIOREQ is removed, the I/O Read state machine returns to state 0. If it reaches state 3, this board has been addressed with an I/O Read and the signal ACKMCIO is returned to the memory controller. From state 3, the state machine unconditionally advances to state 2 which gates the device number in the high order byte, the STATUS register in the next byte, and the low 16 address bits onto the Ridge I/O bus. After state 2, it advances to state 0.

I/O Write State Machine

The I/O Write state machine leaves its null state when it sees an MCIOREQ signal from the memory controller and enters state 1. From this state it will advance to state 3 if four conditions are true:

1. MCIOREQ has remained true.
2. MCIOWT is true (an I/O Write is in progress.
3. The high order byte on the Ridge I/O bus matches the value set in this board's device number switch.
4. CMDINT is false.

If MCIOREQ is removed, this state machine returns to state 0. If it reaches state 3, this board has been addressed with an I/O WRITE and the signal ACKMCIO is returned to the memory controller. From state 3, the state machine unconditionally advances to state 2, which continues to load the COMMAND register with the contents of the I/O bus. It will remain in this state until the signal IODACK is asserted by the memory controller. It will then return to state 0. The effect of this is that the COMMAND register will contain the contents of the high order byte of the I/O bus when the IODACK was issued (which is the I/O Write Data Word). State 2 also sets the CMDINT bit, which prevents further (successful) I/O Writes until it is reset and which interrupts the Z80 through the NMI pin.

DMA LOGIC

The DMA section contains a Word Count Register, a Ridge Address Register, and a Z80 Address Register. The transfers to or from Ridge memory are converted between 4-byte word format and single-byte data by a 74S194 universal shift register.

EXAMPLE: When reading from Ridge memory, the data is loaded into the eight 74S194s on the parallel inputs. The high order byte is available immediately for transfer to the Z80 local memory and successive bytes are available after one shift command because of the arrangement of bits in the shift register chips. Each chip has four bits of the 32 bit word which are 8 bit positions apart (e.g., one chip has bits 0, 8, 16, 24; another has 1, 9, 17, 25). When the transfer is from Z80 memory to Ridge memory, the shift register is loaded by shifting a bit serially from each of the 8 bits of the byte wide Z80 bus into each of the eight 74S194s. Again, only four shifts are required for the conversion between

byte and word. The Z80 memory transfers are coordinated with the Z80 CPU and Z80 DMA chip by obeying the conventions of BUSRQ and BUSAK defined for the Z80 family. When the state machines have control of the Z80 bus, they transfer four bytes at a time.

The DMA section has three interrelated state machines to control it:

1. a four-state sequencer called "MAIN" in the state diagrams which coordinates the startup process;
2. an eight-state sequencer called "RSM", which is the main generator of Ridge I/O bus memory signals; a four state sequencer called "ZREQ", which follows the Z80 BUSRQ and BUSACK rules;
3. and an eight-state sequencer called "DMA", which generates the Z80 memory strobes while doing a Z80 DMA cycle.

Before a DMA sequence begins, the Z80 program must set the direction of the transfer by setting the "Z'Write" bit. The addresses and word count registers also must be set. The Z80 then sets and then clears the "MSTART" bit to begin the DMA process.

When the "MSTART" bit is set, the "MAIN" state machine advances from state 0 to 1. On the next 125-ns clock cycle, the "RSM" advances from state 0 to 5. Normally, certain logic signals would be generated in state 5 but they are blocked because "MAIN" is still in state 1. While "MAIN" is in state 1, the "Z'Write" bit is copied to the "RWT" bit (Ridge Memory Write bit). By now, the DMA process has initiated, but is waiting for the Z80 to clear the "MSTART" bit.

On the next cycle after the "MSTART" bit is cleared, the "MAIN" state machine advances to state 3 for one cycle and then remains in state 2 until the whole DMA transfer is complete. "MAIN" state 2 is a condition in many request signals that originate in "RSM"; until it is true, very little happens. The "MAIN" state 3, which is only true for one cycle, advances the "RSM" state to 7 if "RWT" is clear (the DMA transfer will be from Ridge memory to Z80 memory) so that on the cycle where "MAIN" state 2 first enables everything, "RSM" will be either in state 5 if RWT is set, or in state 7 if RWT is clear.

Ridge to Z80 Memory Transfers

In this case, RWT is clear and the "RSM" begins its work in state 7. In this state, the signal IOMREQ is generated and the daisy-chain signal ACKIOMout is blocked from continuing to lower priority boards. When the board detects ACKIOMin become true, it advances its state to 6, where it removes IOMREQ but asserts the Ridge Memory Address on the bus.

On the next cycle, the RSM enters state 2, where it will stay until the memory controller asserts IODACK. In this state, the 74S194s are given a command to parallel load the Ridge I/O bus. For several cycles, they will be loading zeros (the state of the undriven bus), but on the cycle where IODACK is asserted they will have loaded the data returned by the memory controller. With IODACK, the RSM enters state 3 which is named "UNLOAD". In this state, the RSM

generates a signal "I WANT" (service from the Z80 section of the DMA machines). It will remain in this state until it sees a signal called "INC WC", which comes from the "DMA" state machine.

During the cycle where "INC WC" is asserted, two other signals are examined: the carry-out from the Word Counter and the "ZERO" bit (the most significant bit of the Word Counter). If either "CARRY" or "ZERO" is set, the next state will be state 1, which asserts no signals and is followed immediately by state 0, the null state. Otherwise, the next state is 7, which was discussed in the beginning of this section. The consequences of the signal "I WANT" will be discussed in following sections.

Z80 to Ridge Memory Transfers

In this case, the "RSM" remains in state 5 and various signals become active as "MAIN" reaches state 2. In particular, the signal "I WANT" is generated; this will cause activity in the two remaining state machines ("ZREQ" and "DMA") and will be discussed in the next paragraph. The "RSM" machine will stay in state 5 (named "load-up") until it sees the signal "INC WC". When it does (and the 74LS194s are loaded with a word of data for Ridge memory), the state will progress to state 7 where IOMREQ is generated and ACKIOMout is blocked. It will stay in this state until ACKIOMin reaches this board at which time the state will advance to state 6, also named the MADDR state. Here, the Ridge Memory address will be gated on the Ridge I/O bus. The next clock advances the state the MDATA state (state 4) where the contents of the 74S194s are gated onto the Ridge I/O bus. If the "ZERO" signal (the most significant bit of the Word Counter) is true, the next state will be state 0, the null state. If it is not, the next state is state 5 which was discussed at the beginning of this paragraph.

The "I WANT" signal causes the "ZREQ" state machine to advance from its null state (assuming that the Z80 DMA chip isn't currently asserting BUSREQ) to state 1, where it asserts BUSREQ to the Z80 CPU and Z80 DMA. If it sees BAO (Bus Acknowlege Out) for two cycles, it knows the Z80 CPU is ready to release the Z80 bus and that the Z80 DMA chip isn't currently requesting the bus. The "BAO" signal is the "BUSAK" signal from the Z80 CPU chained through the Z80 DMA chip. "BUSREQ" continues to be asseted and, until it sees "INCWC", the "ZREQ" machine remains in state 2 (where it takes control of the Z80 bus and memory control signals by asserting "ALTernateENable". When "INCWC" is asserted, the ZREQ machine returns to its null state.

The "DMA" machine generates the Z80 memory control signals required to read or write data into the local memory and the "INCBC" and "INCWC" signals which control the address and word counters. It also generates the 74S194 "Shift Left" signal, which converts between word and byte data formats.

On the next cycle after it sees "ALTernate ENable", it leaves its null state and, in state 1, asserts an alternate "MREQ" signal to the Z80 memory system. If "RWT" is true, it also asserts an alternate "RD" signal to the Z80 memory system. It advances to state 3 and asserts the same signals as state 1.

On the next 125ns clock, it advances to state 7 and asserts an alternate "WT" signal to the Z80 memory system if "RWT" is false. On the next clock edge, it advances to state 6 and asserts the same signals as it did in state 7, and also the 74S194 "Shift Left" signal if "RWT" is set.

On the next clock edge, in state 4, it asserts the 74S194 "Shift Left" signal if "RWT" is false, and asserts the "INCBC" signal to advance the address registers. If "INCBC" is true and the two least significant Ridge Address Register bits are both ones (this register contains the full byte address), the "INCWC" signal is generated, which which advances the "ZREQ" and "RSM" state machines as well as incrementing the Word Counter.

On the next clock cycle, the "DMA" state machine advances to its null state. Each time the "DMA" machine is started up, it loops through its states four times and transfers four bytes before the "INCWC" signal is generated, which causes the "ZREQ" state machine to change state back to its null state and remove "ALTernate ENable", which leaves the "ZREQ" machine looping in its null state.

Z80 CPU

The Z80 section is fairly conventional, with the exception that most of the I/O Write strobes are "shaped" by a pair of flip-flop bits that detect the first full cycle of the WT signal and generate pulses that are 125 or 250 ns wide.

Z80 MEMORY SYSTEM

There are four 2732 EPROM chips on the board and a fairly conventional implementation of a 16K-byte dynamic ram subsystem for the Z80. Its unusual aspect is the "alternate" control signals, which can come from the DMA machine described previously.

Z80 DMA CHIP

The Z80 DMA chip's Bus Acknowledge In and Out signals (BAI and BAO) are part of the chain between the Z80 CPU and the "DMA" state machine described earlier. The DMA chip's INT pin is decoded to provide the Terminal Count (TC) signal to the Floppy Disc Chip described next.

FLOPPY DISC CHIP

The floppy disc chip is a NEC uPD765 (Intel 8272). It provides most functions other than buffering, write precompensation, and read clock/data separation.

Write precompensation is performed by a shift register that delays Write Data from the chip by 82.5 ns per stage. The NEC 765 provides Early and Late signals that select between early, normal, and late data. The precompensation

selected also depends on the track (the outer tracks are not precompensated) and whether a single- or double-density operation is taking place (only double-density is precompensated). The normal precompensation time is 250 ns.

Read data/clock separation is done with a phase-locked loop. The raw read data is fed to the first one-shot, whose time period is set to one of two values selected by transistor T401, depending on whether double-density (MFM) or single-density (FM) is selected. In either case, the one-shot's value is set to half the nominal window size for the density selected (0.5 microseconds for MFM and 1.0 microsecond for FM). The output of the Voltage Controlled Oscillator (VCO) is divided by an appropriate constant to generate a square wave with an on-time similar to that of the one-shot.

Two flip-flops, with their clocks and clears cross-coupled to the shaped read data and VCO divided reference clock generate "pump up" and "pump down" signals, which are summed and integrated by an operational amplifier to adjust the frequency and phase of the VCO. The further divided VCO clock becomes the Read Window signal to the NEC 765. A one-shot with a short (20 ns) value triggered off the negative edge of the first one-shot's output is used as the Read Data input to the NEC 765.

The floppy disc chip's interrupt request pin is connected to one of the CTC channels to generate the correct Z80 interrupt responses.

Z80 SIO CHIPS

Two Z80 SIO chips are used in a conventional manner to provide four serial channels. The baud rate clocks are derived from the outputs of four channels of the channels of two Z80 CTC chips. They are used as programmable dividers of a 1.22825 M Hz signal obtained from an oscillator chip.

LINE PRINTERS

The line printer logic includes one four-state sequencer that contains the protocol for the Versatec printers, and one that can be configured to either follow the Data Products or the Centronics protocol. In both cases, when the Z80 attempts to write to the line printer port with an OUT instruction, the state machine is checked to be sure that it is in its null state and that the line printer is not busy. If either of those two conditions are false, no data strobe gets through to the printer. Instead, a signal to a CTC channel is generated and a wait state for the OUT (which is always at least one cycle long) is extended to eight cycles to allow the CTC time to interrupt. The next instruction executed will be that of some interrupt procedure (most likely that of the CTC channel just mentioned). In any case, the OUT is stopped and a stack marker laid down.

The interrupt procedure sets various flags and alters the return address to point beyond the OUT instruction. The Z80 code that generates the OUTs expects that to happen, and checks the flags and the "B" register to determine how many bytes were transferred. This circuitry allows the printers' buffers to be filled at the rate of OTIR instructions rather than a Z80 software loop that

would test the printer state machines. The two CTC channels used for the two-state machines can also be used to generate interrupts when the printers become ready so that, perhaps, a software task to write more data to the printers can be scheduled and eventually run.

FDLP BOARD PROGRAMMING

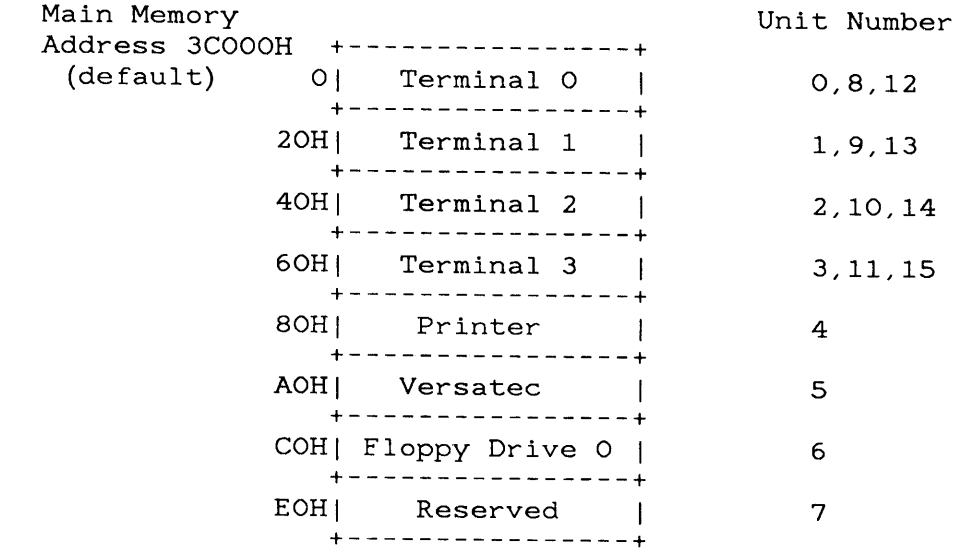
OVERVIEW

The Floppy Disc/Line Printer (FDLP) controller can control two floppy disc drives, four RS-232 ports, a DataProducts/Centronics-type printer port, and a Versatec-type printer/plotter port.

There are 16 units. Terminals have 3 units each.

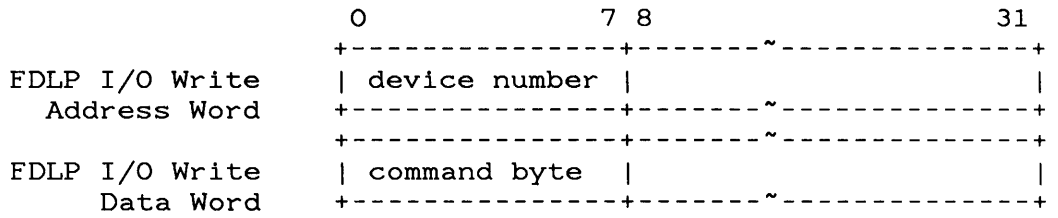
Unit(s)	Use
0 - 3	Terminal Writes
8 - 11	Terminal Reads
12 - 15	Terminal Control and Status
4	Versatec Printer/Plotter
5	DataProducts/Centronics Printer
6	Floppy Disc
7	reserved

The FDLP communicates with the Ridge CPU via Device Control Blocks (DCBs) in Ridge main memory, and command bytes received from I/O Write instructions. Each DCB is 32 bytes long and the eight control blocks for the FDLP are laid out as follows:



By executing a WRITE instruction specifying the FDLP's device number, the FDLP will perform an I/O function. The FDLP uses no information from the

WRITE address word other than the device number (by convention the FDLP is device 1), and the data word is laid out as follows:



In general, a command byte write *starts* an operation. The four least significant bits are the unit; the most significant are usually the hex digit 8 but, for single character reads from the terminal ports, it is the hex digit 9. A few other special bit combinations are also recognized.

Command Byte	Action
80	start I/O (write) on port 0
81	start I/O (write) on port 1
82	start I/O (write) on port 2
83	start I/O (write) on port 3
84	start I/O (write) to Centronics/Data Products
85	start I/O (write) to Versatec
86	start I/O to left floppy
87	start I/O to right floppy
88	start I/O (read) on port 0 (block read)
89	start I/O (read) on port 1 (block read)
8A	start I/O (read) on port 2 (block read)
8B	start I/O (read) on port 3 (block read)
8C	start a control operation on port 0
8D	start a control operation on port 1
8E	start a control operation on port 2
8F	start a control operation on port 3
98	read one character and interrupt port 0
99	read one character and interrupt port 1
9A	read one character and interrupt port 2
9B	read one character and interrupt port 3

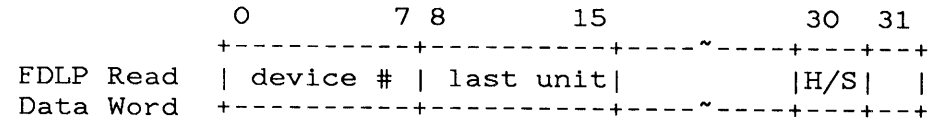
The following special commands are not used in normal operation:

00-7F	write one character on port 0 -- handshake is by bit 30 (special order only used by RBUG)
C0	boot from floppy unit 0
C1	move DCB base to contents of [3C0FC - 3C0FF] handshake by interrupt of 80 (unit 0)
C2	special read from port 0 -- no interrupts handshake is by bit 30

The FDLP buffers command bytes in the local processor's memory; however, it cannot accept them arbitrarily fast. The I/O Write instruction will set a status

bit which the on-board processor will reset when it has accepted the byte. Therefore, the Ridge Operating System must check this bit via an I/O Read before issuing a new I/O Write.

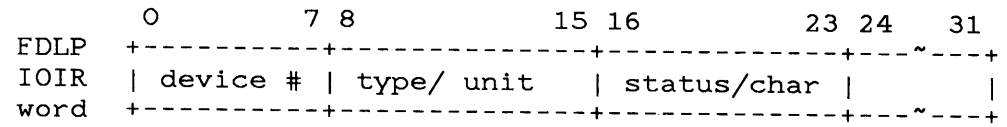
A read command (READ instruction) may be sent to the FDLP at any time. The FDLP returns its device number and last interrupting unit in the following format:



When performing an I/O function, the FDLP first reads the command byte. It usually then gets further information from the DCB and performs the operation. When it finishes, it usually modifies part of the DCB upon completion. Finally, it interrupts the Ridge CPU and passes back information in the I/O Interrupt Read word.

The IOIR values vary with the command byte that started the operation. Single character terminal reads return the character and status in the IOIR (rather than through the DCB). When "external status" interrupts are enabled for the terminal ports, the FDLP will interrupt whenever: 1) break is detected 2) Ring changes state 3) Data Carrier Detect changes state 4) Clear to Send changes state. The SIO status value is the new value of those bits (right after they changed state). Note that "external status" interrupts appear to come from the terminal control ports but have bit 11 set to distinguish them from the control/status command completion. By far the most common IOIR status simply contains the unit number (0-15) and the GSTAT value.

The format of the IOIR word is as follows:



Type/Unit	Status or Character	Class of Interrupt
1000UUUU	status	completion of block requests
1FOP10UU	char	single character terminal read
100111UU	SIO status	RS232 control line changes

U represents a bit of the unit number.

F represents a framing error on terminal reads

O represents a overrun on terminal reads of either the internal 256 byte buffer the local processor maintains or of the serial I/O chip

P represents a parity error on a terminal read

SIO is the Serial I/O chip; Ridge IOIR status bit 16 is Break; bit 18 is CTS; bit 19 is Ring; bit 20 is DCD.

FDLP BOARD PROGRAMMING FOR TERMINALS

The four terminal ports are addressed as twelve units on the FDLP board. There are three units per port so that a) both a read and write may be outstanding and b) so that either of the two operations (read or write) may be aborted without creating a confusing situation with the termination interrupts. As long as the control port exists, certain operations are assigned to the control port that could logically be read or write orders. However, unlike reads or writes, which can take an arbitrarily long time to complete and which may interrupt before they finish (i.e., most writes interrupt when they are started), the control orders are finished when the completion interrupt is generated and do not wait on external events.

The sixteen units are:

0 - 3	terminal writes
4	Centronic/Data Products lineprinter
5	Versatec plotter/printer
6	standard floppy drive (left)
7	additional floppy drive (right)
8 - B	terminal reads
C - F	terminal control

The DCB orders for the terminal write, read, and control are actually intermixed on a unit basis to group the functional parts of a terminal port together.

The DCB structure previously described can actually be interpreted as:

```
(default starting  +----------------------+  UNIT
 address) -> 3C000H|   Terminal 0 Writes  |    0
             0CH|   Terminal 0 Control |    12
             10H|   Terminal 0 Reads   |    8
             20H|   Terminal 1 Writes  |    1
             2CH|   Terminal 1 Control |    13
             30H|   Terminal 1 Reads   |    9
             40H|   Terminal 2 Writes  |    2
             4CH|   Terminal 2 Control |    14
             50H|   Terminal 2 Reads   |    10
             60H|   Terminal 3 Writes  |    3
             7CH|   Terminal 3 Control |    15
             70H|   Terminal 3 Reads   |    11
             80H|   Printer            |    4
             AOH|   Versatec           |    5
             COH|   Floppy Drive       |    6
             EOH|   reserved           |    7
                +----------------------+
```

On the terminal ports, a distinction is made between "block" and "single-character" reads and writes.

The single character write has a GORDER = 3 and has the character in SORDER (to save another data transfer operation from a Ridge memory buffer). If the write character output queue is not full, the character is added and the termination interrupt generated. If the buffer (256 characters) is full, the operation is delayed until the buffer empties.

On block writes, the characters are in a Ridge memory buffer pointed to by the DCB. If the buffer is empty, a copy is made of the buffer in Z80 memory and the operation is started. At that point, the termination interrupt occurs. If the buffer is not empty, the copy is not made until the current writing is finished.

The single character read returns the data in the IOIR word. Those orders are distinguished by the command byte being in the range 98-9B (normally orders are 80 + unit). The block reads are 88 - 8B and transfer the data back to a Ridge memory buffer pointed to by the DCB. Like all buffer pointers in the DCB, these must start on a Ridge word boundary, and not across a page boundary (actually a 64K-byte boundary). The read buffer is 256 bytes so, for a double buffered effect, the block read order should not exceed 128 bytes. The block reads can be terminated by

a. the specified byte count
b. the presence of the character in the TERMINATION CHARACTER MAP for the particular port or
c. a control order to terminate unconditionally (perhaps in response to a time out or process abort in the Ridge).

The MAP is an array of 256 bytes in Z80 memory. The characters in the block read are used as an index into the array and four bits of each byte are flags for the four terminal ports. The leftmost bit of each byte is the flag for port 0, etc. The rightmost four bits are currently unused. The MAP can be read or written by Ridge with control orders. It is initialized by the FDLP to terminate only on char = 3 (ETX) and char = D (carriage-return).

TERMINAL WRITE DCBs

These are command bytes 80-83. All interrupt with IOIR.(8:15) = 80 to 83 The following hex address offsets are added to DCB-BASE (normally 3C000H) plus 000H for terminal port 0, or 020H for terminal port 1, or 040H for terminal port 2, or 060H for terminal port 3.

Hex Addr	Name	Function
0	GORDER	1- block write of up to 256 characters from Ridge memory buffer 3 - single character write of character in SORDER 5 - assert 'break' for the number of character times contained in high-byte-count : low-byte-count
1	SORDER	used with GORDER = 3 as the single character to write
2	GSTAT	not used
3	SSTAT	not used
4	RETRI	Enot used
5-7	RIDGE ADDRESS	This 24 bit byte address must be on word boundary.
8-9	REQUEST BYTE COUNT	1 - 256 bytes
A-B	BYTE COUNT TRANSFERRED	

TERMINAL READ DCBs

These are command bytes 88-89. All interrupt with IOIR.(8:15) = 88 to 89 (possibly Framing error, Overrun, or Parity errors may be set also). The following hex address offsets are added to DCB-BASE (normally 3C000H) plus 010H for terminal port 0, or 030H for terminal port 1, or 050H for terminal port 2, or 070H for terminal port 3.

Hex Addr	Name	Function
0	GORDER	o - block read terminated by byte count or character's presence in MAP
1	SORDER	not used
3	SSTAT	not used
4	RETRIES	not used
5-7	RIDGE ADDRESS	This 24 bit byte address must be on a word boundary

Hex Addr	Name	Function
8-9	REQUEST BYTE COUNT	1 to 256 bytes
A-B	BYTE COUNT TRANSFERRED	

TERMINAL CONTROL DCBs

These are command bytes 8C-8F. All interrupt with IOIR.(8:15) = 8C to 8F The following hex address offsets are added to DCB-BASE (normally 3C000H) plus 00CH for terminal port 0, or 02CH for terminal port 1, or 04CH for terminal port 2, or 06CH for terminal port 3.

Hex Addr	Name	Function
0	GORDER	0 - return termtype in Parm1 1 - set termtype from Parm1 2 - return baud rate in Parm1 3 - set baud rate in Parm1 4 - return present settings of SIO chip registers 4, 3, 5 in Parm1-3 5 - set SIO chip register 4,3,5 from Parm1-3 6 - abort a read in progress 7 - abort a write in progress 8 - read MAP into Ridge address contained in Parm1-3 9 - write MAP from Ridge address contained in Parm1-3
1	PARM1	control/status
2	PARM2	control/status
3	PARM3	control/status

The "term-type" byte consists of the following bits:

bit 0 the most significant bit - If set, the port stops writing when it receives an XOFF character (also called DC3 or control-S, whose hex value is 13H or 93H), and resumes writing when it receives an XON (also called DC1 or control-Q, whose hex value is 11H or 91H). The XON and XOFF characters are not passed to the Ridge CPU.

bit 1 If set, the port inserts an XOFF (or DC3 or control-S) when its input buffer reaches 250 characters out of 256, and inserts an XON (or DC1 or control-Q) when the buffer drops to 150 characters. The characters are inserted into the stream of characters being written on that port as soon as the current character finishes, or immediately if no write is in progress. They will also be written if the output is currently stopped because of having received an XOFF (see bit 0 description).

bit 2 If set, the port clears its Data Terminal Ready line when its input buffer reaches 250 characters, and sets DTR when the buffer drops to 150.

bit 3 If set, interrupts can be sent to the Ridge CPU when the terminal port detects changes in the states of Break, Clear to Send, Data Carrier Detect, and Ring.

bits 4 to 7 reserved.

Special orders to copy Z80 code to or from Ridge memory and to cause the Z80 to call a given address are presently grouped with the Centronics/ Data Products write orders. They are line printer GORDERS of:

3 read the Z80 addresses contained in PARM0:PARM1 into Ridge memory with the address and byte count specified in the usual way.

4 write the Z80 addresses contained in PARM0:PARM1 into Ridge memory with the address and byte count specified in the usual way.

5 call the Z80 address contained in PARM0:PARM1

The Parm0 and Parm1 refer to offsets of 00CH and 00DH within the line printer GORDER.

FDLP BOARD PROGRAMMING FOR THE FLOPPY DISC DRIVE

The Ridge floppy disc drive accepts soft-sectored, single-sided or double-sided, single-density or double-density floppy discs. The FDLP handles eight different formats, which are listed in the DCB table below. There are 77 tracks on a floppy disc, numbered from 0 to 76. Sectors are numbered using 1 as the first sector (1-origined). The FDLP uses an NEC floppy disc controller to handle the details of floppy disc drive control.

The Ridge operating system (ROS) accepts only double-sided, double-density floppy discs with 512 bytes per sector, sixteen sectors per track. Head 0, track 0 is formatted single-density, with sector 1 containing format information for the remainder of the disc: single-sided or double-sided, single or double-density, and number of bytes per sector. Head 1, track 0 contains a UCSD Pascal directory in sectors 1-10. File storage begins in sector 11, track 1, head 0. The capacity of a floppy disc is calculated as follows:

```
Track 0, head 0                                    0 bytes
Track 0, head 1, sectors 11-16.
    6 sectors X 512 bytes/sector =             3,072 bytes
Tracks 1-76.  16 sectors/track X
    2 heads/track X 512 bytes/sector X
    76 tracks                          =   1,245,184 bytes
                                           ---------
                                           1,248,256 bytes
```

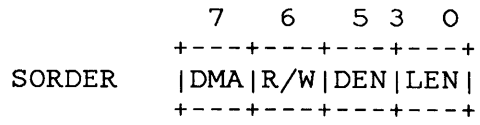
FLOPPY DISC DEVICE CONTROL BLOCK (DCB)

The layout and function of the floppy disc drive device control block is described in the table below:

Hex Addr	Name	Function
0	GORDER	Following are the eleven general orders. For the first eight orders, "read" refers to data being transferred into Ridge main memory, while "write" refers to data being transferred from Ridge main memory to the FDLP. The term "FDLP build" refers to fact that the FDLP will fill in the details of the NEC commands in the CMD00 - CMD08 area described below for read, write and format. Please refer to the NEC uPD765 documentation on orders that are not built by the FDLP. 0 Read, FDLP build, with implied seek. 1 Write, FDLP build, with implied seek. This command is also used for formatting. 2 Read, with implied seek. 3 Write, with implied seek. 4 Read, FDLP build. 5 Write, FDLP build. 6 Read. 7 Write. 8 Seek, requires only HEAD/UNIT and TRACK. 9 Recalibrate, requires only HEAD/UNIT and TRACK. Seeks to track specified. A Return device status (NEC STATUS3) in GSTAT.

Hex Addr	Name	Function

1 SORDER Provides suborder information. This byte is laid out below:



DMA When this bit is set, there is no DMA activity, i.e., the NEC chip only returns status.

R/W When this bit is set, this indicates a write to the NEC chip (Format is a write).

DEN This selects density (this field is used in FDLP build mode only).

0 Double-density, 512 X 16
1 Single-density, 128 X 26; IBM diskette 1
2 Single-density, 256 X 15; IBM diskette 2
3 Double-density, 256 X 26; IBM diskette 1D
4 Double-density, 1024 X 8; IBM diskette 2D
5 Double-density, 512 X 15
6 Double-density, 2048 X 4
7 Double-density, 4096 X 2

LEN NEC command length. This field must be 0 for reads and writes using FDLP build and 6 for formatting with FDLP build.

2 GSTAT General status:

0 OK
1 Not ready
2 Timeout
3 Equipment fault
4 Write protected
5 Ridge double bit error in data or DCB transfer
6 Data overrun
7 Missing address mark
8 Can't find header

Hex Addr	Name	Function
		9 CRC error in header
		A CRC error in data
		FF Illegal parameter in DCB order
3	SSTAT	Special status (from MSB to LSB): Bit 7 Fault Bit 6 Write protected Bit 5 Ready Bit 4 Track 0 Bit 3 Two-sided Bit 2 Head Bit 1-0 Unit
4	RETRIES	The number of retries attempted on this request.
5-7	RIDGE ADDRESS	This address must be on a word boundary. The Ridge address plus the request byte count must not cross a 64KB boundary.
8-9	REQUEST BYTE COUNT	
A-B	BYTE COUNT TRANSFERRED	
C	NEC ORDER	This is the NEC first command byte: 5 Write single-density 6 Read single-density 45H Write double-density 46H Read double-density 4DH Format a track double-density Other orders are available; please refer to NEC uPD765 documentation. The format order requires four bytes of information from Ridge main memory to write into each sector. This allows sector labels to be ordered in any fashion. The four bytes per sector are: 1. Track number 2. Head (0 or 1) 3. Sector number (1-origined) 4. Code for number of bytes per sector: 0 - 128 3 - 1024 1 - 256 4 - 2048 2 - 512 5 - 4096

Hex Addr	Name	Function
D	HEAD/UNIT	Head 0/1 is selected by bit 2, unit number is in bits 0-1 (bits numbered 7 for most significant, 0 for least significant).
E	CYLINDER	Number from 0 to 76 (decimal).
F	SECTOR	1-origined number.
10-	CMD00-	Please refer to NEC uPD765 documentation.
18	CMD08	These bytes are filled in by the FDLP if the "FDLP" build option is specified in GORDER.
19-	STAT00	These are the NEC chip status bytes.
1F	STAT06	Please refer to the NEC uPD765 documentation. These bytes are not valid for the return device status order. For the seek and recalibrate orders, only STAT00 is valid.

VERSATEC AND PRINTER DEVICE CONTROL BLOCK (DCB)

Hex Addr	Name	Function
0	GORDER	Set to 1.
1	SORDER	Sub order (from MSB to LSB): Bit 7 Plot (Versatec only) Bit 6 Simultaneous print & plot (Versatec only) Bit 5 Dataproducts/Centronics mode (not for Versatec), set to one selects Centronics Bit 4 Clear, done before data transfer Bit 3 Reset (Versatec only), done before data transfer Bit 2 RFFED (Versatec only), remote form feed, done after data transfer Bit 1 REOTR (Versatec only), remote end of transfer, done after data transfer Bit 0 RLTER (Versatec only), remote line terminate, done after data transfer
2	GSTAT	0 - OK. 1 Offline 2 Powered down or not connected

Hex Addr	Name	Function
		FF - Byte count too high, or other illegal request
3	SSTAT	Not used.
4	RETRIES	Must be 0.
5-7	RIDGE ADDRESS	This address must be on a word boundary. The Ridge address plus the request byte count must not cross a 64KB boundary.
8-9	REQUEST BYTE COUNT	
A-B	BYTE COUNT TRANSFERRED	

FDLP MONITOR PROGRAM

The floppy disc controller uses a Z80 microprocessor to control many of its functions. It contains a monitor program in read-only memory that can be used as a diagnostic tool to debug the devices connected to the FDLP board. The FDLP monitor has the commands to display and modify Ridge main memory that can be used to check the functioning of the memory controller and memory array boards. The monitor program is activated on terminal 0 port whenever the system is reset, or control-Z is input to the terminal 0 port when RBUG is active. The monitor prints a banner and a prompt, ">" when the system is reset, and a prompt without the banner when activated by control-Z. The FDLP monitor commands are listed below.

Special Characters and Syntax Notation

The characters control-X and control-H terminate input and print "***". Control-S may be typed to suspend output from the monitor; typing any subsequent character except control-X or control-H resumes output. Control-X and control-H terminate output. In he syntax used below "[" and "]" surround optional components, "{" and "}" indicates one from the set is required, and terms enclosed by "<" and ">" are defined following the list of commands. All command names and components are separated by at least one lank. Numeric values are all in hex.

Definition of Terms

<unit>

Unit 0 is the unit number of the unit mounted in the Ridge system.

<density-code>

0 - double-density 512 X 16
1 - single-density 128 X 26 IBM Diskette 1
2 - single-density 256 X 15 IBM Diskette 2
3 - double-density 256 X 26 IBM Diskette 1D
4 - double-density 1024 X 8 IBM Diskette 2D
6 - double-density 2048 X 2
7 - double-density 4096 X 1

<loop-option>

0 - Do one operation.
1 - Scan from track 1 to 76, forever.

Display Z80 Memory Command

D *address [length]*

Displays memory. Read-only memory starts at address 0. Read/write memory occupies addresses C000H through FFFFH. Typing control-S while memory is being displayed temporarily suspends the display. Typing control-X or control-S terminates the display, while typing any other character resumes displaying memory.

Display Z80 Registers Command

DR

Displays the values of Z80 registers at time of the last breakpoint.

Read Diagnostic Floppy Disc Command

DY <unit>

{0} Reads a Dysan Digital Diagnostic Disk in double
{1} Density mode if 0 is specified, single-density mode if 1 is specified. The drive mounted in the Ridge cabinet is <unit> 0.

Format a Floppy Disc Command

FM <unit> <density-code>

Formats a floppy disc in the density specified. <density-code> is described below. Track 0, head 0 is always formatted single-density, 128 X 26. The FDLP senses whether the floppy disc is single-sided or double-sided and formats both sides of a double-sided floppy disc.

Format a Track Command

FT <unit> <density-code> track

Read a Port Command

I *port-number*

Prints the results of a read from the port specified.

Read a Port, Repeatedly, Command

IR *port-number*

Prints the results of a read from the port specified repeatedly, until carriage return is typed.

Jump to Address Command

J *address*

Start executing Z80 code at the specified address.

Download Z80 Memory Command

L*aaaaccccddd...ddd*

Binary data is loaded into memory at address "aaaa" for count "cccc". The address and count are hexadecimal and in Z80 form, i.e., least significant byte, most significant byte. "ddd...ddd" represents the binary data, these bytes immediately follow the address and data bytes.

Modify Z80 Memory Command

M *address*

The byte of data at the specified address is displayed in hex. A new value may then be input, followed by carriage return. After carriage

return, the next sequential address is displayed, and can then be modified. Typing carriage return leaves the value unmodified. Modify mode is ended by typing any non-hex character.

Write to a Port Command

O *port-number value*

Writes "value" to the port specified.

Write to a Port, Repeatedly, Command

OR *port-number value*

"value" is sent to the port specified repeatedly, until carriage return is typed.

Display Ridge Memory (Peek) Command

P *ms-Ridge-address ls-Ridge-address length*

Display Ridge real memory, functions similar to Display Memory command. "ms-Ridge-address" are the 8 most significant Ridge address bits. "ls-Ridge-address" are the 16 least significant Ridge address bits.

Modify Ridge Memory (Poke) Command

PO *ms-Ridge-address ls-Ridge-address*

Modifies Ridge memory, functions similarly to modify memory command.

Print Memory to Data Products Printer Command

PD *address length*

Functions similarly to display memory command, but output is sent to the DataProducts printer port.

Print Memory to Versatec Printer Command

PV *address length*

Functions similarly to display memory command, but output is sent to the Versatec printer/plotter port.

Quiet Mode Command

Q

Puts monitor into "quiet mode", and interrupts the Ridge CPU. Characters read or written from terminal port 0 are now passed to the Ridge CPU. The quiet mode command effectively exits the monitor after entering via control-Z.

Read from Floppy Drive Command

R *head/unit* <density-code> *length* <loop-option>

Reads from a track on a floppy disc and places the data in Z80 memory. "head/unit" is:

0 - unit 0, head 0.
1 - unit 1, head 0.
4 - unit 0, head 1.
5 - unit 1, head 1.

<density-code> and <loop-option> are described in the section following the commands. "length" is the number of bytes to be read. "length" <= 1000H results in the data being placed at location E000H. "length" > 1000H results in the data being placed at location D000H. After typing this command line the monitor prompts:

TRK REC?

The track and record (sector) number should then be typed, separated by spaces. If an error occurs, the NEC status bytes are displayed, otherwise the monitor issues a prompt.

Recalibrate Floppy Drive Command

RC <unit>

"Step" commands are issued until the "track 0" signal from the drive becomes true.

Read Floppy ID

RI <unit>

Prints results of the NEC read ID command. The single-density results are printed, and if this failed, the double-density results are printed on the following line. Please refer to NEC uPD765 documentation for details on the format of this information.

Sense Floppy Drive Command

SD <unit>

Prints results of the NEC sense drive command.

Seek Floppy Drive Command

SK <unit> *track*

Causes the selected unit to seek to the specified track.

Switch RBUG Port Command

SW *terminal port*

Switches the RBUG to the specified port. ???

Transfer Data Between Ridge Memory and Z80 Memory Command

T *ms-Ridge-addr ls-Ridge-addr Z80-addr length direction*

This command transfers a block of data either from Ridge memory to Z80 memory or from Z80 memory to Ridge memory. "ms-Ridge-addr" are the 16 most significant Ridge address bits. "ls-Ridge-addr" are the 16 least significant Ridge address bits. "Z80-addr" is the Z80 memory address. "length" is the number of bytes to transfer. "direction" is 0 for read from Ridge memory, 1 for write into Ridge memory.

Exercise Printers Command

V

{0 }	sends print data to both printer ports. "0"
{1 char}	sends 256-byte writes to the Versatec. "1" sends the Versatec "char" in 4096-byte writes.

Write to Floppy Drive Command

W *head/unit* <density-code> *length* <loop-option>

Writes data from Z80 memory to a track on a floppy disc. Operates similarly to the read from floppy drive command.

Execute Test Programs Command

X *program*

There currently are no test programs for the FDLP (The hard disc monitor does have test programs, however).

Execute Z80 LDIR Command

Y *target-address source-address byte count*

Performs a Z80 LDIR instruction.

Exercise Terminal Ports Command

Z

256-byte writes are sent continuously to terminal ports one through three.

CHAPTER 3
HARD DISC CONTROLLER BOARD
(for 60-142 Mb Disc)

HARD DISC SWITCHES, INDICATORS, AND PINS

HARD DISC BOARD LAYOUT

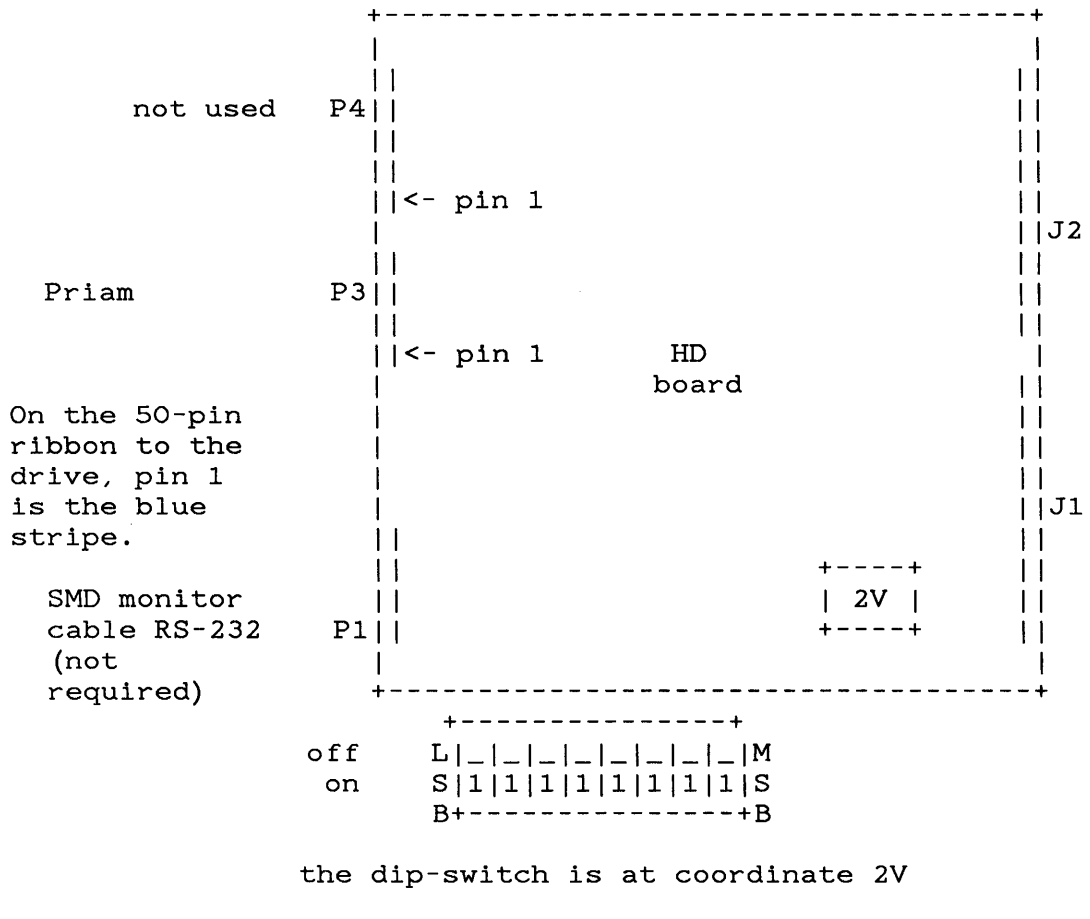


Figure 3-1. Position of Connectors on SMD Board

HARD DISC BOARD CABLING

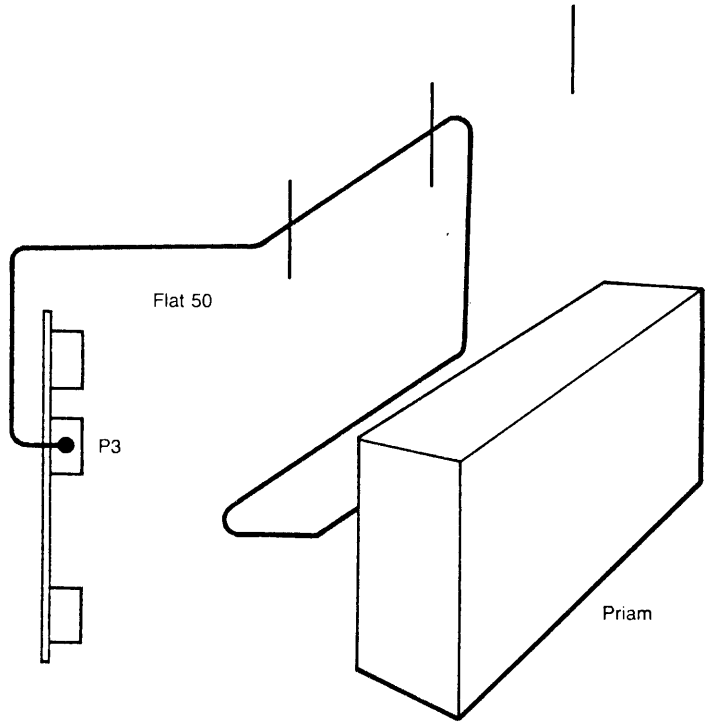


Figure 3-2. Hard Disc Board Cabling

HARD DISC P4 CONNECTOR - ANSI (not normally used)

Pin	Ground Pin	Signal Name
1	10	ground
		control bus:
2	10	Bit 0, Select/Attn. Drive 0
3	10	Bit 1, Select/Attn. Drive 0
4	10	Bit 2, Select/Attn. Drive 0
5	10	Bit 3, Select/Attn. Drive 0
6	10	Bit 4, Select/Attn. Drive 0
7	10	Bit 5, Select/Attn. Drive 0
8	10	Bit 6, Select/Attn. Drive 0
9	10	Bit 7, Select/Attn. Drive 0
11	12	Parity (optional)
13	14	Select out/Attn. In strobe
15	16	Command Request
17	18	Parameter Request
19	20	Bus Direction Out
21	22	Port Enable
23	24	(not used)
25	26	Read Gate
27	28	Write Gate
29	30	Bus Acknowledge
31	32	Index
33	34	Sector Mark
35	36	Attention
37	36	Busy
39	38	Read Data +
40	38	Read Data -
42	41	Read/Reference Clock +
43	41	Read/Reference Clock -
45	44	Write Clock +
46	44	Write Clock -
48	47	Write Data +
49	47	Write Data -
	50	ground

HARD DISC P3 CONNECTOR - PRIAM

Pin	Signal Name	Pin	Signal Name
1	ground	32	ground
2	+ DBUS 0	33	- INDEX
3	+ DBUS 1	34	ground
4	+ DBUS 2	35	- READY
5	+ DBUS 3	36	ground
6	+ DBUS 4	37	- SECTOR MARK
7	+ DBUS 5	38	ground
8	+ DBUS 6	39	+ WRITE DATA
9	+ DBUS 7	40	- WRITE DATA
10	ground	41	ground
11	- READ GATE	42	+ WRITE CLOCK
12	ground	43	+ WRITE CLOCK
13	- RESET	44	ground
14	ground	45	+ READ/REFERENCE CLOCK
15	- WRITE GATE	46	+ READ/REFERENCE CLOCK
16	ground	47	ground
17	- RD	48	+ READ DATA
18	- WR	49	- READ DATA
19	+ AD 1	50	ground
20	+ AD 0		
21	ground		
22	- DRIVE SELECT 1		
23	- DRIVE SELECT 2		
24	- DRIVE SELECT 3		
25	- DRIVE SELECT 4		
26	ground		
27	ground		
28	+5 VOLTS DC (terminator power)		
29	- HEAD SELECT 4		
30	- HEAD SELECT 2		
31	- HEAD SELECT 1		

HARD DISC P1 CONNECTOR

Pin	Signal Name
1	ground
2	Switch 3
3	ground
4	Switch 4
5	ground
6	Switch 5
7	ground
8	Switch 6
9	ground
10	Switch 7
16	ground
20	ground
26	Transmit Data
27	Read Data
28	Request to Send
29	Clear to Send
31	ground
32	Data Carrier Detect
38	Auxiliary Transmit Data
44	Data Terminal Ready
46	Ring
50	ground

Switch 3 through Switch 7 can be read by the on-board microprocessor. When Switch 3 is grounded, the board will use the P4 ANSI connector.

THEORY OF OPERATION

FUNCTIONAL DESCRIPTION

The Hard Disc board can control up to four disc units. The board has two 50-pin connectors, one matching the ANSI X3T9.3 interface, and one matching the Priam interface. Only the Priam interface is fully supported and documented here.

A 50-pin ribbon cable connects the controller's P3 connector to a Priam drive. If there is only one drive, that drive must have a cable terminator installed.

HD INTRODUCTION

The HD board can control from one to four hard discs with a Priam interface. The board can be configured to use the ANSI interface. It has an on-board Z80, which performs a number of functions:

1. It seeks when necessary, including those cases where the data transfers requested span cylinder boundaries.
2. It monitors the state of the circuitry that interfaces with the disc read and write data lines and changes the sector and head values as sectors are successfully transferred.
3. It performs error correction and corrects the data in Ridge memory.
4. It retries operations when they fail.

Much like the FDLP board, the HD board communicates with the Ridge CPU via I/O Writes that start operations and areas in Ridge Memory that contain more detailed information about the request that are called Device Control Blocks (DCBs). The communication between the Ridge CPU and the board is:

1. I/O Write instructions start an operation.
2. The Z80 usually requests further information about the request, which it obtains from a dedicated area in Ridge memory called a Device Control Block (DCB).
3. The operation is performed.
4. A Ridge interrupt is generated and the I/O Interrupt Read that the Ridge CPU will perform will return the board's device number, the unit on the board, and an indication of the success of the requested operation.

HD BOARD LAYOUT

The primary logical parts of the board are:

1. I/O Read, I/O Write, and I/O Interrupt logic
2. A DMA sequencer that generates Ridge memory requests and copies data from or to a pair of fifo chips. It also converts between byte and word formats as it moves data to or from the fifo chips. It has a word count register and an address counter, which has a two stage register on the page portion of the address so that multi-page memory operations can be performed more easily.
3. A serializer/deserializer (SERDES) which converts between the format that comes out of the fifos and the bit format that the disc interface uses. A second pair of fifo chips is also attached to the SERDES and connects to the Z80 (the other pair connects the SERDES to the DMA machine which talks to the memory controller).
4. A sync detector that can be enabled when looking for byte synchronization. It can also time out if sync is not found within 16 byte times of when it is expected. To determine whether the current sector is the one desired, a comparator compares the output of the Z80 fifo and the parallel byte accumulated by the SERDES. The comparator supplies a "no match" signal, which is examined while the sector header is passing through the SERDES, and which is latched elsewhere.
5. An ECC shift register to generate and check the Error Correction Code appended to the end of the header and data portions of every sector. It also can be used to perform the error correction calculation. A part of this circuitry detects when all 32 bits of the ECC register are zero; another part detects 21 zeros for the correction process.
6. A Z80 with an SIO for a Monitor program. Also included in this section are 8 K bytes of EPROM and 2 K bytes of RAM. There are the usual decoders for I/O addresses to access various registers on the board.
7. A "Bit Machine," which runs at the frequency of the disc clock and is responsible for executing very simple microinstructions that can freeze waiting for sector pulses or ECC correction finished or byte synchronization and then define the data formats of the sectors for reading, writing, formatting, etc. A related part this circuitry is a "Bit Machine" status register that, among other functions, latches error conditions such as data overrun, sync timeout, ECC error, compare error, etc.
8. A set of input and output registers for the parallel part of the disc interfaces and differential drivers and receivers for the bit clocks and data lines.

I/O READ, WRITE, AND INTERRUPT LOGIC

The Interrupt state machine is normally in an idle state. The Z80 sets a bit in a register it can write to; this advances the state. When the Z80 resets that bit, the state machine advances the state again and asserts IOIREQ1 or IOIREQ2. It also blocks ACKIOIout so that, when multiple devices request an interrupt at the same time (or before the first device is acknowledged), any lower priority (devices further down the daisy-chain) will not see the ACKIOI signal and will not gate their device numbers but will rather continue to assert IOIREQ1 or

IOIREQ2. When the board finally sees ACKIOIin, it advances the state once more and asserts its device number and the status register on the I/O bus. The state advances to the null state on the next clock unconditionally.

The I/O Read and Write state machine leaves its IONULL state when it sees an MCIOREQ signal from the memory controller and enters the I/O REST state. From this state, it will advance to the ACKMCIO state if the following conditions are true:

1. MCIOREQ must remain true.
2. The device number on the bus must match the device number switch on the board.
3. If the MCIOW bit is set (this is an I/O Write), then the CMDINT latch (command byte interrupt) must not be set. If MCIOREQ is removed, the next cycle will be the IONULL state; otherwise, the machine will remain in the IOREST state.

In the ACKMCIO state, the signal ACKMCIO will be returned to the memory controller. During this state, however, the MCIOREQ and MCIOWT signals remain asserted. If MCIOW is false, the next state will be IOGATE which drives the I/O bus with the board's device number, status register, and low order address bits; the next state from this one is the IONULL state. If MCIOW is true, the state moves from ACKMCIO to IOLOAD where it remains until an IODACK is issued from the memory controller. After the IODACK the next state is IONULL.

DMA LOGIC

The DMA sequencer starts in the MNULL state. The Z80 sets the address registers, word count, and DMA direction. The Z80 sets an MSTART bit, putting the DMA sequencer into Memory Handshake (MHS) state. The Z80 clears the MSTART bit, advancing the sequencer state to MREQ or MWAIT, depending on whether a Ridge-to-disc (memory read) disc-to-Ridge (memory write) operation has been specified.

If MREAD is true, the MHS state leads to the MREQ state. MREQ sends an IOMREQ1 or IOMREQ2 signal to the memory controller and blocks the ACKIOMout from the controller to prevent lower priority boards from detecting the ACKIOMin signal. The DMA state machine advances to MADDR when ACKIOMin is detected. MADDR gates the address register onto the bus. Since the MREAD condition is true, the MADDR state leads to the MLOAD state. MLOAD triggers a flip-flop that generates and extra signal (to meet timing requirements of Ridge I/O bus) which latches the contents of the data bus in four 74LS373s. To confirm that the requested data is on the bus, the memory controller sends IODACK to the DMA machine, moving its state from MLOAD to MWAIT. The machine moves from the MWAIT to PULSE state when the INPUT-READY signal (synchronized to Ridge clocks) from the Ridge pair of fifos becomes true. PULSE generates a SHIFT-IN pulse to the fifos. On the next cycle, the PULSE state changes to MWAIT, where the system waits for INPUT-READY again, if the BC=3 signal is false (the current byte is the last of

a Ridge word). PULSE leads to MNULL if the WC=0 signal is true. If WC=0 is false, PULSE leads to MREQ.

The PULSE state also generates a COUNT-ENABLE condition to the byte counter, which causes it to change on the next clock edge. During a DMA sequence with MREAD set, Ridge memory requests are made and the data is shifted one byte at a time into the fifos while checking for INPUT-READY. This continues until the Word Count register is zero and the last byte of the last word is shifted in to the fifo.

If MREAD is false, the MHS state leads to the MWAIT state. When the synchronized OUTPUT-READY signal from the fifos goes true, the PULSE state is entered, which generates a SHIFT-OUT pulse to the fifo while clocking one byte of the output word register, and which generates the previously described COUNT-ENABLE signals. The state after PULSE depends on the BC=3 and WC=0 signals. If MREAD is false, MREQ leads to MDATA, which which gates the Ridge Data Out register onto the I/O bus. If the WC=0 signal is true, MREQ leads to MNULL. If WC=0 is false, MREQ leads to MWAIT. If MREAD is false during a DMA sequence, the Data Out register is loaded one byte at a time from the output of the fifos while waiting for OUTPUT-READY to indicate that data is ready. After four bytes are assembled, a MCIOREQ1/2 is generated and the data is written to Ridge memory. If WC=0 is false, the process repeats.

SERDES / SYNC DETECTOR / COMPARATOR

The serializer/deserializer (SERDES) converts between serial bit and parallel byte data.

When reading from the disc, the bit stream is sent through a shift register. After every 8 bits enter, the contents are captured in a holding register, the contents of which are available to the Ridge fifo or Z80 fifo. A comparator compares the outputs of the shift register and Z80 fifo and generates a NO-MATCH signal for determining whether the current sector is the desired one.

The bit stream byte boundary is determined by the sync detector, which is triggered when the first one bit is seen or when the comparator signals a match. A wait state is entered while looking for sync. If 128 bits pass in the wait state, the wait state is released but the current sector search will fail. When writing to the disc, the SERDES is given an order to parallel load the contents of either the Ridge fifo or the Z80 fifo once every 8 bit times. During the other 7 bit clocks, the SERDES shifts left. The data out of the SERDES is taken as the most significant bit of the shift register.

ECC CIRCUITS

The ECC register is made up of eight 74LS194As. All four register control modes are used:

- "hold" retains the data after after checking the read data bit stream.
- "shift left" clears the register in four bit times.
- "shift right" unloads and merges the calculated ECC bytes with the data stream when writing data on the disc.
- "parallel load" makes the calculation to generate ECC bytes.

The parallel load connections implement the polynomial division used in the error-correcting code generation and the correction process. The polynomial used is (X**21 + 1)(X**11 + x**2 + 1).

Before beginning the calculation, the ECC register is cleared by setting SHIFT-LEFT to the 74LS194As. In the generation process, the most significant bit of the SERDES is the input the ECC register. During the calculation, the PARALLEL-LOAD mode is set. After the last bit of the data stream has been shifted in, the ECC register contains the desired ECC bytes. At that time, the shift mode of the 74LS194As is set to SHIFT-RIGHT and the mux, which has been supplying the Write Data bit to the disc is switched from the msb of the SERDES to the last bit of the ECC register. After 32 bit times, the ECC bytes have been appended and the sector format programs fill the data stream with a few bytes of zeros.

In the disc read process, the ECC register is cleared by SHIFT-LEFT. The data input in this case is the lsb of the SERDES so that the calculated result will be available 7 bit times earlier. After all the data has been shifted through, the ECC register should be zero if there are no errors. If it isn't, the sector format programs abort and the polynomial remainder is saved in the ECC register by setting the HOLD mode on the shift register controls.

In the error correction process, the ECC register (with the non-zero contents) is shifted with a zero as input until the first 21 bit positions are zero or the shift count exceeds the natural period of the polynomial. If the 21-zero condition is reached, the remaining 11 bits may be XOR'ed with the data in Ridge memory to correct the burst error. The position of the 11 bits in the bit stream is determined by the number of shifts it took to reach the 21-zero condition.

Z80 SECTION

The Z80 section is conventional with two exceptions:

- a pair of flip-flops that generate 250 ns and 125 ns pulses are used to "shape" the decoded register write pulses from the Z80.
- a four-state sequencer that generates shift-in and shift-out pulses to the fifos from Z80 IN's and OUT's, which causes a Z80 WAIT state until the fifos are ready.

A channel of the CTC generates a timeout that unfreezes the Z80 if the fifo is empty or full for too long a time. A side effect of this timeout is an interrupt which will set an error flag.

BIT MACHINE

The bit machine runs at the bit rate of the disc. It is a very simple microprogrammed machine that whose instruction counter can:

- be reset to zero where an IDLE instruction is stored.
- be set (jump) to the contents of the IREG register which can be loaded by the Z80.
- increment its address.
- remain at one address for 256 byte times.

The instruction address never changes faster than once every byte time (8 bit times) so its microcode proms can be relatively slow EPROMS. This section of circuitry also contains a microcode register, which is the latched contents of the proms, a 16 bit counter, which usually counts bytes but can count at the bit rate in the error correction process, a 8 bit PHASE register, which contains a circulating 1, which marks the bit position within the byte, a Bit Machine Status register and various decoders of the microcode register. There are several microcode orders such as "sync search" and "correct error" and "wait for sector pulse" that freeze the PHASE register and release it when a condition becomes true. There are also codes in the microcode register to control the ECC register, Read Gate, Write Gate, generate shift in and shift out pulses to the fifos, etc.

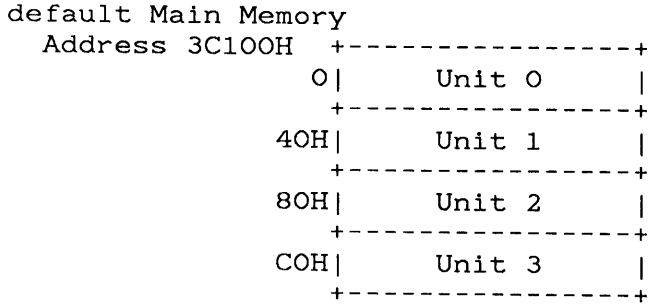
PARALLEL INTERFACE

The parallel interface section is logically quite simple; it consists mainly of buffers and receivers. There is also a register clocked by the bit clock and controlled by the Z80 to shut down or start up the PHASE register.

HARD DISC PROGRAMMING

GENERAL

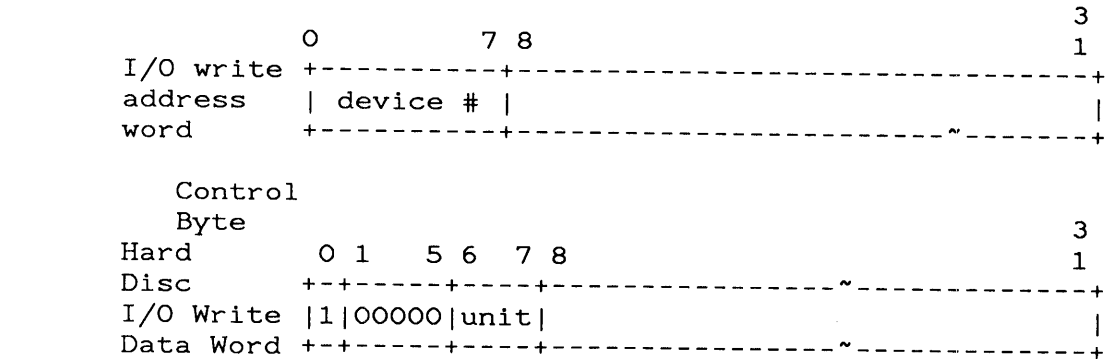
The hard disc controller can control from one to four hard disc drives. Like the FDLP board, it communicates with the Ridge CPU by DCBs. Each DCB is 64 bytes long and the four control blocks for the hard disc controller are laid out as follows:

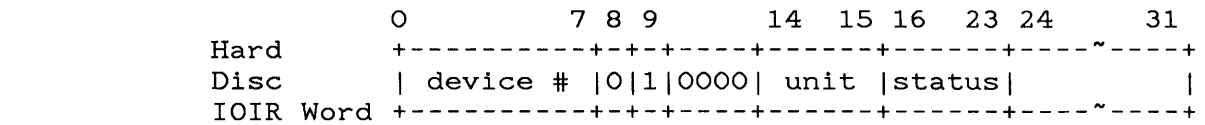


The flow of control is:

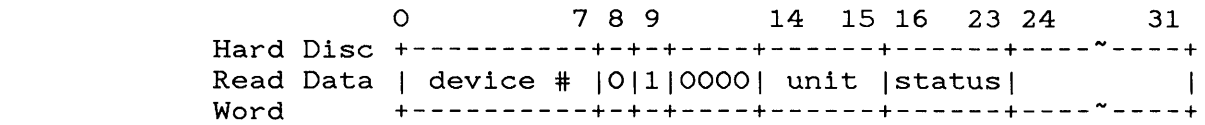
1. The Ridge Operating System builds the DCB request in main Ridge memory at the locations appropriate for the disc unit.
2. A command to start the operation is sent to the hard disc controller using the WRITE instruction.
3. The hard disc controller recognizes its device number from the I/O write address word and accepts the I/O write data word. The board uses the unit number from the I/O write data word to index into the proper DCB.
4. The function is performed.
5. The controller modifies the DCB and interrupts.

The format of the I/O write data word and IOIR word follows:





A read command (READ instruction) may be sent to the hard disc controller at any time. The hard disc controller returns its device number and last interrupting unit in the following format:



The default DCB locations for the Hard Disc board are 3C100H through 3C1FFH, but these may be changed by issuing an I/O write with the control byte in the I/O Write Data Word equal to C0H, after first setting locations 3C1FD through 3C1FF with the new 24-bit Ridge memory address base.

The HD board recognizes various values in the high-order byte of the I/O Write Data Word:

C0H	it internally builds a read request for a 4K byte read from the first page of the drive (head 0, cylinder 0, sector 4 to 7) into Ridge memory address 3E000H, executes it, and interrupts. This command is used to boot the system.
C1H	sets the DCB base to (the value at 3C13E-3C13F) * 256.
C4H	returns the present DCB base value to 3C13E-3C13F.
C2	sets a flag, which is checked as each sector is written, that inhibits writing to the disc.
C3H	turns off the write-inhibitor flag. The pair C2, C3 are intended to be used in a power fail warning trap.

For example, if the board device number were 2, and the desired unit were 0, one would fill the following bytes of Ridge memory to request a read of 4K bytes from cylinder 123H, head 4, sector 5 into Ridge memory 3F000H:

Ridge Memory Locations	**Value**	**Meaning**
3C100	0	read
3C105	3	Ridge memory address
3C106	F0	Ridge memory address
3C107	00	Ridge memory address
3C108	10	4K bytes
3C10D	41	head 4, cylinder 123H
3C10E	23	cylinder 123H
3C10	5	sector 5

Then an I/O Write with address word of 02000000H, and a data word 80000000H should be issued. When the operation is complete, an IOIR word of 024000xxH should be returned, indicating board device number 2, disc unit 0, and GSTAT=OK. If further information about the transfer is desired (especially if STATUS is not OK), the area 3C100 through 3C11F can be examined.

HARD DISC CONTROLLER DEVICE CONTROL BLOCK (DCB)

Hex Addr	Name	Function
0	GORDER	0 Read 1 Write 2 Verify. Reads, but no data transferred to Ridge memory 3 Format a track 4 Seek 5 Return Highest Sector Address. This is the physical address of the last addressable sector in the HDCYL, CYL, and SECTOR fields. The value returned in the Byte Count Transferred is the actual number of bytes between sector marks, which is needed for interpretation of data from the Read Header order. 6 Read Full Sector. Data transfer is always 1040 bytes the transfer count is ignored. The data read is a 12-byte data label, followed by 1024 data bytes, plus a 4-byte checksum. 7 Write Full Sector. The transfer count is ignored, 1040 bytes are transferred in the same format as Read Full Sector. E - Read Header. Priam defect log is transferred into first 9 bytes of DATA LABELS.
1	SORDER	not used.
2	GSTAT	0 OK 1 Not ready 2 Timeout 3 Equipment fault 4 Write protected 5 Ridge double bit error in data or DCB transfer 6 Data overrun 7 Missing address mark (Can't find sync byte) 8 Can't find header that matches 9 CRC error in header A Uncorrectable error in data B Seek error FF Illegal parameter in DCB order
3	SSTAT	Reserved

Hex Addr	Name	Function
4	RETRIES	The number of retries attempted on this request
5-7	RIDGE ADDRESS	This address must be on a word boundary. The Ridge address plus the request byte count must not cross a 64KB boundary.
8-9	REQUEST BYTE COUNT	
A-B	BYTE COUNT TRANSFERRED	
C	Not used.	
D	HDCYL	Four bits of head number, followed by the four most significant bits of the cylinder number.
E	CYL	The least significant eight bits of the cylinder number.
F	SECTOR	Number from 0-17 on 14-inch Priam.
10-1B	DATA LABELS	These twelve bytes are read/written into each sector on a data transfer. This area is also modified by the Read Header GORDER.

MONITOR PROGRAM

The hard disc controller uses a Z80 microprocessor and a monitor program, similar to the floppy disc controller. The hard disc controller monitor is always active, using an RS-232 port located on the front edge connector P1. The monitor prints a banner and a prompt, > when the system is reset. The RS-232 pins are connected as follows:

P1-26	Transmit Data
P1-27	Receive Data
P1-31	Signal ground

The hard disc controller monitor commands are listed below. Commands that are identical to the FDLP monitor are indicated, and their description is found in the section on the FDLP monitor. The hard disc controller varies from the FDLP in that its read write memory is in two discontinuous pieces, from 2000H to 23FFH, and from 3000H to 33FFH, while the Z80 memory on the FDLP is from C000H to FFFFH.

Breakpoint Command

BR *codeloc*

See FDLP monitor description.

Display Z80 Memory Command

D *address length*

See FDLP monitor description.

Display Z80 Registers Command

DR

See FDLP monitor description.

Display FIFO Command

DF *address*

Dumps the contents of the Z-FIFO at the address specified.

Load FIFO Command

FF *address count*

Loads the Z-FIFO with the values from memory.

Format Command

FORMAT

Formats unit 0.

Read a Port Command

I *port-number*

See FDLP monitor description.

Read a Port, Repeatedly, Command

IR *port-number*

See FDLP monitor description.

Jump to Address Command

J *address*

See FDLP monitor description.

Download Z80 Memory Command

L*aaaaccccddd...ddd*

See FDLP monitor description.

Modify Z80 Memory Command

M *address*

See FDLP monitor description.

Write to a Port Command

O *port-number value*

Writes "value" to the port specified.

Write to a Port, Repeatedly, Command

OR *port-number value*

See FDLP monitor description.

Display Ridge Memory (Peek) Command

P *ms-Ridge-address ls-Ridge-address length*

See FDLP monitor description.

Modify Ridge Memory (Poke) Command

PO *ms-Ridge-address ls-Ridge-address length*

See FDLP monitor description.

Read from Hard Disc Command

R *unit ms-Ridge-addr ls-Ridge-addr length head cylinder*

Reads "length" bytes into Ridge memory from the disc track at the specified head and cylinder.

Seek Command

SK *cylinder*

Causes the disc to seek to the specified cylinder.

Transfer Data Between Ridge Memory and Z80 Memory Command

T *ms-Ridge-addr ls-Ridge-addr Z80-addr length direction*

See FDLP monitor description.

Transfer Data Between Ridge and Z80 Memory, Forever, Command

U *ms-Ridge-addr ls-Ridge-addr Z80-addr length direction*

This command is the same as the above command, except that the transfer is repeated forever.

Verify Hard Disc Command

V

(needs two carriage returns)
This command reads the entire disc, displaying all retries and uncorrectable errors.

Write to Hard Disc Command

W *unit ms-Ridge-addr ls-Ridge-addr length head cylinder*

Writes "length" bytes from Ridge memory onto the disc track at the specified head and cylinder.

Execute Test Programs Command

X *program-number*

Executes one of the following test programs according to the "program-number" below:

0	Display device attributes for ANSI drive.
1	Perform maximum length seek.
2	Test parallel data path to ANSI drive.
3	Seek forever.
4	Verify forever.
5	Read forever.
6	Write forever.
7	Execute a DCB forever.
8	Partition track.
9	Spin down a Priam drive.
A	Seek between two cylinders.
B	Perform all possible seeks.
C	Perform all possible seeks, printing as each is performed.

Execute Z80 LDIR Command

Y *target-address source-address byte-count*

Performs a Z80 LDIR instruction.

Chapter 4
SMD CONTROLLER BOARD

SMD CONTROL SWITCHES, INDICATOR LIGHTS, AND PIN ASSIGNMENTS

SMD BOARD LAYOUT

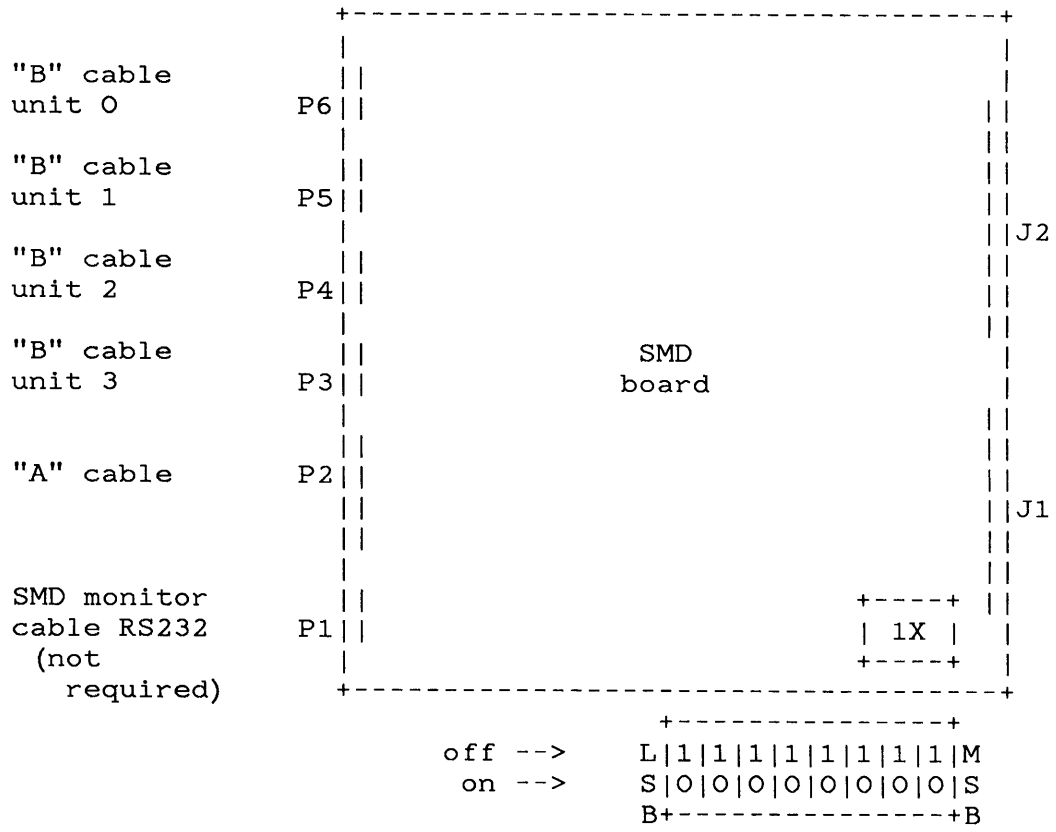


Figure 4-1. Layout of the SMD Board

SMD BOARD CABLING

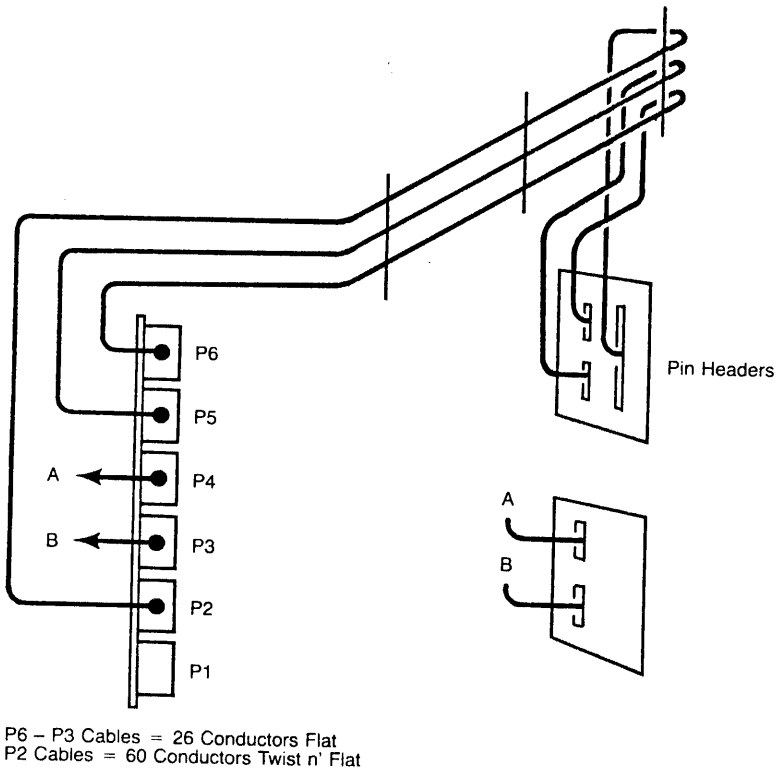


Figure 4-2. SMD Board Cabling

Cables from the lowest "P#" are connected to the left-most I/O connector (when viewed from the outside).

Flat ribbon cables typically have a blue stripe on the wire used for pin 1. On the "A" cable, which consists of 30 twisted pairs, the tan/brown pair is used for pin 1.

SMD BOARD SWITCHES

The only switch on the SMD board is the device number switch, located at position 1X (bottom row, last column). It may be set to any number in the range 0H to 1FH.

SMD "A" CABLE (TAG BUS I/O INTERFACE)

Function	Actual Pin Assignments	Standard SMD Pin Numbering
Unit Select Tag	43 44	22 52
Unit Select 2^0	45 46	23 52
Unit Select 2^1	47 48	24 54
Unit Select 2^2	51 52	26 56
Unit Select 2^3 (always 0)	53 54	27 57
Tag 1	1 2	1 31
Tag 2	3 4	2 32
Tag 3	5 6	3 33
Bit 0	7 8	4 34
Bit 1	9 10	5 35
Bit 2	11 12	6 36
Bit 3	13 14	7 37
Bit 4	15 16	8 38
Bit 5	17 18	9 39
Bit 6	19 20	10 40
Bit 7	21 22	11 41
Bit 8	23 24	12 42
Bit 9	25 26	13 43
Open Cable Detector	27 28	14 44
Index (status 6)	35 36	18 48
Sector (status 7)	49 50	25 55
Fault (status 3)	29 30	15 45
Seek Error (status 2)	31 32	16 46
On Cylinder (status 1)	33 34	17 47
Unit Ready (status 0)	37 38	19 49
Addr. Mark (stat 5) unused	39 40	20 50
Write Protected (status 4)	55 56	28 58
Power Sequence Pick	57	29
Power Sequence Hold	58	59
Busy	41 42	21 51
Spare (optional bus bit 10)	59 60	30 60

SMD "B" CABLE (CABLE INTERFACE)

Function	Actual Pin Numbers	Standard SMD Pin Numbering
Write Data	15 14	8 20
Ground	13	7
Write Clock	11 12	6 19
Ground	10	18
Servo Clock	3 2	2 14
Ground	1	1
Read Data	5 6	3 16
Ground	4	15
Read Clock	9 8	5 17
Ground	7	4
Seek End	19 20	10 23
Unit Selected	18 17	22 9
Ground	16	21
Index	23 22	12 24
Ground	21	11
Index	25 26	13 26
Ground	24	25

SMD MONITOR CABLE

An RS-232 terminal can be connected to the P1 connector to monitor the on-board processor. The pin assignments for P1 are:

Board Pin No.	Signal Name	(25-pin Sub-D connector on 26-pin flat cable) Connector No.
1		1
2		14
3	transmitted data	2
4		15
5	received data	3
6		16
7	request to send	4
8		17
9	clear to send	5
10		18
11		6
12		19
13	ground	7
14	data terminal ready	20
15	data carrier detect	8
16	ground	21
17		9
18	ring	22
19		10
20	ground	23
21	switch 2	11
22	ground	24
23	switch 1	12
24	ground	25
25	switch 0	13
26	ground	--

Switch 0, 1, and 2 can be used as configuration straps to the on-board microprocessor.

THEORY OF OPERATION

FUNCTIONAL DESCRIPTION

Up to four Storage Module Drive (SMD) disc devices can be controlled by the SMD controller. The SMD interface is commonly used for high-performance disc drives.

The "A" Cable

The SMD board uses a 60-conductor daisy-chained cable (the "A" cable). The "A" cable is used to select drives one at a time, to pass commands from controller to drive, and to return drive status to the controller.

The daisy-chain runs from the board to the first drive device, then to the next, up to 4 drives total. If fewer than four drives are connected, the cable must have a "terminator" attached to the end. The terminator is a collection of resistors which match the impedance of the cable to minimize the time for data to settle on the cable.

The "B" Cable

The SMD board also uses "B" cables to connect each drive to the controller (a "star" connection). The "B" cable is for the serial data and clock lines, which are used for reading and writing data on the disc. The "B" cable also contains two status bits from the drive: "seek end" and "unit selected."

SMD INTRODUCTION

The SMD board can control from one to four hard discs with the Storage Module Drive interface. It has an on board Z80 which performs a number of functions:

1. It seeks when necessary, including those cases where the data transfers requested span cylinder boundaries.
2. It closely monitors the state of the circuitry that interfaces with the disc read and write data lines and changes the sector and head values as sectors are successfully transferred.
3. It performs error correction and corrects the data in Ridge memory.
4. It retries operations when they fail.

Much like the FDLP and HD boards, the SMD board communicates with the Ridge CPU via I/O Writes that start operations and areas that contain more detailed information about the request that are called Device Control Blocks (DCBs). Unlike the HD and FDLP boards, however, the DCB's are located in a section of on-board RAM accessible to both the Ridge CPU and the Z80 rather

than in Ridge memory. The communication between the Ridge CPU and the board is :

1. The Ridge CPU fills in all the information about the request it is about to start except the first byte of that request (which contains the basic order).
2. The Ridge CPU fills in the basic order (called the GORDER in the programming section). The Z80, when available to start another operation, is scanning the four GORDER locations corresponding to the four units for more work to do. When it sees one of those locations change from a value of 0FFH (which the Z80 sets when it is finished with an order) to some other value, it knows that the Ridge CPU has built a new request.
3. The operation is performed
4. A Ridge interrupt is generated and the I/O Interrupt Read that the Ridge CPU will perform will return the board's device number, the unit on the board, and an indication of the success of the requested operation. If necessary, the Ridge CPU can obtain more detailed information from the DCB for the unit in the Parm RAM.

SMD BOARD LAYOUT

The primary logical parts of the board are:

1. I/O Read, I/O Write, and I/O Interrupt logic.
2. A DMA sequencer that generates Ridge memory requests and copies data from or to a set of fifo chips. It also converts between double byte and word formats as it moves data to or from the fifo chips. It has a word count register and an address counter which has a two sets of registers on the page portion of the address so that multi-page memory operations can be performed more easily.
3. A serializer/deserializer (SERDES), which converts between the format that comes out of the fifos and the bit format that the disc interface uses. A closely related part of the circuitry is a small RAM (the Header RAM) which can be accessed both by the Z80 and the "Bit Machine" described later. This used to transfer the sync patterns, head, sector, cylinder, and Ridge Data Labels (File IDs) between the two sections.
4. A sync detector which can be enabled when looking for byte synchronization. It can also time out if sync is not found within 16 byte times of when it is expected. A closely related part of the logic is a comparator which looks at the output of the Header Ram and the parallel double byte accumulated by the SERDES; it supplies a "no match" signal which is examined at certain times while the sector header is passing through the SERDES and which is latched elsewhere. This determines whether the current sector is the one desired.
5. An ECC shift register to generate and check the Error Correction Code appended to the end of the header and data portions of every sector. It also can be used to perform the error correction calculation.

A part of this circuitry detects when all 32 bits of the ECC register are zero; another part detects 21 zeros for the correction process.

6. A Z80 with an SIO for a Monitor program. Also included in this section are 8 K bytes of EPROM and 2 K bytes of RAM. There are the usual decoders for I/O addresses to access various registers on the board.
7. A "Bit Machine," which runs at the frequency of the disc clock and is responsible for executing very simple microinstructions that can freeze waiting for sector pulses or ECC correction finished or byte synchronization and then define the data formats of the sectors for reading, writing, formatting, etc. A related part this circuitry is a "Bit Machine" status register which, among other functions, latches error conditions such as data overrun, sync timeout, ECC error, compare error, etc.
8. A set of input and output registers for the parallel part of the disc interfaces and differential drivers and receivers for the interface lines.

I/O READ, WRITE, AND INTERRUPT LOGIC

The interrupt flip-flop is set by a Z80 write to its location. When set, it asserts IOIREQ1 or IOIREQ2. It also blocks ACKIOIout so that when multiple devices request an interrupt at the same time (or before the first device is acknowledged), any lower priority devices (further down the daisy-chain) will not see the ACKIOI signal and will not gate their device numbers but will rather continue to assert IOIREQ1 or IOIREQ2. When the board finally sees ACKIOIin, it clears the interrupt flip-flop and sets the IOR flip-flop which gates the board's device number and the status registers on the I/O bus. The IOR flip-flop is reset on the next clock unconditionally.

The I/O Read and Write operations to the SMD board are actually reads and writes of the Parm RAM, which can also be accessed by the Z80.

Two flip-flops help resolve simultaneous requests by the two ports:

- The MCIOW flip-flop is a copy of the backplane signal MCIOW MCIOW clocked at the major 125 ns clock edge.

- The IORDWT flip-flop, which enables the device number comparator when set. Its input is the signal (MCIOREQ & not ACK flip-flop).

The IT'S_BUSY signal is asserted if the memory controller has this board's device number on the I/O Data lines. The BUSY signal, clocked by the main 250 ns Z80 clock (PHI), indicates whether the Parm RAM is being accessed by the Z80.

The positive edge of the PHI clock is nearly coincident with that of the main Ridge clock on the board, but the PHI clock period is twice as long. On the SMD board, as on all Ridge boards, the three Ridge clock phases are CLK_A, CLK_B, and CLK_C. Most flip-flops on the SMD board are clocked when CLK_C falls; that is negated CLK_C triggers the positive-edge-triggered flip-flops. One exception is the SWITCH flip-flop, which is triggered by the rising

edge of CLK_B and samples the term (not BUSY & IT'S_US). If both signals are true, SWITCH is set; this effectively gives control of the Parm RAM to the Ridge port for the rest of the current main clock, the entire next main clock period, and the first third of the cycle after that.

If the Z80 were to attempt to access the Parm RAM on the next two main clock edges, its circuitry would generate a WAIT state to "stretch" its memory access until the Ridge I/O circuitry was finished. If the Z80 is currently accessing the Parm RAM, the BUSY flip-flop is set and the SWITCH is not. Interference from the Z80 can delay Parm RAM access for up to four main Ridge clocks longer than the minimum, which is still much less than the 16 clock timeout in the memory controller of I/O Reads and Writes.

If SWITCH is set at the rising edge of CLK_B, the ACK flip-flop is set on the next main Ridge clock edge. This generates the ACKMCIO signal to the memory controller. The memory controller will drop MCIOREQ, MCIOW, and the I/O Address Word in the middle of the next main clock cycle. However, the MCIOW flip-flop will preserve the direction of the transfer for one more clock cycle and an eight bit register will hold the lower 8 bits of the I/O Address Word which, in the case of an I/O Write, is the data.

The SWITCH flip-flop has selected a 2:1 mux to gate the I/O Address Word bits 13-23 onto the Parm RAM's address lines and has asserted chip select. With an I/O Read, the Parm Ram is being accessed during this time and the data is available to be clocked at the end of the ACKMCIO cycle; the next cycle sets the IOR flip-flop which gates the device number, the two status registers, and the clocked Parm Ram data onto the Ridge I/O bus. With an I/O Write, the WPULSE flip-flop generates a WT strobe to the Parm RAM a few nanoseconds before ACK. In the case of the write, the data is held for an extra clock phase (until SWITCH drops) and the write pulse drops before the address lines can change.

DMA LOGIC

Memory request signals are controlled by flip-flops and are initially low. The Z80 loads the Ridge memory address and word count registers, then pulses STARTREAD of STARTWRITE. STARTREAD sets ACTIVE and MREQ. ACTIVE remains set until the memory transfer is complete. MREQ generates the MCIOREQ1 or MCIOREQ2 signal to the memory controller and blocks the ACKIOMout signal from reaching lower priority boards (those further from the memory controller on the daisy-chain). Until ACKIOMin is detected, all states remain the same. On the next major clock edge, ADR is asserted, which gates the Ridge Address Register onto the I/O bus. The MCIOREQ1/2 signal will be suppressed during this cycle although MREQ remains set.

On the next major clock edge, the MREQ is reset and MLOAD and L1 are set. The MLOAD flip-flop has two functions:

- While set, it enables of copy of CLK_C called MLATCH to open the 74LS373 latches to capture the contents of the I/O data bus.

- It looks for and would latch the MDNVIO signal in the Double Bit Error flip-flop.

The L1 signal enables the output of the most significant 16 bits of the four 74LS373s onto the fifo input lines.

MLOAD drops on the next cycle after IODACK reaches the SMD board, stopping the generation of MLATC signals and preserving the contents of the bus in the 74LS373's.

The fifos' INPUT-READY signals are combined and synchronized to the Ridge clocks to create a signal called IR. If IR is true when IODACK comes true, the data will be shifted into the fifos in the middle of the next cycle as a side effect of setting the SI flip-flop. If IR is not true at that time, the overall memory state is just the ACTIVE/L1; when IR does come true, the next state is ACTIVE/L1/SI, which generates the shift-in pulse to the fifos.

The ACTIVE/L1/SI state leads to ACTIVE/L2, which gates the lower half of the captured Ridge memory data onto the fifo input lines. The overall memory state remains ACTIVE/L2 until IR comes true; then SI is also set for a main Ridge clock period and another fifo shift-in pulse is generated. During the ACTIVE/L2/SI cycle, the MCIOREQ1/2 signal is generated if the word counter has not asserted the DONE signal. If a lower priority board has an MCIOREQ1/2 outstanding on a previous cycle and the memory controller is now asserting ACKIOMin, the next state will be MREQ/ADR/ACTIVE; otherwise, the next state will be MREQ/ACTIVE.

When reading from Ridge memory, an MCIOREQ1/2 signal is generated which blocks ACKIOMout. When ACKIOMin becomes true, the next state will include ADR/MCIOREQ, which will gate the Ridge address onto the bus. On the next cycle, the four 74LS373s of the Data-In Register are enabled during the last third of every cycle until IODACK is received. The L2 state selects the most significant half of the Ridge word onto the fifos' input lines. If the fifo is asserting INPUT-READY, the SI flip-flop generates a shift-in. On the next cycle, L2 will be set, enabling the least significant half of the Ridge memory data onto the fifos' input lines. Again, when Input-Ready is detected, a shift-in pulse will be generated and the whole cycle repeats until the word-count expires.

When the transfer is from the fifos to Ridge memory, the initial state after the STARTWT pulse is ACTIVE/R1. This state is left when the synchronized Output Ready (OR) goes true; the next state is then ACTIVE/G1 which uses the G1 flip-flop to clock the most significant two 74LS374s of the Output Data Register to capture the output of the fifos and which also generates a shift-out pulse to the fifos. The next state is ACTIVE/R2. When OR goes true, ACTIVE/R2 leads to the G2 state, which clocks the least significant two 74LS374s with the output of the fifos and generates a shift-out pulse. During G2, the board asserts MCIOREQ1/2 and the state advances to ACTIVE/MREQ or ACTIVE/MREQ/ADR depending on whether another lower priority board has previously requested a memory operation and the memory controller is asserting a ACKIOMin for it. The ACTIVE/MREQ state leads to the ACTIVE/MREQ/ADR state on the clock edge after ACKIOMin becomes true. ACTIVE/MREQ/ADR leads to ACTIVE/DATA, which gates the 74LS374s onto the Ridge bus. If the word counter is asserting DONE, the next state has all the

memory request flip-flops reset. If not DONE and OR is true, ACTIVE/DATA leads to ACTIVE/G1; if not DONE and not OR, ACTIVE/DATA leads to R1.

When writing to Ridge memory, the fifo is tested for Output Ready and the two halves of the output data register are loaded one after the other and shift-out pulses are generated to remove the data from the fifo. When both are loaded, a MCIOREQ1/2 signal is generated and the data is written to Ridge memory. If the word count has not expired, the process continues, paced by the Output Ready signal.

SERDES / SYNC DETECTOR / COMPARATOR

The serialiser-deserializer (SERDES) converts between serial bit and parallel byte data.

When reading from the disc, the bit stream is sent through a shift register. After every 16 bits entered, the contents are captured into a holding register, where it is available for shifting into the fifos or for storing into the Header RAM, which can also be accessed by the Z80. A comparator compares the output of the Header RAM and the shift register, and generates a "no-match" signal for determining whether the current sector is the desired one. The byte field boundary in the bit stream is determined by the sync detector. It signals that sync has been achieved when the comparator signals a match. A wait state is entered when looking for sync and if 128 bits have passed while in this state and sync is still not detected, a counter will unfreeze the wait state. However, this condition is latched and the current sector search will fail. When writing to the disc, the SERDES is given an order to parallel load the contents of either the fifos or the Header RAM or zeros once every 16 bit times. During the other 15 bit clocks the SERDES shifts left. The data out of the SERDES is taken as the most significant bit of the shift register.

ECC CIRCUITS

The ECC register is made up of eight 74LS194As. All four register control modes are used:

- "hold" retains the data after after checking the read data bit stream.
- "shift left" clears the register in four bit times.
- "shift right" unloads and merges the calculated ECC bytes with the data stream when writing data on the disc.
- "parallel load" makes the calculation to generate ECC bytes.

The parallel load connections implement the polynomial division used in the error-correcting code generation and the correction process. The polynomial used is (X**21 + 1)(X**11 + x**2 + 1).

Before beginning the calculation, the ECC register is cleared by setting SHIFT-LEFT to the 74LS194As. In the generation process, the most significant bit of the SERDES is the input the ECC Register. During the calculation, the PARALLEL-LOAD mode is set. After the last bit of the data stream has been

shifted in, the ECC register contains the desired ECC bytes. At that time, the shift mode of the 74LS194As is set to SHIFT-RIGHT and the mux which has been supplying the Write Data bit to the disc is switched from the msb of the SERDES to the last bit of the ECC Register. After 32 bit times, the ECC bytes have been appended and the sector format programs fill the data stream with a few bytes of zeros.

In the disc read process, the ECC register is cleared by SHIFT-LEFT. The data input in this case is the lsb of the SERDES so that the calculated result will be available 7 bit times earlier. After all the data has been shifted through, the ECC register should be zero if there are no errors. If it isn't, the sector format programs abort and the polynomial remainder is saved in the ECC register by setting the HOLD mode on the shift register controls.

In the error correction process, the ECC register (with the non-zero contents) is shifted with a zero as input until the first 21 bit positions are zero or the shift count exceeds the natural period of the polynomial. If the 21-zero condition is reached, the remaining 11 bits may be XOR'ed with the data in Ridge memory to correct the burst error. The position of the 11 bits in the bit stream is determined by the number of shifts it took to reach the 21-zero condition.

Z80 SECTION

The Z80 section is mostly very conventional with one exception. The Header RAM's control, data, and address lines are mux'ed between the Z80's lines and the Bit Machine's depending on the Bit Machine's signal READY. The RAM appears to be 8 bits wide to the Z80 and 16 bits wide to the Bit Machine.

BIT MACHINE

The bit machine runs at the bit rate of the disc. It is a very simple microprogrammed machine whose instruction counter can:

- be reset to zero where an IDLE instruction is stored,
- be set (jump) to the contents of the IREG register which can be loaded by the Z80,
- increment its address,
- remain at one address for 256 byte times.

The instruction address never changes faster than once every double byte time (16 bit times) so its microcode proms can be relatively slow EPROMS. This section of circuitry also contains a microcode register, which is the latched contents of the proms, a 16 bit counter which usually counts bytes but can count at the bit rate in the error correction process, a 16 bit PIIASE register which contains a circulating 1 which marks the bit position within the byte, a Bit Machine Status register and various decoders of the microcode register. There are several microcode orders such as "sync search" and "correct error" and "wait for sector pulse" that freeze the PHASE register and release it when a condition

becomes true. There are also codes in the microcode register to control the ECC register, Read Gate, Write Gate, generate shift-in and shift-out pulses to the fifos, etc..

PARALLEL INTERFACE

The parallel interface section is logically quite simple. It consists mainly of differential drivers and receivers with a few registers. It also has a number of flip-flops that combine the servo clock and the read clock from the drive and, as the transition is made from one to the other, "stretches" the composite clock so that there are no glitches in it.

SMD BOARD PROGRAMMING

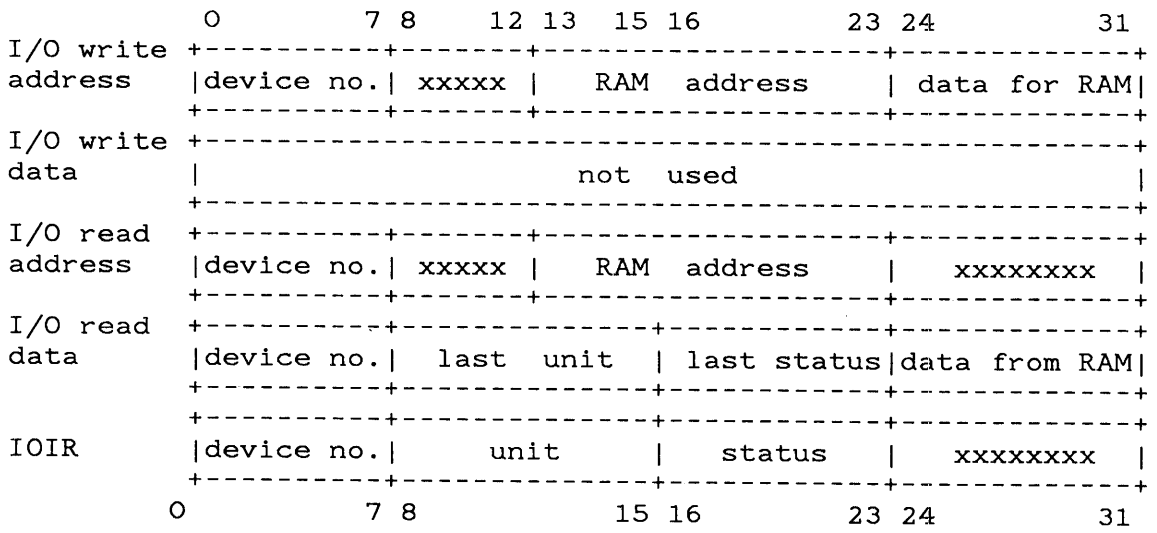
GENERAL

The SMD board has one "A" cable connector and four "B" connectors. It communicates with the Ridge 32 CPU by device control blocks (DCBs). Unlike the FDLP and Hard Disc boards, however, the SMD DCBs are not in Ridge 32 main memory. SMD DCBs are in an area of RAM on the SMD board and can be accessed by the board's local processor and by the Ridge 32 CPU by READ and WRITE instructions. The first 256 bytes of this RAM are divided into four parts; 64 bytes for each of the four units that can be controlled:

Hex Addr		Decimal Addr
	+--------+	
0 - 3F	| unit 0 |	0 - 63
	+--------+	
40 - 7F	| unit 1 |	64 - 127
	+--------+	
80 - BF	| unit 2 |	128 - 181
	+--------+	
C0 - FF	| unit 3 |	182 - 255
	+--------+	

The SMD controller is given Ridge I/O requests in a manner similar to the Floppy Disc/Line Printer (FDLP) and Hard Disc (HD) boards. First, the required parameters (except GORDER) are supplied by the Ridge I/O driver procedure; in this case with WRITEs rather than memory writes. The operation is started; in this case with a WRITE of the GORDER parameter at offset byte 0 of the DCB for the unit (locations 0, 40H, 80H, C0H for units 0, 1, 2, 3). The SMD local processor is scanning those four locations for an order to begin a new request. When it has completed the request, it will set its IOIR status register to the unit that has finished, set the GORDER location to 0FFH, set return status locations on the request and cause an interrupt. The Ridge CPU may now use READ instructions to get return status information related the the request.

The formats of the SMD I/O words are:



For example, it the board device number were 2, and the disc unit number were 3, the following I/O Writes would start a disc Read operation of 4K bytes to Ridge memory address 3F000H, starting at cylinder 123, head 4, sector 5 of the disc:

I/O Write (address part only)	**Function**
0200CF05	sector
0200CE23	low cylinder
0200CD01	high cylinder
0200CC04	head
0200C810	byte count MSB
0200C700	Ridge address LSB (always 0)
0200C6F0	Ridge address middle byte
0200C503	MS byte of Ridge address
0200C000	GORDER=Read (starts operation)

When the operation completes, the board will generate an interrupt and the Ridge CPU will do an I/O Interrupt Read (IOIR) and get 020300xx, signifying device 2, unit 3, GSTAT=00 (ok).

When the board recognizes an I/O Write address word with its device number in the high-order byte, and the low 24 bits equal to 0000C0H, it builds a request for an 8K byte read from the first and second pages of the disc (head 0, cylinder 0, sectors 2 to 5) and executes it. This boots the system.

SMD CONTROLLER DEVICE CONTROL BLOCK

Hex Addr	Name	Function
0	GORDER	0 Read 1 Write 2 Verify. Reads, but no data is transferred to Ridge memory. 3 Format a track. The value in the SECTOR field becomes the first sector after the index mark on the disc. This allows sector skewing. 4 Seek 5 Return Highest Sector Address. This is the physical address of the last addressable sector in the HEAD, CYLHIGH, CYLLOW, and SECTOR fields. The value in the Byte Count Transferred field is the actual number of bytes between Sector Marks which is needed for interpretation of data from the Read Header order. 6Read Full Sector. Data transfer is always 2064 bytes, the transfer count is ignored. The data read is a 12-byte data label, followed by 2048 data bytes, plus a 4-byte checksum. 7 Write Full Sector. The transfer count is ignored, 2064 bytes are transferred 8 Read ERROR HEADER. Data about media defects is stored on the disc by the manufacturer. Before being destroyed by formatting, this data is transferred by the Ridge factory to the "bad blocks" file for subsequent use by the Ridge Operating System. 9 SETTYPE order. Used at the factory to set the drive type in the controller; this number becomes part of each sector header. This number helps the GORDER=5 function return the appropriate value.
1	SORDER	Not used except for SETTYPE order

Hex Addr	Name	Function
2	GSTAT	0 OK 1 Not ready 2 Timeout 3 Equipment fault 1 Write protected 5 Ridge double bit error in data or DCB transfer 6 Data overrun 7 Missing address mark (Can't find sync byte) 8 Can't find header that matches 9 CRC error in header A Uncorrectable error in data B Seek failure C Unknown drive type FF Illegal parameter in DCB order
3	SSTAT	Reserved.
4	RETRIES	The number of retries attempted on this request.
5-7	RIDGE ADDRESS	This address must be on a word boundary.
8-9	REQUEST BYTE COUNT	2048 to 64K bytes (0 is interpreted to mean 64K bytes).
A-B	BYTE COUNT TRANSFERRED	
C	HEAD	Head number. Range depends on the drive.
D	CYLHIGH	Eight most significant bits of the cylinder number.
E	CYLLOW	Eight least significant bits of the cylinder number.
F	SECTOR	Sector number. Range depends on the drive.
10-1B	DATA LABELS	These twelve bytes are read/written into each sector on a data transfer. This area is also used by the read header GORDER to return the manufacturer's data about defective areas of the disc.
20-3D	HEADER	Subsequent page addresses for multi-page reads and writes. The SMD controller can perform "scatter reads" and "scatter writes." These are contiguous areas on disc starting at CYLINDER, HEAD, and SECTOR (and possibly crossing head or cylinder boundaries) which transfer to or from potentially uncontiguous Ridge main memory addresses. The Ridge

main memory address may change every 4K bytes of transfer length. The 0th page address is specified in the usual location. The 1st through 15th page addresses are stored in sequential 16-bit (2-byte) locations starting at 20H of each unit's DCB. The value stored is actually the 24-bit Ridge memory address divided by 256, giving a 16-bit value. The high-order 12 bits of that value is used as the page number. In the case of disc reads, the data labels are the values read from the last sector only, and there is no checking of these values. In the case of disc writes, the lower four bytes of the data label field are internally set to the value specified, but are incremented by the page number (0 to 15) of the request. The upper 8 bytes remain as specified.

Chapter 5
MONOCHROME GRAPHIC DISPLAY INTERFACE BOARD

CONTROL SWITCHES, INDICATOR LIGHTS, AND PIN ASSIGNMENTS

MONOCHROME DISPLAY BOARD LAYOUT

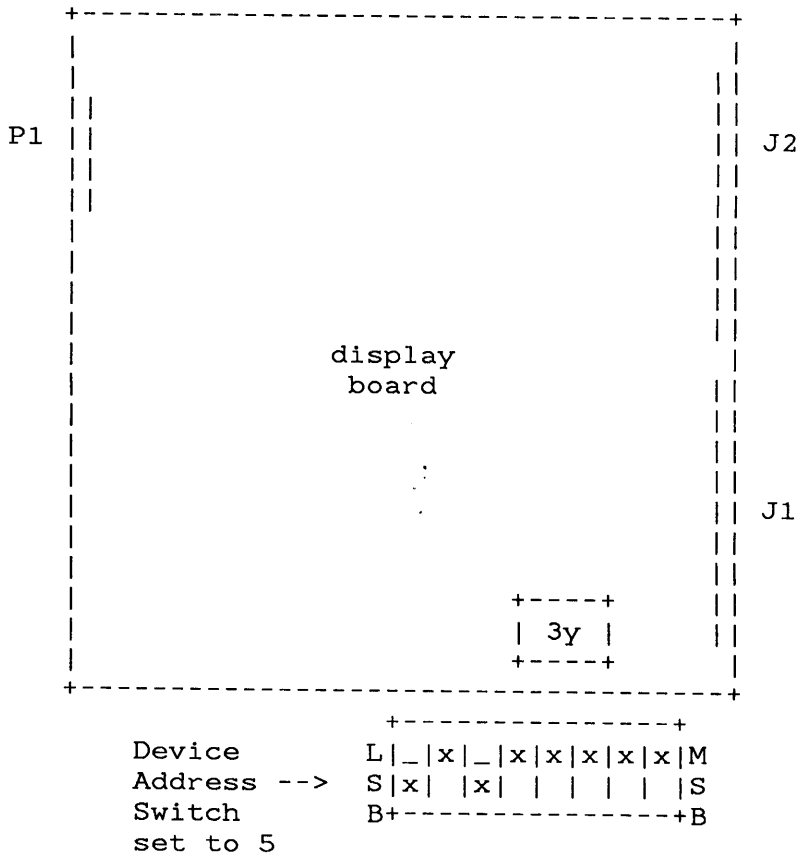


Figure 5-1. Layout of the Monochrome Display Board

Each display is assigned two device numbers. The device switch is set to the desired display device number. The keyboard for that display is then set to that value, less one. (By convention, the first display interface uses 5 for the display and 4 for the keyboard.)

37 for CIO

MONOCHROME DISPLAY BOARD CABLING

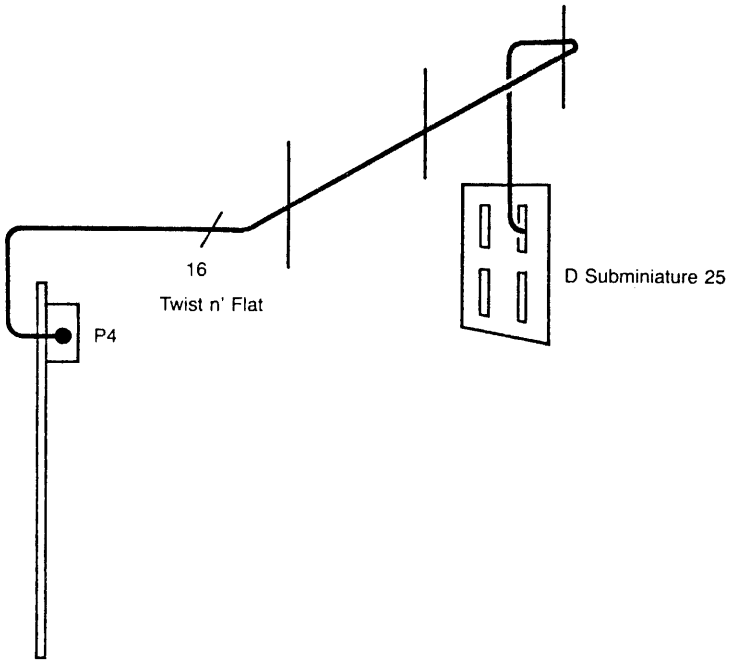


Figure 5-2. Display Board Cabling

Cables from the lowest "P#" are connected to the left-most I/O connector (when viewed from the outside).

THEORY OF OPERATION

(not yet available)

MONOCHROME DISPLAY BOARD PROGRAMMING

The monochrome graphics display interface board supports the graphics display and keyboard. A 128K-byte frame buffer on the board handles display refresh without utilizing Ridge 32 main memory. The refresh buffer uses dynamic RAM chips that are themselves refreshed by the video sweep. The display interface can perform four memory transfer operations:

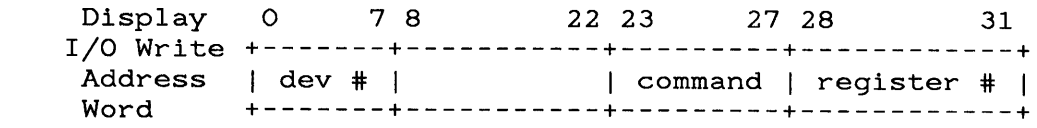
Write Buffer	Move Data from main memory to refresh buffer.
Read Buffer	Move data from refresh buffer to main memory.
Scroll Up	Move data from one place to another in refresh buffer, with increasing addresses.
Scroll Down	Similar to scroll up, except the data is moved with decreasing addresses.

All data operations are multiples of 32 bits, aligned on word boundaries. The video sweep accesses memory in increasing sequential order from 0 to the highest displayable location.

The display interface has four registers to control memory transfers:

1. The memory address register (MAR) is a 24-bit register with two functions; for write/read it contains the main memory source or destination address, for scrolling it contains the destination address in the refresh buffer.
2. The display address register (DAR) is a 16-bit register that contains the refresh buffer source or destination address. For scrolling, it contains the buffer source address.
3. The count register controls the length of a transfer. A count of 0 results in no operation.
4. The status register is used to control display attributes and interrupts, and return information on the display interface's state.

The I/O address and data words to read and write these registers follows:



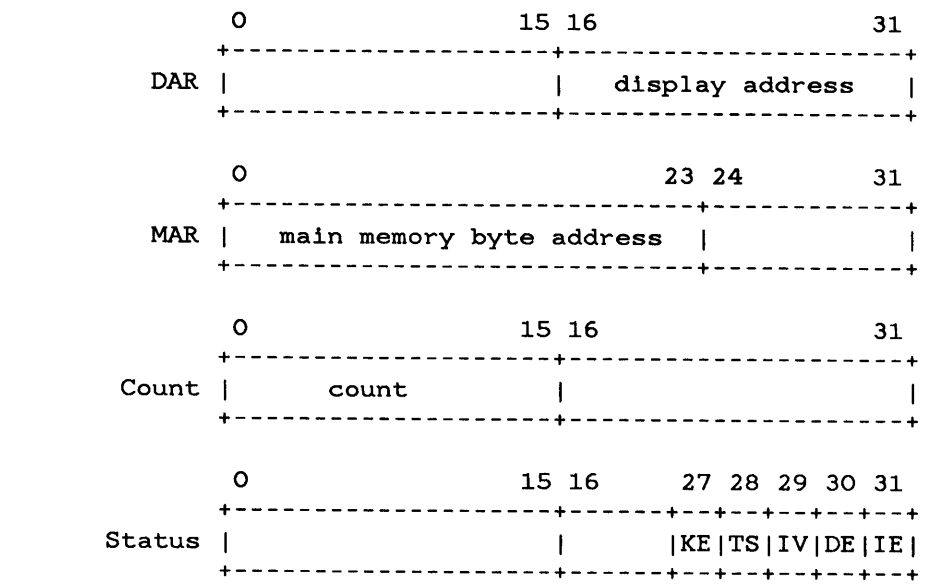
Command — Specifies the operation:

1F - NOP
0E - Write Buffer
0D - Read Buffer
0B - Scroll Up
07 - Scroll Down
0F - Terminate Operation

Register # — Specifies the register to be written.

0 - NOP
1 - DAR
2 - MAR
4 - Count
5 - Both DAR and Count
8 - Status

DISPLAY I/O WRITE DATA WORD



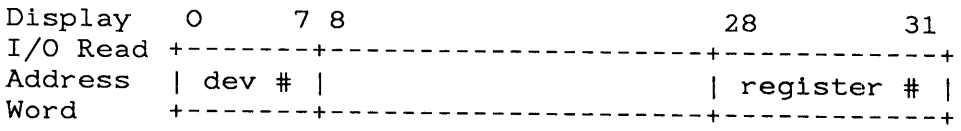
KE Keyboard interrupt enable

TS Top of screen interrupt enable. When set, display interrupts when beam is at top of screen (every 1/60 second).

IV Inverse video. When set, display screen is bright and trace is dark.

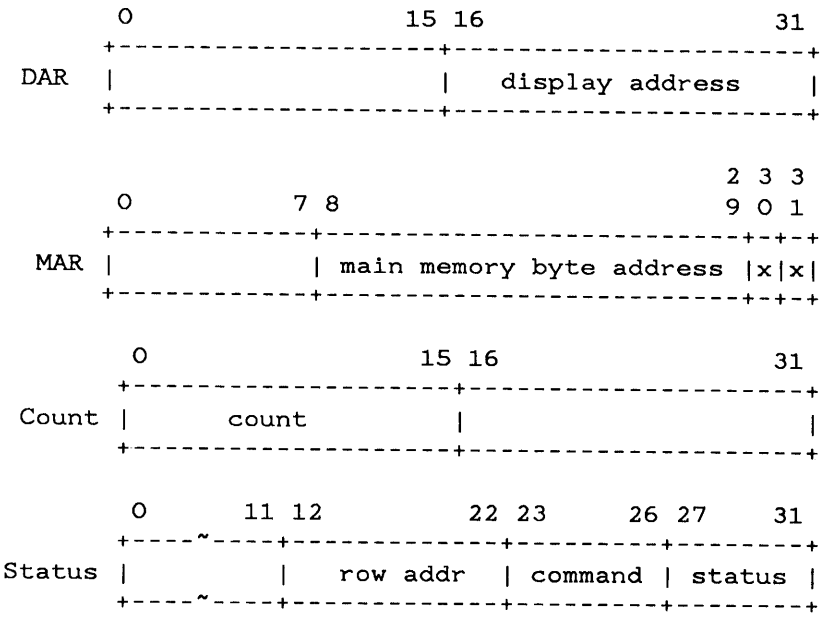
DE Display enable. Turns display on and off. Does not affect refresh of buffer RAM.

IE Interrupts enable. Enables interrupts from both display and keyboard.



Register # - Same as for I/O write address word.

DISPLAY I/O READ DATA WORD



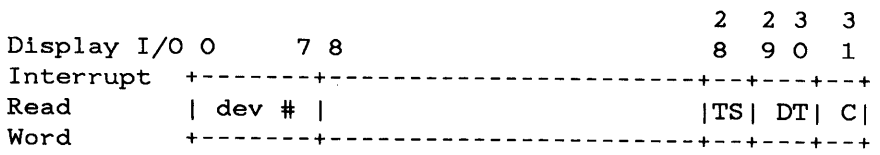
Row addr - Row address of beam.

Command - Operation currently in progress:

1 - Write buffer
2 - Read buffer
4 - Scroll up
8 - Scroll down
0 - Not busy

Status - Same as in I/O write data word.

There are two types of interrupts: display and keyboard. The format of the display and keyboard I/O interrupt read words follow:



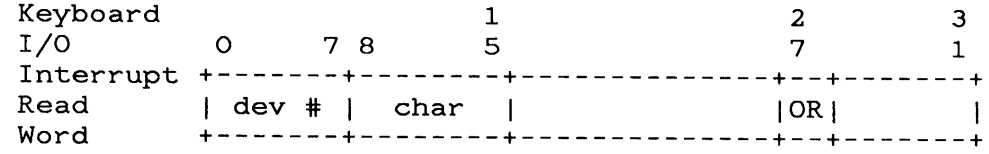
Dev # The least significant bit (bit 7) indicates keyboard or display interrupt. When the display interrupts, bit 7 is set.

TS Beam at top of screen.

DT Display type:
0 - 1024 x 800 horizontal display
1 - 768 x 1024 vertical display
2 - Reserved
3 - Reserved

C Completion. Previous command has finished.

DISPLAY I/O INTERRUPT READ WORD - KEYBOARD



Dev # The least significant bit (bit 7) indicates keyboard or display interrupt. When the keyboard interrupts, bit 7 is clear.

OR Overrun. The display board's 3 word buffer is full, and another character has been received.

Chapter 6
RIDGE 32 TAPE CONTROLLER BOARD

TAPE CONTROLLER SWITCHES AND INDICATOR LIGHTS

TAPE CONTROLLER BOARD LAYOUT

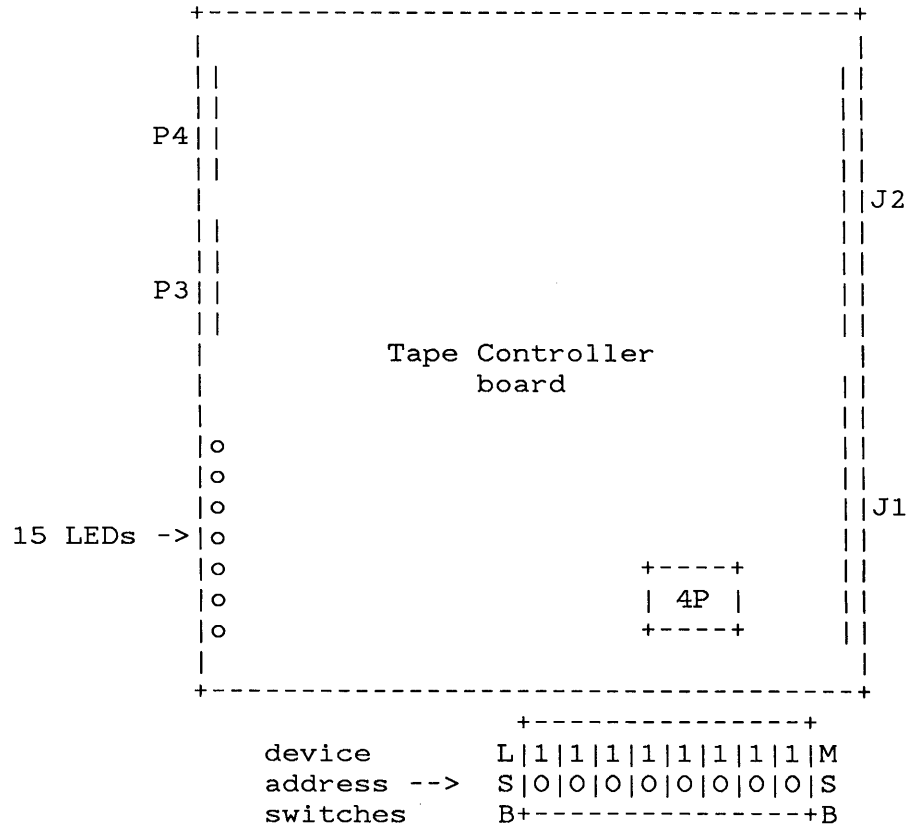


Figure 6-1. Layout of the Tape Controller Board

TAPE CONTROLLER BOARD CABLING

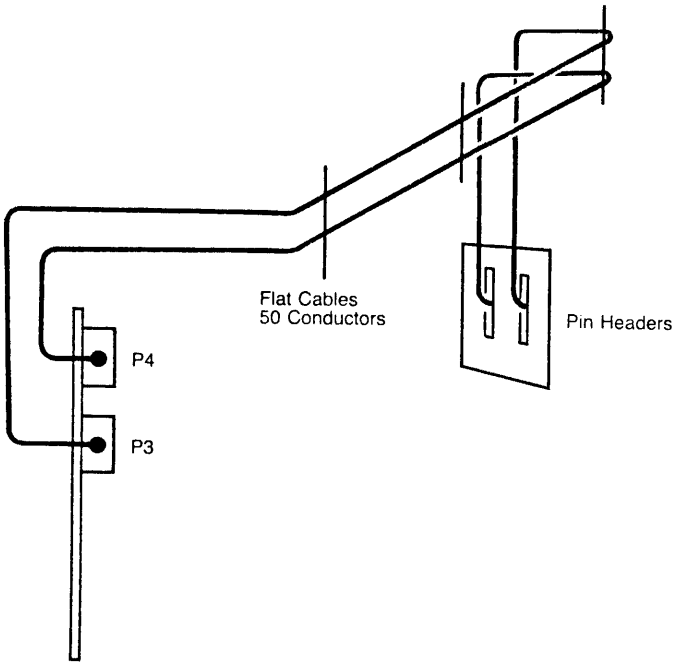


Figure 6-2. Tape Board Cabling

Cables from the lowest "P#" are connected to the left-most I/O connector (when viewed from the outside).

P4 P4 on the tape controller should be connected to P2 on the tape formatter.

P3 P3 on the tape controller should be connected to P1 on the tape formatter.
(Pin 1 on the Ridge is the bottom right pin viewed from the front of the connector)

JUMPERS

Identification The identification code for the tape controller is 20 hexadecimal. Jumpers should be installed in location 6P as shown below:

```
                6P

pin 1     o------o  pin 16   (lsb)
          o------o
          o------o
          o------o
          o------o
          o      o
          o------o
pin 8     o------o  pin 9    (msb)
```

SWITCHES

Device Address — The switches are located in position 4P and a logic zero is produced when the switch is in the "ON" position.

INDICATORS

The LEDs indicate the status of signals from the interface. The LED illuminated corresponds to a logic 1 level.

Signal Name	Status
RDY	The drive is loaded, online, and not rewinding.
FBSY	Formatter busy.
DBSY	Data busy.
FAD	Address of selected formatter.
TAD 0, 1	Address of selected drive.
C0, C1	Last command issued (encoded).

C0	C1	
0	0	GO.
0	1	Rewind.
1	0	Unload.
1	1	Online.

ERASE, WRIT, EDIT, REV, FMK	Operation to be performed with a GO command.
SPEE	High speed selected.
FEN	Formatter enabled.

PIN ASSIGNMENTS

Signals from Controller

Connector Name	Live Pin	Ground Pin	Signal Name ("low true")
P3	4	3	LWD - Last Word
P3	6	5	W4 - Write Data 4
P3	8	7	GO - Initiate Command
P3	10	9	WO - Write Data 0
P3	12	11	W1 - Write Data 1
P3	16	15	-- - reserved
P3	18	17	REV - Reverse
P3	20	19	RWD - Rewind
P3	22	21	WP - Write Data Parity
P3	24	23	W7 - Write Data 7
P3	26	25	W3 - Write Data 3
P3	28	27	W6 - Write Data 6
P3	30	29	W2 - Write Data 2
P3	32	31	W5 - Write Data 5
P3	34	33	WRT - Write
P3	36	35	-- - reserved
P3	38	37	EDIT - Edit
P3	40	39	ERASE - Erase
P3	42	41	WFMK - Write File Mark
P3	14	13	-- - reserved
P3	46	45	TAD0 - Transport Address 0
P4	18	17	FEN - Formatter Enable
P4	24	23	UNLOAD - Rewind/Unload
P4	46	45	TAD1 - Transport Address 1
P4	48	47	FAD - Formatter Address
P4	50	49	SPEED - High Speed Select
P3	2	1	FBSY - Formatter Busy
P3	44	43	-- - reserved
P3	48	47	R2 - Read Data 2
P3	50	49	R3 - Read Data 3

Signals to Controller

Connector Name	Live Pin	Ground Pin	Signal Name ("low true")
P4	1	-	RP - Read Data Parity
P4	2	-	R0 - Read Data 0
P4	3	-	R1 - Read Data 1
P4	4	-	LPT - Load Point
P4	6	5	R4 - Read Data 4
P4	8	7	R7 - Read Data 7
P4	10	9	R6 - Read Data 6
P4	12	11	HER - Hard Error
P4	14	13	FMK - File Mark
P4	16	15	PE - Identification
P4	20	19	R5 - Read Data 5
P4	22	21	EOT - End of Tape
P4	26	25	-- - reserved
P4	28	27	RDY - Ready
P4	30	29	RWD - Rewinding
P4	32	31	PROT - File Protect
P4	34	33	RSTR - Read Strobe
P4	36	35	WSTR - Write Strobe
P4	38	37	DBSY - Data Busy
P4	40	39	SPEED - High-Speed Status
P4	42	41	CER - Corrected Error
P4	44	43	ONL - On-line

THEORY OF OPERATION

The Ridge tape controller is a single printed circuit board assembly that provides a Pertec Formatter Interface for the Ridge 32.

The interface is defined on pages 19 through 39 of the "Cipher Microstreamer Product Description Manual." A brief overview of the controller hardware is presented here.

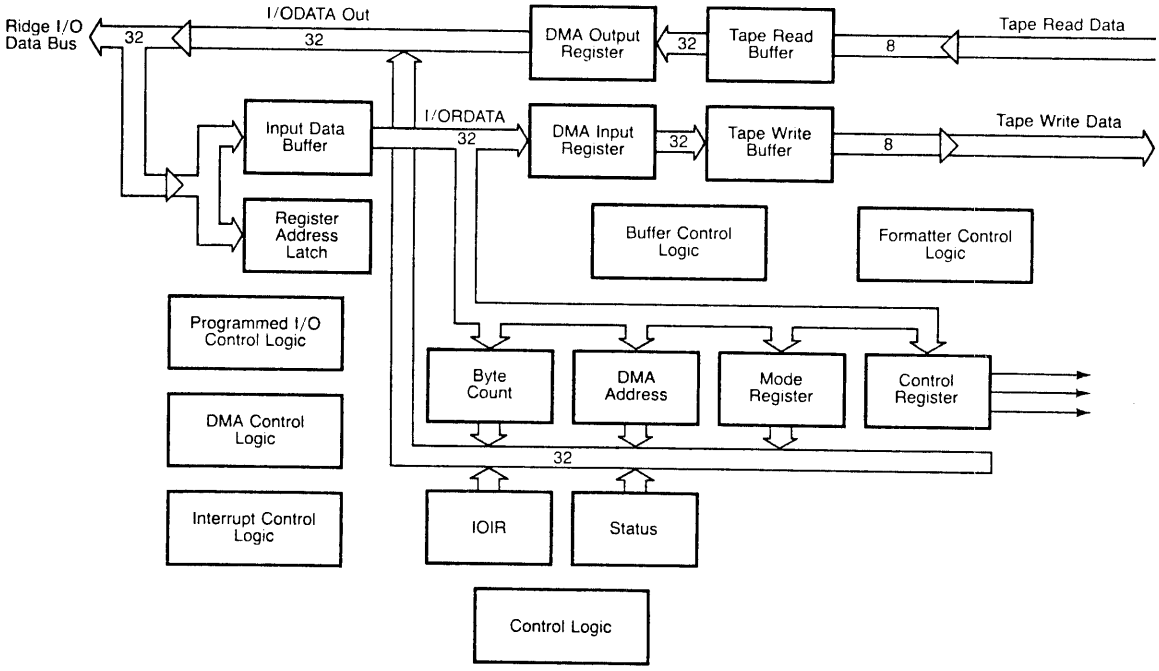


Figure 6-3. Block Diagram of Tape Controller

The Programmed I/O (PI/O) logic allows Ridge software to control and monitor many of the interface signals and to set up data transfers with Ridge I/O Read and I/O Write instructions. Registers are selected with the two least significant bits of the I/O address word, but only the mode register can be modified while a tape Read or Write is in progress.

The PI/O logic follows the standard timing. In a Read, however, ACKMCIO is delayed one cycle to allow data to be enabled on the internal bus before it appears on the Ridge bus.

When IODACK is asserted for a Write, the data is clocked into a temporary buffer before it is latched in the specified register. IODACK is delayed 125 nanoseconds and called "input data available" before it enters the state machine. This reduces the amount of Schottky logic.

The Tape Control Logic synchronizes the clocks from the interface, multiplexes the byte-wide interface busses and the 32-bit Ridge word, and maintains the transfer length counter.

On a tape Read, data is clocked from the Data Input lines to one of four input registers when the leading edge of RSTROBE is detected. On a Tape Write, the contents of one of four output registers is enabled onto the Data Out lines when the trailing edge of WSTROBE is detected.

The Tape Byte Counter is incremented with each 8-bit transfer. When the count overflows, data will no longer be loaded on a Read transfer, and the "last word" signal will be asserted on a write.

The Byte of Word Counter specifies which of the four input or output registers will receive or provide the data byte. The counter is incremented after each byte transfer.

For a tape Read, the "Buffer Serviced" line is asserted after the fourth register has been selected to indicate that the input buffer is full. For a tape Write, the "Buffer Serviced" line is asserted after the fourth register has been selected to indicate that the output buffer is empty.

The Buffer Control Logic transfers data between the Ridge DMA registers and the tape buffers, and instructs the DMA logic when to start a transfer. On a tape Write, the DMA logic tries to keep the input buffer full, so an extra 32-bit Ridge word will probably be fetched when the transfer ends.

The control logic enables the DMA, tape, and Buffer Control hardware. The logic is enabled when a "GO" pulse is generated at the start of a transfer, and disabled when "DBSY" is negated. After a Read, an extra Ridge DMA cycle may be necessary to flush a partially filled input buffer. After a Write, the Ridge DMA machine is allowed to finish filling the input buffer before the hardware is disabled.

Interrupts are generated when the READY signal goes true, or when the control logic finishes a tape Read or Write.

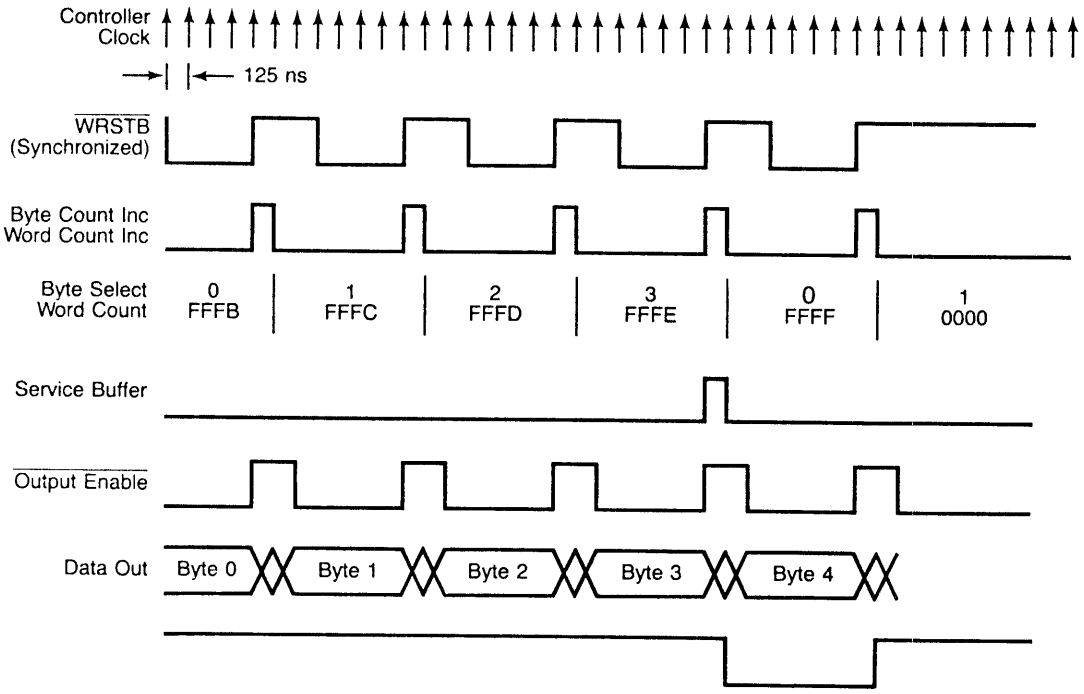


Figure 6-4. A Normal Tape Write

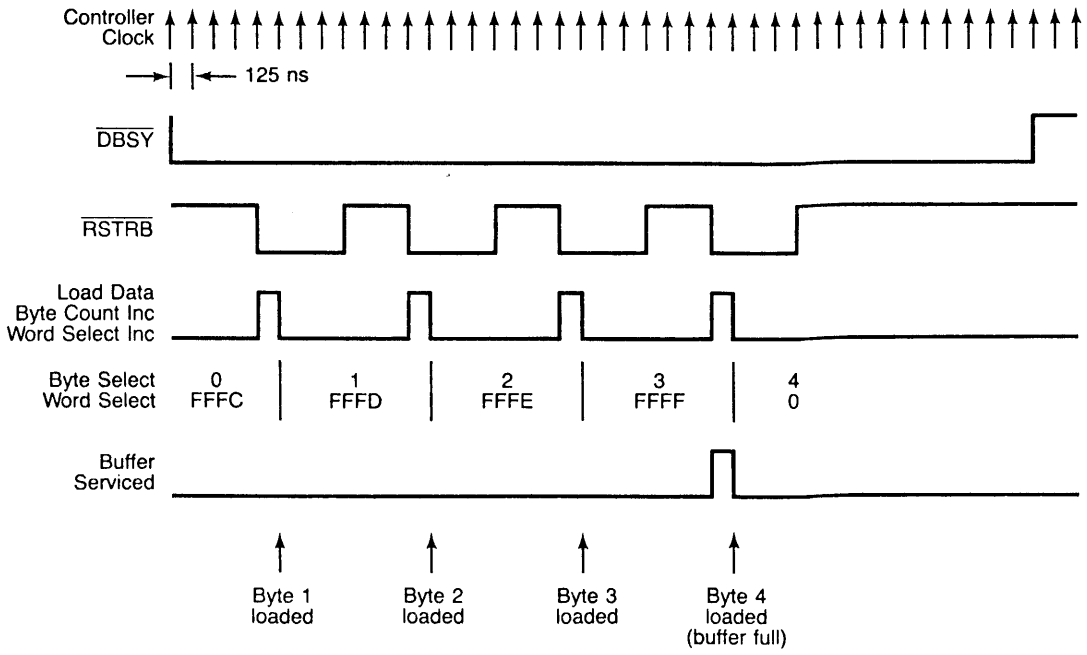


Figure 6-5. A Normal Tape Read

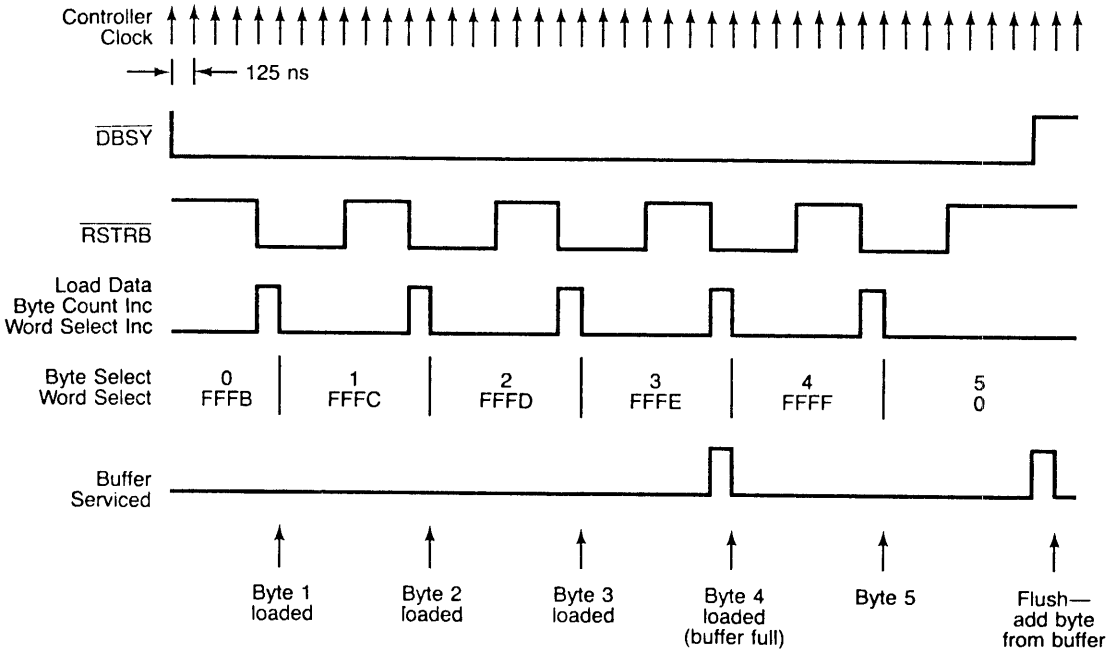


Figure 6-6. An Odd-Length Read Transfer with Buffer Flushing

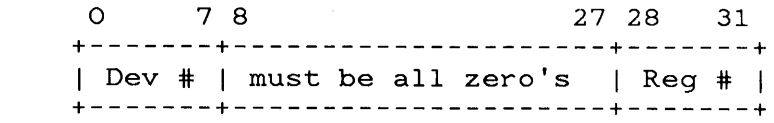
TAPE CONTROLLER PROGRAMMING

The Tape Controller has a standard Pertec interface, which is controlled by the software in the Ridge. With an I/O Read instruction, the status lines from the interface can be sensed, and with an I/O Write, the control lines can be asserted.

Data is exchanged by DMA. The Ridge sets up the count and address registers on the controller and the hardware interrupts when the transfer has completed.

REGISTER DEFINITIONS

Address Word

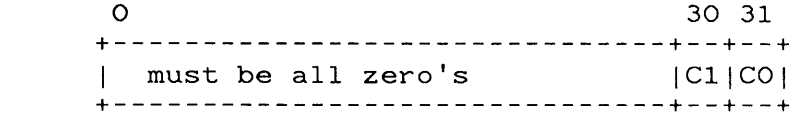


Register #	Read
0	Status
1	Mode
2	DMA Addr
3	Byte Count
	Write
0	Control
1	Mode
2	DMA Addr
3	Byte Count

IODNV (I/O Data Not Valid) status will be returned if the Control, DMA Address, or Byte Count registers are accessed with an I/O write instruction while the "Command in Progress" bit is set.

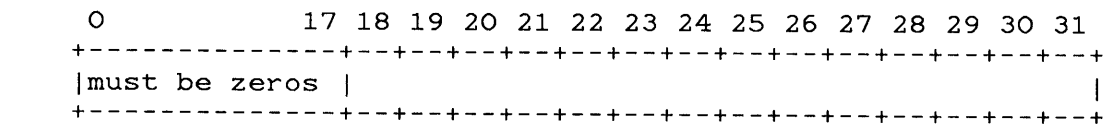
Control Register

The Control register initiates action on the drive. Loading a 2-bit code produces the required pulse on 1 of 4 formatter signal.



C1	C0	Signal Pulsed
0	0	GO
0	1	REWIND
1	0	REWIND & UNLOAD

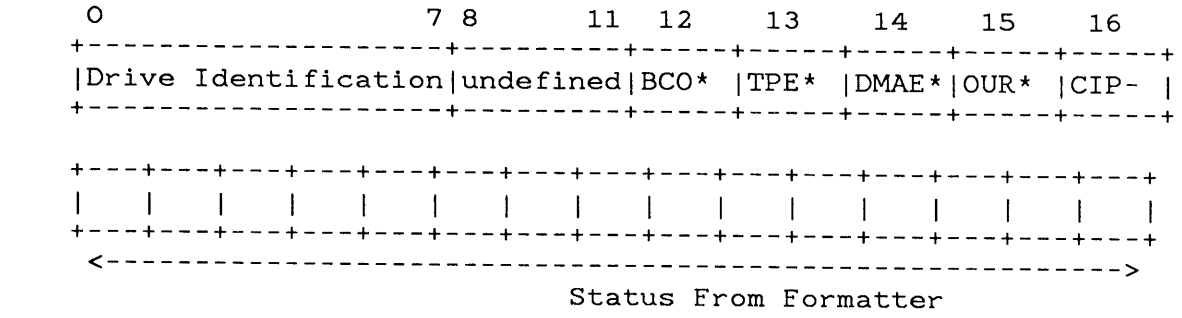
Mode Register



Bit

0-17		- must be zeros
18	IE	- Interrupt Enable
19	DMA	- DMA Enable
20	FEN	- Formatter Enable
21	FAD	- Formatter Address
22	TAD1	- Transport Address LSB
23	TAD0	- Transport Address MSB
24	REV	- Reverse
25	WRITE	- Write
26	ERASE	- Erase
27	EDIT	- Edit
28	FMK	- Filemark
29	SPEED	- Speed
30	P3-36	- Undefined Interface Control Signal Connected To P3 Pin 36
31	P3-14	- Undefined Interface Control Signal Connected To P3 Pin 44

Status



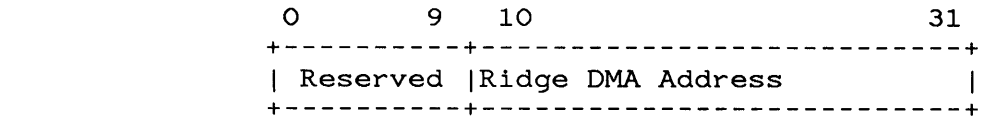
Bit

12	BCO *	- Byte Count Overflow
13	TPE *	- Tape Parity Error
14	DMAE *	- DMA Error
15	OUR *	- Over/Under Run
16	CIP	- Command In Progress

Bit		
17	FMK *	- Filemark
18	HER *	- Hard Error
19	CER *	- Corrected Error
20	IDENT *	- PE Identification
21	P4/26	- Undefined Signal Connected To P4-26
22	P3/44	- Undefined Signal Connected To P3-14
23	HISP	- High Speed
24	RWD	- Rewind
25	EOT	- End Of Tape
26	LPT	- Load Point
27	FPT	- File Protect
28	ONL	- On Line
29	RDY	- Ready
30	FBSY	- Formatter Busy
31	DBSY	- Data Busy

* Cleared with next command.

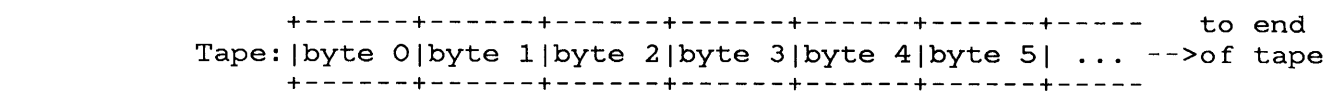
DMA ADDRESS



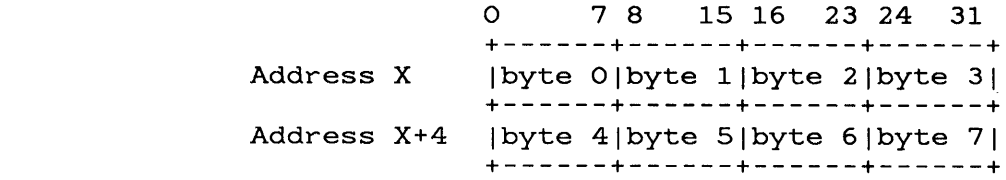
The DMA Address Counter addresses 32-bit Ridge words; therefore, the two least significant bits of the address are not used by the controller.

The address is decremented after each DMA transfer, if the REVERSE bit is set in the mode register. If the bit is "0" the counter is incremented.

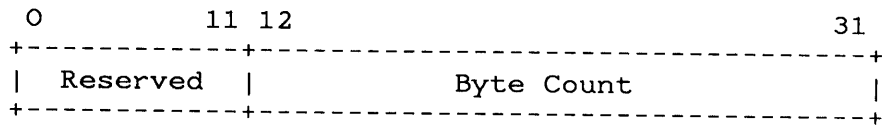
Data is transferred, with the least significant byte of tape data in the most significant part of the Ridge word.



Ridge Memory:



BYTE COUNT



The number of bytes to be transferred must be loaded in 2's complement.

FFFFF will produce a 1 byte transfer. 00000 will produce a 2**20 byte transfer.

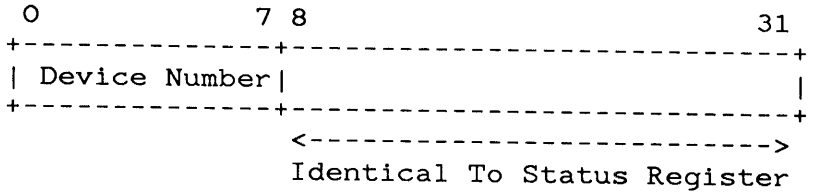
The byte counter is always enabled after a "GO" command is issued even if the DMA logic is not enabled.

When the count is incremented past 00000, the Byte Count Overflow bit, in the status register will be set and the DMA Logic will be disabled until the read operation stops.

The length of a record can be determined by loading 00000 into the counter and initiating a read operation with DMA disabled. When the operation finishes the byte counter can be read to determine the record length.

INTERRUPTS

Interrupts are generated when either the "READY" signal is asserted or the "Command in Progress" bit goes false.



If the interrupt enable bit is disabled after the controller has made an interrupt request to the Ridge, the Ridge interrupt handshake will continue to completion.

DIAGNOSTICS FOR TAPE CONTROLLER BOARD AND DRIVE

The Ridge Tape Controller Diagnostic has two modes; Auto Debug mode checks the hardware and the drive, and Manual Debug mode permits the user to read and write tapes.

RUNNING THE DIAGNOSTIC

This diagnostic can be run under the control of **systest**, **sus**, or individually. In these examples, the diagnostic is run individually.

Before using the tapedriver diagnostic with **systest** or individually, log in as **root** and remove all lines from the **/ros/conf** file *except*:
`:1:/drivers/fdlp`
When finished with the diagnostic, re-insert the line(s).

AUTO DEBUG MODE

There are three tests in the auto debug mode.

In the register test, a one and a zero is moved through all bit positions of the registers. If the contents of a register is not equal to the expected value, an error message is written.

In the loopback test, most of the control signals of the interface are tested. A special loopback connector must be installed in place of the cables that are attached to the drive.

The DMA test exercises both the tape controller and the drive. Records are written on the tape, read, and checked. Spacing operations are performed and verified, and tests with oversized and undersized I/O buffers check most of the hardware.

Source	(Connector/ Pin)	Receiver	(Connector/ Pin)
LWD	(P3/2)	FBSY	(P3/4)
GO	(P3/8)	LDP	(P4/4)
	(P3/14)	HER	(P4/12)
REV	(P3/18)	FMK	(P4/14)
RWD	(P3/20)	EOT	(P4/22)
WRT	(P3/34)	RDY	(P4/28)
	(P3/36)	RWD	(P4/30)
EDIT	(P3/38)	FPT	(P4/32)
ERASE	(P3/40)	DBSY	(P4/38)
TAD0	(P3/46)		(P3/44)
FEN	(P4/18)	IDENT	(P4/16)
RWU	(P4/24)		(P2/26)
TAD1	(P4/46)	ONL	(P4/44)
FAD	(P4/48)	CER	(P4/42)

The output of the program running the register and loopback tests appears below: User inputs are in bold.

```
$ tape                                  <<to run the diagnostic>>

Tape Diagnostic 14-Jan-84

Found tape as device 000000FF           <<address of tape controller>>
Do you want automatic debug (Y)?        y [RETURN]
DoAutoDebug
Do you want to do register bit tests (N)? y [RETURN]
If connected, make sure
drive is not online, then hit <Return>  [RETURN]
Walking ones thru DMA register
Walking zeros thru DMA register
Walking ones thru Byte Count register
Walking zeros thru Byte Count register
Walking ones thru Mode register
Walking zeros thru Mode register
Do you want to do loopback tests (N) ?  y [RETURN]
Starting Loopback tests
Attach loopback connector, then hit <Return>  [RETURN]
LoopBack test completed
Do you want to do DMA tests (Y) ?       n [RETURN]

$<<end of program>>
```

The output from the diagnostic running the DMA test appears below: User responses are in bold.

$ tape

Tape Diagnostic 14-Jan-84

Found tape as device 000000FF
Do you want automatic debug (Y) ? **y [RETURN]**
DoAutoDebug
Do you want to do register bit tests (N) ? **y [RETURN]**
Do you want to do loopback tests (N) ? **n [RETURN]**
Do you want to do DMA test (Y) ? **y [RETURN]**
Enter Address of Drive: **0 [RETURN]**
Do you want to do high speed IO (N) ? **n [RETURN]**
Enter IO Block Size (hex) : **1234**
Starting DMA tests
Install Cipher flat cables, then hit <Return>
Insert test tape in drive with write ring installed. Then hit <Return>
Rewinding tape
Rewind completed
writing block
writing block completed
writing block
writing block completed
writing file mark
writing file mark completed
writing bloc
writing block completed
writing file mark
Rewinding tape
Rewind completed
reading block
reading block completed
Comparing blocks
Compare complete
reading block
reading block completed
Comparing blocks
Compare complete
reading block
reading block completed
reading block
reading block completed
Comparing blocks
Compare complete
reading block
reading block completed
Comparing blocks
Compare complete
reading block
reading block completed

<<theses two messages are reminders>>

<<the number of bytes to be written in a record for this test>>

<<the tape controller controls up to eight drives (two formatters with four drives>>

<<the operation being performed>>

```
reading block
reading block completed
Skipping  backward file
Skipping  backward file
Skipping  backward record
Skipping  backward record
Reading Long block into short buffer
reading block
reading block completed
Comparing blocks
Compare complete
Skipping  forward file
writing block
writing block completed
writing file mark
writing file mark completed
writing file mark
writing file mark completed
Skipping  backward file
Skipping  backward file
Skipping  backward file
Skipping  forward file
Reading Short block into long buffer
reading block
reading block completed
Comparing blocks
Compare complete
Do you want to unload the tape (N) ?          y [RETURN]
Rewinding and Unloading tape

$
```

MANUAL DEBUG MODE

In Manual Debug mode, the operation of the drive can be controlled by entering command codes. The codes available are:

E	or Q	Exit
	H	Help
	B	Backspace files
	C	Change block size
	G	Get file from tape
	R	Rewind
	S	Skip files
	T	Dump tape controller status register
	U	Unload tape
	V	Change verbose level
	W	Write file onto tape

In the example below, a file called FILE1 is written to the tape, in 1000 byte blocks, using the "W" command code; the tape is rewound with "R" and the file is read with "G" into file2.

$ tape

```
Tape Diagnostic 14-Jan-84
Found tape as device 000000FF
Do you want automatic debug (Y) ?                    n [RETURN]
DoManualDebug
Enter Address of Drive:                              0 [RETURN]
Do you want to do high speed IO (N) ?                n [RETURN]
Enter IO Block Size (hex) :                          1000 [RETURN]
Enter command code (H for Help) :                    w [RETURN]
Enter disc file name :                               file1 [RETURN]
30000 bytes )
60000 bytes ) <-------------------------------------- <<total bytes written>>
Wrote 89732 bytes
Enter command code (H for Help) :                    r [RETURN]
Rewinding tape
Enter command code (H for Help) :                    g [RETURN]
Enter disc file name :                               file2 [RETURN]
Record size (hex) is: 00001000 |
Record size (hex) is: 00001000 |
Record size (hex) is: 00001000 |
Record size (hex) is: 00001000 |
Record size (hex) is: 00001000 |
Record size (hex) is: 00001000 | <-------------------- <<the records being read>>
Record size (hex) is: 00001000 |
Record size (hex) is: 00001000 |
30000 bytes                    |
Record size (hex) is: 00001000 |
Record size (hex) is: 00001000 |
Record size (hex) is: 00001000 |
```

```
Record size (hex) is: 00001000
Record size (hex) is: 00001000
Record size (hex) is: 00001000
Record size (hex) is: 00001000
60000 bytes
Record size (hex) is: 00001000
Record size (hex) is: 00001000
Record size (hex) is: 00001000
Record size (hex) is: 00001000
Record size (hex) is: 00001000
Record size (hex) is: 00001000
Record size (hex) is: 00000E84
Record size (hex) is: 00000000
Full blocks = 21, Partial blocks = 1, Bytes in file = 89732
Enter command code (H for Help) :                    u [RETURN]
Rewinding and Unloading tape
Enter command code (H for Help) :                    q [RETURN]

$
```

If for the "get" command, the block size, specified by the user, were smaller than the record on the tape, then the actual record size would have been printed with a warning message. The "get" function should be repeated; however, the buffer size should be modified with the "C" command. If the buffer specified is larger than the actual record the "get" function would operate properly.

The warning message from the "get" command:

```
Record size (hex) is: 00002000  READ INCOMPLETE - IO Buffer is too SMALL
Record size (hex) is: 00002000  READ INCOMPLETE - IO Buffer is too SMALL
Record size (hex) is: 00002000  READ INCOMPLETE - IO Buffer is too SMALL
30000 bytes
Record size (hex) is: 00002000  READ INCOMPLETE - IO Buffer is too SMALL
Record size (hex) is: 00002000  READ INCOMPLETE - IO Buffer is too SMALL
Record size (hex) is: 00002000  READ INCOMPLETE - IO Buffer is too SMALL
Record size (hex) is: 00002000  READ INCOMPLETE - IO Buffer is too SMALL
60000 bytes
Record size (hex) is: 00002000  READ INCOMPLETE - IO Buffer is too SMALL
Record size (hex) is: 00002000  READ INCOMPLETE - IO Buffer is too SMALL
Record size (hex) is: 00001E84  READ INCOMPLETE - IO Buffer is too SMALL
Record size (hex) is: 00000000
Full blocks = 0, Partial blocks = 11, Bytes in file = 89732
```

Changing the I/O Block size and repeating the "get" command:

```
Enter command code (H for H lp) :          c [RETURN]
Enter IO Block Size (hex) :                2000
Enter command code (H for Help) :          r [RETURN]
Rewinding tape
Enter command code (H for Help) :          g [RETURN]
Enter disc file name :                     file2[RETURN]
Record size (hex) is: 00002000
Record size (hex) is: 00002000
Record size (hex) is: 00002000
        .
        .
        .
```

The "v" command controls the amount of information printed when a command is executing. Shown below is a "w" command with the verbose level set to 1; the previous examples had the verbose level set to the default value, 2.

Changing the verbose level:

```
Enter command code (H for Help) :          v [RETURN]
Verbose level = 00000002. Enter new level : 1 [RETURN]
Enter command code (H for Help) :
```

The "w" command with the new verbose level:

```
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
30000 bytes
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
```

```
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
60000 bytes
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
writing block
writing block completed
writing file mark
writing file mark completed
writing file mark
writing file mark completed
Skipping  backward record
Skipping  backward file
Skipping  forward file
Wrote 89732 bytes
Enter command code (H for Help) :
```

Chapter 7
DR11 INTERFACE CONTROLLER

DR11 INTERFACE SPECIFICATIONS

INTRODUCTION

The Ridge DR11 is an asynchronous parallel interface for peripheral equipment or an intercomputer link. The interface emulates the DR11W defined by the Digital Equipment Corporation; however, not all of the functions and modes have been implemented, and the timing has been more rigorously defined. The maximum transfer rate of the Ridge DR11 is 2 Mbytes per second.

INTERFACE SIGNALS

From Peripheral Equipment

DI0-15	DATA IN - A 16-bit unidirectional data bus from the peripheral. The high order 8 bits are lines DI7 through DI0 and low order byte is DI15 through DI8.
CYCLE REQ A,B	CYCLE REQUEST A and B - Either line can initiate a one word transfer, provided READY and BUSY are negated.
ATTN	ATTENTION - Halts the data transfer, sets READY, and generates an interrupt to the host CPU.
STATUS A,B,C	Status or control signals to be defined by the peripheral device.

To Peripheral Equipment

DO0-15	DATA OUT - A 16-bit unidirectional data bus to the peripheral. The high order 8 bits are lines DO7 through DO0 and the low order byte is DO15 through DO8.
INIT	Initialization or reset command.

FUNCT 1,2,3	Control signals to be defined by the peripheral device.
GO	A pulse generated at the beginning of a DMA block transfer.
READY	Reset with GO; set when the DMA transfer has completed.
BUSY	When BUSY is asserted, the host CPU is processing a CYCLE REQUEST. Another CYCLE REQUEST cannot be issued until BUSY is negated with READY asserted.
END CYCLE	A pulse generated when BUSY is negated.
ACLO FUNCT2	Same as FUNCT 2.

Signals Not Implemented

The following signals are included in the DEC DR11W but are not implemented in the Ridge version of the interface:

CO CNTHL
C1 CNTHL
WC INC ENB
BA INC ENB
AOO
BURST

Electrical Interface

The receivers are Motorola MC3437, National DS8837 or Signetics 8T37 and the drivers are 74S38.

There is one receiver for each input, and there is one driver for each output. The two FUNCT 3 lines each have a driver, and there is a receiver for both CYCLE REQ A and CYCLE REQ B. STAT C is received only on J1 pin 25.

220/330 Ohm termination is installed at both the drivers, the receivers, and all unimplemented lines.

Data Transfers

The DR11 Interface operates in either normal or link mode and data is transferred either by DMA cycles or Programmed I/O.

Normal Mode DMA Cycles

At the beginning of a block transfer of 1 to 64K words, the controller pulses "GO" and negates "READY". The user device initiates the exchange of each word by pulsing either "CYCLE REQUEST A" or "CYCLE REQUEST B" and the controller asserts "BUSY" while data is being exchanged with the memory in the host machine.

At the conclusion of a cycle, the controller produces an "END" pulse and the user device either supplies the next word on the "DATA IN" bus for a read or latches the contents of the "DATA OUT" bus for a write.

After the last word of the block has been transferred, or the user aborts the operation by pulsing "ATTN", the controller will assert "READY".

The interface will not support byte transfers, read-modify-write cycles or overlapping I/O cycle requests; however, DMA cycles can be primed, so that the first word can be transferred when "GO" is pulsed without a "CYCLE REQUEST".

Normal Mode Programmed I/O Transfers

Data, Control, or Status can be exchanged under direct program control of the host Ridge computer. The protocol, using the Function and Status lines, is to be defined by the peripheral device.

Link Mode

An interprocessor link can be formed by cross-coupling the cable between two DR11Ws. The interface functions as previously defined, but the polarity of BUSY is inverted so the "transfer complete" or "not Busy" condition on one system will generate a cycle request to the other.

To begin a block transfer DMA, the transmitting DR11W must make the first transfer when GO is asserted.

DR11 CONTROL SWITCHES, INDICATOR LIGHTS, AND PIN ASSIGNMENTS

DR11 BOARD LAYOUT

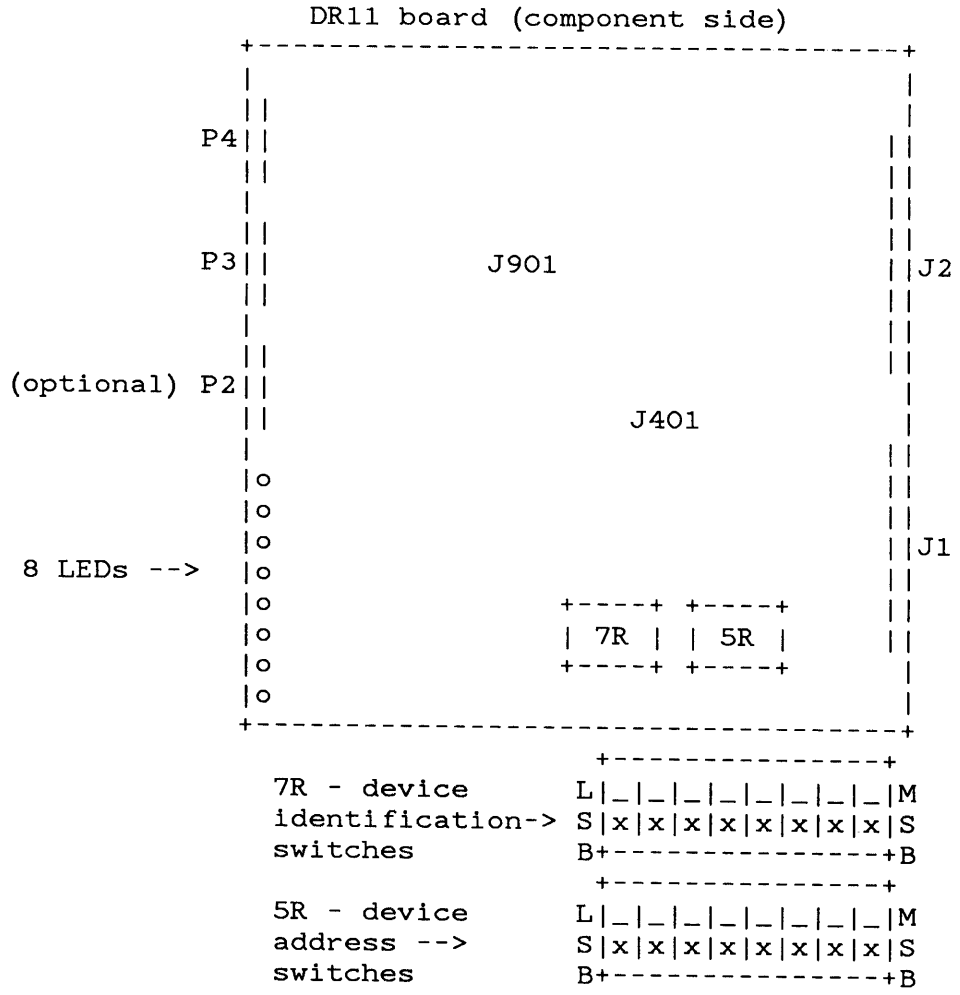


Figure 7-1. Layout of the DR11 Board

DR11 CABLING

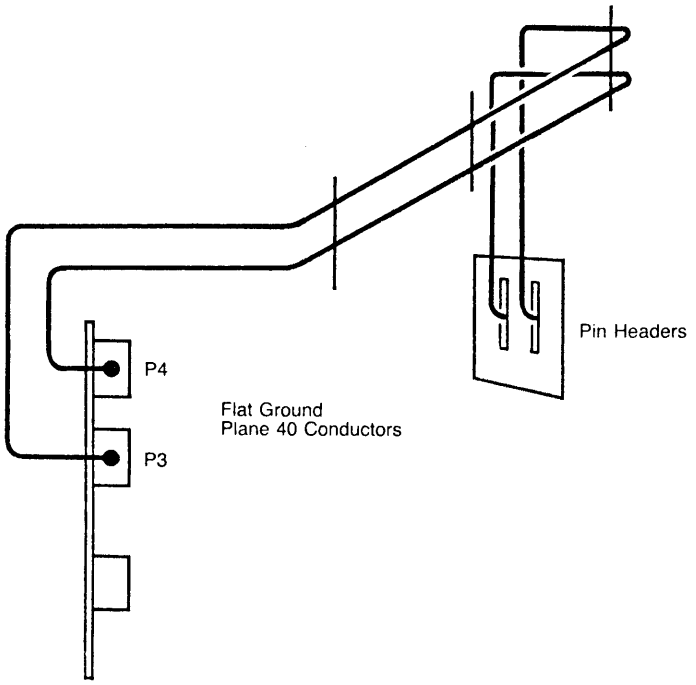


Figure 7-2. DR11 Board Cabling

Cables from the lowest "P#" are connected to the left-most I/O connector (when viewed from the outside).

JUMPERS

Keyboard enable — With J402 installed, the keyboard logic will be enabled and the controller will occupy 2 device addresses. J402 is located between 5L and 6L.
The jumper should be omitted if the keyboard logic (chips 10L, 10M, 10N ...) is not present

DR11 Link Mode — For the DR11 logic to operate in "link mode" the jumper should be installed. The jumper is located between 6F and 6G.
The Ungermann-Bass NIU Ethernet device operates in either mode.

SWITCHES

Address — Card addressing depends on jumper 402. If it is omitted, the card will respond to the address selected on the switch; however, if the jumper is installed, the controller will occupy the 2 addresses specified by the seven most

significant switches. An odd address selects the keyboard port and an even will enable the DR11 logic.
The switches are located in position 5R and a logic zero is produced when the switch is in the "ON" position.

Identification

The card identification code depends on the device attached to the controller.

Card I.D.	Device
30 (hex)	Metheus display controller
31	Metheus contrllr with keyboard
32	Ungermann-Bass NIU-150
3C-3F	user DR11 devices

The switches are at position 7R. A logic zero results with the switch in the "ON" position.

CONNECTORS

P4

P4 is for DR11 data being sent to the Ridge from the user device. It is labeled DI on the board and it should be connected to J5 on the Metheus, to "output" on the Raster, and to the rightmost connector (when viewed from the rear) on the UB NIU-150.

P3

P3 is for DR11 data being sent to the User device from the Ridge. It is labeled DO on the board and it should be connected to J6 on the Metheus, to "input" on the Raster, and to the leftmost connector on the NIU-150.

Pin 1 on the Ridge is the bottom right pin viewed from the front of the connector and pin 1 on the Metheus and Raster is on the is on the right side of the connector.

P2

P2 is the Ridge Keyboard interface.

INDICATORS

The eight LEDs indicate the status of signals from the DR11 interface. They can help troubleshoot problems. The LED illuminated corresponds to a logic 1 level.

READY — When "RDY" is illuminated, a DR11 block transfer is not in progress.

BUSY — BUSY indicates a DMA cycle is in progress. Under normal conditions, the LED will be on for only several hundred nanoseconds. If is stays on, the DMA daisy chain should be checked or the card cage should be examined for a missing card.

F1,F2,F3 — The function lines are defined by the user device.

For the Metheus:

F1 Transfer direction:

0 <=> to the Metheus.
1 <=> to the Ridge.

F2 undefined.

F3 Abort transfer and interrupt.

For the Raster, the lines are encoded as:

F3	F2	F1	
0	0	0	Transfer from Ridge.
0	1	0	Transfer to Ridge.
1	0	0	"Warm reset".

For the NIU-150, the lines are encoded as:

F3	F2	F1	
0	0	0	Reset
0	0	1	Preview frame (read up to 1st 32 bytes)
0	1	0	Output frame
0	1	1	Read frame
1	0	0	Flush frame
1	1	0	Idle

Commands are presented to the NIU by first placing the code on the function lines, then pulsing the GO line. The NIU terminates commands by pulsing the ATTN line.

SA,SB,SC

The Status lines are defined by the user device.

For the Metheus:

SA Device Ready.
SB Data is available for the Ridge.
SC Diagnostics are in progress.

For the Raster:

SA Device Not present.

SC	SB	
0	0	Error condition.
0	1	Transfer from Raster has completed.
1	0	Transfer interrupted by terminal.
1	1	Ready.

For the NIU-150:

SA NIU error
SB Frame ready for the Ridge
SC NIU ready for data from Ridge

The Ridge controller is supplied with two 25-foot 40-conductor cables.

PIN ASSIGNMENTS

Pin	(DEC)	J1 Signal	J2 Signal
1	(VV)	+DO 15	+DI 15
2	(UU)	+DO 00	+DI 00
3	(TT)	+DO 14	+DI 14
4	(SS)	+DO 01	+DI 01
5	(RR)	+DO 13	+DI 13
6	(PP)	+DO 02	+DI 02
7	(NN)	+DO 12	+DI 12
8	(MM)	+DO 03	+DI 03
9	(LL)	+DO 11	+DI 11
10	(KK)	+DO 04	+DI 04
11	(JJ)	+DO 10	+DI 10
12	(HH)	+DO 05	+DI 05
13	(FF)	+DO 09	+DI 09
14	(EE)	+DO 06	+DI 06
15	(DD)	+DO 08	+DI 08
16	(CC)	+DO 07	+DI 07
17	(BB)	GROUND	GROUND
18	(AA)	GROUND	GROUND
19	(Z)	+CYCLE RQ B	GROUND

Pin	(DEC)	J1 Signal	J2 Signal
20	(Y)	GROUND	GROUND
21	(X)	+END CYCLE	+GO
22	(W)	GROUND	GROUND
23	(V)	+STATUS C	+FNCT 1
24	(U)	GROUND	GROUND
25	(T)		
26	(S)	GROUND	GROUND
27	(R)	+STATUS B	+FNCT 2
28	(P)	GROUND	GROUND
29	(N)	+INIT	
30	(M)	GROUND	GROUND
31	(L)	+STATUS A	+FNCT 3
32	(K)		+FNCT 3
33	(J)		
34	(H)	GROUND	GROUND
35	(F)	+READY	
36	(E)	GROUND	GROUND
37	(D)	+ACLO FNCT 2	+ATTN
38	(C)	GROUND	GROUND
39	(B)	+CYCLE RQ A	+BUSY
40	(A)	GROUND	GROUND

THEORY OF OPERATION

This section explains the operation of the Ridge DR11 Controller. First a description of the Digital Equipment DR11W Interface is presented, then the architecture of the controller is shown.

DR11W INTERFACE

DR11W is a 16-bit parallel, asynchronous interface between a VAX or PDP11 machine and a user device. The interface consists of two unidirectional data buses and 21 control signals, and it supports half duplex block transfers of 1 to 64K words.

At the beginning of a block transfer, the controller pulses "GO" and negates "READY". The user device initiates the exchange of each word by pulsing either "CYCLE REQUEST A" or "CYCLE REQUEST B" and the controller asserts "BUSY" while data is being exchanged with the memory in the host machine.

At the conclusion of a cycle, the controller produces an "END" pulse and the user device either supplies the next word on the "DATA IN" bus for a read or latches the contents of the DATA OUT bus for a write.

After the last word of the block has been transferred, or the the user aborts the operation by pushing "ATTN", the controller will assert "READY".

There are two options to the basic DR11 transfer.

- DMA cycles can be "primed;" the first word can be transferred when "GO" is issued without a "CYCLE REQUEST."

- The interface can operate in "Link Mode" so that the polarity of "BUSY" is complemented from the normal mode.

Not all the signals in the interface are explained here. Six signals, "FUNCTION 1, 2, 3", and "STATUS A, B, C" are defined by the user device and some signals control the DEC UNIBUS or are involved in byte transfers which are not supported by the RIDGE DR11 Controller.

Timing diagrams are presented below and more detailed information about the interface can be obtained from the DEC manual, "DR11W Direct Memory Interface Module."

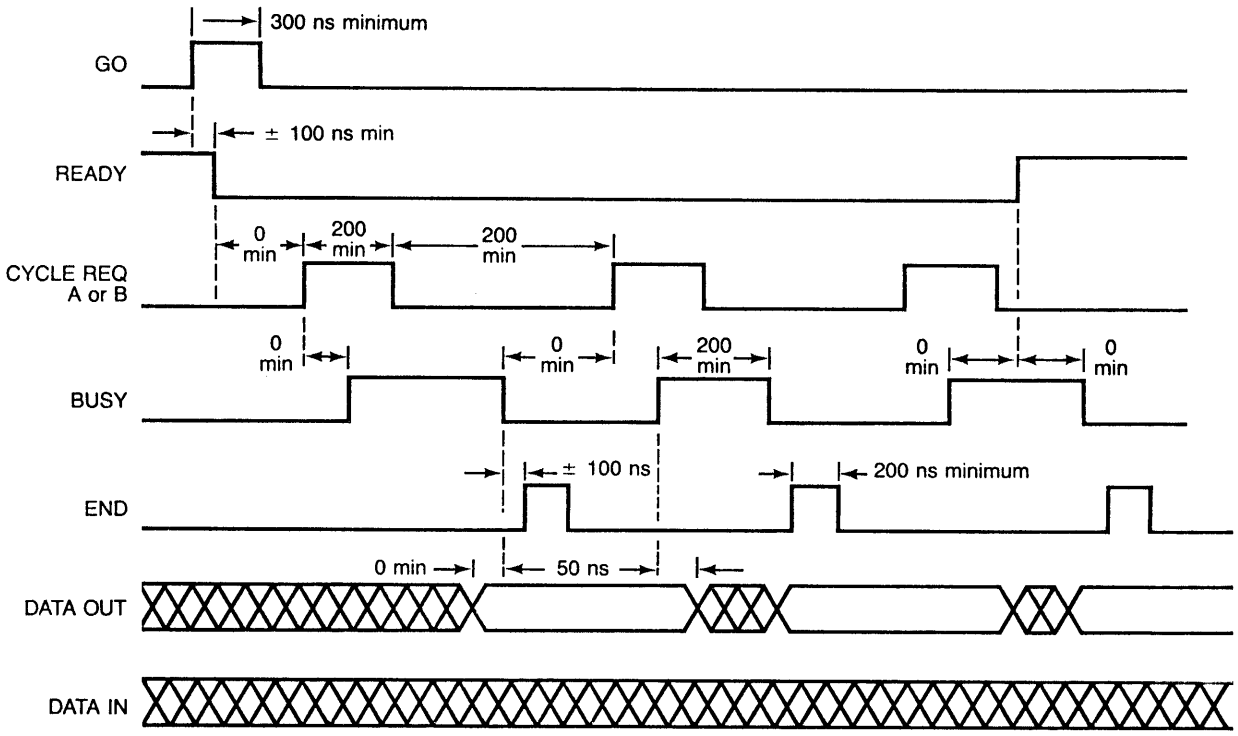


Figure 7-3. DMA Transfer to Peripheral Equipment

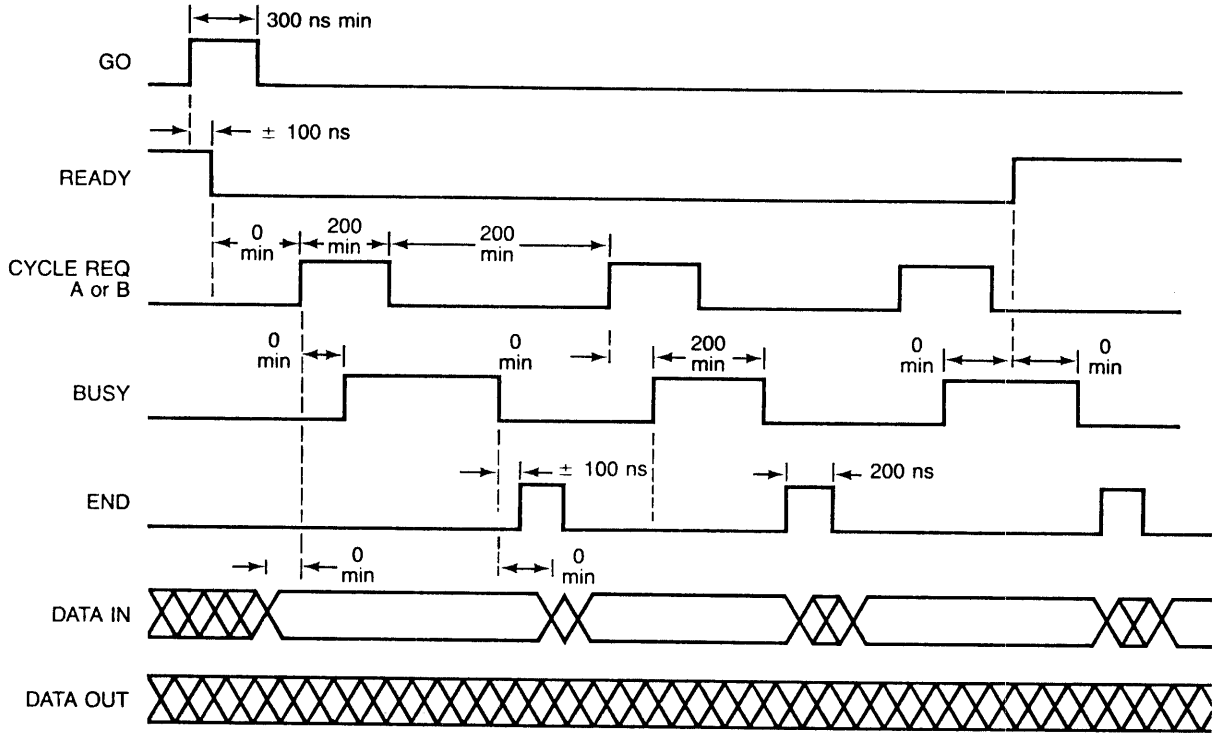


Figure 7-4. DMA Transfer from Peripheral Equipment

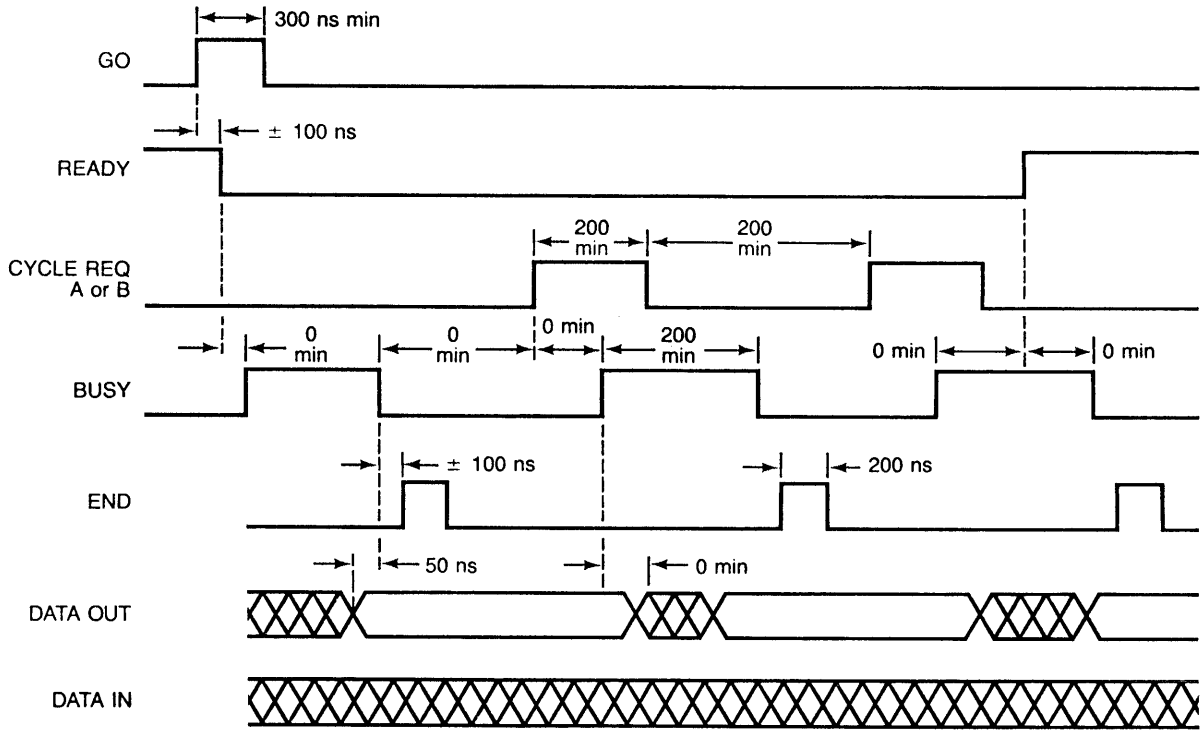


Figure 7-5. Primed DMA Transfer to Peripheral Equipment

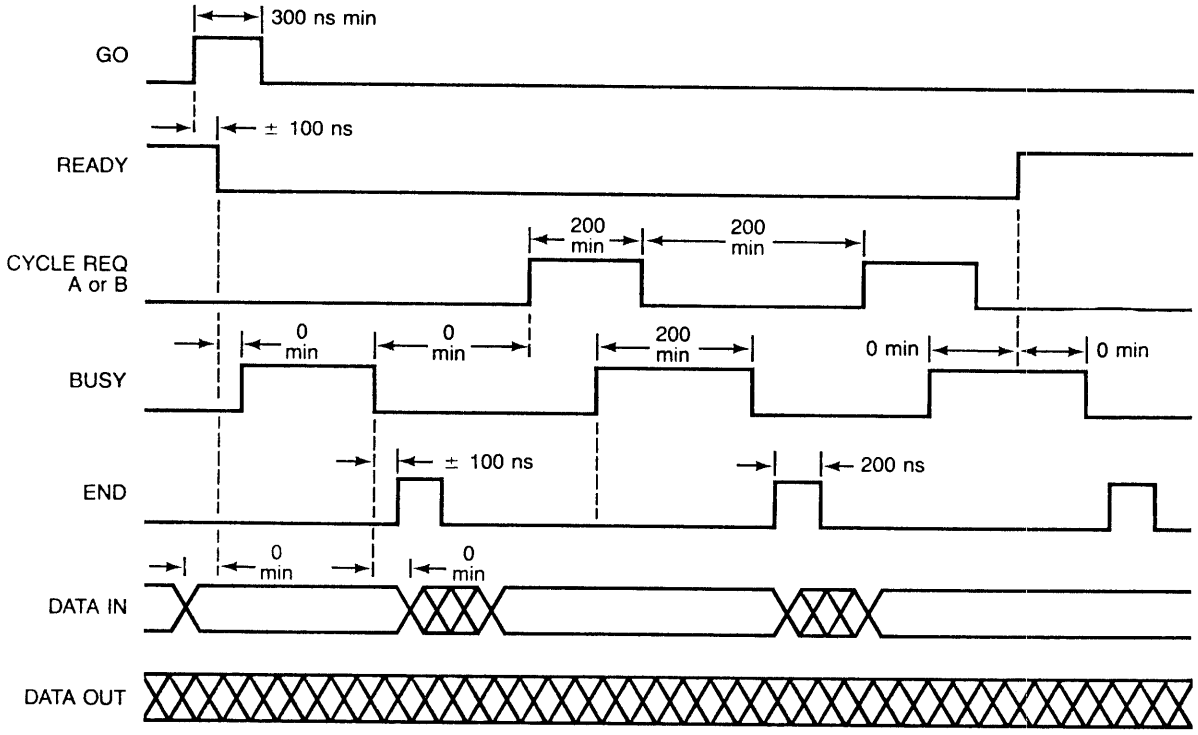


Figure 7-6. Primed DMA Transfer from Peripheral Equipment

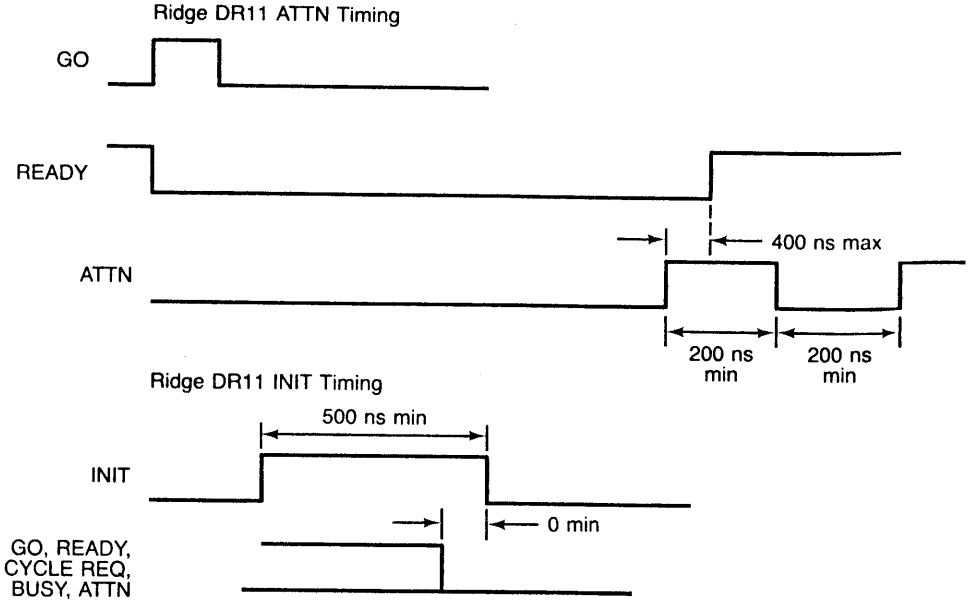


Figure 7-7. DR11 ATTN Timing

THE RIDGE CONTROLLER

The controller consists of programmed I/O logic, data transfer hardware, a serial input port, and interrupt logic. A block diagram of the controller and a description of each section is given below:

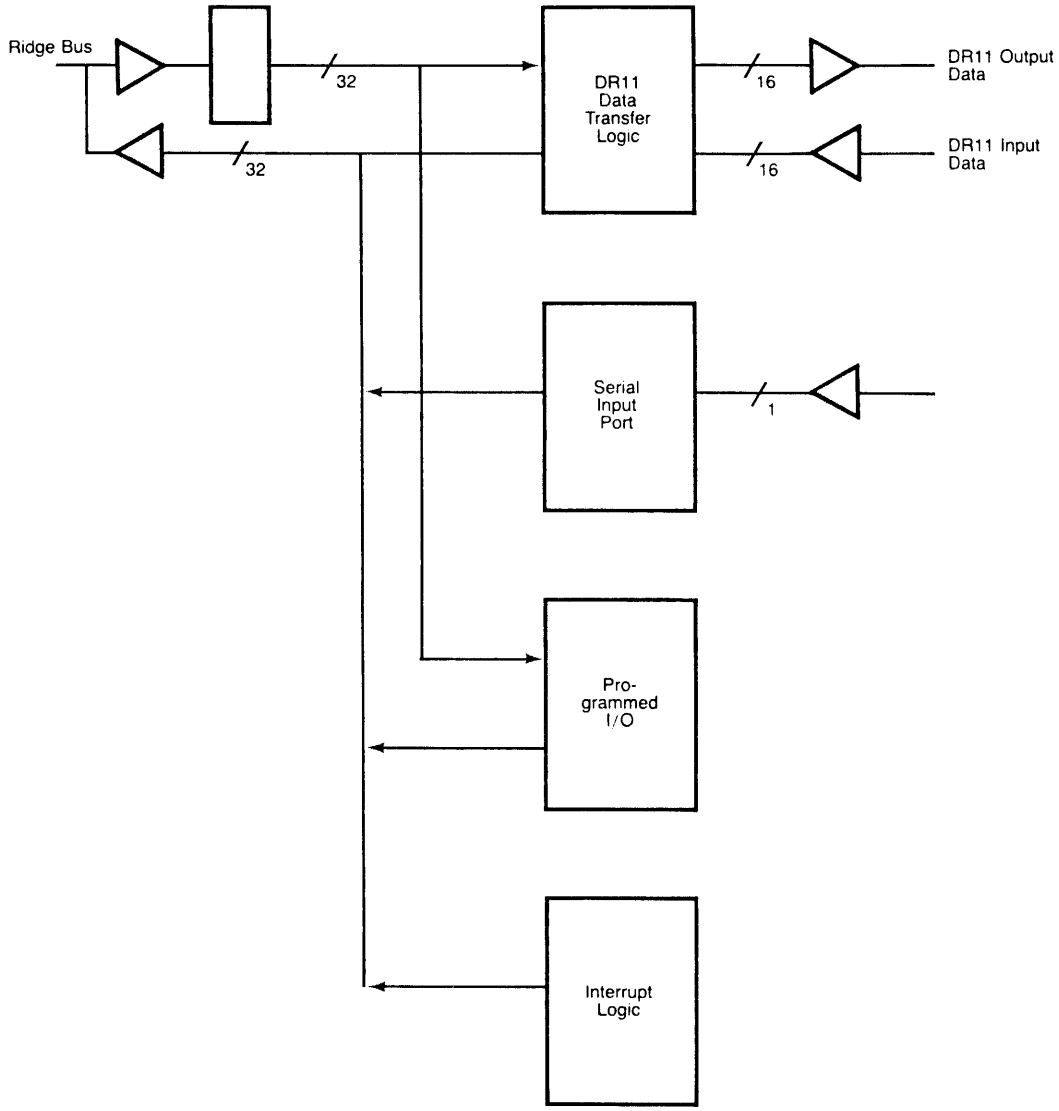


Figure 7-8. Ridge DR11 Block Diagram

PROGRAMMED I/O LOGIC

The controller contains registers that can be accessed by the RIDGE CPU with I/O Read and I/O Write instructions to control the operation of the interface.

Registers are accessed by placing the card address in bits 0-7 and the register number in the least significant bit positions of the I/O Address word of the input/output instruction.

Card addressing depends on jumper J401. If it is omitted, the card responds to the address selected by the card address DIP switch; however, if the jumper is installed, the controller will occupy two addresses specified by the most significant seven switches. An odd address will select the serial port and an even address will enable the DR11 registers.

J401	Bits 0 - 7 of I/O Address Word (Sx=value of address switch x)	Registers Selected
INSTALLED	S7 S6 S5 S4 S3 S2 S1 0	DR11
INSTALLED	S7 S6 S5 S4 S3 S2 S1 1	SERIAL INPUT PORT
OMITTED	S7 S6 S5 S4 S3 S2 S1 S0	DR11

Read access is always permitted; however, write access to the DR11 byte counter, address register, and the ODR is prohibited and IODNVM is returned when an attempt is made to modify these registers when a block transfer is in progress.

The PIO logic follows the standard timing; however, in a read, ACKMCIO is delayed one cycle to allow data to be enabled on the internal bus before it is placed on the RIDGE bus.

For a write, the data is clocked into a temporary register, when IODACK is asserted, before it is latched in the specified register. IODACK is delayed 125 nanoseconds and called "INPUT DATA AVAIL" before it enters the state machine. This reduces the number of schottky parts.

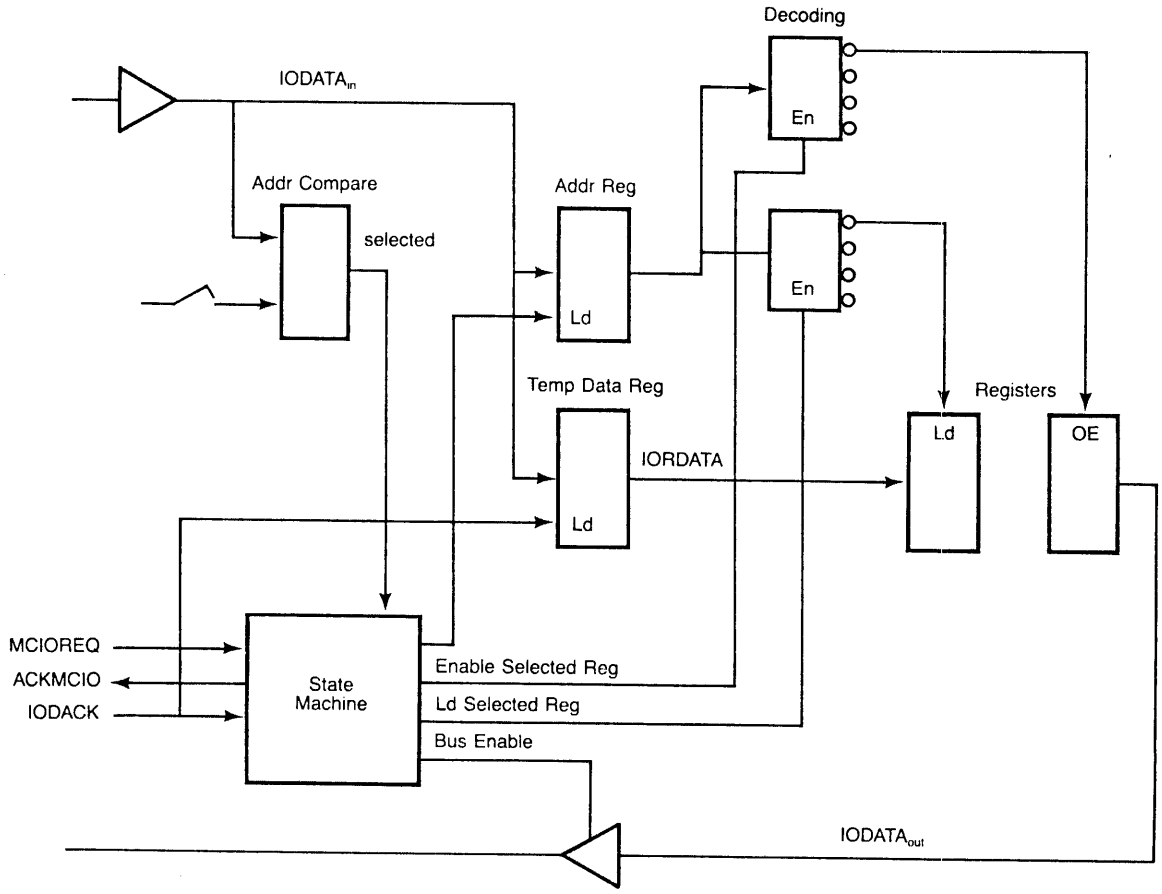


Figure 7-9. Programmed I/O Logic

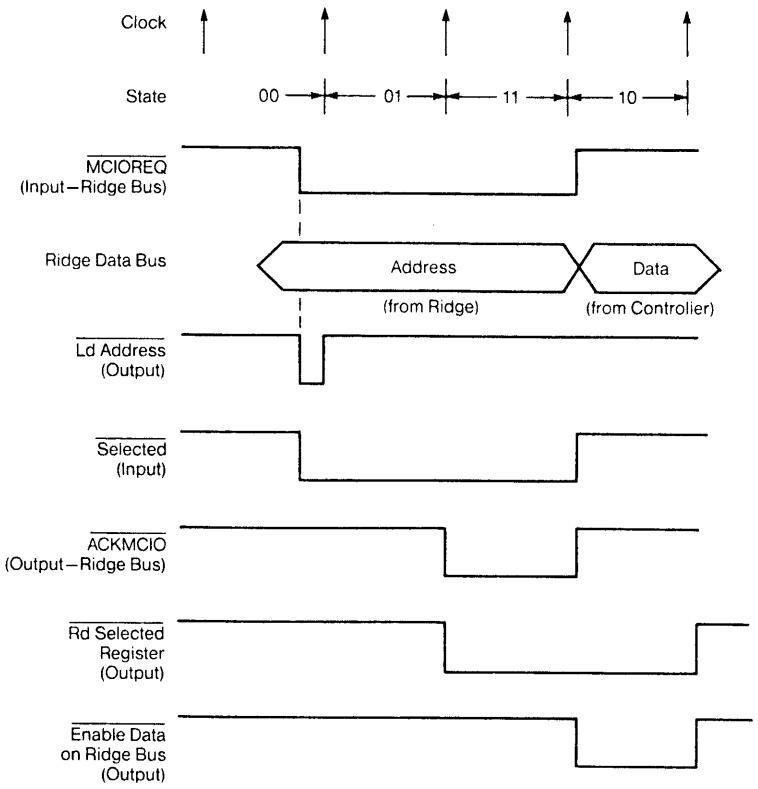


Figure 7-10. Programmed I/O - Read

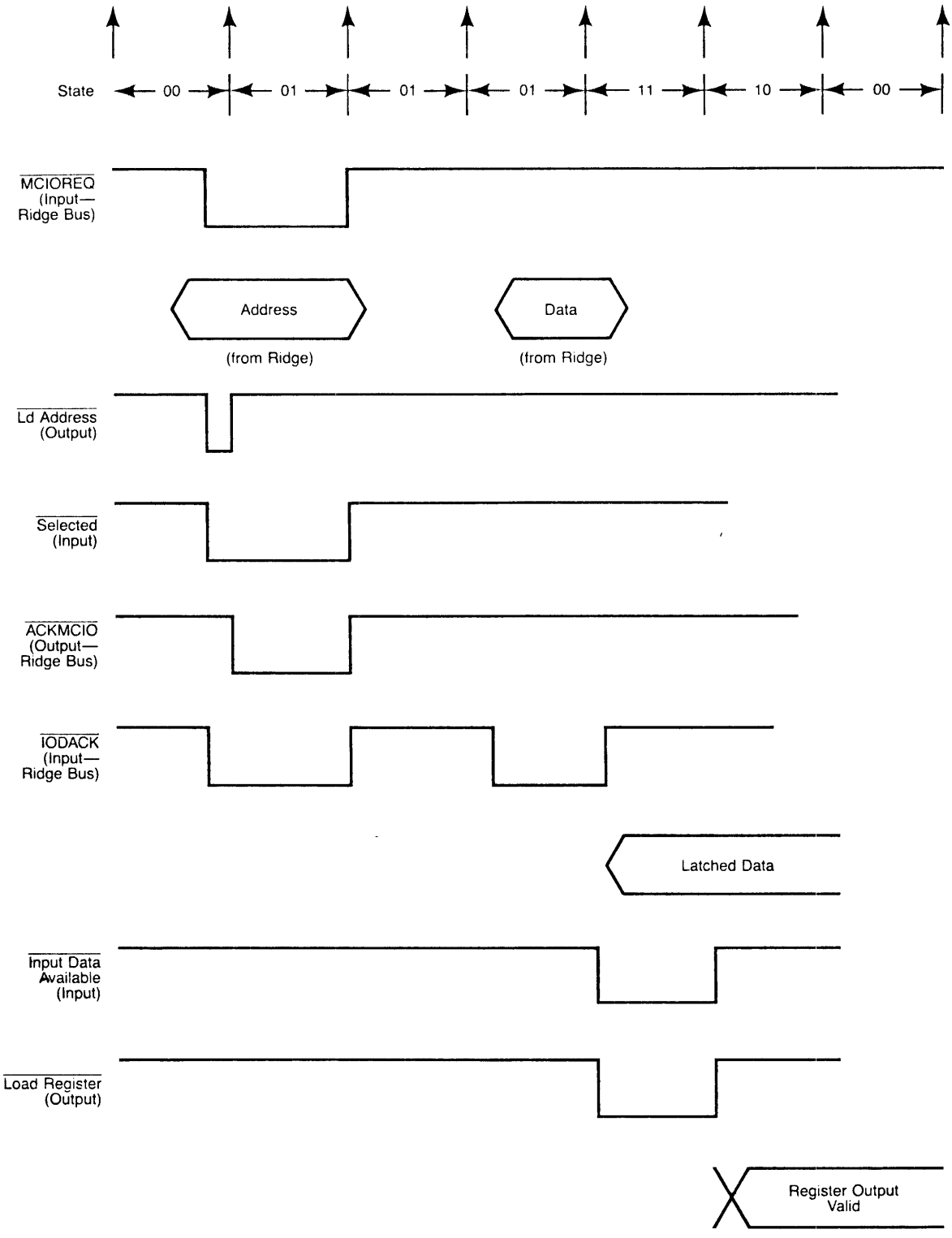


Figure 7-11. Programmed I/O - Write

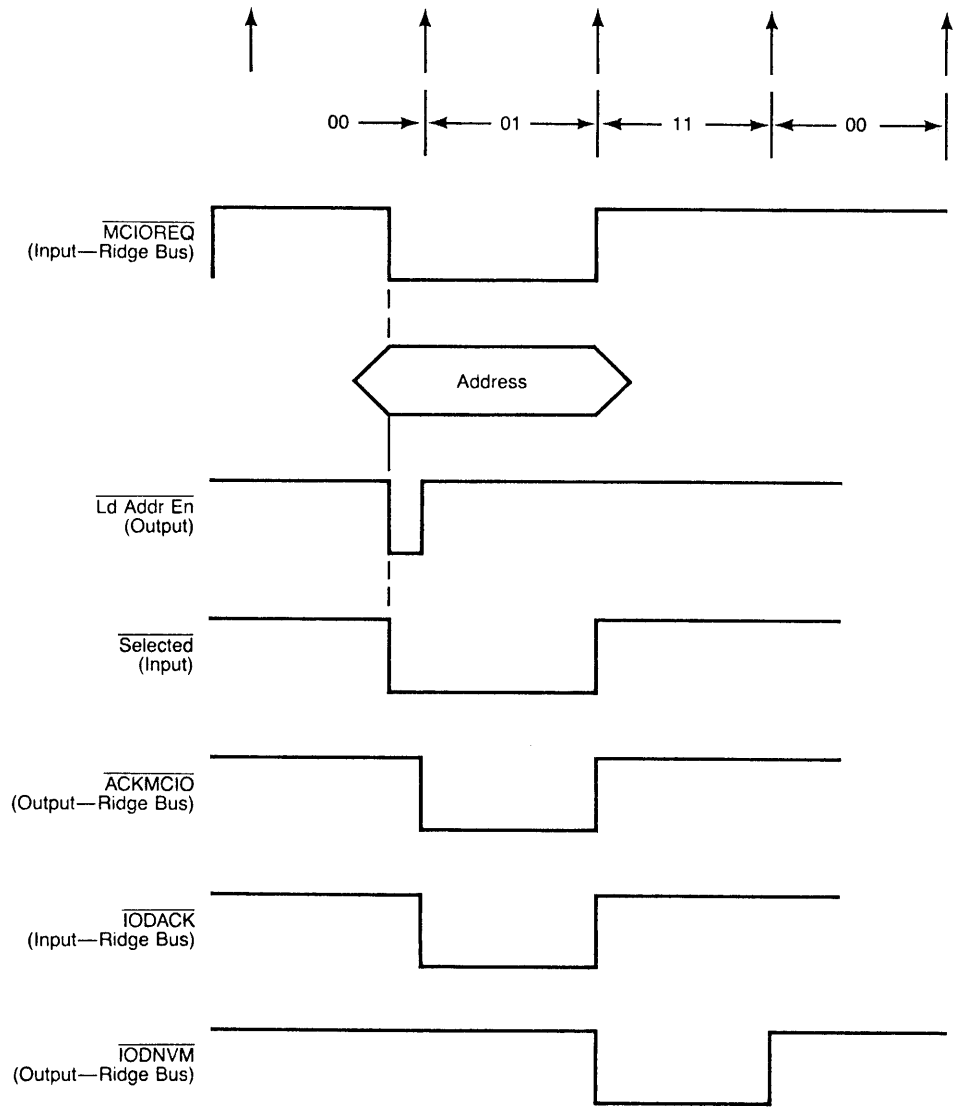


Figure 7-12. Programmed I/O - Write, But Not Permitted

DATA TRANSFER LOGIC

The data transfer logic has a separate double buffered data path for input and output plus three state machines to control the data flow.

For a DR11 write transfer, the Buffer Control Logic tells the RIDGE DMA state machine when to load the input buffer and it informs the DR11 Control logic when the output buffer is full.

For a DR11 read transfer, the buffer control logic informs the DR11 machine when the input buffer can be loaded and it tells the RIDGE DMA logic when there is data available.

The DR11 control logic is pictured below: The PLA state machine uses Ready to determine the beginning and end of a block transfer and the positive transition of either "CYCLE REQUEST" line to start a word transfer.

The Word Select Counter specifies which half of the RIDGE word is involved in the 16-bit DR11 transfer and it indicates to the state machine, at the end of a DR11 read transfer, if an additional transfer to the RIDGE is required to flush the buffer.

The "COMMAND in PROGRESS" signal is for software status and the signal, "XFER ENABLED" allows the Buffer Control Logic to transfer data.

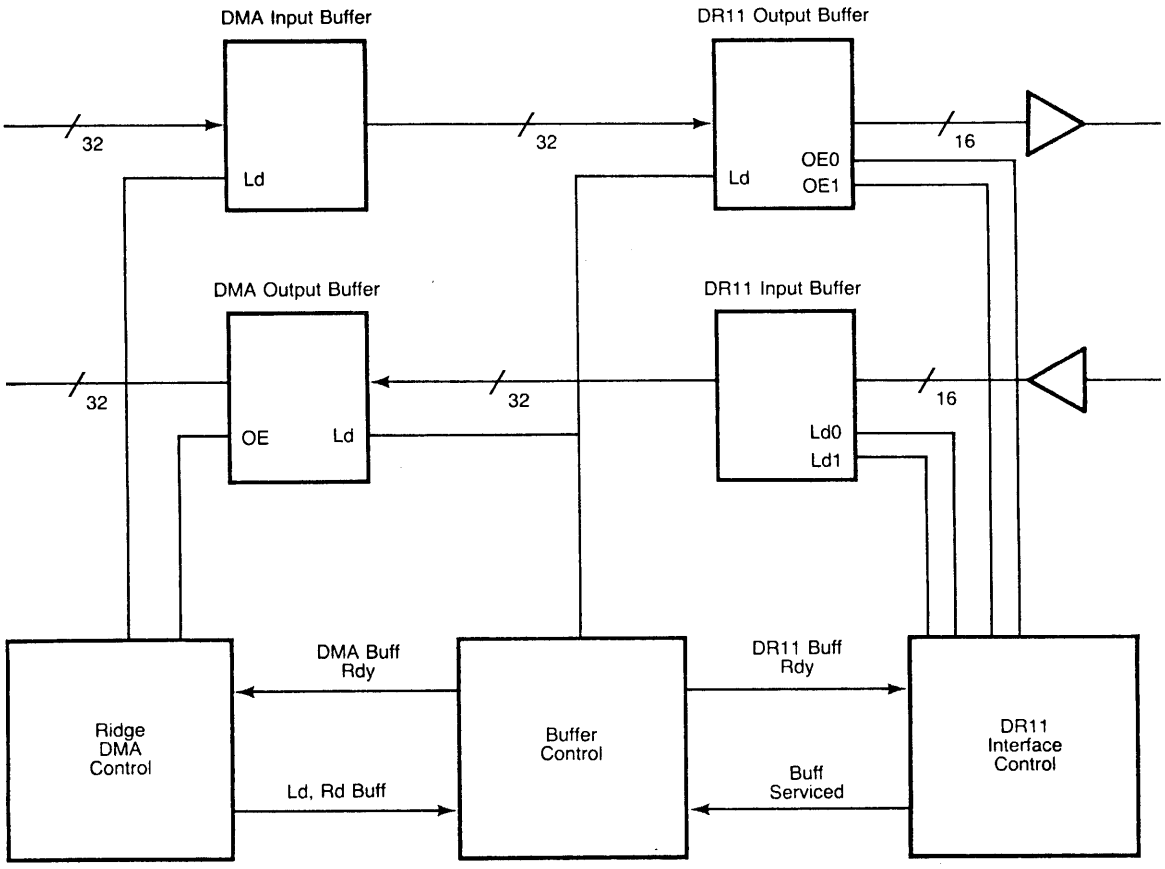


Figure 7-13. Data Transfer Logic

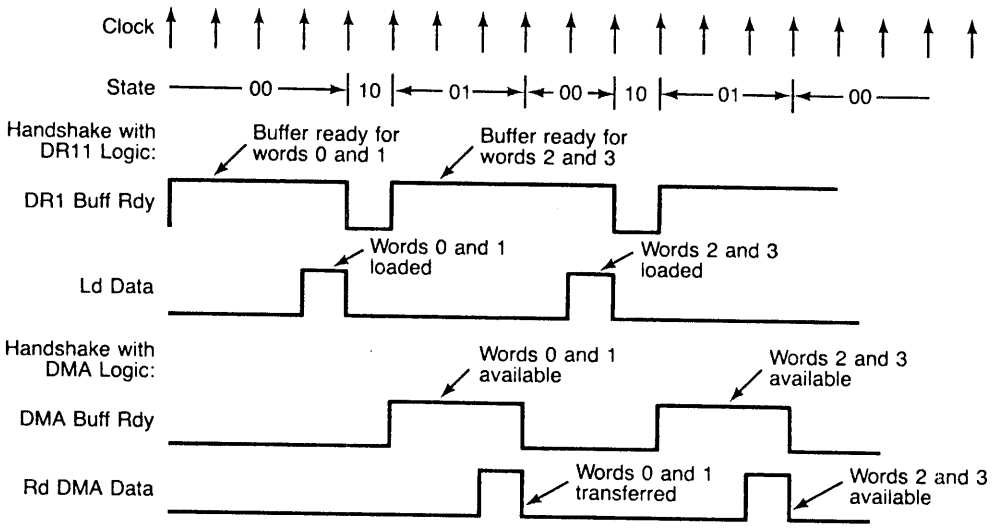


Figure 7-14. Buffer Control Logic - DR11 Read

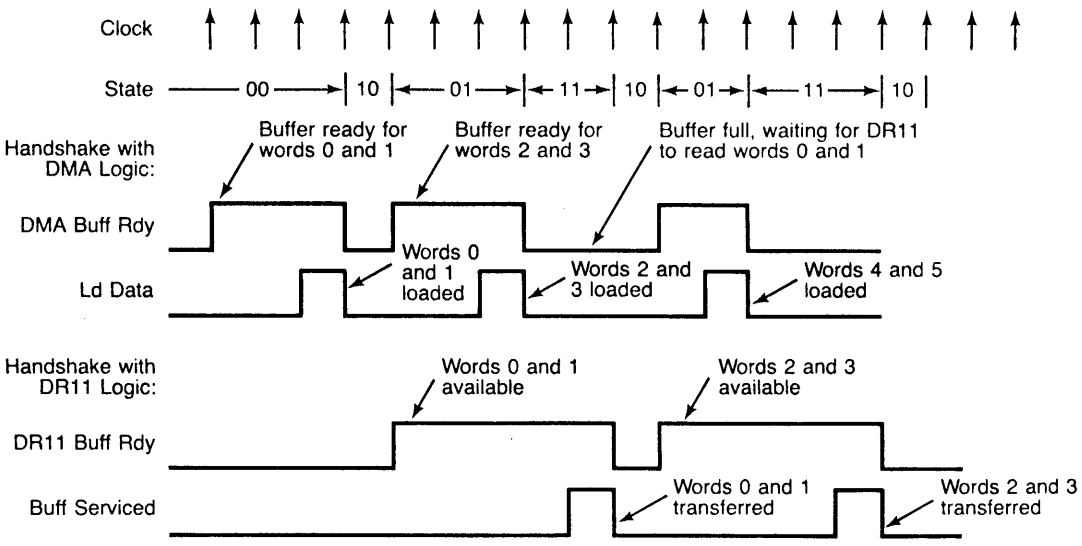


Figure 7-15. Buffer Control Logic - DR11 Write

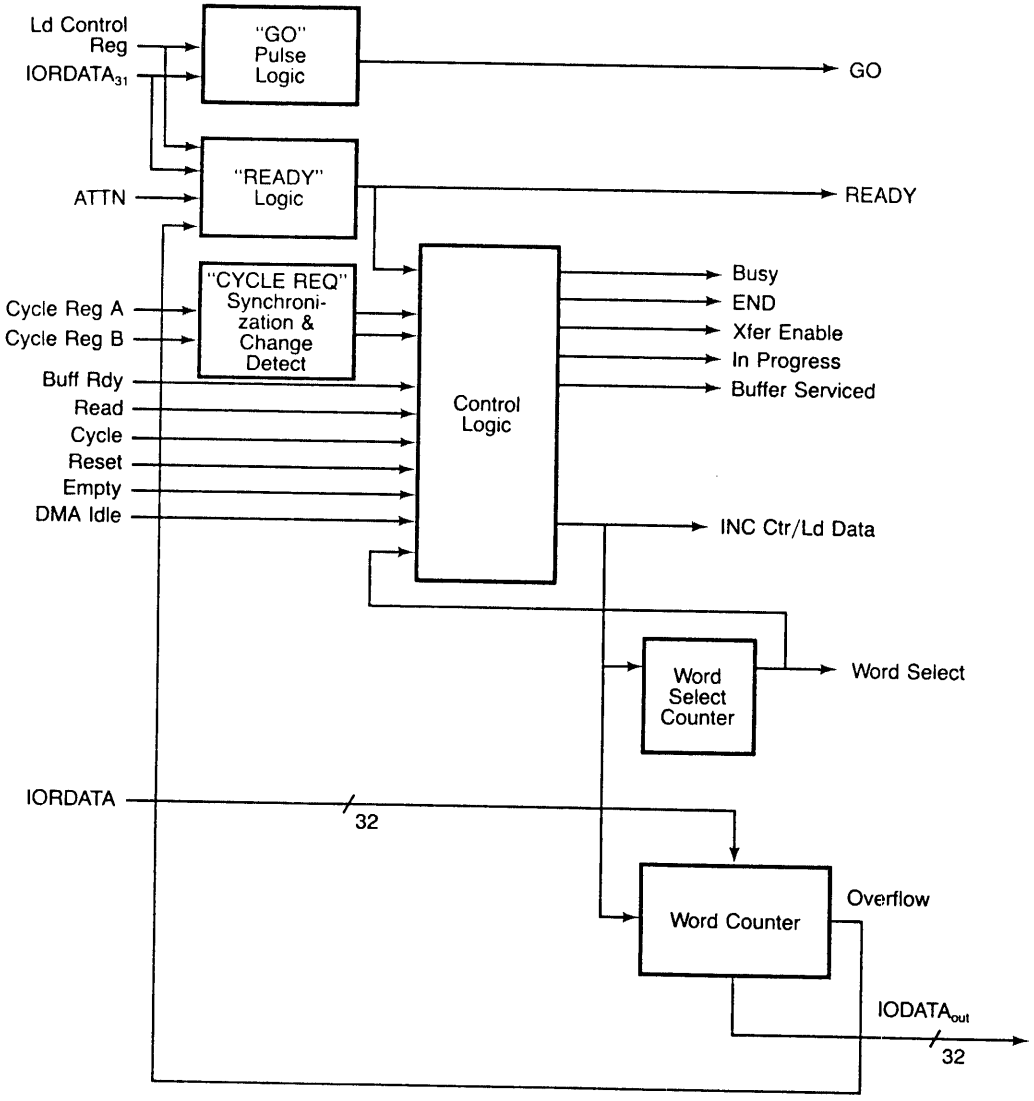


Figure 7-16. DR11 Control Logic

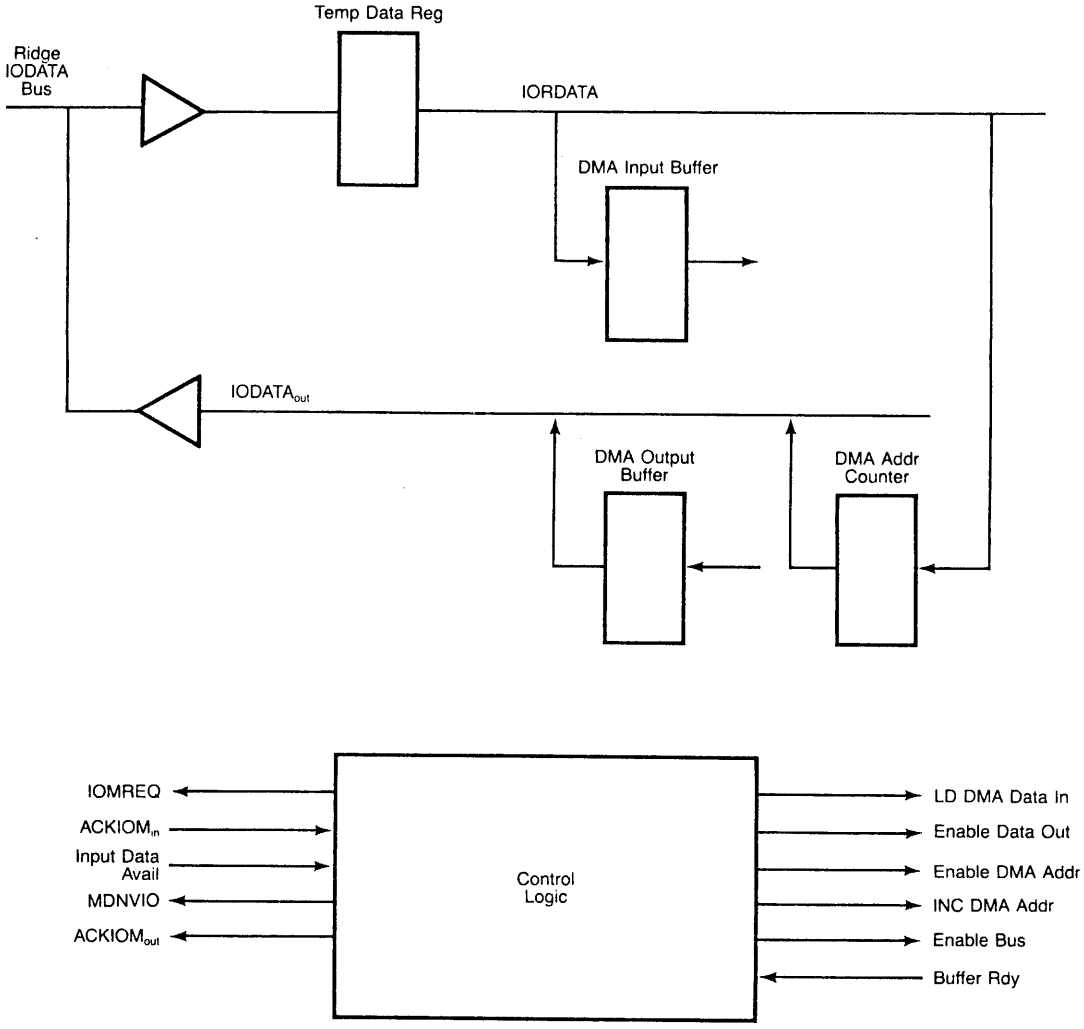


Figure 7-17. DMA Logic

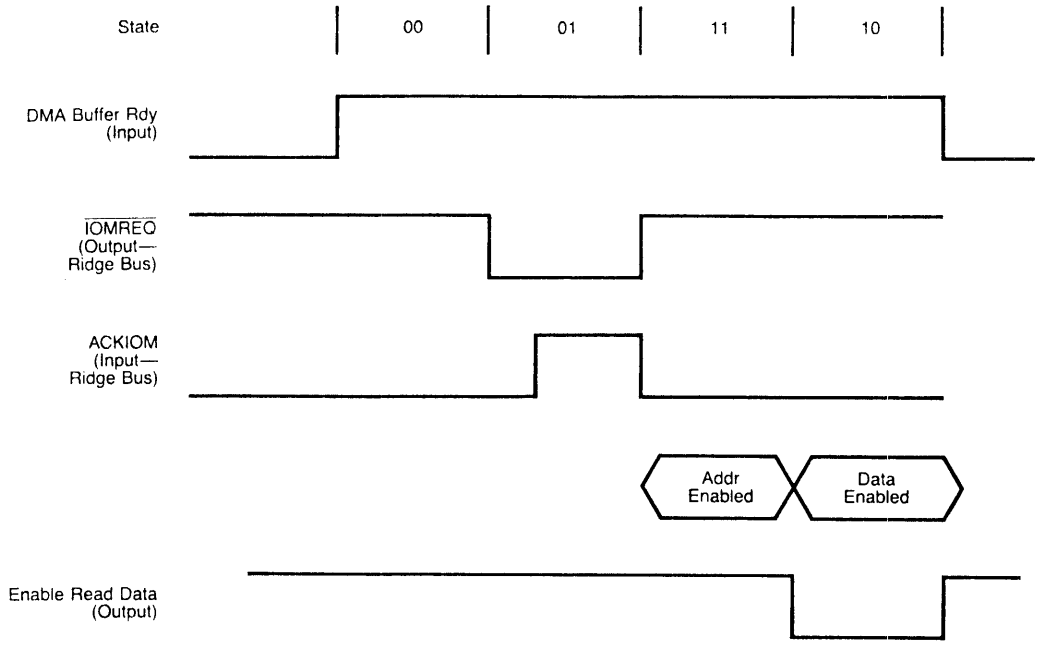


Figure 7-18. Ridge DMA Write Timing

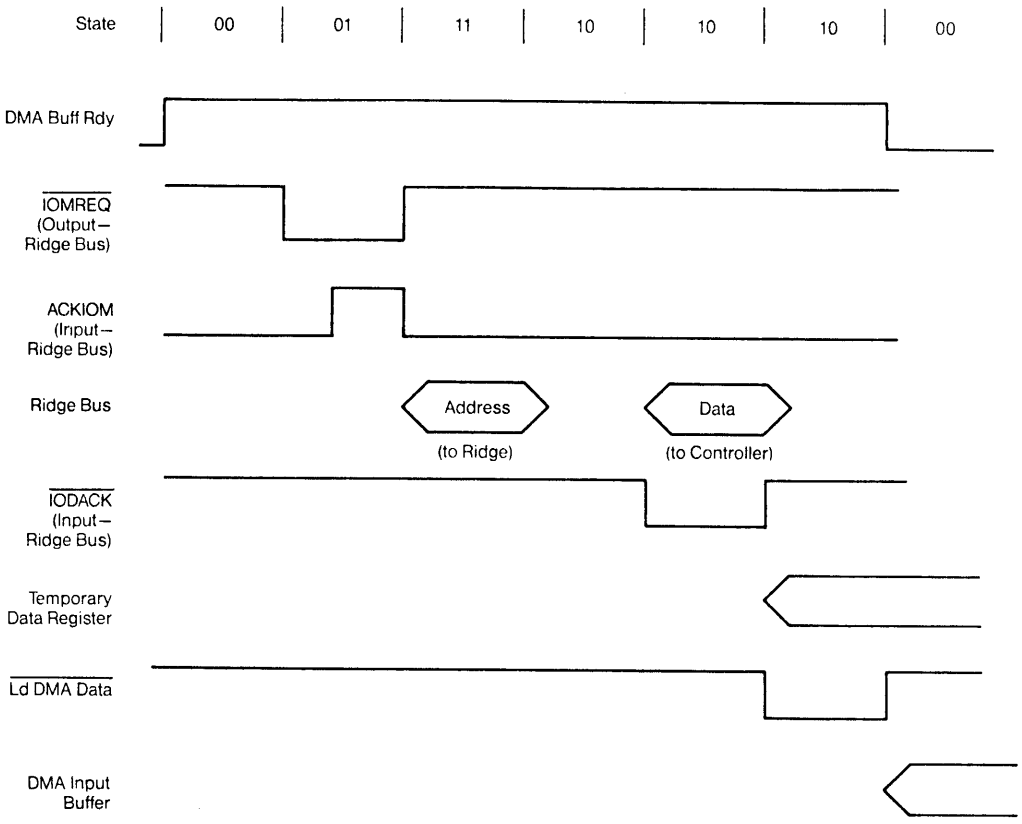


Figure 7-19. Ridge DMA Read Timing

SERIAL INPUT PORT

The serial input port is a UART with a differential receiver and two levels of buffering. A flip flop delays the UART signal, RDA (receive data available), one bit interval to create set-up time for the parity overrun and framing error status bits.

The Buffer Control Logic is two state machines. The UART Buffer Control Logic loads a character from the UART to the to the first buffer and the UART interrupt logic moves data from the first to the second buffer. If a character is received when the buffers are full, the character will not be read from the UART. Overrun status is handled by the UART.

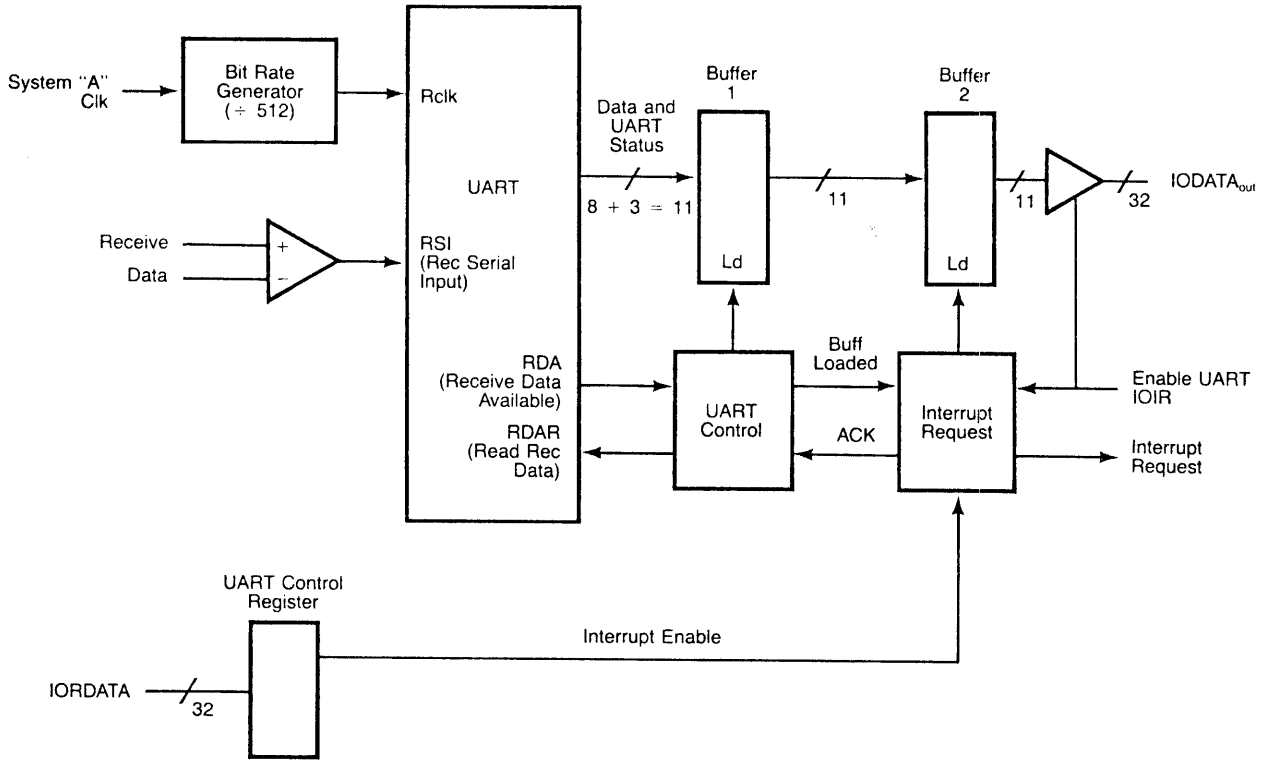


Figure 7-20. Serial Input Port

INTERRUPT LOGIC

Interrupts can be generated by both the DR11 hardware and the serial input port. The completion of a block transfer, or the user device asserting 'ATTN' will produce a DR11 interrupt, and the serial port will create an interrupt when a character has been received.

There is a handshake between the requesting logic and the interrupt hardware. Both the DR11 and the serial port assert a request to the interrupt logic until their IOIR data is enabled onto the RIDGE bus.

If both the DR11 and the serial port are making a request, the serial port will be serviced first.

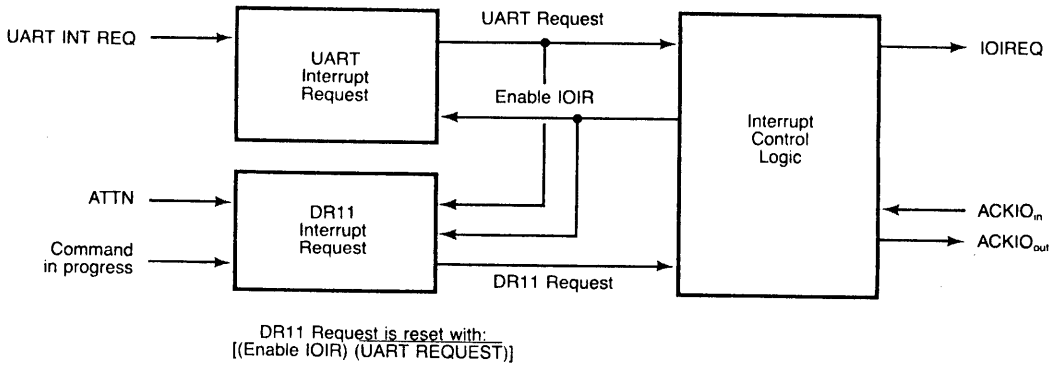


Figure 7-21. Interrupt Logic

DR11 PROGRAMMING

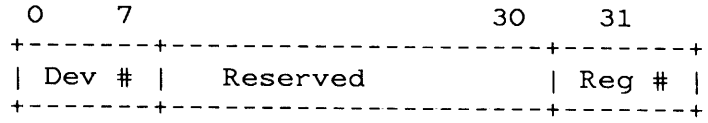
The Ridge DR11 contains the same registers as the DEC DR11W; however, the bits are organized differently in the Control/Status R gister and the signals that pertain to the Unibus have been deleted, and a serial asynchronous input port is included.

The board occupies one address position except when the serial port is enabled, with a hardware jumper, the card will respond to two addresses. An address ending in O selects the DR11 registers, and an address ending in 1 selects the UART circuitry.

The operation of the interface is defined by the peripheral device.

DR11 REGISTER DEFINITIONS

I/O Address Word

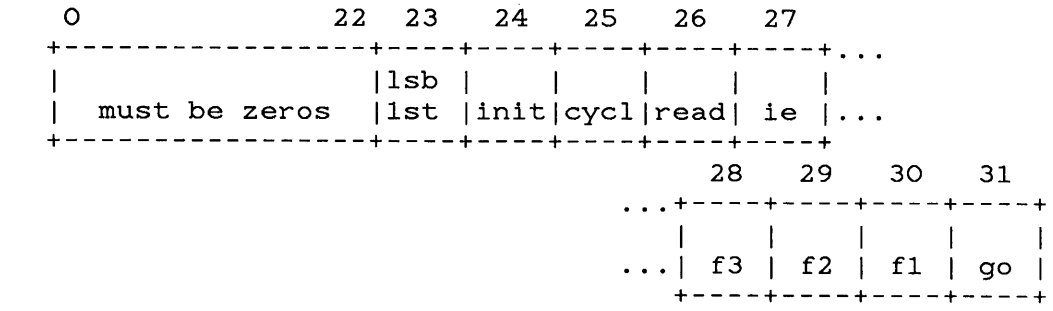


Register #	Read	Write
0	Status	Control
1	IDR	ODR
2	DMA Addr	DMA Address
3	Byte Count	Byte Count

IODNV (IO Data Not Valid) status will be returned if the ODR, DMA Address, or Byte Count registers are accessed with an I/O write instruction while the "Command in progress" bit is set.

Reserved bits must be loaded with zeros.

Control Register



LSB First (23) This bit controls the placement of the 16-bit DR11 word in the 32-bit Ridge word. When set, the first DR11 word will be stored or read from bits 16-31 of the Ridge word, and the second word will involve bits 0-15. When reset, the first DR11 word goes to and from Ridge bits 0-15, and the second to bits 16-31.

INIT (24) Initialization Command. Resets the peripheral and Ridge DR11 circuitry.

CYCLE (25) Allows the first word of a DMA transfer to be exchanged without a Request from the peripheral.

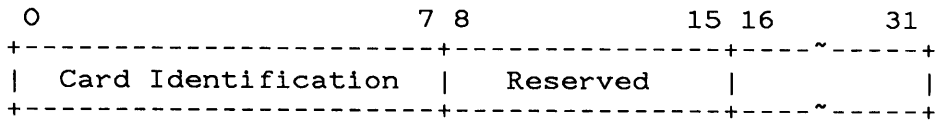
READ (26) Specifies the Direction of the data transfer from the DR11 to the user device.

IE (27) Interrupt enable.

F3,F2,F1 (28, 29, 30) User defined interface Function lines.

GO (31) Generates the 300 nsec interface GO pulse that starts a transfer.

Status



Bits

23-31 are from the control register.

20 STAT C - User defined status.

21 STAT B,- User defined status.

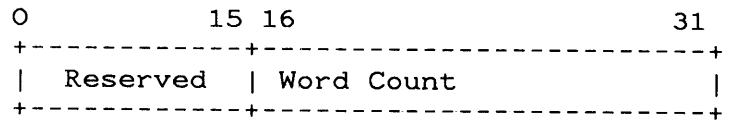
22 STAT A,- User defined status.

19 ATTN - Interrupt request from user device.

18 DE - Ridge DMA Error.

17 CIP - Command in progress. Set with GO and reset on completion of DMA or ATTN asserted. This is similar to "READY" on the DEC DR11W but the polarity has been inverted.

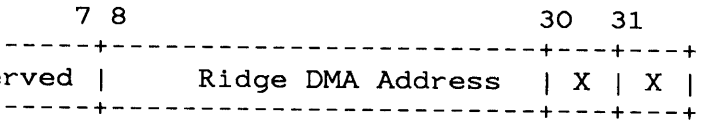
Word Count



The number of 16-bit words to be transferred must be loaded in 2's complement.

FFFF will produce a 16-bit transfer. 0000 will produce 64k 16-bit transfers.

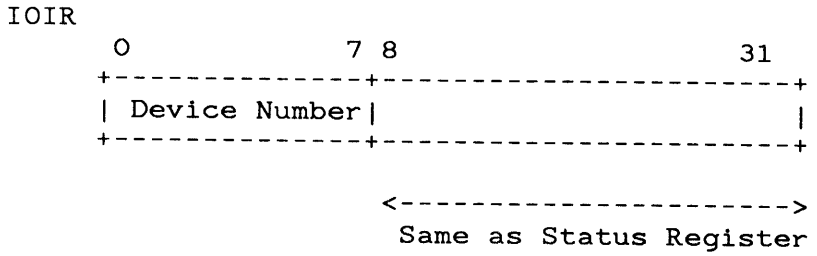
DMA Address



The DMA Address Counter addresses 32-bit Ridge words; therefore, the two least significant bits of the address are not used by the controller. On a read, bit 30 will be undefined and bit 31 is the DMA Read line.

DR11 Interrupts

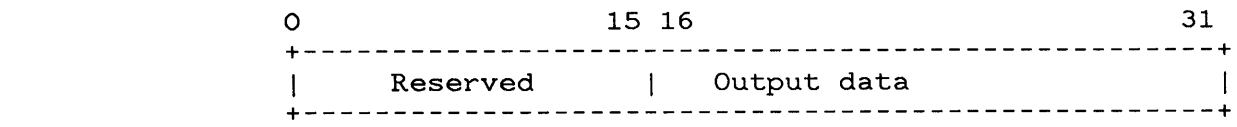
Interrupts are generated when either the "ATTN" signal, from the user device, is asserted or the "Command in Progress" bit goes false.



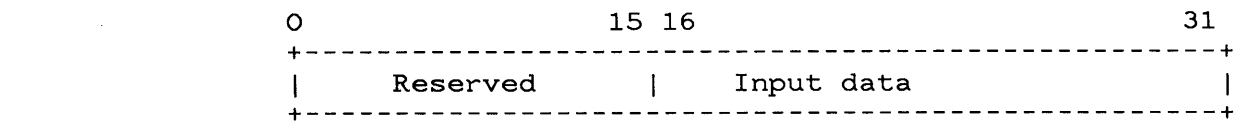
If the interrupts are disabled after the controller has made an interrupt request to the Ridge, the Ridge interrupt handshake will continue to completion.

Programmed I/O Data Registers

Output Data Register (ODR)

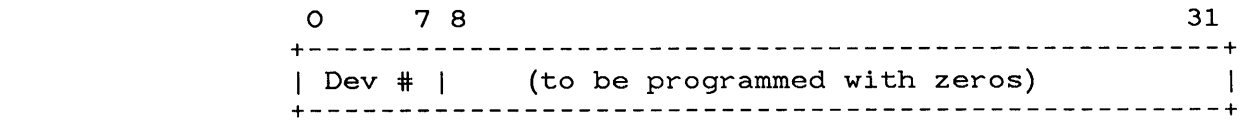


Input Data Register (IDR)



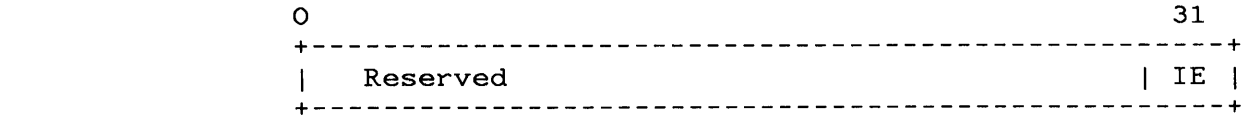
SERIAL PORT - REGISTER DEFINITIONS

I/O Address Word



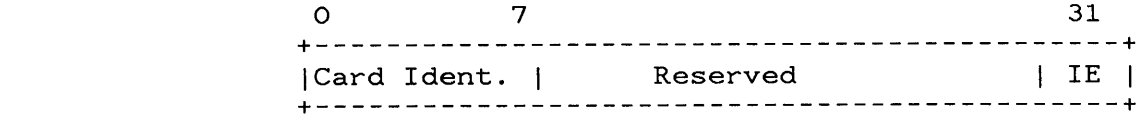
Write = Control Register.
Read = Status Register.

Control Register



IE = Interrupt enable

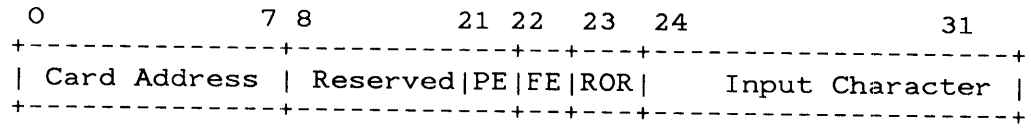
Status Register



IE = Interrupt enable status from
control register.

IOIR

When the interrupts are enabled, the serial input port generates an interrupt each time a character is received. The character is available in the IOIR word.



ROR Receiver over run. If a second character has been received before the first character has been read, then the first character will be lost and the ROR bit will be set.

FE Frame error. The "stop bit" was defective on the received character.

PE Parity error. The parity bit did not agree with the calculated parity.

DR11 CONTROLLER DIAGNOSTICS

The DR11 Controller Diagnostic tests the operation of a Ridge DR11 Interface Controller. The program consists of six parts: a register, a loopback, a keyboard interface, a Metheus, and an Ungermann-Bass NIU test. It requires use of the Ridge Operating System (ROS).

The name of the program is "dr11diag" and its operation is controlled by a series of questions. The first input required is the device number which is the number set on the address switches on the controller board.

The next several questions are related to specific tests and the last ones are concerned with halting execution when an error is detected and the number of times the selected tests are to be repeated.

From the example below, the address of the controller is FF, it is to be connected to a Metheus color display, and only the registers are to be tested; the program will run 3 times, but it will stop if an error is detected.

All answers are to be followed by pressing **[RETURN]**, and all user inputs are in bold print in the figures. Default answers are given in parenthesis; and if the default is desired, the question can be answered with just **[RETURN]**.

RUNNING THE DIAGNOSTIC

This diagnostic may be invoked by **systest**, **sus**, or individually. In these examples, the diagnostic is invoked individually.

Before running the **dr11diag** with **systest** or individually, log in as **root** and remove all lines from the **/ros/conf** file *except*:
`:1:/drivers/fdlp`
After using the diagnostic, re-insert the line(s).

```
$ dr11diag                                  [RETURN]

DR11 Diagnostic     Version 1.xx

Enter DR11 device number (hex) :            ff [RETURN]

Device Type = 30 (hex) : Metheus.

Do you want to do register bit tests        (Y)?[RETURN]
Do you want to do loopback tests            (Y)?n [RETURN]
Do you want to do the keyboard test         (N)?n [RETURN]
Do you want to do the Metheus test          (N)?n [RETURN]
Do you want to stop errors                  (Y)?y [RETURN]
Loop count:                                 3 [RETURN]
```

DR11 REGISTER TEST

In the register test, a one and a zero is moved through all bit positions of all registers except the ODR/IDR and if the contents of a register is not equal to the expected value, an error message is written.

When the registers are being tested, the device connected to the DR11 controller should either be disconnected or off-line.

A sample output from one pass of the test appears below:

```
DR11 Diagnostic - Walking ones thru DMA register
DR11 Diagnostic - Walking zeros thru DMA register
DR11 Diagnostic - Walking ones thru Word Count register
DR11 Diagnostic - Walking zeros thru Word Count register
DR11 Diagnostic - Walking ones thru Control, Status registers
DR11 Diagnostic - Walking zeros thru Control, Status registers
```

DR11 LOOPBACK TEST

In the loopback test, a special connector must be installed in place of the cables. The signals are connected has shown below:

Source	(Connector/ Pin)	Receiver	(Connector/ Pin)
DO 0	(P3/2)	DI 0	(P4/2)
DO 1	(P3/4)	DI 1	(P4/4)
DO 2	(P3/6)	DI 2	(P4/6)
DO 3	(P3/8)	DI 3	(P4/8)
DO 4	(P3/10)	DI 4	(P4/10)
DO 5	(P3/12)	DI 5	(P4/12)
DO 6	(P3/14)	DI 6	(P4/14)
DO 7	(P3/16)	DI 7	(P4/16)
DO 8	(P3/15)	DI 8	(P4/15)
DO 9	(P3/13)	DI 9	(P4/13)
DO 10	(P3/11)	DI 10	(P4/11)
DO 11	(P3/9)	DI 11	(P4/9)
DO 12	(P3/7)	DI 12	(P4/7)
DO 13	(P3/5)	DI 13	(P4/5)
DO 14	(P3/3)	DI 14	(P4/3)
DO 15	(P3/1)	DI 15	(P4/1)
F2	(P4/27)	REQ B	(P3/19)
INIT	(P3/29)	SC	(P3/23)
BUSY	(P4/39)	SB	(P3/23)
READY	(P3/35)	SA	(P3/31)
F1	(P4/23)	REQ A	(P3/39)
F3	(P4/32)	ATTN	(P4/37)

The test exercises most of the DR11 hardware. First, data is walked through the ODR and read back in the IDR. This checks the IDR/ODR registers and the data drivers and receivers.

The interrupt logic is tested by pulsing the F3 line which is connected to ATTN through the loopback.

In the DMA Input tests, data moves from the ODR through the input buffers to Ridge memory and for the output tests, data moves from memory through the output buffers to the IDR. Cycle Request pulses are generated with F1 and F2.

The loopback test asks for the number of bytes to be transferred and the status of the J901 Link Mode jumper. A sample of the output from the loopback program appears below:

```
$ dr11diag                                          [RETURN]

DR11 Diagnostic     Version 1.xx

Enter DR11 device number (hex) :                    ff [RETURN]

Device Type = 30 (hex) : Metheus.

Do you want to do register bit tests                (Y) ?n [RETURN]
Do you want to do loopback tests                    (Y) ?y [RETURN]
Do you want to do the keyboard test                 (N) ?n [RETURN]
Do you want to do the Metheus test                  (N) ?n [RETURN]
Do you want to stop errors                          (Y) ?y [RETURN]
Is the Link Mode jumper, J901, installed            (Y) ?y [RETURN]
DMA Transfer Length:                                123 [RETURN]
Loop count:                                         2 [RETURN]
Attach loopback connector, then press <return>.     [RETURN]

DR11 Diagnostic - Walking ones thru IDR, ODR registers
DR11 Diagnostic - Walking zeros thru IDR, ODR registers
DR11 Diagnostic - Doing write protect test.
DR11 Diagnostic - Doing ATTN signal Test.
DR11 Diagnostic - Doing DMA output test.
DR11 Diagnostic - Doing DMA Input test.
DR11 Diagnostic - Doing DMA Input test.
DR11 Diagnostic - Doing DMA Input test.
DR11 Diagnostic - Doing DMA Input test.
DR11 Diagnostic - Doing LSBFirst test.
DR11 Diagnostic - Doing "primed" transfer test.  (Cycle bit set.)

DR11 Diagnostic - Walking ones thru IDR, ODR registers
DR11 Diagnostic - Walking zeros thru IDR, ODR registers
DR11 Diagnostic - Doing write protect test.
DR11 Diagnostic - Doing ATTN signal Test.
DR11 Diagnostic - Doing DMA output test.
DR11 Diagnostic - Doing DMA Input test.
```

```
DR11 Diagnostic - Doing DMA Input test.
DR11 Diagnostic - Doing DMA Input test.
DR11 Diagnostic - Doing DMA Input test.
DR11 Diagnostic - Doing LSBFirst test.
DR11 Diagnostic - Doing "primed" transfer test. (Cycle bit set.)

$
```

METHEUS TEST

In the Metheus test a block of 1024 bytes is written, using the "Write Rectangle" command, to the Metheus Omega 400 and with "Read Rectangle" the data is returned to the Ridge through the DR11. If there are any differences between the data sent and the data received, an error message is printed.

With each pass of the Metheus test, the write block/read block routine is repeated 64 times with different data.

An example of the Metheus test appears below:

```
$ dr11diag                                          [RETURN]

DR11 Diagnostic     Version 1.xx
Enter DR11 device number (hex) :                    ff [RETURN]
Device Type = 30 (hex) : Metheus.

Do you want to do register bit tests                (Y) ?n [RETURN]
Do you want to do loopback tests                    (Y) ?n [RETURN]
Do you want to do the keyboard tests                (N) ?n [RETURN]
Do you want to do the Metheus tests                 (N) ?y [RETURN]
Do you want to stop errors                          (Y) ?y [RETURN]
Is the Link Mode jumper, J901, installed            (Y) ?y [RETURN]
Loop count:                                         1 [RETURN]

DR11 Diagnostic - Metheus Test
DR11 Diagnostic - Metheus is running power-up diagnostics
waiting for Metheus to become ready
waiting for the transfer to conclude
waiting for Metheus to become ready
waiting for Metheus to have data ready
starting read transfer
waiting for read transfer to complete
checking received data
end of test
waiting for Metheus to become ready
waiting for the transfer to conclude
waiting for Metheus to become ready
waiting for Metheus to have data ready
starting read transfer
waiting for read transfer to complete
checking received data
```

end of test

.
.
.

KEYBOARD TEST

The keyboard test displays the received characters and any framing, or overrun errors. If characters are not input within several seconds a timeout message will be printed.

For the test to function, the the keyboard logic and jumper, J401, must be installed and the device address supplied to the diagnostic must be the address of the DR11 hardware, NOT the serial port.

$ dr11diag **[RETURN]**

DR11 Diagnostic Version 1.xx

Enter DR11 device number (hex) : **fe [RETURN]**

Device Type = 30 (hex) : Metheus.

Do you want to do register bit tests (Y) ?**n** [**RETURN**]
Do you want to do loopback tests (Y) ?**n** [**RETURN**]
Do you want to do the keyboard test (N) ?**y** [**RETURN**]
Do you want to do the Metheus test (N) ?**n** [**RETURN**]
Do you want to stop errors (Y) ?**n** [**RETURN**]
Loop count: **3** [**RETURN**]

DR11 Diagnostic - Doing Keyboard Test
$$ Character = 113 (q). Framing Error = 0 Overrun =0
$$ Character = 119 (w). Framing Error = 0 Overrun =0
$$ Character = 101 (e). Framing Error = 0 Overrun =0
$$ Character = 114 (r). Framing Error = 0 Overrun =0
$$ Character = 116 (t). Framing Error = 0 Overrun =0
$$ Character = 121 (y). Framing Error = 0 Overrun =0

DR11 Diagnostic - Doing Keyboard Test
$$ Character = 49 (1). Framing Error = 0 Overrun =0
$$ Character = 50 (2). Framing Error = 0 Overrun =0
$$ Character = 51 (3). Framing Error = 0 Overrun =0
$$ Character = 52 (4). Framing Error = 0 Overrun =0
$$ Character = 53 (5). Framing Error = 0 Overrun =0
$$ Character = 54 (6). Framing Error = 0 Overrun =0

DR11 Diagnostic - Doing Keyboard Test
$$ Character = 33 (!). Framing Error = 0 Overrun =0
$$ Character = 64 (@). Framing Error = 0 Overrun =0
$$ Character = 35 (#). Framing Error = 0 Overrun =0
$$ Character = 36 ($). Framing Error = 0 Overrun =0

```
$$ Character = 37 (%).  Framing Error = 0  Overrun =0
$$ Character = 94 (^).  Framing Error = 0  Overrun =0

$
```

UNGERMANN-BASS NIU-150 TEST

The Ungermann-Bass NIU-150 test checks the operation of the UB NIU-150. Turn on or reset the NIU and wait for its status light to blink. If the light does not blink within one minute, the NIU has failed its self-test.

The UB test first checks the initial state of the status lines. These should indicate that there is no NIU error, that a frame is ready from the NIU, and that the NIU can accept a frame from the Ridge.

Next, the data path from the NIU is checked by reading the boot request frame which is queued in the NIU and comparing it to its expected contents.

Finally, the data path to the NIU is checked by sending it a download frame, and verifying that a proper acknowledgement is returned.

An example of the UB test appears below.

```
$fr11diag                                          [RETURN]

DR11 Diagnostic    Version 1.xx
Enter DR11 device number (hex) :                   ff [RETURN]
Device Type = 32 (hex) : UB NIU-150.

Do you want to do register bit tests               (Y) ?n [RETURN]
Do you want to do loopback tests                   (Y) ?n [RETURN]
Do you want to do the UB test                      (N) ?y [RETURN]
Do you want ot stop on errors                      (Y) ?y [RETURN]
Loop count:                                        2 [RETURN]
Reset NIU, wait for status light to blink, press <return>.
                                                   [RETURN]

DR11 Diagnostic - UB Test
Waiting for frame ready from UB
Waiting for frame ready from UB

DR11 Diagnostic - UB Test
Waiting for frame ready from UB
Waiting for frame ready from UB

$
```

CHAPTER 8
SYSTEM DIAGNOSTICS

ON-LINE AND OFF-LINE DIAGNOSTICS

There are two very general types of diagnostics: on-line and off-line.

The on-line diagnostics are called **systest** and are run on an alive-and-well system. They reside on the system disk.

The off-line diagnostics called **SUS** are run when ROS cannot be booted, so **sus** must be booted off the floppy disk after the system fails, or before the Ridge Operating System is booted.

The diagnostic subsystems, such as the tape diagnostics or DR11 diagnostics, are named the same in both **systest** and **sus**. The dialogue for a board diagnostic is described in that board's Diagnostic section.

TO RUN SYSTEST

Before running any on-line **systest** diagnostic, log in as **root** and remove all lines from the **/ros/conf** file *except*:

```
:1:/drivers/fdlp
```

After using **systest**, reinsert any lines that were removed from **/ros/conf** and reboot the system.

After logging in as **root**, enter

```
# cd /usr/test/sys
# systest
```

The question:

```
Do you want Standard Configuration? (y)
```

will be displayed. If you answer "y" to this question, *systest* will run the standard default tests. If you wish to run a specific set of diagnostic tests, type "n" and answer the questions about what tests are to be done.

TO RUN SUS

Insert the **sus** floppy in the drive and boot the **sus** by holding down the DEVICE 2 button and pressing the LOAD button until the red light on the floppy disk goes on.

The screen will show a **SUS>** prompt when **sus** is ready for sub-command input. Sub-commands are listed on the following pages.

STAND–ALONE UTILITY SYSTEM (SUS)

INTRODUCTION TO SUS

The Stand-Alone Utility System (SUS) allows diagnosis of the Ridge Computer without the use of the hard disk. SUS contains a memory diagnostic, Ridge I/O board diagnostics, and a set of system utilities (most of which are disk utilities).

SUS is used primarily by manufacturing to test and repair memory and I/O boards. The technician can run and set breakpoints inside each of the diagnostics. This allows the technician to stop the machine at a point where the board fails.

The SUS is booted from floppy and then resides in the first one megabyte of main memory. This allows manufacturing to test memory and I/O boards with a minimum configuration test station: the processor set, a one-megabyte array, one FDLP board, and the floppy drive.

SUS is also used, to a lesser extent, by Field Service for analyzing and repairing hard disk problems. The system engineer will primarily use the disk utilities to repair bad spots on the hard disk created by power glitches, reset during disk write, or faulty hardware.

SUS includes an interpretive language which gives the user looping constructs. These constructs can be used both interactively and from a script file resident on floppy.

STRUCTURE OF SUS

Figure 8-1 shows the overall structure of SUS. SUS is primarily composed of a monitor that interprets input command lines and then runs the appropriate routine. The monitor and the basic utilities (such as cat, rm, and text) are compiled and linked into SYSU. The monitor calls the basic utilities as local procedures and calls the high-level utilities and diagnostics as external procedures. This facilitates the addition of new utilities and diagnostics. The high-level utilities and diagnostics are compiled separately and then the code is extracted. The extracted code is put into files on the SUS floppy. The monitor overlays the code from the file into real memory and then executes it. To create the SUSYSTEM, SYSU is linked with a modified version of the Ridge kernel. The SUS is created by writing filler, RBug, and SUSYSTEM to a floppy. Both the utilities and diagnostics are written in Pascal, except for a small portion that is written in Assembler.

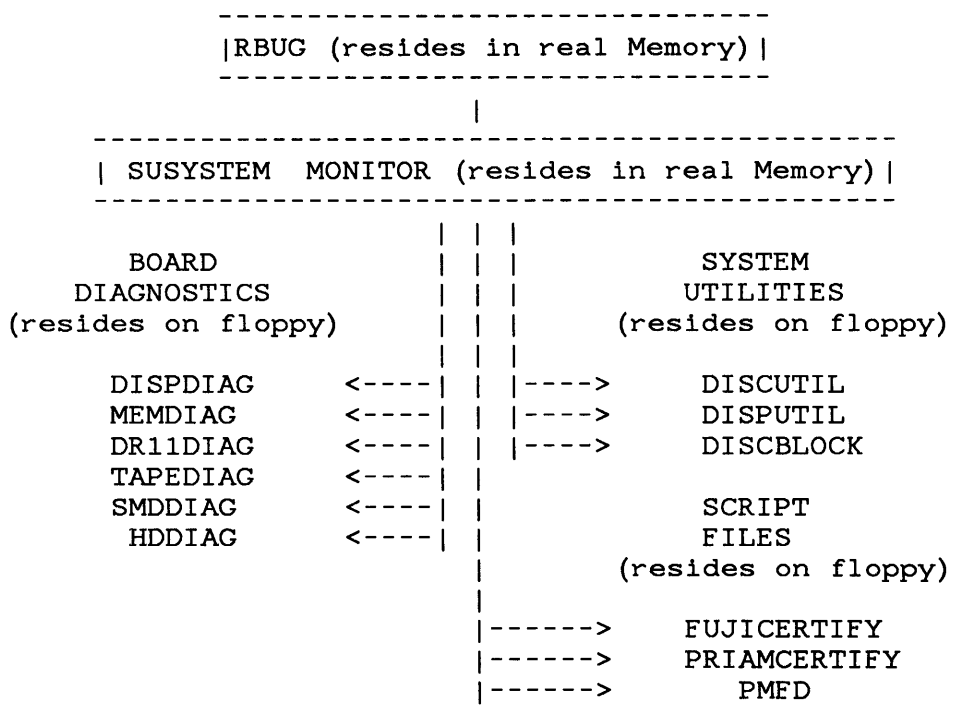


Figure 8-1. Susystem Structure

When writing utilities and diagnostics that are to be added to the SUS, remember that the amount of available data stack and heap space is limited to 80 pages and 60 pages, respectively. If you use more than what is available, the SUS will page fault and will end up in Rbug. The SUS runs in virtual mode, but can not handle page faults. Another problem is passing the input and output file pointers. To do this, add the modifications in the example shown in Figure 8-2.

```
            PROGRAM NameDiagnostic(input,output,stderr);
insert>     Procedure NameDiag(Var Input  : Text;
                               Var Output : Text;
                               Var Stderr : Text);
            Declarations
            Body of Program

 mod>       End(.);   change period to semicolon at end of
                      original program
 add>       Begin

 add>         NameDiag(Input, Output, Stderr);

 add>       End.
```

Figure 8-2. Header of a New Diagnostic Program

When writing a diagnostic, remember that, because the diagnostic is called as a procedure, any heap space that is allocated will not be automatically deallocated. Therefore, heap space should be explicitly deallocated at some point within the diagnostic. This is necessary when allowing the SUS to loop the diagnostic.

There are shell command files which do all the compiling and linking. The *mkdiags* file will have to be modified to compile and link NameDiagnostic along with the original diagnostics. Run *mksysu* and then run *mkdiags*. To make the SUS floppy, put a floppy in the drive and enter

```
# /usr/test/sus
# make
```

SUS INTERPRETER

The SUS Interpreter allows the user to write programs using the SUS commands. The SUS Interpreter understands several looping constructs, plus one conditional statement. The syntax for each of these is illustrated in the flow charts below. The three constructs (FOR, REPEAT, and WHILE) are similar to the forms found in the C and Pascal programming languages. The conditional statement is similar to the form found in the Unix Shell language. In each of these constructs, DO and DONE are used to group commands together. The syntax of expressions can be found in the Expressions section of this manual. Some examples of programs written using these constructs can be seen in the Script Examples section of this manual.

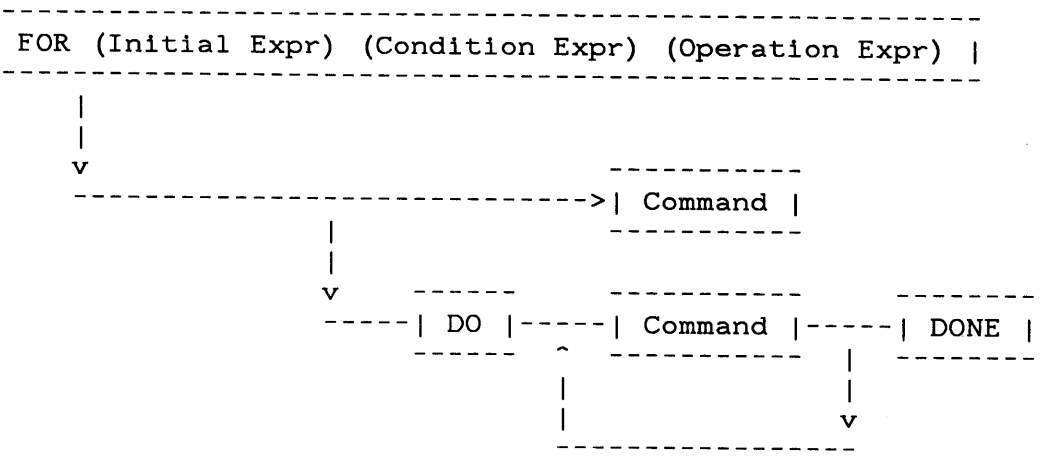


Figure 8-3. FOR Loop Syntax

In the above FOR loop, the initial expression is evaluated first. Then, as long as the condition expression is true, the command(s) are executed. The operation expression is evaluated at the end of each loop.

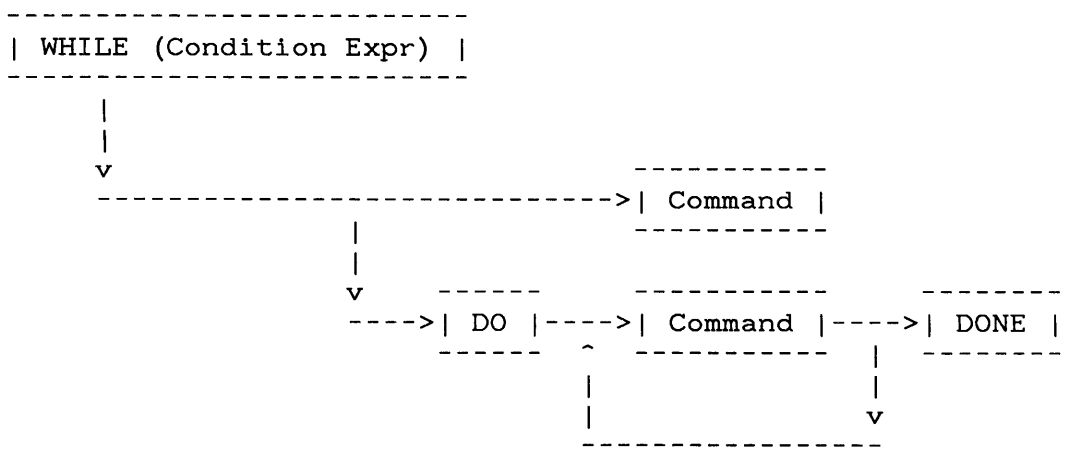


Figure 8-4. WHILE Loop Syntax

In the above WHILE loop, the command(s) are executed as long as the condition expression is true.

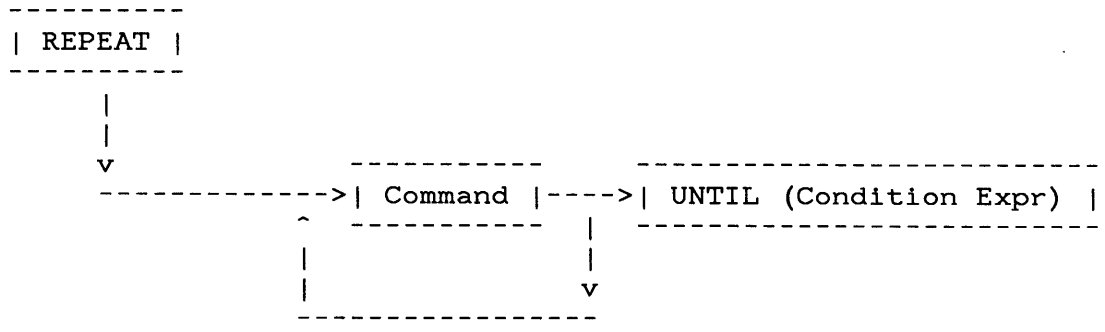


Figure 8-5. REPEAT Loop Syntax

In the above REPEAT loop, the command(s) are executed until the condition expression is true. The conditional expression is evaluated at the end of the loop.

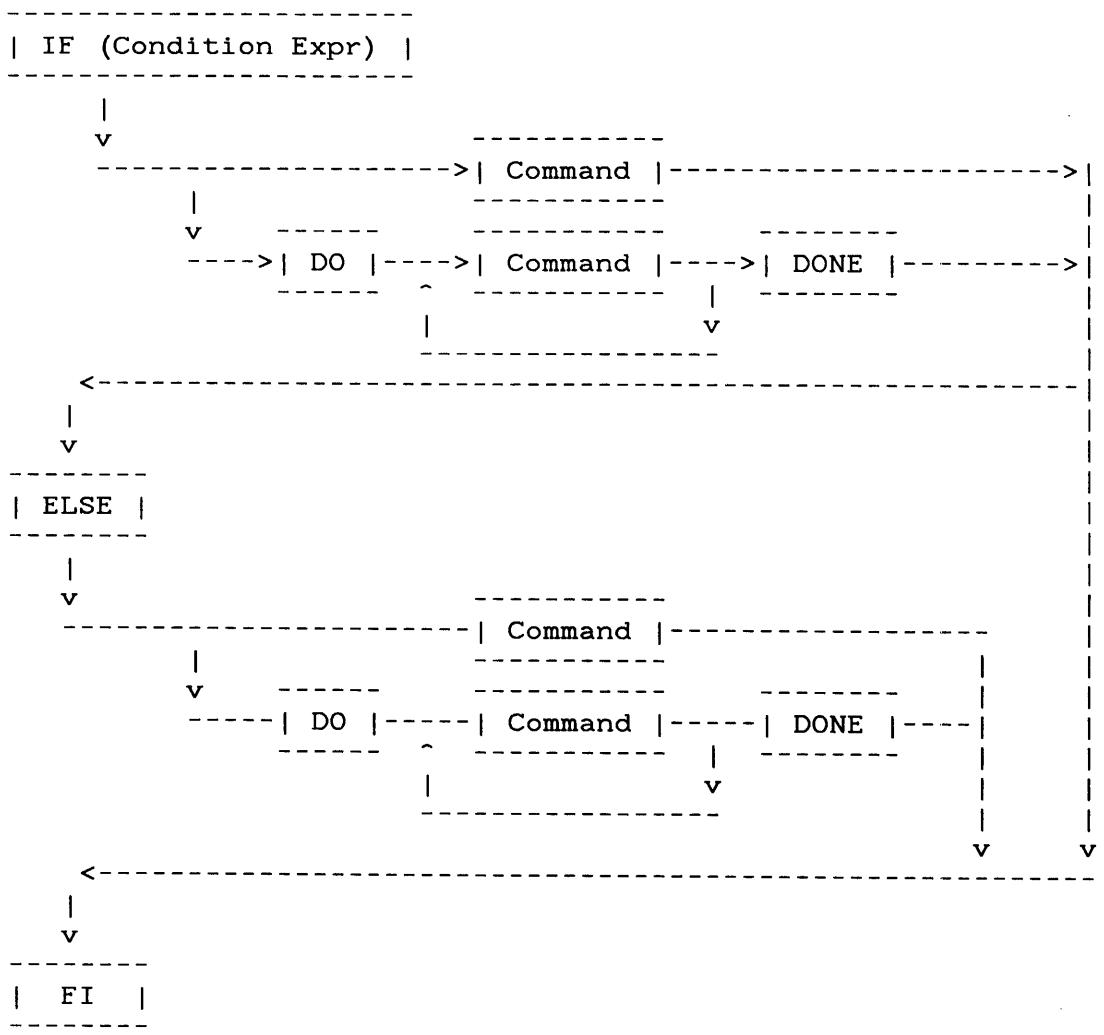


Figure 8-6. IF Conditional Syntax

SUS EXPRESSIONS

The SUS contains an expression handler. It's job is to analyze an argument and return an integer as the result. Expressions are made up of operators, functions, and variables. Expressions can be used in either decimal or hexadecimal mode, which is selectable with the dec and hex commands, respectively. The expression handler does not detect arithmetic overflow.

The operators, which are defined below, are similar in form to the operators in the C programming language. The nine functions available are TRUE, LDEN, RAND, MBRD[Expr], MEMR[Expr], MHRD[Expr], IORD[Expr], WINT[Expr], and LRRD[Expr]. Each of the functions are defined below. There are twenty-six variables that can be used in expressions, labeled A through Z. These variables are of type integer (32 bits). You can produce the real address of a variable by using the '&' in front of the variable label. For example, '&X' would produce the

real address of variable X. These real addresses can be used to pass SUS variables to assembly language programs loaded into the code space area.

To run an assembly language program in real mode use the RUN command. Because of these variables, it is important to protect the hexadecimal numbers A through F with a leading decimal number. Values can be assigned to variables by using the '=' operator inside expressions, or interactively by using the **read** command. The contents of variables can be printed by the **echo** and **echoln** commands. For further explanation of these commands, refer to the SUS Commands section of this manual.

In expressions, true and false are treated the same as they are in the C programming language. False is zero and true is anything but zero. Functions, such as LDEN, and Conditional operators, such as '==', return values with all bits set for true and no bits set for false. This facilitates the use of the '!' operator, which is a bit-wise "not." This allows you to use !LDEN to check for when the Load Enable switch is off.

Operators:

+	- Add
-	- Subtract
*	- Multiply
/	- Divide
%	- Mod (Remainder)
!	- Not (Bit wise)
&&	- And (Bit wise)
||	- Or (Bit wise)
>>	- Logical Shift Right
<<	- Logical Shift Left
==	- Equal
!=	- Not Equal
<	- Less than
>	- Greater than
<=	- Less than or Equal
>=	- Greater than or Equal

Functions:

TRUE	Returns true. (FFFFFFFF Hex)
LDEN	Returns true if Load Enable Switch is on.
RAND	Returns 32 bit random number.
MBRD[*Address*]	Returns a byte from main memory using a real address. Address is an expression.

MEMR[*Address*] Returns a word from main memory using a real address. Address is an expression.

MHRD[*Address*] Returns a half word from main memory using a real address. Address is an expression.

LRRD[*Address*] Returns value of Logging Ram read of Address. Address is an expression.

IORD[*Address*] Returns I/O read data from device number contained in Address. Address is an expression.

WINT[*Time*] Wait for Interrupt and return IOIR data from device that has been acquired by the ADEV command. Time can be used to specify the number of seconds to wait before timing out. If time out occurs then a message is printed on the screen. Time is an expression.

Expression Format:

Value = Number or Variable or Function

[!] [(] Value [Operator] [Value] [)] [Operator] ...

SUS COMMANDS

When SUS is booted, it looks for two files, in order:

DangerOn If this file exists, a flag is set to cause SUS to issue warnings before any dangerous command is executed, and to allow you to abort that command. The DangerOn file may be removed by the RM command to keep the DangerOn flag from being set. The DangerOn file can be re-created by the Text command.

default If this file exists, SUS will execute the commands contained in it. This file does not exist unless you create it. The **Default** file is especially useful when testing large batches of boards. By setting the Default file up to run the diagnostics necessary for testing a particular board, the technician can just install the board in the Ridge and power up. The SUS will boot and run the tests.

The following is a list of SUS Commands and their functions. A list of the commands can be obtained from the SUS by typing an 'H' after the SUS prompt, as follows:

```
SUS> H
```

ADEV *device* Acquire device for receiving interrupts. Only one device can be acquired at one time. (see RDEV and WINT)

B *address* Sets Rbug break point at the given relative address. The given address is added to the code base. The RBug break point base is set so relative addresses can be used when setting break points in RBug.

CAT *fromfile* [*tofile*]
Cat floppy file to another floppy file, if no *tofile*, then cat to screen.

DATE [*YYMMddhhmm*]
Set or display date and time.

DEBUG Toggles SUS tracing mode on and off.

DEC Tells the monitor that expressions typed in are decimal.

DIR Prints the directory of the floppy.

DISCBLOCK Runs Disk Control Block Utility. Allows the sending and receiving of user specified device control blocks with either a HD Control board or a SMD Control board. Only works with Televideo type of terminal.

DISCUTIL Runs Disk Utility. See list of commands in Disk Utility Command section.

DISPDIAG Runs Ridge Display Diagnostic.

DISPUTIL Runs Ridge Display Utility. Allows user to write and read display buffer.

DR11DIAG Runs DR11 board diagnostic.

ECHO [*var*] [*'text'*]
Prints contents of variables and text enclosed in either single quotes or double quotes.

ECHOLN [*var*] [*'text'*]
Prints contents of variables and text enclosed in either single quotes or double quotes. Then issues a line feed carriage return.

END Causes command lines to stop being added to the subroutine area. See **SUB** command.

EVAL,E *expression*

Evaluates an expression. See Expression section of this manual for syntax.

GOSUB *linenumber*

Continue executing command lines starting at linenumber and continuing until RETURN command is encountered.

H Prints the list of valid commands.

HDDIAG Runs HD board diagnostic. Do not run this test unless the drive connected to the HD board is a scratch drive. The test will write over the disk.

HEX Tells the monitor that expressions typed in are hexadecimal.

HEXDUMP *filename*

Performs a hexdump of a floppy file.

IORD *address*

Reads I/O using the given address. Prints to screen if in verbose mode.

IOWR *address data*

Writes I/O using the given address and data. Prints to screen if in verbose mode.

LINES Toggles display command line numbers mode.

LOAD *filename* Load code from floppy *filename.*

LRRD *address* Does Logging Ram read using the given address.

LRWR *address data*

Does Logging Ram write using the given address and data.

MBRD *address* Reads a byte from main memory using a real address. Prints to screen if in verbose mode.

MBWR *address data*

Writes a byte to main memory using a real address. Prints to screen if in verbose mode.

MC *reladdr* Allows modification of code loaded in code space area, a byte at a time.

MEMDIAG Runs the memory diagnostic on each memory array board in the system except for board 0.

MEMR *address* Reads a word from main memory using a real address. Address must be on a word boundary. Prints to screen if in verbose mode.

MEMW *address data*
Writes a word to main memory using a real address. Address must be on a word boundary. Prints to screen if in verbose mode.

MHRD *address* Reads a half word from main memory using a real address. Address must be on a half word boundary. Prints to screen if in verbose mode.

MHWR *address data*
Writes a half word to main memory using a real address. Address must be on a half word boundary. Prints to screen if in verbose mode.

RASM [*-l fname*] *source.s*
Run Ridge assembler on source file and put binary code into source.o file. Code can be loaded by using the LOAD command and executed by using the RUN command. The -l option allows you to get a listing of the assembled source file. See section on "Using RASM" of this manual for more information.

RDEV Release device that was acquired by the ADEV command.

READ *var* Waits for expression to be entered and places that value in the variable var. Var can be A through Z.

RETURN Used to resume executing command lines at the location after the GOSUB command was encountered. Only valid after doing a GOSUB command.

RM *filename* Removes *filename* from floppy.

RUN [*offset*] Executes code loaded in by the LOAD command. Code is executed in Kernel mode (using real addresses). The optional argument can be used to start execution at a location other than zero. See RASM command.

SAVE *filename* Saves code from main memory into floppy *filename.*

SH *filename*	Executes commands from floppy shell command file.
SMDDIAG	Runs SMD board diagnostic. Do not run this test unless the drive connected to the SMD board is a scratch drive. The test will write over the disk.
START	Resets current command line number to one.
SUB	Causes subsequent command lines to be added to the subroutine area until the END command is encountered.
TAPEDIAG	Runs tape board diagnostic. Do not run this test unless the tape in the drive is a scratch tape. The test will write over the tape.
TEXT *filename*	Allows text to be entered into a floppy file. To get out, hit del.
VERBOSE	Toggles verbose mode on and off.
WAIT *seconds*	Wait for a specified number of seconds, can be expression.
WINT [*timeout*]	Wait for interrupt from device. Must use ADEV command before using this command. The optional timeout factor can be used to specify a number of seconds to wait before timing out. If time out occurs, a message is printed on the screen.
WHAT	Prints device number, device ID number, and name of device for all responding I/O devices.

USING RASM

The purpose of RASM on the SUS is to allow the user to write, assemble, and execute Ridge assembly routines. The SUS interpreter can handle most programming jobs but there are instances where either the interpreter lacks the proper command or a very tight loop is needed. RASM gives the user access to any Ridge machine instruction. The drawback in using Ridge assembly routines is the difficulty of interactive communication between the user and the program. With this in mind, some modifications were made to the standard version of RASM for use on the SUS. The following description assumes that the reader is familiar with the standard version of RASM and the Ridge machine instruction set. The reader can become familiar with both by reading the Assembler (AS) section of the ROS Programmer's Guide.

RASM on the SUS has the same argument structure as the standard RASM. RASM can be given a single argument which must end in ".s". The single argument is the users source file to be assembled. To get an assembly listing, the -*l*filename option can be used. Below is a simple example of how communication

between the SUS interpreter and an assembly routine can be established.

```
;   R0 - Value of x
;   R1 - Address of x
;   R2 - Value of y, Address of z
;   R3 - Address of y
;
    LADDR   R1,<&x>,L  ;Get address of variable x
    LOAD    R0,R1      ;Get value of variable x
    LADDR   R3,<&y>,L  ;Get address of variable y
    LOAD    R2,R3      ;Get value of variable y
    STORE   R0,R2      ;Write x to location y
    LOAD    R0,R2      ;Read location y into R0
    LADDR   R2,<&z>,L  ;Get address of variable z
    STORE   R0,R2      ;Write R0 into variable z
    RET     R11,R11    ;Return to SUS interpreter
```

The purpose of this assembly routine is to write and then read data from main memory using specified data and address. The first line of the assembly routine contains a nonstandard expression "<&x>". This expression is interpreted as the real address of the SUS variable *x*. The value of the SUS variable *x* could also be obtained by using a similar expression "<x>". The value of a variable is less useful, since it is only the value of the variable at the time of the assembly. Once the address of a variable is contained in a register, it is a simple matter to get the current value of the variable by executing a LOAD. The value of variable *y* is loaded into R2 in the same way. The first STORE instruction writes the value of variable *x* into the memory location specified by the value of variable *y*. The next instruction reads the same memory location into R0. To pass this data back to the SUS interpreter, the address of variable *z* is loaded into R2 and then R0 is written into R2. A return with R11 is used to get back to the SUS interpreter. One rule to keep in mind when writing an assembly routine is to use registers R0 - R7, rather than R8 - R15. Registers 8 - 15 are used by the Pascal compiler to keep track of the stack and procedure calls. If these registers are used, there is no way to get back to the SUS interpreter without crashing.

Now that the assembly routine has been explained, let's assemble and then run it. Assuming the name of the floppy file that contains the routine is *temp.s*, the routine can be assembled by typing *rasm temp.s*, followed by a carriage return. This will produce a file named *temp.o*. The *temp.o* file is an executable binary object file. Before running the routine, the SUS variables *x* and *y* need to be loaded with values. An interpreter program for doing this and running the assembly routine is shown below.

```
echo "Enter Data to write: "
read x
echo "Enter Location for write/read: "
read y
load temp.o
run
echoln "Data read was: " z
```

The first four lines of the program allow the user to specify the data and

location that the assembly routine will use. The load command gets the code from the *temp.o* file and the **run** command executes it. When the **run** command is used, the code is executed in kernel mode, which means all addresses are real addresses, rather than virtual addresses. This gives the user the ability to map out memory in a variety of ways and to write directly to device control blocks. Keep in mind that the SUS takes up the first one mega byte of memory so any addresses above 100000 hex can be used freely. The run command can be given an argument which tells it to start executing at a location other than 0. This allows the ".o" file to contain several different routines that can be accessed by run with an offset. The offset can be found by looking at a RASM listing. The last line of the program just prints out the value of variable *z*. As with diagnostic code, the assembly code can be modified after being loaded using the modify code command (MC) and break points can be set by using the break command (B).

DISK UTILITY COMMANDS

The Disk Utility contains four lists. Initially they are empty. These lists hold information about bad spots on the hard disk. The Bad Block List and Bad Page List both contain bad spot information that was determined by the disk manufacturer. The only difference between these two lists is the way in which the information is formatted. The Bad Block List describes the bad spot by track number, head number, byte offset number, and bit length number. The Bad Page List describes the bad spots as disk page number, with head to head and track to track skew taken into account.

The Suspect Block List and the Suspect Page List are in the same format as the lists above, except the information contained in these lists is determined by running one of two utility commands, **certify** or **verify**. Because of this, the information contained in these lists tends to be about "new" bad spots on the disk.

To run the Disk Utility, type:

```
SUS> discutil
```

The system will respond with a DU prompt, which indicates that the disk utility is active.

The following is a list of Disk Utility Commands and their functions. A list of the commands can be obtained by typing an 'H' after the DU prompt as follows:

```
DU> H
```

ABP *page* Add bad page to bad page list.

CAT *fromfile* [*tofile*]
Cat floppy file to another floppy file, if no tofile then cat to screen.

CERTIFY [*startpage* [*numberofpages*]] [**L**]
Certify is similar to verify, except that each block on the disk is read and saved, then that block is written over with the MFM worst case pattern. This block is then read again to check if it is still readable. The original contents of the block is then written back. This command should be proceeded by a MBB or RBB command. It is very important not to do a reset or powerdown during a certify. To get out of a certify safely, turn load enable off. This allows certify to continue to run until it reaches a safe exit point.

CVH *pagenumber* Converts a page number to head, track, and sector.

CVP *head track byteoffset*
Converts a defect location on the hard disk to a page number.

CVS *head track* Convert Head Track to Skew for that Head Track.

DATE [*YYMMddhhmm*]
Set or display date and time.

DEBUG Toggles tracing mode on and off.

DEC Tells the monitor that expressions typed in are decimal.

DEV [*devicenumber*]
Display or set (if device number is given) the device number of the hard disk board to be used by the disk utilities. Default is device 2.

DP *page offset* [*cnt*]
Displays disk data via a page buffer.

DS *head track sector* [*offset cnt*]
Displays disk data via a sector buffer.

ECHO [*var*] ['*text*']
Prints contents of variables and text enclosed in either single quotes or doublequotes.

ECHOLN [*var*] ['*text*']
Prints contents of variables and text enclosed in either single quotes or double quotes. Then issues a line feed carriage return.

END Causes command lines to stop being added to the subroutine area.

EVAL expression Evaluates an expression. See Expression section of this manual for syntax.

FORMAT [*head track*]
Format a track at a time. If head and track are not given then format the whole disk.

GOSUB *linenumber* Continue executing command lines starting at linenumber and continuing until RETURN command is encountered.

H Prints the list of valid commands.

HEX Tells the monitor that expressions typed in are hexadecimal.

LBB Print bad block list.

LBP List bad pages using skew information.

LINES Toggles display command line numbers mode on and off.

LSB Print suspect block list.

LSP List suspect pages using skew info

MBB Make bad block list from information on the hard disk. In verbose mode, bad block information is printed on the screen.

MP *page offset* Modifies disk data via a page buffer.

MS *head track sector* [*offset*]
Modifies disk data via a sector buffer.

MSKEW Displays current skews and then allows the user to modify them.

PCI Prints the hard disk configuration information.

Q Quit Disk Utility, Go Back to SUS.

RBB [*filename*] Read bad block list from floppy. In verbose mode, bad block information is printed on the screen.

READ *var* Waits for expression to be entered and places that value in the variable var. Var can be A through Z.

RETURN Used to resume executing command lines at the location after the GOSUB command was encountered. Only valid after doing a GOSUB command.

RM *filename* Remove file from floppy.

SEEK *track* Seek to the specified track. In verbose mode, a seek message is printed on the screen.

SH *filename* Executes commands from floppy shell command file.

SP *page* Scans a disk page a sector at a time and reports any errors.

START Resets current command line number to one.

SUB Causes subsequent command lines to be added to the subroutine area until the END command is encountered.

UNIT [*unitnumber*] Display or set (if unit number is given) the unit number of the drive to be used by the disk utilities. Default is unit 0.

V [*startpage* [*numberofpages*]] [**L**]
Verify hard disk by reading the whole disk, skipping any blocks in the bad block list. Should be proceeded by a MBB or RBB command. Any errors found while reading a block will cause that block to be added to the suspect block list. The L option causes the verify to loop until load enable is turned off. Start page and number of pages allow the verification of a particular area of the disk. By leaving them out, the whole disk is verified.

VERBOSE Toggles verbose mode on and off.

WAIT *seconds* Wait for a specified number of seconds, can be expression.

WBB [*filename*] Write bad block list to floppy file.

WBP Write bad page list to floppy file (3.0 Style).

WBSB [*filename*] Write the combined contents of the bad block list and suspect block list to floppy.

WDDF *page offset* [*cnt*]
Write disk data to floppy file.

WFDD *page offset [cnt]*
Write floppy data file to disk. If parameters are not given, they are taken from the header of the floppy file.

WHAT Prints device number, device ID number, and name of device for all responding I/O devices.

WSB [*filename*] Write suspect block list to floppy file.

WSP Write suspect page list to floppy file.

DISK REPAIR PROCEDURE

The symptoms of a bad spot on a system hard disk are I/O errors pertaining to a particular device number, unit number, and page on unit number. When errors of this type are displayed, it is important to write down this information before going to the SUS. With this information, the hard disk can be analyzed by using the disk utilities on the SUS. After booting the SUS, run DISCUTIL. Then make sure the device and unit numbers are set correctly. The DU defaults to device 2 and unit 0, which is normal for a single volume system. If the system is a multiple volume system, check the error message for the correct device number and unit number. The device number and unit number can be modified by using the commands DEV and UNIT.

The next step is to find the particular sector or sectors that are bad. This is done by entering the scan page (SP) command, followed by the number of the badpage, as follows:

```
DU> SP badpage
```

Note that the numbers you type in are assumed to be in decimal unless the command HEX is typed in; after which, numbers will be assumed to be hexadecimal. Scan page will read each sector in that page and report any errors along with the head, track, and sector numbers.

Having found the bad sector, the next step is to write over that sector to repair it. This is done by using the modify sector command (MS) as follows:

```
DU> MS head track sector
```

MS should display the first character it read and then an equal sign. The next step is to type in a non-hexadecimal character such as '/' to cause MS to write over the disk sector. If MS encountered an error when it read the sector (which it should in this case), a message will warn you that the buffer about to be written to the disk may contain invalid data and ask you if you wish to continue. This is not a problem since that sector cannot be read. Type in 'yes' to direct the MS to write to disk. Now, scan page can be used to check and see if the bad sector has been repaired. Repeat this procedure for each bad page. Note that,

even if every bad page is repaired on a disk, there is no guarantee that the system will come up again. If the bad spots occurred in a vital area, such as in the directory, there may be no way to bring the system back up without reloading the software.

SCRIPT EXAMPLES

FujiCertify Script File (Must be run from Discutil):
Used to certify a Fujitsu Disk Drive

```
dec
eval u=0
unit u
repeat
 echo 'Enter Number of Units to test : '
 read y
until (y>0)&&(y<5)
pci
echo 'Enter Number of Pages  on Drive ' u ' : '
read a
echo 'Enter Number of Tracks per Head on Drive ' u ' : '
read b
echoln
if y==1
 do
   echoln 'Insert BadBlock Floppy for this Fujitsu Drive'
   echo   ' Hit Return to Continue ... '
   read z
   echo 'Reading BadBlock file ...'
   rbb
   echoln
   echoln 'Reinsert SUS Floppy '
   echo   'Hit Return to Continue ...'
   read z
 done
else
 do
   echoln 'For Multiple Drive Certify the SUS floppy must'
   echoln ' contain a BadBlock file for each drive to be '
   echoln ' tested.  Each of these  file names should be '
   echoln ' appended with the unit number. BadBlocks[0-3]'
   echo   ' Type return to continue ...'
   read z
 done
fi
repeat
 if y>1
  do
   echo 'Reading BadBlocks for Drive ' u ' ...'
   if u==0
    rbb BadBlocks0
   else
```

```
   if u==1
    rbb BadBlocks1
   else
    if u==2
     rbb BadBlocks2
    else
     rbb BadBlocks3
    fi
   fi
  fi
  echoln
 done
fi
echoln
echoln '***Starting Certify of Fujitsu Disk Drive ' u '***'
echoln '    Can be terminated with LoadEn Off     '
eval z=0
repeat
 seek rand>>16%b
 certify z 1
 eval z=z+1
until (z>=a)||(!lden)
if y==1
 wsb suspectblocks
else
 do
  if u==0
   wsb suspectblocks0
  else
   if u==1
     wsb suspectblocks1
   else
     if u==2
       wsb suspectblocks2
     else
       wsb suspectblocks3
     fi
   fi
  fi
 done
fi
if z>=a
 echoln '>>>>  Certify Complete on Drive ' u '  <<<<'
fi
lsp
eval u=u+1
unit u
until (u>=y)||(!lden)
unit 0
echoln 'If Page Count is not Zero then there are new Bad Spots on this Disk'
echoln 'which were not in the BadBlock file,  SuspectBlock file has been'
echoln 'written to floppy.'
```

PMFD Script File:
Does random seeks on floppy drive to clean heads

```
hex
eval y=46000001
eval x=1
if true
do
 echoln 'Insert cleaning diskette into floppy drive'
 echo   'Hit return to continue ...'
 read z
 adev 1
 repeat
  memw 3c0c0 0
  memw 3c0c4 00100000
  memw 3c0c8 01000000
  memw 3c0cc x<<8||y
  eval x=rand+1>>16%4c
  iowr 01000000 86000000
  wint 1
 until (!lden)
 rdev
done
fi
```

APPENDIX A
Z80 REGISTERS

SMD WRITES

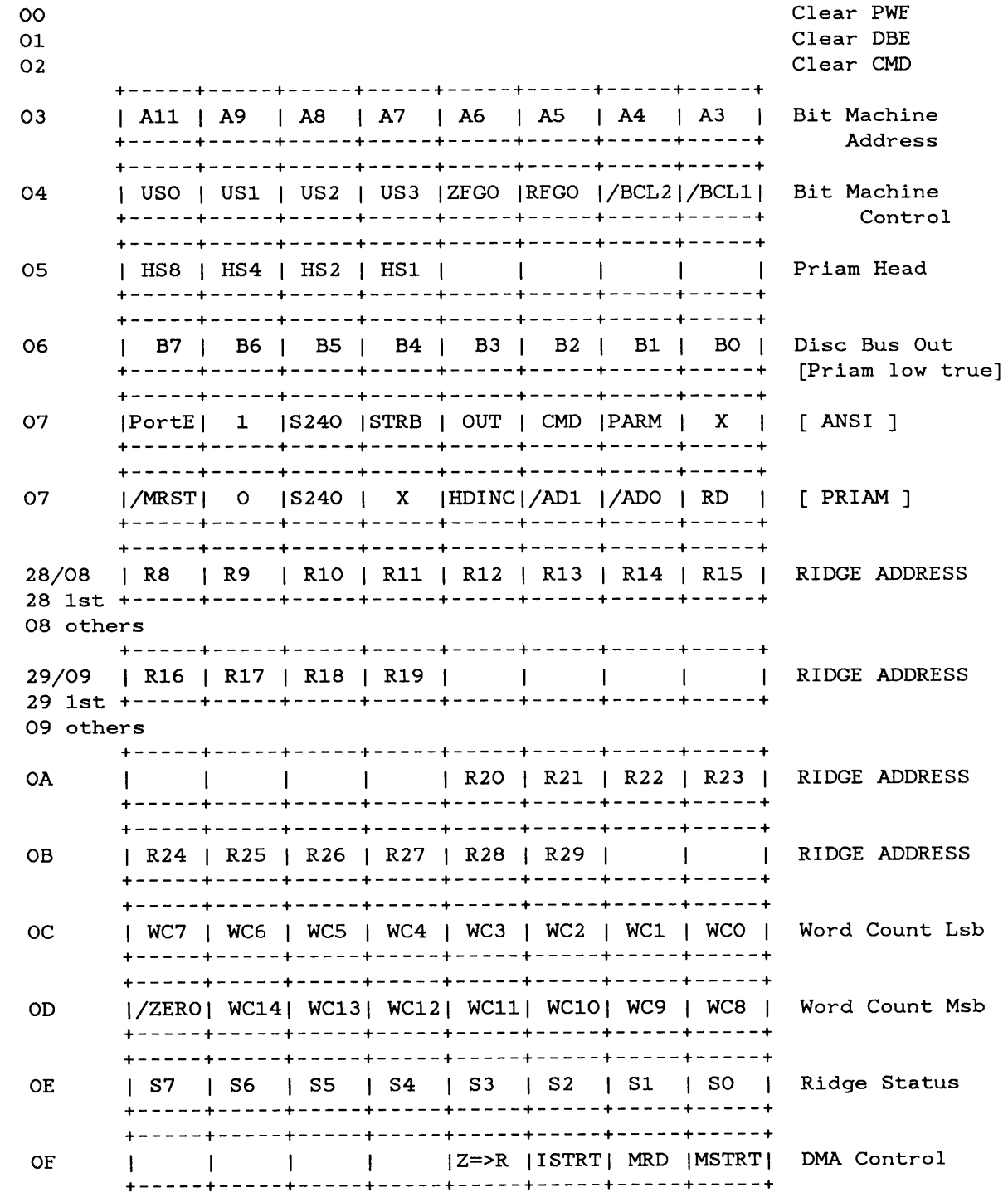
addressbits

7xx0	WC7	WC6	WC5	WC4	WC3	WC2	WC1	WC0	WCLow
7xx1	~DONE	WC14	WC13	WC12	WC11	WC10	WC9	WC8	WCHigh
7xx2	R24	R25	R26	R27	R28	R29	R22	R23	ADLow
7xx3					R20	R21			ADHigh
7xx4	R16A	R17A	R18A	R19A	R16B	R17B	R18B	R19B	PAGLow
7XX5	R8A	R9A	R10A	R11A	R12A	R13A	R14A	R15A	PAGHigh
7xx6	R8B	R9B	R10B	R11B	R12B	R13B	R14B	R15B	ALTPAGe
7xx7									----
6XX0	R16	R17	R18	R19	R20	R21	R22	R23	STATLow
6XX1	R8	R9	R10	R11	R12	R13	R14	R15	STATHigh
6XX2	I10	I9	I8	I7	I6	I5	I4	I3	IREG
6XX3	RELES	STRBL	STRBE	RTZ	AMenb	FLTcl	SERV+	SERV-	CONTROL (TAG3)
6XX4	CYL7	CYL6	CYL5	CYL4	CYL3	CYL2	CYL1	CYL0	CYL (TAG1)
6XX5	US1	US0	USTAG	CLKen	TAG5	TAG4	CYL9	CYL8	CONTROL1
6XX6	PHLOD	BITGO	FIFGO	SCTOO	BITMD	PICKH	ZCYL	ZSTRB	CONTROL2
6XX7	HD7	HD6	HD5	HD4	HD3	HD2	HD1	HD0	HEAD

SMD REGISTER READS/STROBES

5xx0 W									NEWInstruction
5xx1 W									STARTbitmachine
5xx2 W									STARTWT (dma)
5xx3 W									STARTRD (dma)
5xx4 W									CLRDBE
5xx5 W									CLRPON
5xx6 W									SETINT
5xx7 W									----
5xx0 R	SECTR	INDEX	(AM)	WP	FAULT	SKERR	ONCYL	URDY	STAT0-7
5xx1 R	RDY	COMP error	ABORT	DATA error	ECC error	FIFO error	BAD SYNC	EXEC-UTING	BitStateMachine Status
5xx2 R	ACTIV	MNULL	INULL	DBE	BUSY	SW2	SW1	SWO	MISCSTAT
5xx3 R	SKEND 3	SKEND 2	SKEND 1	SKEND 0	SEL 3	SEL 2	SEL 1	SEL 0	SeekEND
5xx4 R	ECC7	ECC6	ECC5	ECC4	ECC3	ECC2	ECC1	ECC0	ECCL
5xx5 R	ECC15	ECC14	ECC13	ECC12	ECC11	ECC10	ECC9	ECC8	ECCH
5xx6 R	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0	BCNTL
5xx7 R	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BCNTH

HD I/O PORTS (WRITES)



HD I/O PORTS (READS)

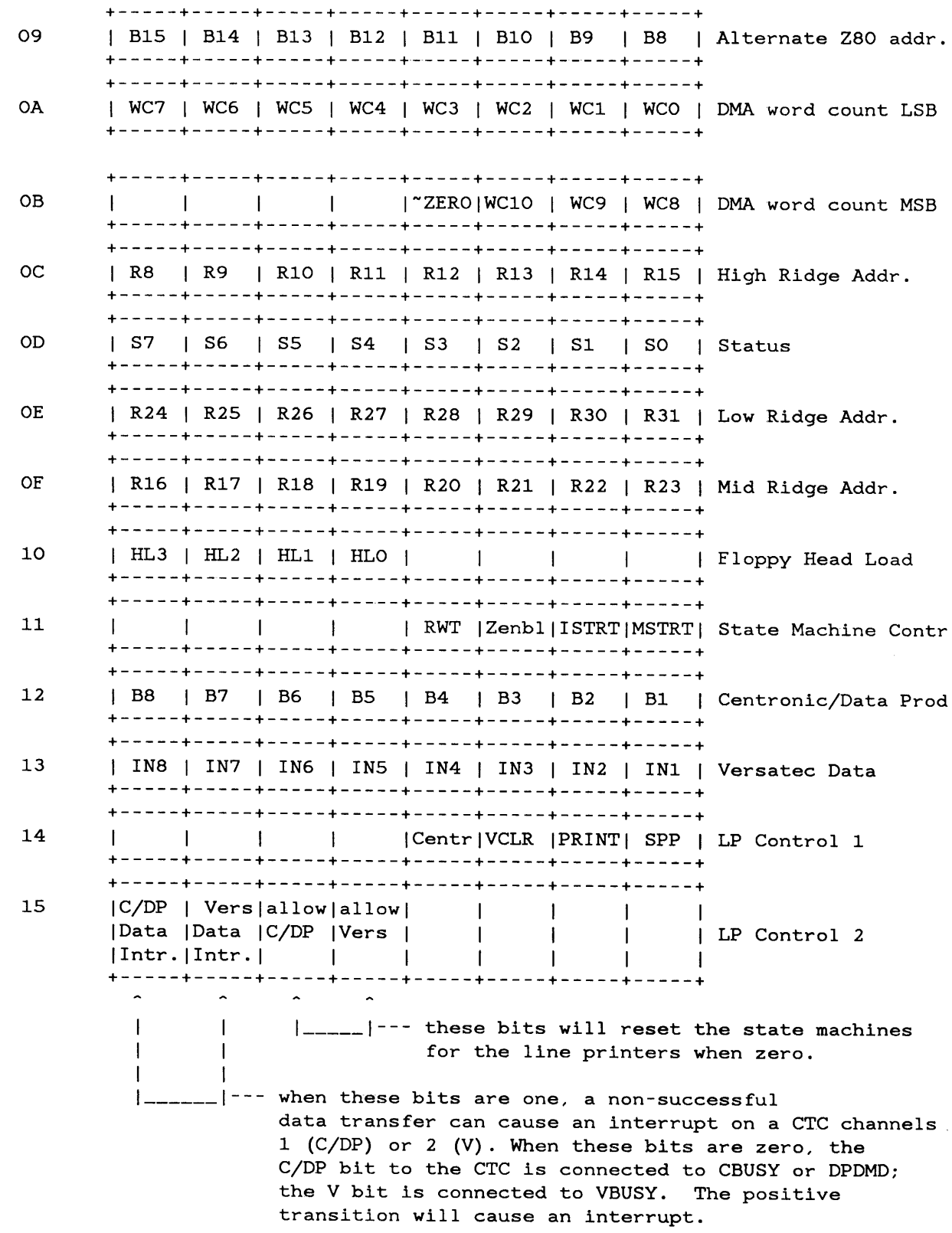
40	msb							lsb	Device Number
41	msb							lsb	Ridge Command
42	WC14	WC13	WC12	WC11	WC10	WC9	WC8	WC7	Word Count
43	MNULL	MHS	INULL	IHS	/ZIR	/ZOR	/RIR	/ROR	SM/Fifo Stat
44	SW7	SW6	SW5	SW4	SW3	CMDIN	PON	P.E.	Misc Status
45	ECC0	ECC1	ECC2	ECC3	ECC4	ECC5	ECC6	ECC7	ECC Byte
46	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0	Byte Cnt Lsb
47	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	Byte Cnt Msb
48	ECCNZ	No SYNC	ECC error	COMP error	ABORT	FIFO error	RDY	XEQN	Bit Status
49	B7	B6	B5	B4	B3	B2	B1	B0	Disc Bus In [Priam is low true]
4A	START	X	/RDY	NEWI	ATTN	BUSY	BUSAK	X	ANSI
4A	START	X	/RDY	NEWI	X	X	X	/REDY	PRIAM
4D									PRIAM WT
4E									Set 'START'
4F									Set 'NEWI'

HD I/O PORTS (OTHERS)

10	msb							lsb	Zfifo Write
18	msb							lsb	Rfifo Write
50	msb							lsb	Zfifo Read
58	msb							lsb	Rfifo Read
80-83	msb							lsb	SIO
C0-C3	msb							lsb	CTC

FD/LP I/O PORTS (WRITES)

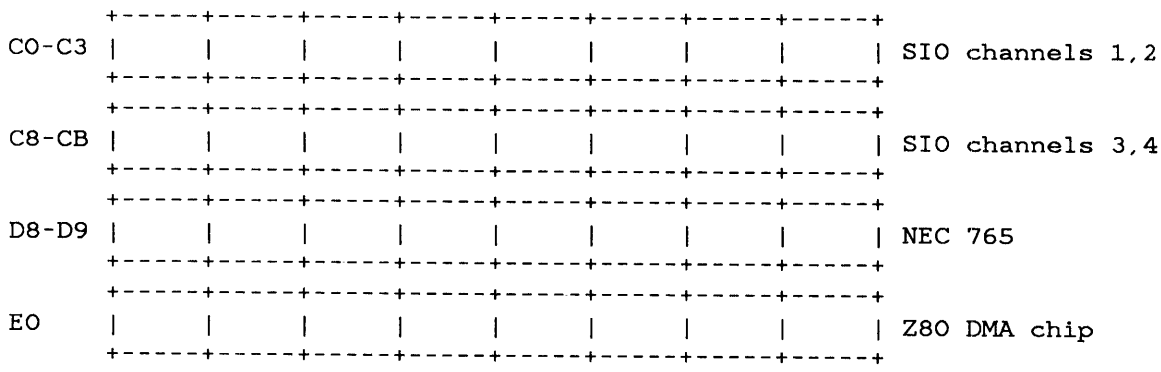
00				CLEAR	RESET	RFFED	REOTR	RLTER	LPReset
01									----
02									----
03									CLRPE
04									CLRDBE
05									CLRCMDINT
06									CLRPON
07									----
08	B7	B6	B5	B4	B3	B2	B1	B0	Alternate Z80 addr.


09 | B15 | B14 | B13 | B12 | B11 | B10 | B9 | B8 | Alternate Z80 addr.
0A | WC7 | WC6 | WC5 | WC4 | WC3 | WC2 | WC1 | WC0 | DMA word count LSB
0B | | | | | ~ZERO | WC10 | WC9 | WC8 | DMA word count MSB
0C | R8 | R9 | R10 | R11 | R12 | R13 | R14 | R15 | High Ridge Addr.
0D | S7 | S6 | S5 | S4 | S3 | S2 | S1 | S0 | Status
0E | R24 | R25 | R26 | R27 | R28 | R29 | R30 | R31 | Low Ridge Addr.
0F | R16 | R17 | R18 | R19 | R20 | R21 | R22 | R23 | Mid Ridge Addr.
10 | HL3 | HL2 | HL1 | HL0 | | | | | Floppy Head Load
11 | | | | | RWT | Zenbl | ISTRT | MSTRT | State Machine Contr
12 | B8 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | Centronic/Data Prod
13 | IN8 | IN7 | IN6 | IN5 | IN4 | IN3 | IN2 | IN1 | Versatec Data
14 | | | | | Centr | VCLR | PRINT | SPP | LP Control 1
15 | C/DP Data Intr. | Vers Data Intr. | allow C/DP | allow Vers | | | | | LP Control 2
these bits will reset the state machines for the line printers when zero.
when these bits are one, a non-successful data transfer can cause an interrupt on a CTC channels 1 (C/DP) or 2 (V). When these bits are zero, the C/DP bit to the CTC is connected to CBUSY or DPDMD; the V bit is connected to VBUSY. The positive transition will cause an interrupt.

FD/LPD I/O PORTS (READS)

	B7	B6	B5	B4	B3	B2	B1	B0	
40	B7	B6	B5	B4	B3	B2	B1	B0	Ridge Command
41	B7	B6	B5	B4	B3	B2	B1	B0	Device Number
42	INTR. SM 1	INTR. SM 0	MemSM 1	MemSM 0	DBE	PWF	CMD-INTR.	Z80 ParEr	State Machine Status
43	C/DP SM 1	C/DP SM 0	V SM 1	V SM 0	-CACK	+C BUSY	+DP DMD	+V BUSY	LP STATUS 1
44	+DPon line	+DP ready	+DP BOF	-C FAULT	+CSEL ECT	+C PAPER	+VOn line	+VNo Paper	LP STATUS 2
45					SW3	SW2	SW1	SW0	SWITCHES

FD/LPD I/O PORTS (CHIPS)



RIDGE

Reader Feedback

Manual Title: _____

Overall rating of this document	☐ Excellent	☐ Adequate	☐ Poor
Readability of text	☐ Very Clear	☐ Adequate	☐ Difficult
Usefulness of information	☐ Helpful	☐ Adequate	☐ Not Useful
Any errors?	☐ Yes	☐ No	

If so, identify error and page number _____

Additional comments: _____

Would you like more information on Ridge products? _____

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1590 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Ridge Computers

Publications Department
2451 Mission College Blvd.
Santa Clara, CA 95050-8290

Software Release Description
Standalone Utility Sustem (SUS), Release 3.5
April 15, 1987

Contents of Media

This is the release of the 3.5 version of the Standalone Utility System (SUS) diagnostics and utilities. The changes in this release were required for the support of the new Ridge 3200, CIO board, Extended Tape board, and Extended DR11 board. This release also supports RX/V with the mvtoc command in Discutil. The previous 3.3.2 version of the SUS is not bootable on a 3200 system.

There are 19 files contained on the SUS floppy. Their names are:

Operating System:	Text Files:	Utilities:
filler	dangeron	dir
rbug	fujicertify	discblock
susystem	pmfd	discutil
	priamcertify	disputil
		hexdump
		rasm

There are 12 files contained on the SUS cartridge tape.
Their names are:

Operating System:	Text Files:	Utilities:
susystem	fujicertify	discutil
	priamcertify	disputil
		hexdump

These files are common to both floppy and cartridge tape.

Diagnostics:	Product Title	Part No.	Versions
dispdiag	PCA,Displ,15	002-8058	V3C/V2D
	PCA,Displ,19	002-8064	V3D
dr11diag	PCA,DR11	000-8579	V2B
	PCA,DR11.XTD	003-0776	V3
hddiag	PCA,HD	001-4760	V3/V2B
memdiag	PCA,1MA,3311	000-8535	V1
	PCA,4MA,3312	002-0084	V1
	PCA,16MA,3314	003-5059	V1A
	PCA,MC	002-2298	V4C
	PCA,EMC	003-8356	V1C
	PCA,HW.CMC	003-4633	V4A
smddiag	PCA,SMD	004-1256	V2E/V4B
	PCA,SMD.XTD	004-1262	V6B
tapediag	PCA,TAPE	000-8507	V2B/V1C
	PCA,TAPE.XTD	003-4405	V3A

Booting the SUS from floppy

To boot the SUS follow the steps below.

1. Insert SUS floppy into floppy drive.
2. Hold front panel device switch down in position 2.
3. Press front panel load switch momentarily.
4. Continue to hold front panel device switch down until the floppy drive access light comes on.

Booting the SUS from cartridge tape

To boot the SUS follow the steps below.

1. Insert ROS 3.5 cartridge tape into drive.
2. If system is a Ridge 32 slow boot system to get RBug prompt. If system is a Ridge 3200 do a manual boot to get RBug prompt.
3. At the RBug prompt type 'c 2<cr>'.

SUS Documentation

The SUS is documented in the Stand-Alone Utility System section of the Ridge Hardware Reference Manual or in the Systems Diagnostics section of the Hardware Installation and Maintenance Manual.