Heurikon 68000 Microcomputer

HK68 (tm)

User's Manual

Heurikon Corporation
3001 Latham Drive
Madison, WI 53713

(608)-271-8700

January 1983

Rev B

User's Maual - Contents

1.0 INTRODUCTION

The purpose of this manual is to document the features and to present implementation examples of the Heurikon HK68 (tm) microcomputer board.

This manual covers the unique features of the HK68 board. Although general information such as DMAC, MMU, MPU, CIO and SCC programming is discussed, more detailed information is available directly from the chip manufacturers.

For more information, contact:

Item(s)	Manufacturer	
MPU, MMU	Motorola Inc., 3501 Ed Bluestein Blvd., Austin, Texas 78721	(512)-928-7113
DMAC	Hitachi, Ltd., 1800 Bering Drive San Jose, Calif 95112	(408)-292-6404
CIO, SCC	Zilog Inc., 10340 Bubb Road Cupertino, Calif 95014	(408)-446-4666
Winchester Controllers	Priam Corporation, 3096 Orchard Drive San Jose, Calif 95134	(408)-946-4600
	Shugart Associates, 435 Oakmead Pkwy Sunnyvale, Calif 94086	(408)-733-0100
	Western Digital Corporation, 3128 Red Hill Ave. Newport Beach, Calif 92663	(714)-557-3550
	Xebec, 432 Lakeside Drive Sunnyvale, Calif 94086	(408)-735-1340
DRAM Controller	National Semiconductor Corp., 2900 Semiconductor Dr. Santa Clara, Calif 95051	(408)-721-5000

Don't be shy about calling us with questions, too. We are prepared to help with your application.

1.1 Disclaimer

1.2 Trademarks

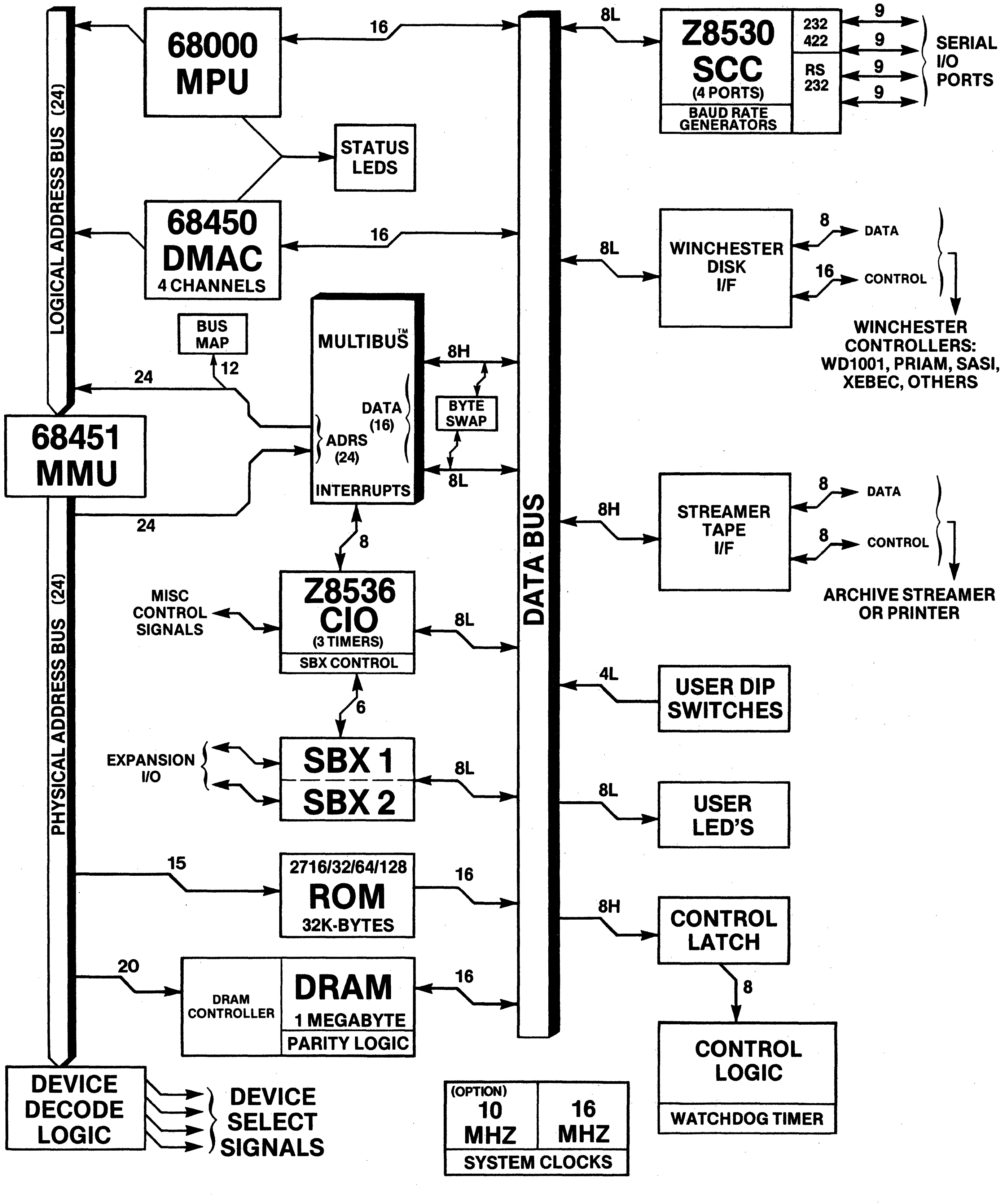
Name	Trademark of
HK68	Heurikon Corporation
Multibus, iSBX	Intel Corporation
Smart I/F	Priam Corporation
Zilog, Z-80	Zilog Corporation
CP/M-68K	Digital Research Corporation
Unix	Bell Labs
UniPlus+	UniSoft Corporation

2.0 HK68 FEATURE SUMMARY

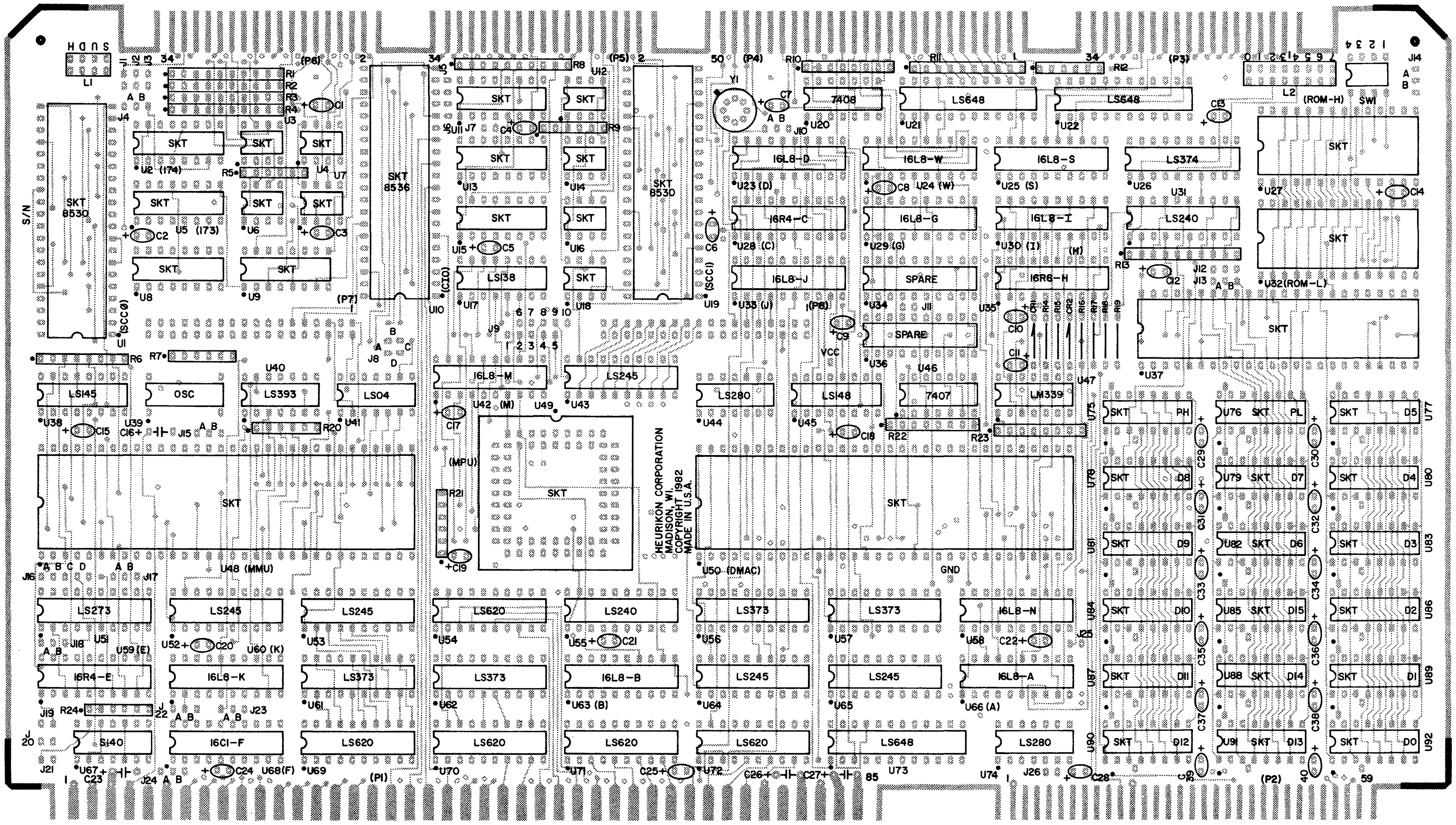
MPU	Motorola 68000 uProcessor chip (or equiv) 32-bit internal architecture, 16-bit external data path 24 address lines, 16 megabyte range
DMAC	Motorola 68450 chip (or equiv) Four channels 8-bit or 16-bit data transfers
MMU	Motorola 68451 chip (or equiv) Logical to physical address translation Operates in segment mode. Segment sizes of from 256 bytes to 16 megabytes Separates supervisor, user, DMAC, bus memory areas Implements write protection
RAM	128K, 256K, 512K, 1 megabyte capacities Single or double-decker configuration Two parity bits per word Uses 64K x 1 or 256K x 1 DRAM, with Hardware Refresh
ROM	Two ROM sockets (2716, 2732, 2764, 27128) 32K byte capacity
Multibus	24-bit addressing (16 megabyte range) 16-bit data bus, compatible with 8-bit boards Eight bus interrupts, bi-directional (via CIO) Master/Slave modes On-card byte swap buffer
Serial I/O	Two Zilog Z8530 Serial Communication Controllers Four serial I/O ports Separate baud rate generators for each port Asynchronous, synchronous modes RS-232C I/F (all ports) RS-422 I/F (optional, one port)
Winchester	8-bit I/F for Western Digital, Priam, or SASI (Shugart, Xebec) May be used as a general purpose 8-bit port
Streamer Tape	8-bit I/F for Archive streamer tape May also be used for Centronics type printer I/F
LED's	Eight user LED's under software control Four MPU/DMAC status LED's
DIP Switches	Four user definable DIP switches
SBX	Two expansion SBX connectors for APU, SIO, PIO, FDIO, etc. One single width space, one double width space.
CIO	Zilog Z8536 Counter/Timer and Parallel I/O Unit Three 16-bit counter/timers Three parallel ports for on-card control functions
Other	Winchester and Streamer Tape ports may be combined to form a 16-bit general purpose parallel I/O port. Built-in board serial number to promote software security

2.1 Block Diagram

HEURIKON HK68 SINGLE BOARD MICROCOMPUTER



CONTROL SIGNAL BUS
NOT SHOWN



HEURIKON CORPORATION
MADISON, WI.
COPYRIGHT 1982
MADE IN U.S.A.
SKT 8530 (SCC0) U1
SKT 8536 (CIO) U10
SKT 8530 (SCC1) U19
U2 (174)
U5 (173)
OSC
LS145
LS393
LS04
16L8-M
LS245
(MPU) U49 SKT
U48 (MMU)
U50 (DMAC)
7408
LS648
16L8-D
16L8-W
16L8-S
LS374
16R4-C
16L8-G
16L8-I
LS240
16L8-J
SPARE
16R6-H
LS138
LS280
LS148
7407
LM339
U27 (ROM-H)
U32 (ROM-L)
SKT
LS273
LS620
LS373
16L8-N
16R4-E
16L8-K
16L8-B
16L8-A
S140
16C1-F
(P1)
(P2)
(P3)
(P4)
(P5)
(P6)
(P7)
(P8)
GND
VCC
S/N
L1
L2
SW1
Y1
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
PH
PL

2.2 Major Component Locations

```
          SIO B & A     SIO D & C     Winchester I/F      Tape I/F

  ____     ________     _________     _______________     _________     ____
 /    |___|   P6   |___|   P5    |___|      P4       |___|   P3    |___|    \
|  Status LED's    |   |        |    |  Winc & Tape I/F   LED's              |
|                  |   | RS-232 |    | 10 MHz Osc         DIP Sw  | ROM H |  |
|   ___   RS-422 I/F|CIO|  I/F   |SCC|                            |_______|  |
|  |   |  RS-232 I/F|   |        |(1)|  System Control Logic      | ROM L |  |
|  |SCC|           |___|         |___|                            |_______|  |
|  |(0)| _________                  ____________             ____________  |
|  |___||_________|                |____________|           |DRAM Control| |
|          SBX P7                       SBX P8               |____________| |
|                                                          . --------------- |
|    System Clock            _______                       |                 |
|   ___________________     |       |    ___________________|                |
|  |                   |    |  MPU  |   |                   ||                |
|  |       MMU         |    |       |   |       DMAC        ||    Dynamic     |
|  |___________________|    |_______|   |___________________||      RAM       |
|                                                          |                 |
|          MMU Bus Gates            DMAC Bus Gates         |                 |
|                                                          |                 |
|  Bus Arbitrator       Multibus I/F                      |                 |
|_                                              ______    |_________________|
  |____________________________________________|      |____________________|
                     P1                                      P2
                  Multibus                              Auxiliary Bus
      (20 Address, 16 Data, 8 Interrupts)             (4 Address, 1 Int)
             (15 Control, Power)                          (1 Reset)
```

3.0 GETTING GOING Read This!

Here's what you need to get the Heurikon HK68 "on-the-air":

Hardware:	Heurikon HK68 Microcomputer board Heurikon HK68 Monitor pROM Card cage and power supply (e.g., Heurikon MLZ-804) Serial I/F cable (RS-232) CRT Terminal	
	Optional equipment:	Winchester and/or Floppy disk drives Expansion memory
Software:	Heurikon HK68 Monitor in pROM	
	Optional programs:	UniPlus+ (Unix) Operating System CP/M-68K Operating System Application programs

3.1 Installation Steps

All products are fully tested before they are shipped from the factory, so we know they work. When you receive your first board, follow these steps to assure yourself that the system is operational:

1. Visually inspect the board(s) for loose components which could be the result of shipping vibrations.

2. Visually inspect the chassis and all cables.
   Be sure all boards are seated properly in the card cage.
   Be sure all cables are securely in place.

3. Connect a CRT terminal to Serial Port B. (Connector P6)
   If you are making your own cables, refer to section 13.
   Set the terminal as follows:
   9600 baud, full duplex
   Eight data bits (no parity)
   Two stop bits for transmit data
   One stop bit for receive data
   If your terminal doesn't have separate controls for transmit and receive stop bits, select one stop bit for both transmit and receive.

4. Connect AC power and turn the system on.

5. Push the system RESET button. A sign-on message and prompt from the monitor should appear on the screen. If not, check your power supply voltages and CRT cabling.

6. If you've read this so far, you must not mind reading manuals. Now's the time to read the monitor manual and the operating system literature.

7. Reconfigure the jumpers, etc, as necessary for your application.

"There's no sense reading directions to something before you understand a little about it because they don't mean anything to you. You have to know enough about something to be confused before directions help." - Andy Rooney

3.2 Troubleshooting Guide

In case of difficulty, use this checklist:

1. Inspect the power cables and connectors. If the HK68 board has power, the SCC and CIO chips will feel warm to the touch. They're the big ones near P5 and P6.

2. Verify that the DIP switches are set correctly. (Refer to the 'hbug' monitor manual.)

3. Check your power supply for proper DC voltages. If possible, look for excessive power supply ripple or noise using an oscilloscope. Connect your ground probe to the "GND" test point just below the DMAC chip socket.

Voltage	Test Point
Vcc (+5)	"VCC" TP above U45 (above the DMAC)
+12	U16 pin 8 (Top-left corner pin)
-12	U16 pin 5 (Top-right corner pin)

4. Check the chips to be sure they are firmly in place. Look for chips with bent or broken pins. In particular, check the pROMs.

5. Check your terminal switches and cables. Be sure the P6 connector is on properly. The cable stripe (wire #1) should be toward the center of the HK68 board and the cable should flow toward the rear. The port B portion of the cable is on the wire #34 side. If you have made your own cables, pay particular attention to the cable drawings in section 13.

6. Check the jumpers to be sure your board is configured properly. Use this list of critical jumpers: (See section 17.2 for help in locating the jumpers and for jumper functions.)

Jumper	Position
J1, J2, J3,	open (no shunt plugged in)
J11	open
J12	A
J13	A
J14	B
J15	A
J17	A
J22	A (or "B" if using 256K x 1 RAMS)
J23	A (or "B" if using double decker RAM)
all others	don't care

7. After you've checked all of the above items, call us at (608)-271-8700 and ask for help. Have the following information handy:

   a. The state of the four status LED's (near P6)
   b. The state of the eight user LED's (near P3)
   c. The monitor program revision level (part of sign-on msg.)
   d. The HK68 p.c.b. revision level (silk screened near MPU)
   e. The HK68 p.c.b. serial number (scribed along short edge)

3.3 Monitor Summary ('hbug')

The HK68 monitor program is contained in two 2732 pROMs. It is intended to provide a fundamental ability to check the memory and I/O devices, to manually enter a program and to down-line load or bootstrap a larger program into memory. Advanced features and utilities may be loaded from media or via an operating system.

Refer to the 'hbug' manual for details on the commands and command formats.

4.0 MPU SUMMARY INFORMATION

This section details some of the important features of the 68000 MPU chip and, in particular, those items which are specific to the implementation on the Heurikon HK68. Refer to appendix A for additional information. See section 7.6 for memory timing data.

4.1 MPU Interrupts

The MPU can set an interrupt priority level in such a way that interrupts of a lower priority will not be honored. Interrupt level seven, however, cannot be masked off.

Level	Interrupt
7	Power Fail Interrupt (P2-19) Highest Priority, Non-maskable
6	MMU Interrupt
5	CIO Interrupt (sub-priority: timer 3, port A, timer 2, port B, timer 1)
4	DMAC Interrupt
3	SCC Interrupt (sub-priority: port A, port B, port C, port D) (sub-sub-priority: rcv ready, tx ready, status change)
2	(not assigned)
1	(not assigned)
0	(idle, no interrupt)

When an interrupt is recognized by the MPU, the current instruction is completed and an interrupt acknowledge sequence is initiated whose purpose it is to acquire an interrupt vector from the interrupting device. The vector number is used to select one of 256 exception vectors located in reserved locations in lower memory (see section 4.2 for a listing.) The exception vector specifies the address of the interrupt service routine.

The SCC, CIO and DMAC devices on the HK68 are capable of generating more than one vector, depending on the particular condition which caused the interrupt. This significantly reduces the time required to service the interrupt because the program does not have to rigorously test for the interrupt cause.

4.2 MPU Exception Vectors

Exception vectors are memory locations from which the MPU fetches the address of a routine to handle an exception (interrupt). All exception vectors are two words long (four bytes), except for the reset vector which is four words. The listing below shows the vector space as it appears to the Heurikon HK68 MPU. It varies slightly from the Motorola 68000 MPU manual listing due to particular implementations on the HK68 board. Refer to the MPU ducumentation for more details. The vector table occupies the first 1024 bytes of memory. Unused vector positions may be used for other purposes (e.g., code or data).

Vector	Address		
Number	Dec	Hex	Assignment
0	0	000	Reset: Initial SSP (Supervisor Stack Pointer)
1	4	004	Reset: Initial PC (Supr Program Counter)
2	8	008	Bus Error (Parity, Watchdog Timer, MMU Fault)
3	12	00C	Address Error
4	16	010	Illegal Instruction
5	20	014	Divide by Zero
6	24	018	CHK Instruction (register bounds)
7	28	01C	TRAPV Instruction (overflow)
8	32	020	Privilege Violation (STOP, RESET, RTE, etc)
9	36	024	Trace (Program development tool)
10	40	028	Instruction Group 1010 Emulator
11	44	02C	Instruction Group 1111 Emulator
12	48	030	(reserved)
13	52	034	(reserved)
14	56	038	(reserved)
15	60	03C	Uninitialized Interrupt from DMAC or MMU
16-23	64 -95	040 -05F	(reserved-8)
24	96	060	Spurious Interrupt, not used
25	100	064	Level 1 Interrupt Autovector, not assigned
26	104	068	Level 2 Interrupt Autovector, not assigned
27	108	06C	Level 3 Interrupt Autovector, not used
28	112	070	Level 4 Interrupt Autovector, not used
29	116	074	Level 5 Interrupt Autovector, not used
30	120	078	Level 6 Interrupt Autovector, not used
31	124	07C	Level 7 Interrupt, Power Fail Detect (P2-19)
32-47	128 -191	080 -0BF	TRAP Instruction Vectors (16)
48-63	192 -255	0C0 -0FF	(reserved-16)
64-255	256 -1023	100 -3FF	User Interrupt Vectors (192)

Autovectoring is not used except for power fail detect since interrupts from all other devices can be programmed to provide a vector number (which would likely point into the "User Interrupt Vector" area, above). Vectors 25 through 30 may be used as user interrupt vectors, if desired.

The table on the following page gives suggested interrupt vectors for each of the possible device interrupts which could occur. Note that the listing is in order of interrupt priority, highest priority first.

Int Lvl	Suggested Vector #	Device	Condition		
7	31	PFIN	Power Fail Interrupt (Must be vector 31)		
6	30	MMU	Write violation, Segment access, Undefined segment access		
5	96	CIO	Timer 3		
	79	CIO	XINT0	SBX module interupt	ref: section 11.1
	77		XINT1	SBX modle interrupt	
	75		ARDY	Archive Streamer Tape Ready, et al.	
	73		AEXC	Archive Streamer Tape Exception, et al.	
	71		SA-IO	SASI Winchester I/O, et al.	
	69		SA-CD	SASI Winchester C/D, et al.	
	67		WINTR	Winchester I/F	
	65		DOG	Watchdog Timer	
	98	CIO	Timer 2		
	78	CIO	INT7	Multibus Interrupt 7	ref: section 11.2
	76		INT6	" " 6	
	74		INT5	" " 5	
	72		INT4	" " 4	
	70		INT3	" " 3	
	68		INT2	" " 2	
	66		INT1	" " 1	
	64		INT0	" " 0	
	100	CIO	Timer 1		
	102		Timer, Error		
4	104	DMAC	Channel 0, Normal vector		Note: DMAC Channel priorities may be specified when programming the DMAC.
	105		Channel 0, Error vector		
	106		Channel 1, Normal vector		
	107		Channel 1, Error vector		
	108		Channel 2, Normal vector		
	109		Channel 2, Error vector		
	110		Channel 3, Normal vector		
	111		Channel 3, Error vector		
3	92	SCC	Port A, Receive character available		
	94		Port A, Special receive condition		
	88		Port A, Transmit buffer empty		
	90		Port A, External/Status change		
	84		Port B, Receive character available		
	86		Port B, Special receive condition		
	80		Port B, Transmit buffer empty		
	82		Port B, External/Status change		
	93		Port C, Receive character available		
	95		Port C, Special receive condition		
	89		Port C, Transmit buffer empty		
	91		Port C, External/Status change		
	85		Port D, Receive character available		
	87		Port D, Special receive condition		
	81		Port D, Transmit buffer empty		
	83		Port D, External/Status change		

The suggested interrupt vectors for the CIO and SCC devices take into account that the lower bit and upper four bits of the vectors are shared, e.g., all CIO Port A vectors have five bits which are the same for all interrupt causes.

Each device contains interrupt enable and control bits which allow the actual interrupt priority levels to be modified under program control by temporarily disabling certain devices.

Of course, fewer vectors may be used if the devices are programmed not to use modified vectors or if interrupts from some devices are not enabled.

If you use the suggested vector numbers in the above table, the proper values to load into the SCC and CIO vector registers are:

SCC0 (Ports A & B):	50 (hex)
SCC1 (Ports C & D):	51 (hex)
CIO, Port A:	41 (hex)
CIO, Port B:	40 (hex)
CIO, C/T vector:	60 (hex)

Making your way through the Zilog CIO and SCC manuals in search of details on the interrupt logic is quite an experience. To give you a head start, begin your quest with these recommended readings from the appendices to this manual:

Device	Item
SCC	Z8530 Technical Manual, appendix D
	Port priorities: App D, section 3.2.2, table 3-5
	Vector register: App D, section 4.1.3
	Vectors: App D, section 4.1.10, table 4-3
CIO	Z8536 Technical Manual, appendix E
	Vector register: App E, section 2.10.1
	Bit priorities: App E, section 3.3.2

4.3 Status LED's

(located near P6 edge)

There are four status LED's which give a visual indication of the MPU/DMAC status. These LED's continuosly show the state of the board as follows:

LED	Name	Meaning
S	Supr	The MPU is in the Supervisor state
U	User	The MPU is in the User state
D	DMAC	The DMAC or Multibus has control of the facilities
H	Halt	The MPU has halted (double bus fault, odd stack address or the system reset line is active) See section 8.1.

4.4 Instruction Set Summary

(See next page.)

For information on instruction execution speeds, refer to appendix A and section 7.6.

INSTRUCTION SET OVERVIEW

The MC68000 instruction set is shown in Table 10. Some additional instructions are variations, or subsets, of these and they appear in Table 11. Special emphasis has been given to the instruction set's support of structured high-level languages to facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words and most instructions can use any of the 14 addressing modes. Combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned multiply and divide, "quick" arithmetic operations, BCD arithmetic and expanded operations (through traps).

TABLE 10 — INSTRUCTION SET

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
B$_{CC}$	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DB$_{CC}$	Test Condition, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
S$_{CC}$	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

TABLE 11 — VARIATIONS OF INSTRUCTION TYPES

Instruction Type	Variation	Description
ADD	**ADD**	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	**AND**	Logical And
	ANDI	And Immediate
CMP	**CMP**	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	**EOR**	Exclusive Or
	EORI	Exclusive Or Immediate
MOVE	**MOVE**	Move
	MOVEA	Move Address
	MOVEQ	Move Quick
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	**NEG**	Negate
	NEGX	Negate with Extend
OR	**OR**	Logical Or
	ORI	Or Immediate
SUB	**SUB**	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

5.0 DMAC USAGE

The DMA Controller chip will support four separate channels of data transfer. Each channel may be initialized by the MPU to monitor different I/O devices and transfer data to or from memory when the particular device is ready. This allows concurrent operations with multiple I/O devices.

For readers who are not familiar with the workings of a DMA, there is an excellent paper describing some of the 68450 features in section 5.7.

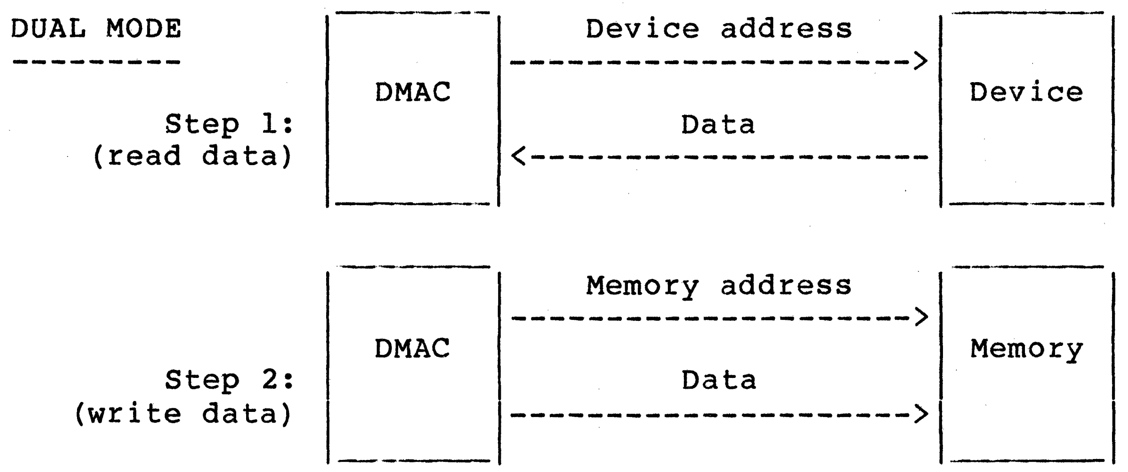
There are numerous modes with which the DMAC can be programmed to transfer the data, depending on the device speed, data configuration, etc. For example, a channel may be programmed to transfer a block of data without releasing the on-card bus between cycles, until the entire block has been transferred. Also, there are two addressing modes which may be selected. The choice is dictated by the hardware configuration of the HK68 board.

The DMAC data transfers may be either 8 or 16 bits per cycle. Memory-to-memory transfers could be either, depending on whether you want to move a byte or a word value. I/O-to-memory transfers would usually be eight bits because the devices only produce (or use) eight bits at a time. If there was a 16-bit I/O device, e.g., on the Multibus, then a 16-bit transfer would be OK. The DMAC has four channels. Each channel is independent of the other, and may be progammed as desired for 8 or 16 bit transfers.

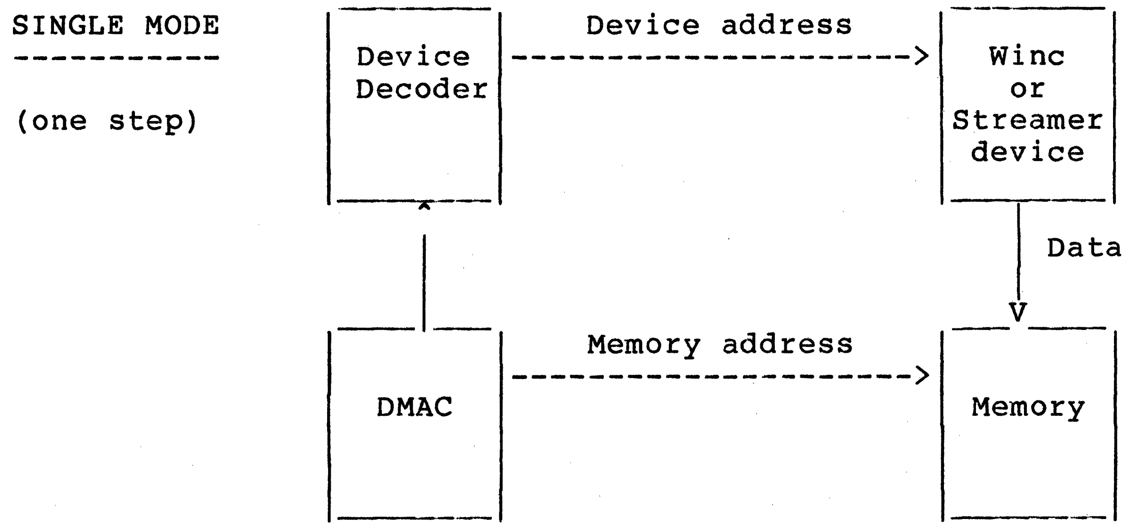
The two addressing modes are detailed below:

Mode	Description (refer to diagrams, below)
Dual	The DMAC does TWO memory cycles for each byte or word transfer. First, the DMAC puts out the address of the source byte (or word) and reads the data, loading it into a temporary (internal) holding register. Then, the DMAC turns around and puts out the destination address along with the data it had read in the previous cycle. The DMAC acts as a buffer for the data between the read and write operation. Note that the whole process requires the DMAC to consume TWO bus cycles on the HK68 because the read and write addresses are different values.
Single	The DMAC is sneaky. In cooperation with some special external logic on the HK68, the DMAC is able to transfer a byte or word data value in only ONE cycle. This is done by the external hardware ASSUMING that either the source (or the destination) address is fixed, and it is a particular device being addressed. On the HK68 we assume that DMAC channel 0 will be accessing the Winchester port, and that channel 1 will be talking to the Streamer Tape port. The external hardware can tell if a DMAC cycle is in progress as well as which channel is active. Then, the DMAC provides the memory address for the data while the external hardware automatically "tweaks" the assumed device to get (or store) the data. The data has a direct path between the device and memory. The DMAC does not actually receive the data.

The single adddressing mode is capable of transferring data at double the rate of the dual addressing mode. However, only certain devices (the Winchester or Steamer Tape ports) may be addressed in the single mode due to the need for special hardware decoding of the device enable signal. Also, since the Winchester and Tape ports are only eight bits wide and are associated with either the upper or the lower half of the data bus, the data organization in memory requires skipping every other byte. For example, 512 bytes of data read from the Winchester port using the single addressing mode will require a 1024 byte RAM buffer. Only the low half of each memory word would be significant.



- -



Any port may be addressed in dual mode by any channel, while the Winchester and Streamer ports may optionally be used in single mode by channels 0 and 1. For those devices, it's up to the programmer as to which mode to use. Jumper J11 must be set according to the selected mode.

The data transfer rate is controlled by the I/O device with an upper limit determined by bus arbitration and memory delays. See section 5.2.

The DMAC outputs one of eight function codes during each cycle to allow the MMU to identify the proper memory segment. Function code 7 is unique in that no RAM parity checking is performed when that function code is used. If the parity logic is enabled (via jumper J12) use of function code 7 will allow a partially initialized block of RAM to be transferred without an error. See section 5.1 for a complete list of function codes.

Refer to appendix B for additional information on the DMAC.

5.1 DMAC Channel Assignments

Channel	Use	Addressing Mode	Jumpers
0	Winchester (P4)	Single	Install J11
		Dual	Remove J11
1	Streamer Tape (P3)	Single	J9-1,2 & 6,7 & J11
	" "	Dual	J9-1,2 & 6,7 & no J11
	SCC Port A	Dual	J9-3,7
	SBX P7	Dual	J9-7,8
2	SCC Port A	Dual	J9-3,4
	SBX P7	Dual	J9-4,8
	SBX P8	Dual	J9-4,5
3	SBX P7	Dual	J9-8,9
	SBX P8	Dual	J9-5,9
	SCC Port B	Dual	J9-9,10

All channels may be programmed for memory-to-memory data transfers using the dual address mode.

Note that channel 0 is dedicated to the Winchester port because the "ready" signal for that channel is wired directly to the Winchester I/F logic. Of course, channel 0 could also be used for memory-to-memory transfers.

5.2 Data Transfer Rates

Since the DMAC and MPU must contend for the use of the on-card bus, the data transfer speed is dependent on the state of the MPU and DMAC when the device issues it's ready signal. The DMAC must request the bus from the MPU and be synchronized with the device and memory. These factors influence the DMA timing:

1. On-card Bus Arbitration Delay. When the DMAC requests the bus from the MPU, the MPU completes the current memory cycle and releases the control signals.

2. MMU Translation Delay. The MMU, if present, translates the logical device or memory address from the DMAC into the physical memory address.

3. Multibus Arbitration Delay. If the physical memory address is off-card, and if the Multibus is not already owned by this board, there is a delay associated with the acquisition of the Multibus.

4. Memory or Device Access Delay. Some finite time is required to perform the read or write operation at the target address. The DMAC is synchronized by the on-card DTACK signal in the same manner as the MPU.

5. Parity Checking Time. If the RAM parity logic is enabled, the parity of each byte must be computed before the DMAC is allowed to read data from on-card RAM.

6. Minimum DMAC Cycle Time. The DMAC requires four or five cycles for a transfer, depending on the transfer direction.

Refer to the DMAC technical literature in appendix B for additional information. Also, see section 7.6 on "Memory Timing."

5.3 Register Summary (DMAC)

The internal DMAC registers consist of a General Control Register (one for the whole DMAC) and four groups of 16 registers (one group per channel.)

Register Name		Chnl 0	Chnl 1	Chnl 2	Chnl 3	Function
		Register Address				
CSR		FE9000	FE9040	FE9080	FE90C0	Channel Status Reg
CER		FE9001	FE9041	FE9081	FE90C1	Channel Error Reg
DCR		FE9004	FE9044	FE9084	FE90C4	Device Control Reg
OCR		FE9005	FE9045	FE9085	FE90C5	Operation Control Reg
SCR		FE9006	FE9046	FE9086	FE90C6	Sequence Control Reg
CCR		FE9007	FE9047	FE9087	FE90C7	Channel Control Reg
MTC	16	FE900A	FE904A	FE908A	FE90CA	Memory Transfer Counter
MAR	32	FE900C	FE904C	FE908C	FE90CC	Memory Address Register
DAR	32	FE9014	FE9054	FE9094	FE90D4	Device Address Reg
BTC	16	FE901A	FE905A	FE909A	FE90DA	Base Transfer Counter
BAR	32	FE901C	FE905C	FE909C	FE90DC	Base Address Reg
NIV		FE9025	FE9065	FE90A5	FE90E5	Normal Interrupt Vector
EIV		FE9027	FE9067	FE90A7	FE90E7	Error Interrupt Vector
CPR		FE902D	FE906D	FE90AD	FE90ED	Channel Priority Reg
MFC		FE9029	FE9069	FE90A9	FE90E9	Memory Function Codes
DFC		FE9031	FE9071	FE90B1	FE90F1	Device Function Codes
BFC		FE9039	FE9079	FE90B9	FE90F9	Base Function Codes
GCR			FE90FF (Global)			General Control Reg

16 means 16-bit register (bytes are individually addressable)
32 means 32-bit register " " " "
others are 8-bit registers

5.4 Software Example

These are the DMA test routines. They set up DMA channel 2 to do a memory-to-memory transfer and interrupt upon completion. The interrupt service routine sets up the channel again and the process repeats. Once started, the program returns to the monitor while the interrupts keep the transfer going.

There are two sections to this example: the 'C' code portion and the machine code portion. First, the 'C' code:

```
/*
 *      dma.c
 *      DMA test routine
 */

#define DMABASE 0xFE9040        /* Base address of channel 2 of DMAC */
#define LEDS    0xFEC001

#define VECTOR  0x40
#define ERRVECT 0x41

#define out(port, data) *(char *) (port) = data
#define in(port) *(char *) (port)

main()
{
    initdma();
    initvector();
    dotransfer();
}

/*
 *      dotransfer()
 *      Transfers 0x1000 bytes from memory at 0x020000 to memory at 0x030000.
 */

dotransfer()
{
    dodma(0x020000, 0x030000, 0x1000);
}

/*
 *      initvector()
 *      Initializes the memory locations in the interrupt vector table to
 *      point to the interrupt service routines.
 */

initvector()
{
    int normint(), errint();

    *(int (**)()) (VECTOR << 2) = normint;
    *(int (**)()) (ERRVECT << 2) = errint;
}
```

```
/*
 *      initdma()
 *      Initializes the DMAC for memory-to-memory transfers in auto request
 *      at limited rate mode.
 */

initdma()
{
    register int index;
    static unsigned char dmatable[] = {
        0x04,   0x88,                   /* External request mode irrelevant
                                           since auto requests used, 68000
                                           device type (memory), 16 bit
                                           port, PCL undefined (not used). */

        0x05,   0x10,                   /* Direction memory to device (since
                                           both are essentialy the same the
                                           direction need never change like
                                           with Winchester DMA), Word
                                           transfers (word to word DMA), no
                                           chaining, auto request at rate
                                           limited by the GCR (allowing the
                                           MPU to run at the same time).   */

        0x06,   0x05,                   /* Both addresses increment.       */

        0x2D,   0x00,                   /* Channel priority is zero since
                                           not urgent task.                */

        0x25,   VECTOR,                 /* Normal interrupt vector. Upon
                                           completion of the DMA cycle the
                                           DMA will interrupt using this
                                           vector.                         */

        0x27,   ERRVECT,                /* Error vector.                   */

        0x29,   0x00,                   /* Memory function code.           */

        0x31,   0x00,                   /* Device function code.           */

        0xFF,   0x0F,                   /* The general control register. This
                                           value will generate long bursts
                                           with long spaces in between.    */

        0x00
    };

    for (index = 0; dmatable[index] != 0x00; index += 2)
        out(DMABASE + dmatable[index], dmatable[index + 1]);
}
```

```
#define STATUS      DMABASE             /* Status register                */
#define ERR_REG     DMABASE + 0x01      /* Error register                 */
#define COMMAND     DMABASE + 0x07      /* Command register               */
#define T_COUNT     DMABASE + 0x0A      /* Transfer count register        */
#define DEV_ADDRS   DMABASE + 0x14      /* Device address register        */
#define MEM_ADDRS   DMABASE + 0x0C      /* Memory address register        */

/*
 *      dodma(source, dest, length)
 *      Uses the DMA to transfer "length" bytes from "source" to "dest".
 */


static char ledcnt = ~0;

dodma(source, dest, length)
unsigned short int *source, *dest;
int length;
{
    out(STATUS, 0xFF);
    *(unsigned short int **) (MEM_ADDRS) = source;
    *(unsigned short int **) (DEV_ADDRS) = dest;
    *(unsigned short int *) (T_COUNT) = (length >> 1);
    out(COMMAND, 0x88);
    out(LEDS, ledcnt--);
}

/*
 *      error()
 *      Invoked by any error interrupt from the DMAC. Outputs the error
 *      register on the led port.
 */

error()
{
    out(LEDS, in(ERR_REG));
}
```

This is the machine code portion of the DMA sample software:

```
                .text
                .even

_normint:       movem.l         d0-d7/a0-a6, -(a7)
                jsr             _dotransf
                movem.l         (a7)+, d0-d7/a0-a6
                rte

_errint:        movem.l         d0-d7/a0-a6, -(a7)
                jsr             _error
                movem.l         (a7)+, d0-d7/a0-a6
                rte

                .globl          _normint
                .globl          _errint
```

5.5 Relevant Jumpers (DMAC)

Jumper	Function	Pos	Notes
J9	DMA Request/Ack	J9-1,2	DMA Ack1 to Tape Acknowledge
		J9-3,4	SIO A Ready to DMA Req 2
		J9-3,7	SIO A Ready to DMA Req 1
	See section 5.1	J9-4,5	SBX P8 Ready to DMA Req 2
	for more about J9	J9-4,8	SBX P7 Ready to DMA Req 2
		J9-5,9	SBX P8 Ready to DMA Req 3
		J9-6,7	Tape Ready to DMA Req 1
		J9-7,8	SBX P7 Ready to DMA Req 1
		J9-8,9	SBX P7 Ready to DMA Req 3
		J9-9,10	SIO B Ready to DMA Req 3
J11	Winc/Tape DMA Mode	Install for Single Address transfers to the Winchester or Tape ports. Remove for Dual Address transfers	

5.6 DMAC Bypass

The HK68 will operate without the DMAC chip. If it is removed from the board, the following jumpers must be installed in its place.

Connect U50 pin 25 to 26
Install J9-1,2

If the Watchdog Timer is enabled (jumper J25 removed) the software can determine if the DMAC chip is installed. Any attempt to access a non-existant DMAC will result in a Watchdog timeout and a Bus Error Exception. See section 8.1.

5.7 DMAC Application Info

The following paper was written by Thomas W. Cantrell, of Hitachi, for presentation at WESCON 1982. It makes for some very good reading about the DMAC chip. Reprinted with permission.

THE HD68450—A VERSATILE DMA CONTROLLER FOR HIGH PERFORMANCE SYSTEM DESIGN

Thomas W. Cantrell
Microprocessor and Peripheral Marketing
Hitachi America, Ltd.
1800 Bering Drive
San Jose, CA 95112

INTRODUCTION

High performance DMA (Direct Memory Access) capability is required for new, powerful systems including...

Multiuser, Multiprogramming General Purpose Computers.

Real Time Control Systems.

Robotics.

Data Communications.

High Speed Data Acquisition.

The HD68450 (figure 1) by combining innovative DMA protocols, a complete HD68000 bus interface and extremely high data transfer rates simplifies the design of high performance systems. Indeed, the integration of so much capability into one LSI device opens the door for many new product designs.

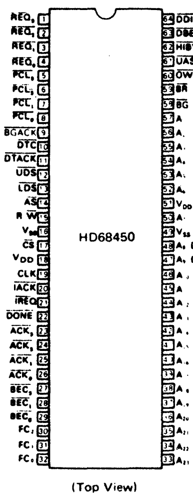


Fig. 1

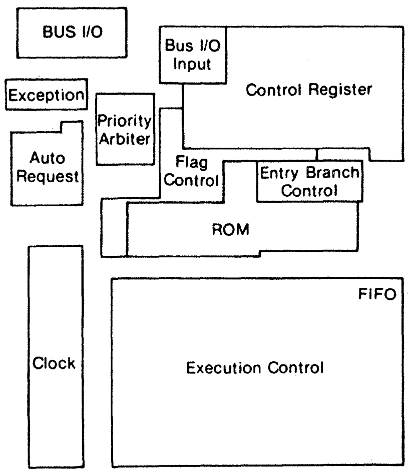


Fig. 2

The HD68450 is one of the first completely new complex peripherals for the HD68000 family. The DMAC contains over 40 thousand transistors (figure 2) implemented using the Hitachi 3 micron depletion load NMOS process technology. A 64 pin package is used. No compromise which might sacrifice performance or flexibility was considered.

WHAT IS A DMA CONTROLLER ?

Essentially, a Direct Memory Access Controller is a processing unit which is optimized for a special class of applications, namely that of moving data within and between system memory and peripheral devices. Peripheral devices which are often interfaced (figure 3) using DMA include disk storage (flexible and rigid), tape storage, video interfaces, CRTs, printers and other data comm devices (i.e. local area networks, IEEE-488, UARTs, etc.).

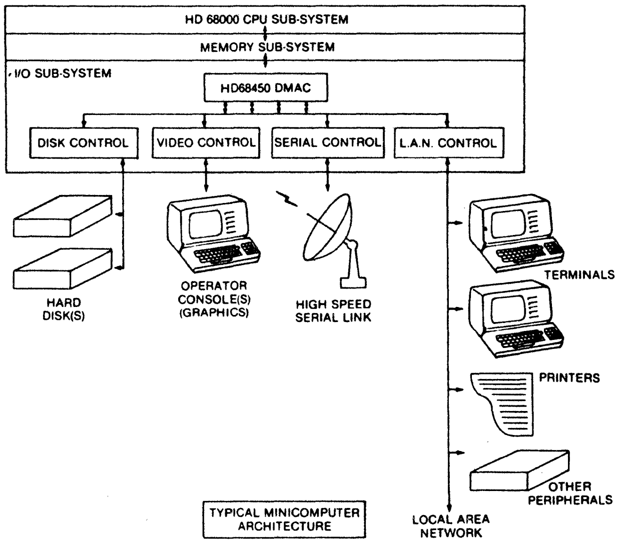


Fig. 3

Like a general purpose CPU (i.e. HD68000), a DMAC acquires use of the system bus, exerting control over the existing data, address and status/control lines.

Once in control, the DMAC in effect executes a short program which accomplishes the high speed data transfer. In this regard, there are two extremes to be considered.

In older DMAC designs, the data transfer 'program' is implemented in hardware. In this case, the main CPU is burdened with the task of initializing the DMAC with every parameter for each transfer.

At the other extreme, the DMAC is no different than a general purpose CPU. The DMAC is responsible for accessing and executing an external data transfer program as well as fetching and storing the data to be transferred. The problems with this approach include making the programmer responsible for all low level control of the data transfer, and significantly reduced performance due to the replacement of dedicated hardware with software.

The HD68450 lies between these extremes, incorporating a fairly high level of intelligence to reduce main CPU overhead, while retaining dedicated hardware to accomplish the data transfer at the highest possible speed.

By considering a DMA controller as a specialized CPU, evaluation of the device's characteristics can be performed using criteria familiar to many designers (figure 4). Utilizing this technique will show that the HD68450 DMAC exhibits the same flexibility, general

Criteria	Processor Type	
	DMA Controller	General Purpose CPU
DATA ACCESS CAPABILITY	Channel Organization DMA Protocols	Register Organization Addressing Modes
BUS INTERFACE	Address Space Memory Protection Exception Processing	Address Space Memory Protection Exception Processing
PERFORMANCE	Data Throughput DMA Latency	Instruction Cycle Interrupt Response

Fig. 4

purpose architecture, and very high performance that have made the HD68000 CPU the processor of choice for demanding applications.

FRAMEWORK FOR EVALUATION

The next level of detail for the above framework is shown in figure 5. In this figure, the relevant aspects of the HD68450 are compared with those for other popular DMA controllers. This figure also outlines the format for the remainder of the paper, i.e. the following topics will be examined in more detail.

Channel Organization

DMA Protocols/Address Space/Data Types

Bus Interface/Exception Handling

Performance

Clearly, the HD68450 is the most powerful, highest performance LSI DMAC currently available.

	HD68450	i8089	HD68B44	i8257	AMD9517A-2
PERFORMANCE					
Clock rate	8mhz	5mhz	2mhz	2mhz	5mhz
No. of channels	4	2	4	4	4
Address Space (bytes)	16M	1M	64K	64K	64K
Maximum single burst transfer (bytes)	256K	64K	64K	16K	64K
Data types	byte, word long word	byte, word	byte	byte	byte
Maximum transfer rate (M bytes/sec.)	4	1.2	2	0.5	1.6
DMA MODES					
Various including memory to memory I/O to memory memory to I/O auto-request external request	YES	YES	YES	YES	YES
Programmable bus bandwidth	YES	YES	NO	NO	NO
Address increment or decrement	YES	YES	YES	NO	YES
Round robin priority	YES	YES	YES	YES	YES
Programmable priority	YES	YES	NO	NO	NO
Hardware chaining	YES	NO	NO	NO	NO
SYSTEM BUS					
Bus matching	YES	YES	NO	NO	NO
Bus exception handling	YES	NO	NO	NO	NO
Cycle Retry	YES	NO	NO	NO	NO
Programmable vectored interrupts	YES	NO	NO	NO	NO
Memory protection	YES	YES	NO	NO	NO

A>

Fig. 5

CHANNEL ORGANIZATION

In order to best serve the needs of the broadest range of applications, the HD68450 incorporates four completely independent channels into one package. The HD68450's programming model is shown in figure 6. Like the HD68000, the HD68450 does not cut corners when it comes to register resources to get the job done...over 100 bytes of registers are used. A brief discussion of the purpose of and method of use for each register follows...

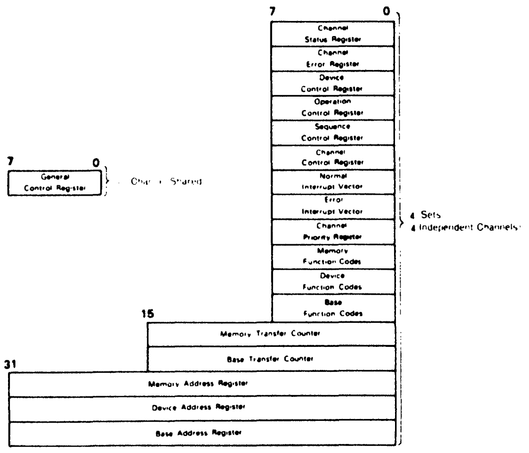


Fig. 6

General Control Register

This is the only register that is shared by all four channels. It is responsible for controlling one of the major innovative factors of the HD68450, namely the device's capability for programmable bus bandwidth utilization.

Consider a large, multiuser minicomputer implemented using the HD68000 and multiple HD68450 DMACs. Such a system will typically have a large number of I/O devices, including high priority devices, particularly rigid disk storage, and low priority devices, like terminals and printers.

A key consideration in a sophisticated computer involves the operating system's scheduling of programs and management of I/O resources. This task is complicated by the fact that the execution mix of the job stream will typically vary between being I/O bound and compute bound.

It is well known that even powerful mainframes can 'choke' if the software and hardware are not correctly 'tuned' to accommodate the dynamic changes in the computing environment. Common symptoms include poor terminal response time, I/O thrashing, excessive execution delays for small, compute bound tasks, increased response latency to asynchronous real time events, etc.

This is a difficult problem to solve. One common solution is to force users to predefine the resources they expect their job to use. The scheduler then attempts a priori to balance the job stream it initiates, This solution does require dedicated hardware (to monitor the computing environment in real time) and software (the scheduler).

Unfortunately, this constraint can negatively impact users. The ability to dynamically allocate memory and I/O under program control is often eliminated, and user programs may become data (size, device used, etc.) dependent.

In order to eliminate much of the complexity typically required to resolve the above issues, the HD68450 provides the on chip capability to determine the amount of DMA activity in the system and automatically adjust the rate at which it makes requests for the system bus. This prevents I/O devices from 'hogging' the bus and degrading main CPU performance. The OS scheduling algorithms can dynamically allocate more or less bus bandwidth to individual DMACs and rearrange individual channel priorities as required to maximize system performance (figure 7).

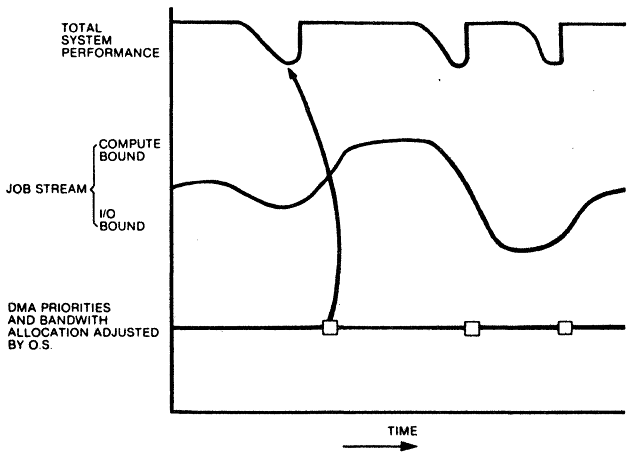


Fig. 7

The General Control Register defines the portion of bus bandwidth the DMAC can use. Sophisticated circuitry within the DMAC automatically monitors DMA activity in the system and dynamically moderates its own requests for the bus.

The unique programmable bus bandwidth utilization capability of the HD68450 can be used to provide minicomputer performance in low cost, LSI based systems.

<u>Memory Address, Device Address and Transfer Counter Registers</u>

These define source, destination and length of the DMA transfer. Note that like the HD68000, provision for a 32 bit implementation has been made (32 bit address registers). A single DMA burst can transfer up to 256K bytes (64K operands x long word (4 byte) operands).

<u>Channel Status, Error, Control and Priority Registers</u>

These registers are used for the basic operation of each of the four channels contained in the HD68450.

The Status Register allows the CPU to interrogate the current state of a DMA operation. Indicators include operation complete, device termination normal, channel active, channel error, etc.

The Error Register indicates the cause of an error. These include external errors such as a hardware bus error or external DMA abort and also internal errors such as an attempt to incorrectly configure DMAC operation. The HD68450 provides much useful information to the system programmer about the integrity of the system software and hardware.

The Channel Control Register is used to start, stop, continue and abort a DMA operation. Also, the CCR can enable or disable interrupts generated by the HD68450 programmable interrupt generation logic.

The Channel Priority Register allows the individual setting of each channel's priority (0-3). On chip priority arbitration logic automatically manages a 'round-robin' service mechanism for channels of equal priority.

<u>Device Control Register</u>

The Device Control Register defines the nature of the peripheral(s) associated with a channel. The HD68450 is quite flexible in this area, supporting both 8 and 16 bit peripheral chips (figure 8). Devices which use a READY line are directly provided for. Also, a 'peripheral control line' (PCL) on the DMAC can be programmably defined to be a status, status with interrupt on transition and abort input, or a start I/O pulse output.

<u>Channel Operation and Sequence Control Registers</u>

These registers define the basic nature of the DMA operation. The direction of the transfer, size of the operands, and sequence of addressing are programmably controlled using these registers. The HD68450 can manipulate 8 bit, 16 bit or 32 bit operands. The DMAC will automatically perform bus matching, i.e. a 16 bit word can be fetched from memory and sent as two 8 bit transfers to an 8 bit peripheral. Also, both the memory and device addresses can be programmably defined to either increment, decrement or not count at all. The latter mode is useful for high speed initialization of memory or I/O devices.

The method of DMA request initiation can be specified to be auto request at maximum rate, auto request at limited rate (utilizing the programmable bus bandwidth feature), or external request. An innovative mode of auto requesting the first operand and following with external requests is also provided.

Finally, another new, unique and powerful capability of the HD68450 is defined. This is the ability to

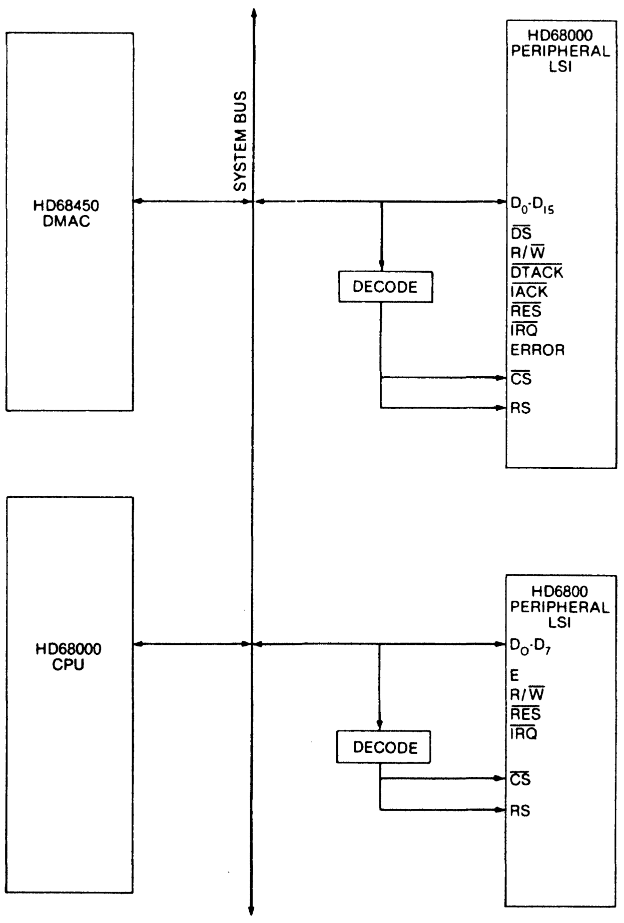


Fig. 8

specify that DMA operations utilize one of two 'chaining' modes, array or linked list.

Chaining

The array chaining mechanism (figure 9) works as follows. The DMAC automatically fetches the address and length of blocks to be transferred from an 'array' or 'table' in memory. After the operation defined by the first entry in the array is complete, the DMAC reloads the address and count registers from the second entry and so on. Only after all required entries in the table have been processed does the DMAC terminate the operation.

In a slightly different manner, linked list chaining processes command sequences, not in a linear array, but contained in a linked list format (figure 10). Similar to array chaining, address and transfer count information are contained in the list, but in addition, each entry is followed by a pointer to the next. The DMAC automatically fetches each succeeding entry based on the link pointer to the next.

The facilities for system design provided by the HD68450 chaining capabilities are quite significant. First, the amount of driver software required to control

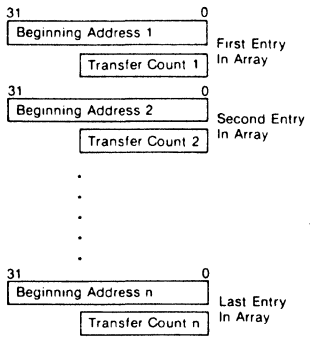


Fig. 9

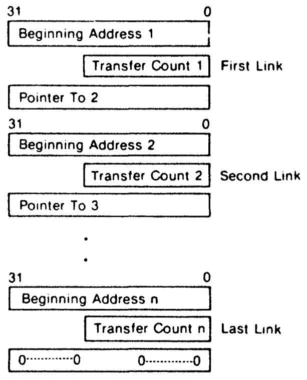


Fig. 10

DMAC operation is reduced. Also, since the main CPU isn't burdened with excessive 'handholding' of the DMAC, system performance is improved.

In fact, the implications of hardware DMA chaining go past the low level driver software, and can significantly reduce the cost and improve the performance of higher level system software if exploited.

An ideal example is the common text editor software required of all general purpose computers. Since a CRT screen is conceptually a 'contiguous' piece of memory (i.e. column 1 precedes column 2, line 5 precedes line 19, etc.), all text editors must be able to present text data in a contiguous manner. Some text editors actually maintain data in memory in a sequential manner. If the user inserts a line of text all other text is moved to accommodate the new entry. More sophisticated text editors may utilize linked list techniques, but the main CPU is burdened with presenting the linked list to the CRT in a sequential manner.

Now, utilizing the HD68450 linked list chaining mode, figure 11 shows how the design of a text editor can be simplified and performance enhanced. The text data can physically be stored in non-contiguous blocks. The text editor simply maintains a linked list defining

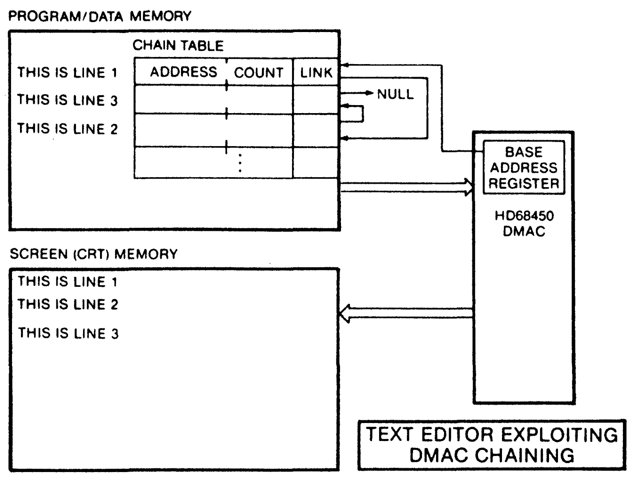


Fig.11

the proper sequential order and the HD68450 automatically 'gathers' the data properly for transmission to the CRT.

Another use for the chained DMA modes is what is commonly known as 'garbage collection.' Many OS scheduling and memory allocation systems are faced with the problem of memory fragmentation. This refers to a situation in which many blocks of free memory are scattered or fragmented (non-contiguous) within the memory map. The HD68450 can easily consolidate these small blocks into one larger contiguous memory space (figure 12).

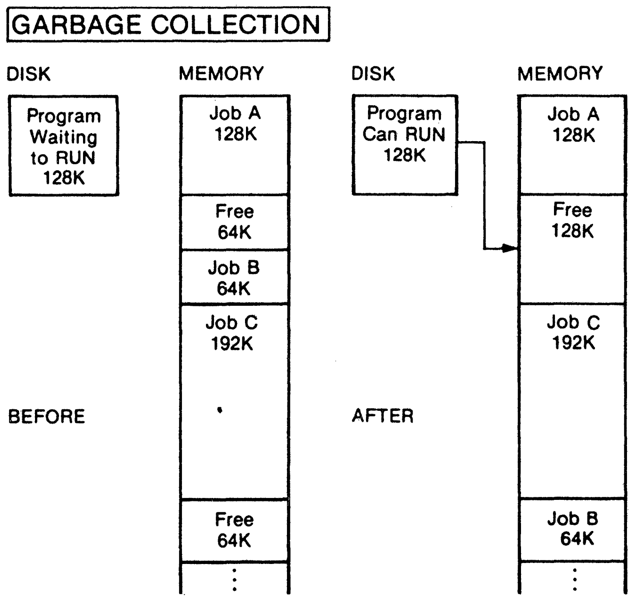


Fig. 12

Note also that the linked list can be circular in that the last entry can point to the first. This allows repetitive operations like CRT or dynamic memory refresh to occur continuously with no main CPU overhead.

In fact, a single entry with a link pointing to itself mimics the 'autoload' capability of more primitive DMACs. Note that most DMAC's autoload feature typically requires the dedicated use of two channels, one for the transfer and one for the autoload information.

Finally, chaining allows massive amounts of data to be moved in 256K byte bursts, with only a small latency (less than 5 microseconds) between bursts. This supports very large, high speed data acquisition (i.e. image, signal processing, etc.) applications better than more limited DMACs.

This type of hardware support for 'scatter/gather' operations has only been the luxury of large computers until now.

Normal and Error Interrupt Vector Registers

The HD68450 fully supports the powerful vectored interrupt mechanism of the HD68000. The programmer can define which interrupt vectors are associated with each channel and which interrupt vector to use when a DMA operation error occurs on any channel. An on chip priority interrupt controller resolves interrupt requests between channels of different or equal priorities.

Base Address and Transfer Counter Registers

Used with the chaining modes, the base address register points to the start of the array or linked list of commands. For array chaining, the base transfer counter determines the number of entries in the array to be processed.

Memory, Base and Device Function Code Registers

Larger HD68000 based systems will often implement the CPU's memory protection facilities. Fortunately, the HD68450 DMAC fully provides for operation in such systems. These registers allow each access to memory to be defined (USER or SUPERVISOR, etc.) exploiting the same function code (FCO-FC2) scheme used by the CPU. This allows complete flexibility in choosing the way in which the DMAC will operate in a protected environment.

For instance, one configuration might define the source of a DMA operation as USER memory, the destination as a SUPERVISOR peripheral. Another might define the memory and peripheral operands as USER, but restrict DMAC accesses to the base table (containing chaining sequences) to SUPERVISOR memory.

Upon investigation, most previous integrated CPU/DMAC architectures have significant shortfalls (if any provision at all!) for managing DMA in a protected environment.

DMA PROTOCOLS/ADDRESS SPACE/DATA TYPES

The HD68450 incorporates a very flexible DMA protocol selection. These include...

Memory to I/O, I/O to memory

Memory to memory

Software halt and continue mode

Array and linked list chaining

External DMA request

Auto request, maximum rate

Auto request, limited rate

Cycle steal with bus hold or relinquish

Auto increment, decrement or no change of addresses

The flexibility of the DMAC use in a system is only limited by the imagination of the designer, not a lack in number or capabilities of the DMAC's resources.

This page has been intentionally left blank.

6.0 MEMORY MANAGEMENT UNIT (MMU)

This section explains some of the relevant features of the 68451 MMU chip. Refer to appendix C for more details.

The MMU automatically enters a "transparent" mode following a system reset. Thus, all logical addresses and physical address will be the same. The MMU must be programmed and enabled before any address translations will begin.

The MMU allows 32 memory segments to be defined. Each segment may be from 256 bytes to the full 16 megabyte space. Certain users may be given access to the entire memory space, while other users may be restricted to a smaller segment or be prohibited from writing to a segment.

6.1 Function Code Definitions

The table below shows the MPU and DMAC function codes which are generated for each memory reference. They indicate to the MMU the particular type of reference being made, and are used to index into the MMU Address Space Table (AST). Ultimately, the function codes determine the logical to physical mapping and the protection levels for the operation (e.g., write protect, user/supervisior space).

Hex	FC3	FC2	FC1	FC0		
0	0	0	0	0	\	
1	0	0	0	1	|	
2	0	0	1	0	|	
3	0	0	1	1	|	
4	0	1	0	0	> DMA Function Codes	
5	0	1	0	1	|	
6	0	1	1	0	|	
7	0	1	1	1	/ (no parity checking, code 7)	
8	1	0	0	0	(reserved)	\
9	1	0	0	1	User DATA	|
A	1	0	1	0	User PROGRAM	|
B	1	0	1	1	(reserved)	> MPU Function
C	1	1	0	0	(reserved)	| Codes
D	1	1	0	1	Supervisor, DATA	|
E	1	1	1	0	Supervisor, PROGRAM	/
F	1	1	1	1	Access from Multibus	

These code assignments do not correspond to the example suggested in the MMU technical manuals. (FC3 is inverted and interrupt acknowledge does not consume a code.) Disregard the MMU literature in this respect.

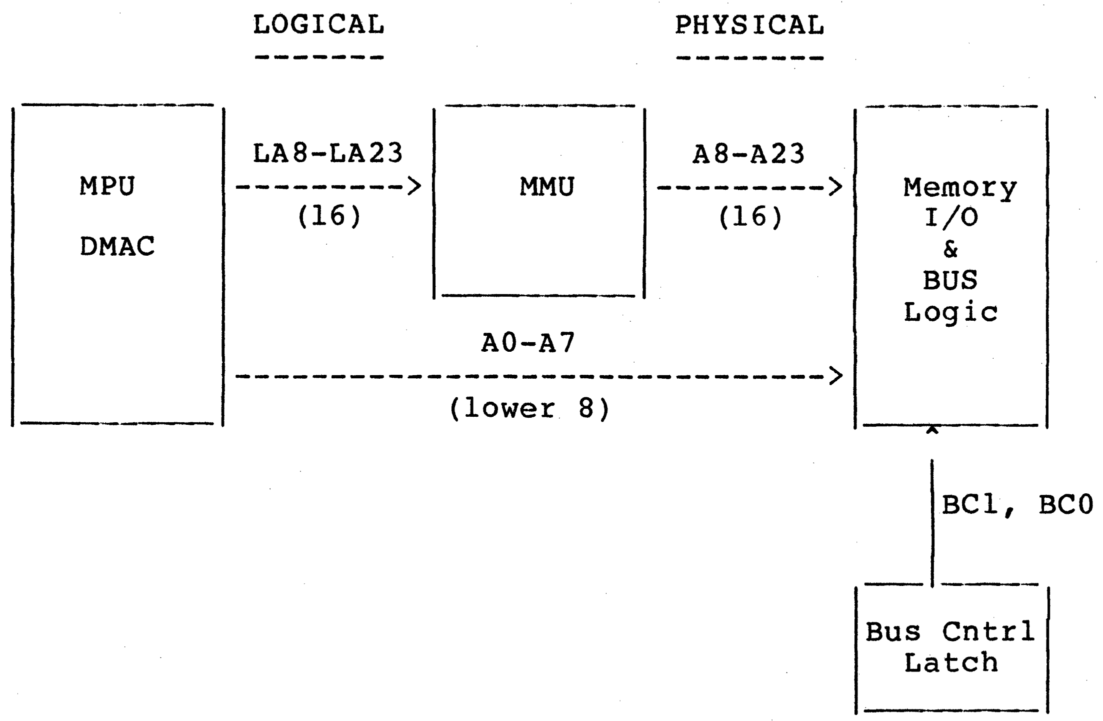
6.2 Address Line Coordination

Physical	MPU/DMAC	Access from bus	Access to bus
	mapped from		mapped to
A23	LA23	xx	ADR17
A22	LA22	xx	ADR16
A21	LA21	xx	ADR15
A20	LA20	xx	ADR14
		mapped from	
A19	LA19	ADR13	ADR13
A18	LA18	ADR12	ADR12
A17	LA17	ADR11	ADR11
A16	LA16	ADR10	ADR10
A15	LA15	ADRF	ADRF
A14	LA14	ADRE	ADRE
A13	LA13	ADRD	ADRD
A12	LA12	ADRC	ADRC
		direct from	
A11	LA11	ADRB	ADRB
A10	LA10	ADRA	ADRA
A9	LA9	ADR9	ADR9
A8	LA8	ADR8	ADR8
	direct from		direct to
A7	LA7	ADR7	ADR7
A6	LA6	ADR6	ADR6
A5	LA5	ADR5	ADR5
A4	LA4	ADR4	ADR4
A3	LA3	ADR3	ADR3
A2	LA2	ADR2	ADR2
A1	LA1	ADR1	ADR1
	derived from	derived from	generates
A0	UDS, LDS	ADR0 & CONV	ADR0
			via CONV

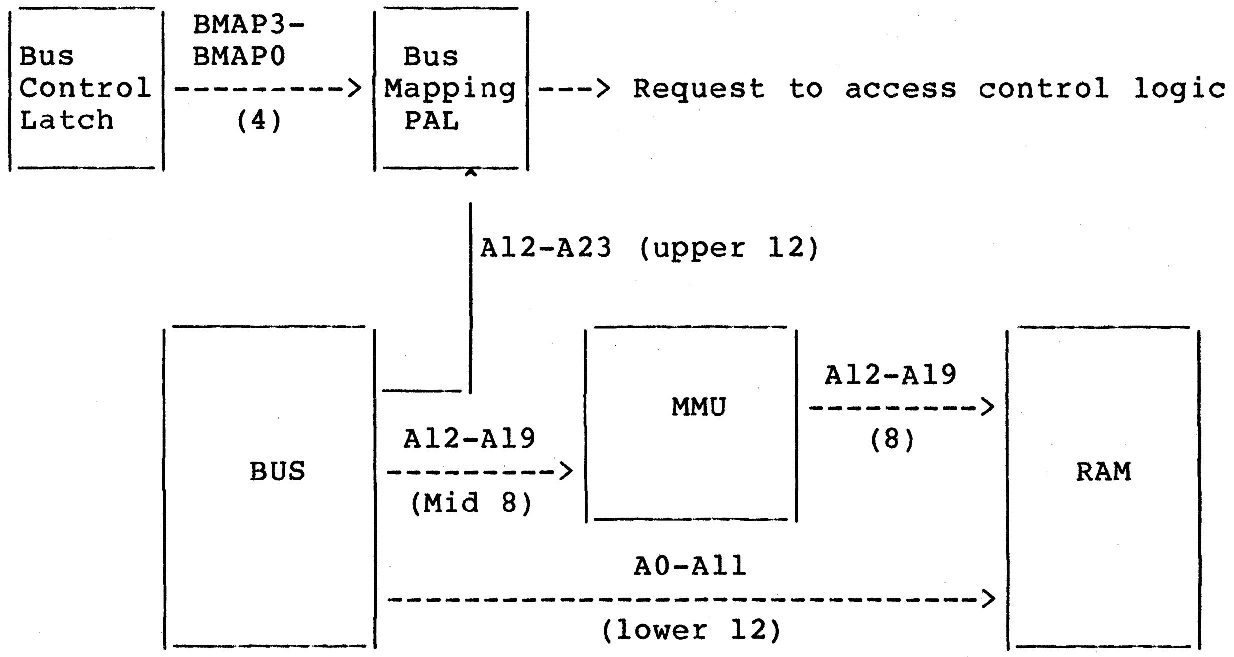
Key:	"Ax"	refers to the physical, on-card address signal
	"LAx"	refers to the logical address output from the MPU and DMAC
	"ADRx"	refers to the Multibus address signal (note that the numbering convention for the Multibus is hex, while the on-card signals use decimal.)
	"UDS"	is the MPU/DMAC upper data strobe signal
	"LDS"	is the MPU/DMAC lower data strobe signal
	"CONV"	refers to the data CONVENTION in use. See "Bus Data Conventions," section 10.8, for details.
	"xx"	means the input is undefined (treat as "zero")

6.3 MMU Address Line Block Diagram

MPU/DMAC Access to memory or to the bus



Memory access from the bus

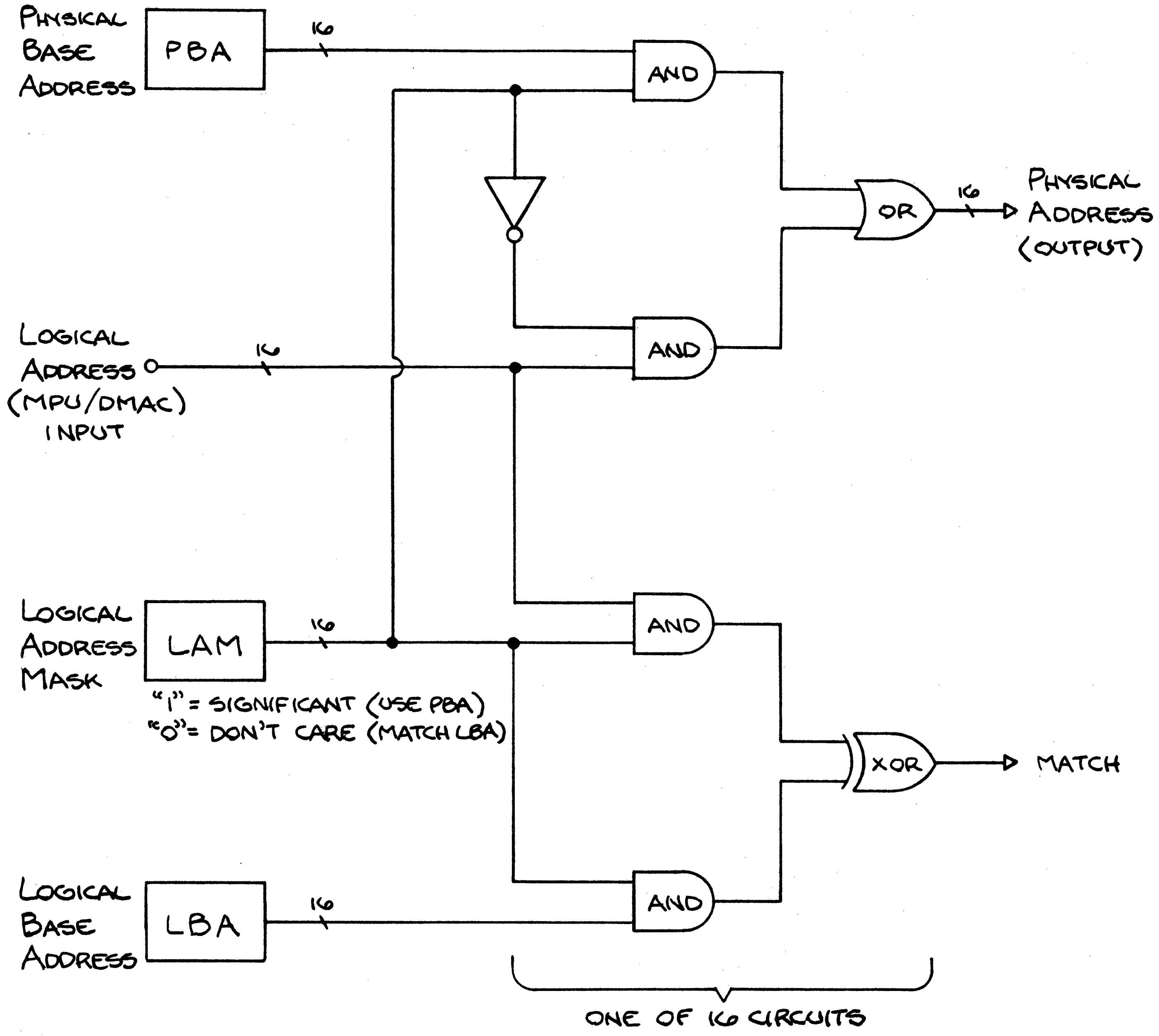


During accesses from the bus, the logical A20 through A23 inputs to the MMU are not defined. Therefore, the upper four bits of the MMU's "LAM" (Logical Address Mask) register should be programmed with zeros (don't cares).

6.4 Register Summary (MMU)

The MMU registers may be read or written by the MPU. Some addresses map into actual commands. 16-bit registers may be accessed as either a 16-bit word or as a pair of 8-bit bytes, i.e., each half of the word registers may be operated on independently.

Register	Address	Size (bits)	Function
AST0	FE8000	8	DMA code 0
AST1	FE8002	8	DMA code 1
AST2	FE8004	8	DMA code 2
AST3	FE8006	8	DMA code 3
AST4	FE8008	8	DMA code 4
AST5	FE800A	8	DMA code 5
AST6	FE800C	8	DMA code 6
AST7	FE800E	8	DMA code 7
AST8	FE8010	8	MPU, (reserved)
AST9	FE8012	8	MPU, User data
AST10	FE8014	8	MPU, User program
AST11	FE8016	8	MPU, (reserved)
AST12	FE8018	8	MPU, (reserved)
AST13	FE801A	8	MPU, Supervisor data
AST14	FE801C	8	MPU, Supervisor program
AST15	FE801E	8	Access from Multibus
AC0/AC1	FE8020/1	8/8	LBA & Translation Adrs
AC2/AC3	FE8022/3	8/8	LAM (Logical Adrs Mask)
AC4/AC5	FE8024/5	8/8	PBA & Translation Reg
AC6	FE8026	8	ASN (Adrs Space Nmbr)
AC7	FE8027	8	Segment Status
AC8	FE8028	8	ASM (Adrs Space Mask)
DP	FE8029	8	DP (Descriptor Pointer)
IVR	FE802B	8	IVR (Interrupt Vector Reg)
GSR	FE802D	8	GSR (Global Status Reg)
LSR	FE802F	8	LSR (Local Status Reg)
SSR	FE8031	8	SSR & Transfer Descrip Oprn
IDP	FE8039	8	IDP (Int Descrip Pointer)
RDP	FE803B	8	RDP (Result Descrip Pointer)
(command)	FE803D	n/a	Direct Translation Operation
(command)	FE803F	n/a	Load Descriptor Operation
	(other)	n/a	Null Operation


PHYSICAL BASE ADDRESS
PBA
16
AND
OR
PHYSICAL ADDRESS (OUTPUT)
LOGICAL ADDRESS (MPU/DMAC) INPUT
AND
LOGICAL ADDRESS MASK
LAM
"1" = SIGNIFICANT (USE PBA)
"0" = DON'T CARE (MATCH LBA)
AND
XOR
MATCH
LOGICAL BASE ADDRESS
LBA
AND
ONE OF 16 CIRCUITS

MMU REGISTER RELATIONSHIPS

6.5 Relevant Jumpers (MMU)

Jumper	Function	Pos	Notes
J17	MMU Mode Select	J17-A	Mode S1 (factory adjustment)
		J17-B	Mode S2

Do not fiddle with this jumper. See "Memory Timing," section 7.6, for information on MMU timing.

6.6 MMU Bypass

The HK68 will operate without an MMU chip. If the MMU is removed from the board, the following jumpers must be installed in its place.

```
        Chip U48        Connect pin 2 to 13
                        Connect pin 5 to 6 to 7
                        Connect pin 48 to 49
                           "     "  47 TO 50
                           "     "  46 TO 51
                           "     "  45 TO 52
                           "     "  44 TO 53
                           "     "  43 TO 54
                           "     "  42 TO 55

                           "     "  40 TO 57
                           "     "  39 TO 58
                           "     "  38 TO 59
                           "     "  37 TO 60
                           "     "  36 TO 61
                           "     "  35 TO 62
                           "     "  34 TO 63
                           "     "  33 TO 64
                           "     "  32 TO 1
```

If the Watchdog timer is enabled (jumper J25 removed) the software can determine if the MMU chip is installed. Any attempt to access a non-existant MMU will result in a Watchdog timeout and thus a Bus Error Exception. See section 8.1.

6.7 Sample MMU Software

```
#define MMU     0xFE8000
#define MMUACC  (MMU + 0x20)    /* Accum adrs offset */
#define MMULD   (MMU + 0x3F)    /* Load Descriptor */
#define MMUAST9 (MMU + 0x12)    /* Adrs Space Table 9 */

mmuinit()
{
        char i, j, k;
        int *p;
        char *q ={ MMULD };     /* Load Descriptor Command */

            /* The MMU Table contains groups of values for loading the
               descriptors.  These values map logical addresses from
               0x008800 through 0x0FFFFF to the same physical addresses. */

        static int mmutable[] ={
                0x0088,         /* LBA */       /* 00 8800 to 00 8FFF   2K */
                0xFFF8,         /* LAM */
                0x0088,         /* PBA */

                0x0090, 0xFFF0, 0x0090, /* LBA, LAM, PBA    00 9000 to
                                                            00 9FFF      4K */

                0x00A0, 0xFFE0, 0x00A0,                  /* 00 A000 to
                                                            00 BFFF      8K */

                0x00C0, 0xFFC0, 0x00C0,                  /* 00 C000 to
                                                            00 FFFF     16K */

                0x0100, 0xFF00, 0x0100,                  /* 01 0000 to
                                                            01 FFFF     64K */

                0x0200, 0xFE00, 0x0200,                  /* 02 0000 to
                                                            03 FFFF    128K */

                0x0400, 0xFC00, 0x0400,                  /* 04 0000 to
                                                            07 FFFF    256K */

                0x0800, 0xF800, 0x0800                   /* 08 0000 to
                                                            0F FFFF    512K */
        };
        for (k=0, i=0; i<sizeof(mmutable)/6; i++) {
                p = MMUACC;
                for (j=0; j<3; j++)
                        *p++ = mmutable[k++];    /* Load MMU Accum regs */
                *p++ = 0x0101;                   /* ASN, SSR */
                *p = 0xFF01 + i;                 /* ASM, DPn */
                if (*q != 0x00)                  /* Load despriptor */
                        return(ERROR);
        }
        q = MMUAST9;                    /* User AST9, must use pointer to char */
        *q = 0x01;                      /* Load AST9, User data */
        *(q+2) = 0x01;                  /* Load AST10, User pgm */
        return(0);
}
```

6.8 MMU Primer

This section explains the some of the basics of memory management.

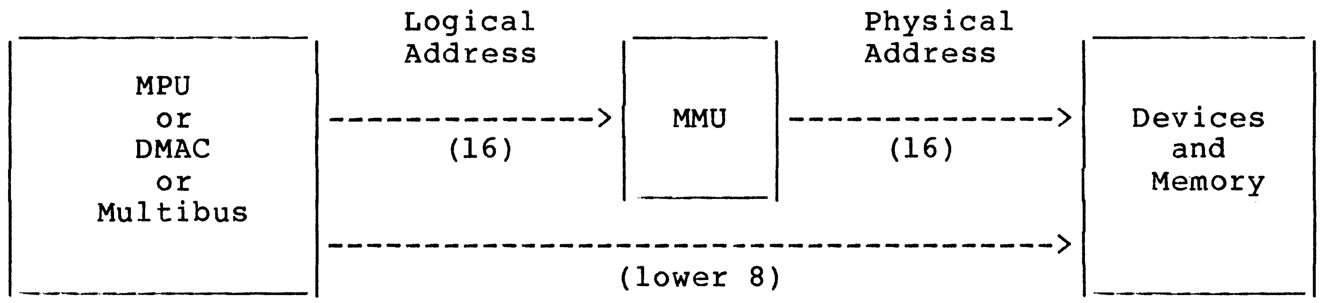
1. Logical vs. Physical Addresses:

   a. A "logical" address is the value used by a program. A "physical" address is the actual state required on the address lines to access the intended memory or device. Physical addresses are determined by the hardware, and usually cannot be changed except by modifying the circuitry. A logical address is where the program "thinks" the memory is; the physical address is the actual location.

      In a computer system which does not have an MMU, the logical and physical addresses for any location will always be the same. The programmer must determine in advance which address values are to be used to access, for example, a certain memory block.

      If an MMU is used, the logical addresses for a memory block need not be equal to the physical addresses, although they commonly are.

   b. On the HK68, all address values are run through the MMU, even for I/O device accesses. Actually, an I/O device access is identical to a memory access since all I/O is "memory mapped." The only difference is in the physical address values. We have reserved the low physical addresses for memory and the very top addresses for I/O devices. But, by using the MMU, a programmer may adjust the logical addresses for devices and memory; so that, for example, a particular memory block is in the top address space and the I/O devices at lower addresses.



   c. The logical to physical address conversion operates slightly differently depending on whether the HK68 access is being controlled by the MPU or some other board on the Multibus. First of all, the Multibus can only access on-card RAM. There is no direct method by which another CPU board can touch an on-card I/O device. Secondly, not as many address lines (only 8) are sent to the MMU for mapping when the access is from another board. Thus, the MMU maps in 256 byte blocks when the 68000 MPU or DMAC is in charge and in the larger 4K blocks when the bus is controlling the cycle.

d. Don't confuse the function of the MMU and the Bus Mapping PAL. They are completely independent devices. The Bus Map monitors the Multibus and detects whether or not a valid address (one which is intended for the HK68) is present on the bus address lines. The MMU converts incoming logical addresses to physical values. The Bus Map acts as the "doorman," while the MMU translates the addresses, once the door has been openned. The Bus Mapping PAL can be programmed in a variety of ways; however, it only monitors the upper 12 Multibus address lines. This means that the HK68 can occupy 4K bytes, minimum, of bus address space. By ignoring some of the upper address lines, the space occupied can be enlarged (up to the on-card RAM size, in space sizes which are increments of powers of two).

   Refer to section 10.3 for more details about accesses from the Multibus.

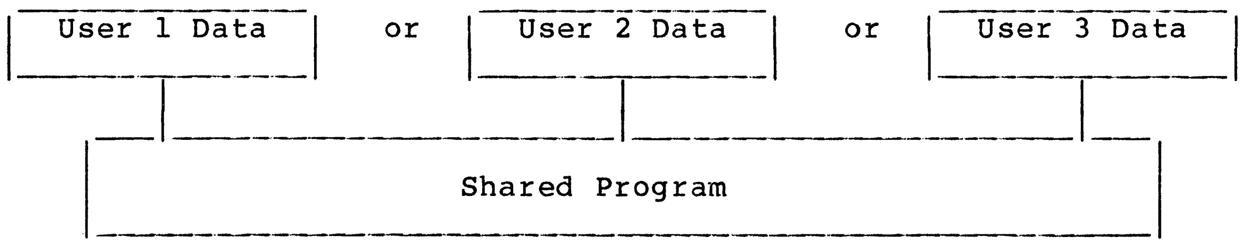
e. MMU implementations operate in different ways since there is, unfortunately, no standard method. On the HK68, the MMU allows the upper 16 bits (out of 24) to be adjusted according to certain rules, which must be followed by the programmer.

   The MMU operates by changing some of the upper bits of a (logical) address to another specified value, which becomes the physical address. The lower address lines get through unscathed. The number of bits to change and, of course, the new values are specified by the programmer and loaded into the MMU's internal registers.

2. Functions:

   a. Easily the most important function of the MMU is to provide memory access protection. The MMU can tell whether an access is generated by the MPU, the DMAC or the Multibus. Furthermore, for MPU accesses, the MMU can determine if the memory access is controlled by the "supervisor" program, which has certain privileges not afforded to other ("user") programs. The MMU can prevent a user program from writing (or even reading) from specified memory blocks or I/O device groups. This function is vital to a multiuser operating system, such as Unix, to prevent one user from clobberring another, or to protect data from unauthorized use or alteration.

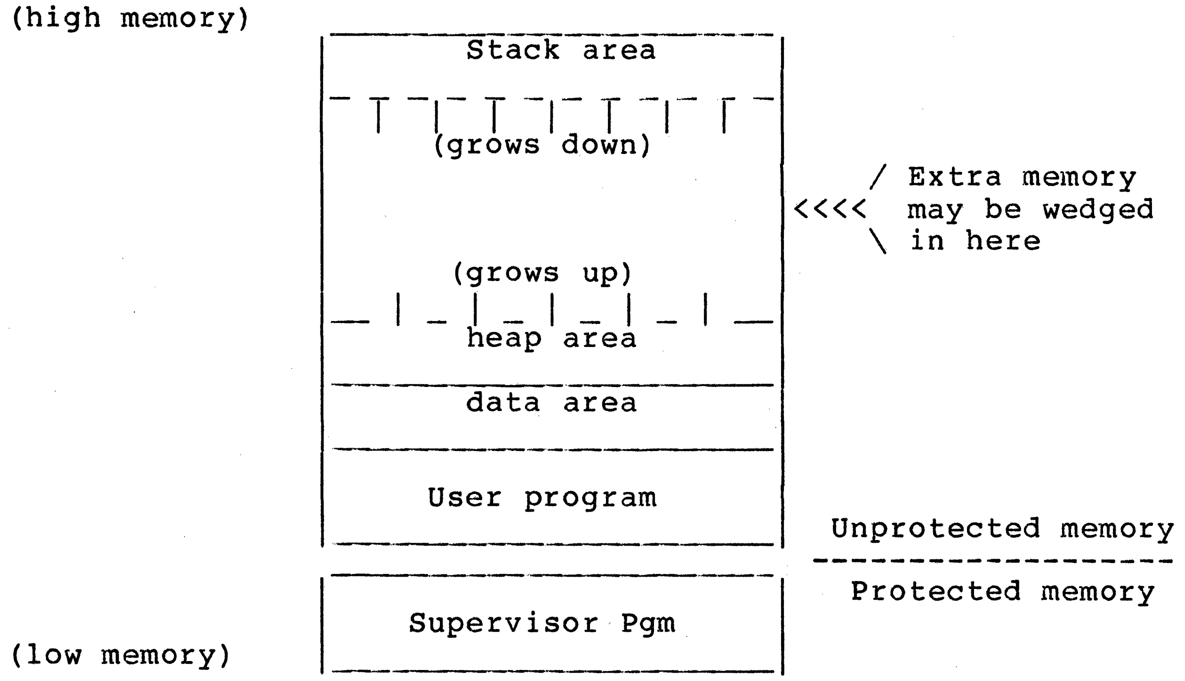
b. Another function performed by the MMU is to allow a Multi-user operating system some freedom in memory allocation. Two or more users could be executing the same program segment but have separate memory areas for storage of variables. The operating system is responsible for switching in the correct memory block, depending on which user is executing. The program doesn't have to worry about locating the specific RAM block belonging to the active user. In this example, each user would have a different physical memory block for variables; but, when one or the other block is being accessed by the program, the program would always find the active block at the same logical address. It is a simple procedure for the operating system to switch the blocks, since all it has to do is change a few registers inside the MMU. The result of all this is a big time savings for the MPU, since the MPU is able to effectively "move" a large block of memory without actually moving each byte.



c. Or, let's say you have some programs which, for some reason, must be at certain addresses when executing. Such programs are called "absolute" as opposed to "relocatable." If two users each want to run one of these programs, and if the programs should both demand the same memory area to run in, the operating system has a problem. It can either run one program and make the second user wait; or, it can swap back and forth between the two programs, running each one a little bit at a time, so both users are being serviced. But, swapping programs is very time consuming and leaves less time for the programs to actually execute. Through the use of the MMU logic, however, the operating system can load both programs into different physical memory segments, but yet allow them to execute at the logical address which each program desires. The swapping is accomplished very quickly "logically" instead of physically.

d. Use of an MMU also provides the ability to gracefully enlarge a memory segment without having to completely rearrange memory. Modern programs (those coming out of compliers) typically have a "text" segment which is the actual executable code, a "heap" or "data" segment which is used for variable storage (as discussed above) and a "stack" area for program return addresses and certain parameters which have a relatively temporary existance. The stack usually starts near the end of memory and "grows" toward the text segment, as necessary, during program execution. The operating system allocates a certain amount of space for the heap and stack when the program is loaded; but, the operating system really doesn't know if it has allocated enough space. Indeed, most programs themselves don't even know, prior to execution, precisely how much memory will be needed.

So, what happens if the stack grows down and is about to bump into the data or text segments? One solution would be to move all the contents of the stack up to higher memory and thus open up additional space between the stack and the rest of the program. With an MMU, all that needs to be done is to map in a new memory segment and LOGICALLY move the stack contents. The logical address of the stack is changed while leaving the physical addresses alone. No data movement is necessary; the new segment is "wedged" in, right between the existing memory areas.



e. Another memory management function, but one not required on the HK68, is to expand the address range of a CPU chip. For example, the Zilog Z-80 has only 16 address line outputs, which limits its addressing range to 65,536 bytes. In order to address more memory, a memory mapping device may be designed which uses three or four of the upper CPU address lines as inputs and provides a dozen output lines. Those outputs are then combined with the remaining CPU address lines to expand the total physical address lines to, for example, 24.

f. MMU Functional Summary:

1. Read and/or write protection of special memory areas to prevent unauthorized use or alteration of programs or data.
2. Relieve the MPU of doing a physical swap of programs and data in order to add memory within a data area.
3. Locate an absolute program at a particular execution address, even though that physical address is unavailable.
4. Address line expansion.

If you followed more than 80% of all that, you may consider yourself an MMU expert. A written certification examination will be available. Study up!

7.0 MEMORY CONFIGURATION

The Heurikon HK68 microcomputer will accommodate a variety of RAM and ROM configurations. There are two ROM sockets (one for the high byte, one for the low byte) and 18 RAM sockets (including two parity bits per word).

7.1 ROM

ROM occupies a fixed 32k byte physical address space starting at FE0000 (hex). At power-on, the ROM is also mapped into address 000000 in order to allow the MPU to fetch the reset exception vector and begin execution. Execution may proceed at base 000000 or FE0000. There will be no RAM until the MPU accesses any physical address at or above FE0000, at which time the ROM image based at 000000 will be turned off and RAM turned on. It is acceptable for the reset vector to point directly to an initialization routine in block FE0000, in which case the RAM will be turned on immediately after the vector has been fetched. (Caution: Accessing an I/O device after power-up will also turn off the lower ROM image.) ROM access time must be 435 nsec or less.

ROM type	Board Capacity	ROM type	Board Capacity
2716	4K bytes	2764	16K bytes
2732	8K bytes	27128	32K bytes

7.2 RAM

RAM must be turned on following power-up, as described above. On-card RAM starts at physical address 000000 and is contiguous through the end of RAM. Once the RAM has been turned on, it cannot be turned off; it may only be logically relocated via the MMU.

The HK68 can accommodate a "double-decker" RAM matrix, for a total of 36 chips, maximum. This is accomplished by using special IC sockets which allow two chips to be mounted in the same chip position, one above the other. This method allows a full megabyte of memory, plus parity, when using 256k x 1 parts. Jumper J23 selects the number of decks in use. If the J23 shunt is removed, on-card RAM will be disabled and all physical space below FE0000 will be assigned to the Multibus. See also "RAM Parity Logic," section 8.2, below.

RAM	Board Capacity (bytes)	
type	1 Deck	2 Decks
64K x 1	128K	256K
256K x 1	512K	1 Meg

7.3 Multibus

All physical addresses from the end of on-card RAM to the beginning of the ROM at FE0000 are assumed to be off-card, on the Multibus (see the memory map, below). Bus arbitration is automatic.

It is not possible to access the physical bus addresses occupied by the on-card RAM or any physical bus address above FE0000. However, in systems using multiple HK68 processors, each board can map its on-card RAM into different address spaces by use of the bus mapping PAL and the MMU logic. This will allow all processors access to each other's RAM. The bus cannot be used following a power-up until the MPU has accessed above FE0000. See "Bus Control," section 10, for more information concerning the Multibus.

7.4 Physical Memory Maps

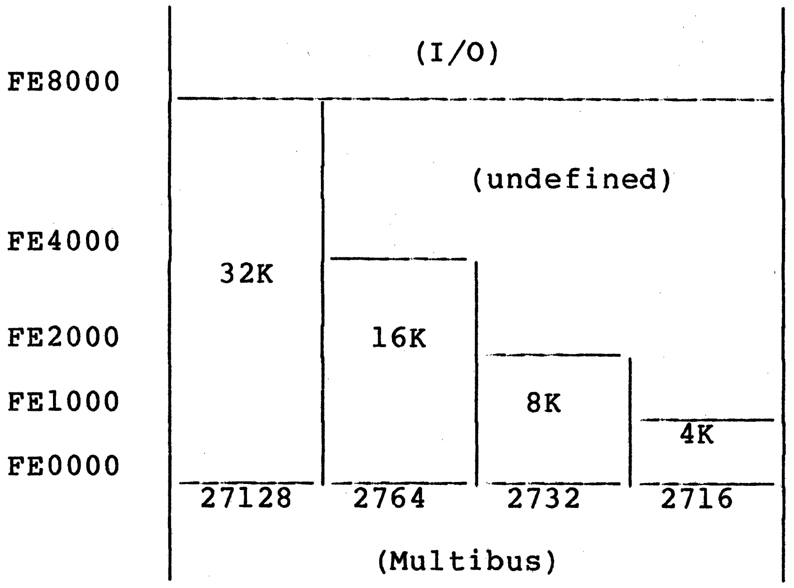
(The maps are not shown to scale)

```
                  Normal Use                          At Power-up
                  ----------                          -----------

     FFFFFF  -----------------           FFFFFF  -----------------   Any access
             |        ~       |                  |        ~       |  in this
             |      (I/O)     |                  |      (I/O)     |  region
             |        ~       |                  |        ~       |  will turn
     FE8000  -----------------                   -----------------   on the
             |      ROM       |                  |      ROM       |  "normal"
             |     (32k)      |                  |     (32k)      |  map.
     FE0000  -----------------           FE0000  --------------------------------
             |                |                  | (see sect 7.5) |
             |                |          FB0000  -----------------
             |    Multibus    |                  |                |
             |     Memory     |                  |                |
        ~    ~        ~       ~                  ~        ~       ~
        ~    ~        ~       ~                  ~        ~       ~
                                                 ~        ~       ~
             |                |                  |                |
     100000  ---------        |                  |                |
             | RAM    |       |                  |   (undefined)  |
             | 512k   |       |                  |                |
             | Deck2  |       |                  |                |
             |        |       |                  |                |
     080000  ---------        |                  |                |
             | RAM    |       |                  |                |
             | 512k   | ------- |                | ---------------|
             | Deck1  | 128-2 |                  |                |
             |        | 128-1 |                  |      ROM       |
     000000  -----------------           000000  -----------------
              256kx1  64k x 1
```

See section 15 for a more detailed map which includes a breakdown of the "I/O" space.

ROM Map Detail:



7.5 RAM Size Determination

It is possible for a program to determine the on-card RAM configuration without using a trial and error block test procedure. Follow these steps to ascertain the settings of J22 and J23:

1. After power-on (or reset) but BEFORE transferring control to block FE0000, which would turn RAM on:

    a. Read the long word at address 000004 (hex) and compare with the long word at address FC0004 (hex). If they are equal, J22 is set "A," indicating 64K x 1 type RAM chips are installed. If equal, skip to step 1.d., below. The particular value of the long word is not significant. (It will be the PC portion of the reset vector.)

    b. Read the long word at address 000004 and compare with the long word at address FD0004. If they are equal, J22 is set "B," meaning you've got 256K x 1 type RAM chips installed. If equal, skip to step 1.d., below.

    c. If both tests above failed, J22 is not installed. This indicates that there is no on-card RAM. Skip to step 2.

    d. Read the long word at address 000004 and compare with the long word at address FB0004. If they are equal, J23 is set "B", which would be the case for a double-decker RAM configuration. If the long words are not equal, J23 is set "A," for a single deck RAM.

    Summary: The long word at address 000004 will compare equal to the long word at the indicated addresses, based on the settings of J22 and J23, according to the table below: (Note: "#" means not equal, "=" means equal.)

	Address			Jumpers		Top
FD0004	FC0004	FB0004	RAM type, decks, Total bytes	J22	J23	Adrs
#	#	#	No on-card RAM	?	none	
#	#	=	No on-card RAM	?	none	
#	=	#	64K x 1, single, 128K	A	A	01FFFF
#	=	=	64K x 1, double, 256K	A	B	03FFFF
=	#	#	256K x 1, single, 512K	B	A	07FFFF
=	#	=	256K x 1, double, 1 meg	B	B	0FFFFF
=	=	#	(invalid)	?	?	
=	=	=	(invalid)	?	?	

2. Temporarily store the results in an MPU register (you don't have any RAM yet.)

3. Turn on the on-card RAM by transferring control to ROM at block FE0000. (Refer to section 7.1.)

4. Save the results of the tests in RAM for later use.

Once RAM has been turned on, this procedure will no longer be effective. If you want to determine RAM size using this method, you must do so prior to turning on the RAM.

To determine the amount of off-card memory, a conventional algorithm, which writes and reads off-card addresses looking for read errors and/or a Watchdog timeout, could be used.

7.6 Memory Timing

The HK68 memory logic has been carefully tuned to give optimum memory cycle times under a variety of conditions. Considerations have been given to these factors:

1. RAM parity checking requires additional time following the normal RAM access delay. (Parity Computation Time.) DTACK cannot be generated early to the MPU because DTACK cannot be cancelled if the parity computation subsequently determines that an error occurred. No delay is required, however, for write cycles.

2. The MMU, if present, delays the generation of stable physical addresses. (Translation Time.)

3. Typical access times for pROMs are 100 to 200 nanoseconds longer than RAM; however, parity checking is not required, so DTACK may be delivered to the MPU early. Since most programs will be in RAM (or could at least be copied to RAM for execution,) ROM timing need not be optimized.

4. Dynamic memory refreshing must be fast enough that a lengthy (or infinite) bus access cycle will not cause loss of the RAM contents. If a long access FROM the bus occurs, which would be terminated by the Watchdog Timer, refreshing must resume and a complete refresh cycle must be done before the maximun refresh time allowed by the RAMs expires. Refreshing operates normally during accesses TO the bus. The HK68 uses a hardware refresh.

Accordingly, the HK68 design has the following operating characteristics:

Extra clock cycles are inserted in memory references to synchronize the MPU and DMAC with the MMU and memory. The number of extra clock cycles required for a memory read or write may be found in the table below. These cycles are in addition to the four cycles built into the instruction timing charts provided in appendix A. Each clock cycle is 125 nanoseconds.

	MMU Configuration		Jumper
	with MMU	w/o MMU	J12
RAM reads, with parity enabled:	5	3	B
RAM reads, with parity disabled:	3	1	A
RAM writes:	3	1	
ROM (all cycles):	4	2	

Typical RAM access times:	150 nanoseconds
Typical ROM access times:	250 - 350 nanoseconds
Maximum ROM access time:	435 nanoseconds

Typical instruction execution times (RAM with parity, without MMU):

Instruction		MPU Timing	Actual Cycles	Speed
move	register to register	4(1/0)	7	875 nsec.
add	register to register	4(1/0)	7	875 nsec.
add.w	memory to register	8(2/0)	14	1.75 usec.
add.w	register to memory	12(2/1)	20	2.50 usec.
multiply		70(1/0)*	73*	9.125 usec.*
divide	unsiged, register	140(1/0)*	143*	17.875 usec.*

*maximum

If the MMU is installed, add 250 nanoseconds for each memory access (total of the parenthesizes numbers).

The HK68 uses hardware logic to control refreshing of the dynamic memory. The DRAM refresh rate is controlled by a 125KHz clock signal. This gives a refresh rate of one millisecond for 128 row, 64K x 1 RAMs or two milliseconds for 256 row, 256K x 1 RAMs. This is about twice as fast as necessary to keep the RAMs alive. Spare system clock cycles are used whenever possible to prevent the memory refreshing from slowing the MPU. Hidden refreshes occur during I/O, ROM or off-card accesses. Also, refreshing can occur during lengthy instructions (e.g., rotates or divides) without slowing execution. If no spare cycles are available, the worst case impact of RAM refreshing on memory timing is the addition of three clock cycles every 8 microseconds, or a net speed reduction of 4.7%.

7.7 Relevant Jumpers (Memory Configuration)

Jumper	Function	Pos	Notes
J12	RAM Speed/Parity	J12-A	Fast (no parity)
		J12-B	Slow, Parity Enabled
J13	ROM Type Select	J13-A, J14-A	2716 type
J14	ROM Type Select	J13-A, J14-B	2732, 2764 type
		J13-B, J14-B	27128 type
J22	RAM Device Size Select	J22-A	64K x 1
		J22-B	256K x 1
J23	RAM Deck Select	J23-A	Single Deck
		J23-B	Double-Decker
		(none)	on-card RAM disabled

8.0 SYSTEM ERROR HANDLING

There are numerous events which could cause an error to occur. The responses to these events are carefully controlled.

8.1 Error Conditions

The following error conditions may arise during MPU or DMAC cycles:

Type	Error	Reason	Disposition
H/W	RAM Parity	Incorrect parity was detected during a read cycle from on-card RAM memory. This may be due to a true parity error (RAM data changed,) or because the memory location was not initialized prior to the read and contained garbage. See section 8.2.	The memory cycle is terminated, the bus error exception is taken by the MPU or the bus error code is set in the DMAC.
H/W	Collision	An attempt was being made to use off-card facilities at the same time as the Multibus was requesting use of the on-card memory. Neither the MPU/DMAC nor the bus could complete its cycle because both sides of the bus interface are "busy." This is known as the "kitchen door effect."	The on-card request by the MPU or DMAC is suspendend, the bus request is honored and then the MPU or DMAC request is reissued. No muss, no fuss.
H/W S/W	Watchdog	During an access, usually to the bus, no XACK (bus ack) was received within a fixed time interval defined by a hardware timer. (About one millisecond.) This is usually the result of no bus device being assigned to the specified bus address or if an MMU or DMAC access is attempted without the MMU or DMAC chips installed. A timeout could also occur if an access from the bus is not terminated by the bus master. See also section 8.3.	For an access TO the bus, the memory cycle is terminated, the BERR (Bus Error) exception is taken by the MPU and execution resumes at the location specified by the exception vector. If the CIO has been so programmed, the DOG bit in CIO port A will be set. If an access FROM the bus was in progress, no BERR exception occurs, but the Bus Control Latch will be cleared to terminate the request. The DOG bit will be set.
H/W	Power Fail Interrupt (NMI)	Pin P2-19 has been brought low by an external device. This pin is intended for use by the power fail detection logic, but it may be used for a different function if desired. This is a non-maskable interrupt.	The MPU completes the current instruction and then initiates exception processing. Vector 31 is read and control is transferred to the specified adrs.

Error Conditions, continued...

Type	Error	Reason	Disposition
H/W S/W	Double Bus Fault	Another bus error (parity or timeout) occurred during the processing of a previous bus error, address error or reset exception. This error is the result of a major software bug or a hardware malfunction. A typical software bug which could cause this error would be an improperly initialized stack pointer, which points to an invalid address. Also, if the bus error exception vector has not been initialized, a parity error could have occurred when the vector was read.	The MPU enters the HALT state. Processing stops. Dead. The HALT status LED will come on. The only way out of this condition is to issue a hardware reset.
S/W	Odd Stack Address	The stack pointer contains an odd address value. The LSB of the stack pointer must be zero.	Same as Double Bus Fault, Above. The MPU will HALT.
S/W	MMU Fault	The MMU has detected a write violation or an undefined segment address.	The memory cycle is terminated and the BERR (Bus Error) exception is taken.
S/W	Divide by Zero	The value of the divisor for a divide instruction is zero. Even the almighty 68000 can't compute to infinity.	The instruction is aborted and vector 5 is used to transfer control.
S/W	Privileged Violation	A program executing in the user state attempted to execute a privileged instruction.	The instruction is not executed. Exception vector 8 is used to transfer control.
S/W	Address Error	An odd address has been specified for an instruction or a word or long word memory operand.	The bus cycle is aborted and vector 3 is used to transfer control.
S/W	Illegal or Unimplemented Instruction	The bit pattern for the fetched instruction is not legal.	The instruction is not executed. Exception vector 4, 10 or 11 is used to transfer control

As the above table indicates, there are numerous causes for a BERR (Bus Error) exception. In order to determine the cause of the exception, make the following tests, in the order given:

1. Test the DOG bit in CIO Port A.
2. Test the Fault status bits in the MMU.
3. If it wasn't a Watchdog timeout or an MMU Fault, the bus error was the result of a RAM parity error.

If a bus error occurs during a DMA cycle, the DMAC sets an internal error flag

8.2 RAM Parity Logic

Each byte of the on-card RAM has an associated parity bit which is automatically written with odd parity. If a byte is read by the MPU or DMAC which has an incorrect parity bit, a bus error is generated. If the memory cycle was controlled by the DMAC, the bus error is recorded by the DMAC and the bus cycle is terminated. No parity checking is done on bytes read by the bus or if the DMAC is using function code 7. Parity errors could result from numerous conditions, as outlined below.

Cause	Discussion
Defective Memory Chip	This "hard" error is the result of a bad bit cell or group of cells in the memory chip. Bad locations always return the same value, regardless of what has been written. It can be temporarily tolerated by the software, if the program has the smarts to "map around" the bad RAM area. The correct fix would be to replace the bad memory chip.
Alpha Particles	This is a "soft," random error. This type of error only affects one bit, and does not cause permanent damage. An alpha particle (helium nuclei produced by the environment or the chip package itself) penetrates the memory cell and causes a charge reversal, thus altering the data stored at that location. Rewriting the data will correct the error.

Much concern has been given to the alpha particle phenomena which came to light with the modern dynamic memory chip technology. Memory manufacturers have been very successful in reducing this problem. It is difficult to predict when alpha particle errors will occur because they are random events. Also, the error must occur in an "active" memory Location, otherwise you'll never know an error has occurred. Error rates are a function of chip design, size and capacity, as well as the size of the memory array. The mean time between failures (MTBF) for a fully used, 1 megabyte RAM array is better than 6 months.

Uninitialized Memory	Some programs read data from uninitialized areas of memory (RAM locations which have not been written since power was applied). A program might do this, for example, when block moving a non-contiguous set of bytes. It is usually easier to program the transfer of a contiguous group of bytes, including the intermediate uninitialized data. This practice would probably generate a parity error. The fix is to clear ALL memory immediately after power-on. Also, the DMAC can use function code 7 to ignore parity during a DMA data transfer.
Lengthy Access FROM the Bus	If the Watchdog Timer is disabled (jumper J25 installed), a long access from the bus could cause loss of RAM contents because memory refreshing ceases during such an access.

Incidentally, use of a parity bit actually decreases the MTBF by 12%. This is because the parity bit itself could be the victim of a soft error. There are more targets for the alpha particles to shoot at. Using a parity bit simply means you'll KNOW when an error occurs and thus improve the system reliability; but, alas, the parity bit also increases the chances of having an error. For critical applications, an external memory board with error correction capability could be used.

The parity logic may be disabled by installing jumper J12-B.

For a good discussion of memory error concepts, see "Choose the Right Level of Memory-Error Protection," <u>Electronic Design</u> magazine, February 18, 1982, page ss27.

8.3 Watchdog Timer

Whenever an access is made to or from the Multibus, the timing of the access is controlled by the bus control signals. The Watchdog's function is to prevent the HK68 board from being stopped by some malfunction of the Multibus. The following conditions will trigger the Watchdog Timer:

During an access TO the bus:	The Multibus is continuously busy. The addressed device does not exist. The addressed device does not respond with XACK.
During an access FROM the bus:	The master processor does not release the control signals following an access. There was an attempt to write into a memory segment wich was write protected by the MMU.
Other:	An access has been attempted to a non-existant MMU or DMAC chip.

All cases result in the DOG bit in CIO port A being set. This bit may be used to generate an interrupt. Except for accesses FROM the bus, the Bus Error exception is taken if a timeout occurs during an access.

The Watchdog is required to protect against a long request FROM the bus causing loss of RAM data. The use of the Watchdog is optional for requests TO the bus since refreshing continues normally during accesses to the bus.

The timer is factory set to approximately one millisecond. It may be disabled by installing jumper J25.

8.4 Relevant Jumpers (System Errors)

Jumper	Function	Pos	Notes
J12	RAM Speed/Parity	J12-A	Fast (no parity)
		J12-B	Slow, Parity Enabled
J25	Watchdog Disable	Install	to disable Watchdog Timer

9.0 MISCELLANEOUS ON-CARD DEVICES

This section describes the characteristics of the user DIP Switches, LED's and bus control latch.

9.1 Ring Detect Input Port (physical address FEC000, upper data, read only)

Each serial I/O interface has one RS-232 line connected to this input port. The function of these bits is normally for a modem Ring Detect signal; however, they may be used to monitor a different interface signal, if desired.

D15	D14	D11	D12	D11	D10	D9	D8
x	x	x	x	RD-D	RD-C	RD-B	RD-A
				P5-34	P5-17	P6-34	P6-17

Bit	Function	Meaning
11	Modem Ring Detect input, SIO Port D	"0" means true
10	Modem Ring Detect input, SIO Port C	"0" means true
9	Modem Ring Detect input, SIO Port B	"0" means true
8	Modem Ring Detect input, SIO Port A	"0" means true

The default condition of these signals (when no cable is attached) is dependent on jumper J15.

9.2 DIP Switch Input Port (physical addrs FEC001 hex, lower data, read only)

This input port allows the MPU to test the settings of four user DIP switches and four on-card status signals.

D7	D6	D5	D4	D3	D2	D1	D0
WESTDG	PRIAM	MPS-1	MPS-0	DIP-1	DIP-2	DIP-3	DIP-4
P4-41	P4-1	P8-8	P7-8				

Bit	Function	Meaning		
7	Western Digital Indicator	WESTDG	PRIAM	P4 Cable type
6	Priam Winchester Indicator			
		0	0	no cable
		0	1	Priam cable
		1	0	Western Dig
		1	1	SASI cable
5	SBX Module 1 present	"1" = module 1 (P8) present		
4	SBX Module 0 present	"1" = module 0 (P7) present		
3	User DIP switch, #1 (MSB)	"1" means ON, "0" means OFF		
2	User DIP switch, #2			
1	User DIP switch, #3			
0	User DIP switch, #4 (LSB)			

Note that the Ring Detect port and the DIP Switch port can be treated as one 16-bit device at address FEC000.

9.3 Bus Control Latch (physical address FEC000 hex, upper, write only)

These eight control bits affect the way the HK68 board interacts with the Multibus.

D15	D14	D13	D12	D11	D10	D9	D8
MISC	SPLIT	BC1	BC0	BMAP3	BMAP2	BMAP1	BMAP0

Bit	Function	Meaning
7	Miscellanous Control	J16-B, Data Convention "0" = Motorola "1" = Intel/Zilog J16-C, Write Protect "0" = disabled "1" = enabled
6	16 bit word split function	See "Bus Control," section 10
5	Bus Control 1	See "Bus Control," section 10
4	Bus Control 0	
3	Bus Map Select, bit 3	See "Bus Control," section 10
2	Bus Map Select, bit 2	
1	Bus Map Select, bit 1	
0	Bus Map Select, bit 0	

All control bits are set to zero ("0") at power on, after a system reset, as a result of executing an MPU 'RESET' instruction or if the Watchdog timer expires.

9.4 User LED Output Port (physical adrs FEC001 hex, lower data, write only)

There are eight LED's (located near the P3 board edge) whose meanings may be defined by the program.

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

Writing a one ("1") turns the corresponding LED on. A zero ("0") will turn the LED off. The state of the LED latch is not determined at power-on and is not changed by a RESET.

The control port described above and the LED port can be treated as a single 16 bit output device at address FEC000.

See section 4.3 for information on the four system status LED's.

9.5 Address Summary (Miscellaneous Ports)

Port	Direction	Address	Data
Ring Detect	Input	FEC000	Upper*
DIP Switch	Input	FEC001	Lower
Bus Cntrl latch	Output	FEC000	Upper*
User LED	Output	FEC001	Lower

*If a byte operation is done between a data register and address FEC000, the lower 8 bits of the data register will automatically be transferred on the upper 8 data lines. Therefore, the data should be in bits D7 to D0 of the MPU register. On the other hand, if a 16 bit word is transferred, the upper 8 bits will be used for port FEC000 and the lower byte for FEF001.

9.6 Relevant Jumpers (Miscellaneous Control)

Jumper	Function	Pos	Notes
J15	RS-232 Status Default	J15-A	True
		J15-B	False
J16	Miscellaneous Control (See MISC bit, above)	J16-A	Convention = Motorola
		J16-B	Convention = latched D15 ("0" = Motorola)
		J16-(not A or B)	Conv = Zilog
		J16-C	Bus Wr Prot = latched D15 ("0" = disable)
		J16-D	Bus Write Protect disabled
		J16-(not C or D)	Bus WProt enabled

9.7 Built-in Board Serial Number

In order to promote software security, the Heurikon HK68 microcomputer has a built-in serial number (in a PAL). The intent of this feature is to allow an operating system or application program to uniquely identify the host board, configure itself for that board and then refuse to execute properly on any other host. This will help prevent the unauthorized duplication and transfer of software between machines. For obvious reasons, this feature is not detailed in this manual. Consult the factory for more information.

OEM customers who need to know the details concerning this feature will be asked to sign a non-disclosure agreement.

10.0 BUS CONTROL

The control logic for the Multibus (a.k.a. IEEE-796) allows numerous bus masters to share the resources on the bus. The control logic for the Multibus is divided into the following sections:

1. On-card going off (access TO the bus)
2. Off-card coming on (access FROM the bus)
3. Data Convention control (relative byte locations)
4. Bus Interrupts

10.1 Bus Control Signals

The following signals on connector P1 are used by the Multibus arbitration logic: (refer to section 19.1 for a complete listing of connector P1 and P2 signals.)

Pin	Name	Function
P1-13	BCLK/	Bus Clock. An 8 MHz (10 MHz optional) clock generated by the highest priority master board on the bus. This signal is used to synchronize all bus requests and arbitration. The 10 MHz option requires the addition of crystal oscillator Y1.
P1-15	BPRN/	Bus Priority In. A low level indicates that no higher priority master needs the bus.
P1-16	BPRO/	Bus Prioity Out. A low level indicates that neither this board nor any higher priority board needs the bus.

BPRN/ and BPRO/ form a daisy chain for priority resolution. BPRO/ of each processor board is connected to the BPRN/ pin on the next lowest priority processor. BPRN/ of the highest priority board is tied low by installing jumper J20 on that board.

Pin	Name	Function
P1-17	BUSY/	Bus Busy. A low level indicates that the bus is in use.
P1-18	BREQ/	Bus Request. A low level indicates that this board needs the bus. This signal may be used to implement a parallel priority arbitration scheme instead of a daisy chain. BREQ/ for each slot on the bus is independent of all other BREQ/ lines; i.e., this signal is not bused.

Bus Control Signals, continued...

Pin	Name	Function
P1-25	LOCK/	Bus Lock. This signal is used to prevent the target board from releasing the facilities between a pair of bus accesses. This is necessary to implement "test and set" types of instructions which use read/ modify/write cycles. If true (low) during an access FROM the bus, the HK68 board will not release the on-card bus to the MPU or DMAC between bus cycles, unless the Watchdog timer expires. During an access TO the bus, LOCK will be true whenever the MPU/DMAC Address Strobe (AS) signal is on. Not all Multibus compatible boards support this function. If not required, remove jumper J26.
P1-29	CBRQ/	Common Bus Request. This signal is common for all processors in a system. A low level indicates that there is a bus request pending from a processor which is not already using the bus, regardless of priority. This signal allows a processor to maintain control of the bus, whether actively using the bus or not, until such time as there is another processor needing the bus. This method reduces the bus arbitration time in the absence of multiple bus requests, since the processor last using the bus can "keep" it until another board actually needs it.
P1-31	CCLK/	Constant Clock. The highest priority bus master provides this signal to the bus. It is used by the other bus masters to arbitrate for use of the Multibus. The HK68 provides an 8Mhz clock signal. A 10MHz clock is available as an option.
P1-27	BHEN/	Byte High Enable. This signal, when true, indicates that a 16-bit bus operation is in progress. Otherwise, bus data transfers are 8-bit bytes, on the lower eight bits.
P1-23	XACK/	Transfer Acknowledge. At the completion of a bus operation, the target board (slave) generates this signal to indicate that the operation has been completed. Data is valid for a read or has been written for a write. XACK synchronizes all transfers over the bus and allows devices of various speeds to use the bus.
P1-xx	AACK/	Advanced Acknowledge. Obsolete, not used.

Bus Control Signals, continued...

Pin	Name	Function
P1-14	INIT/	Initialize. This is the hardware reset line. It may be either an input or an output, as determined by the setting of jumper J24. When used as an output, the MPU can activate this signal by executing a 'reset' instruction.
P2-19	PFIN/	Power Fail Interrupt. This is a non-maskable interrupt input to the MPU. It may be used to signal any function which must have a super-high priority. See section 8.1
P2-38	RESET/	Reset. This input pin may be used to reset only one board even though there are other boards on the bus.
P1-19	MRDC/	Memory Read Control.
P1-20	MWTC/	Memory Write Control. These two signals control memory reads and writes. They indicate that the bus address is valid and, for writes, that the data bus is valid. The master processor waits for XACK/ before terminating the command.
P1-21	IORC/	I/O Read Control.
P1-22	IOWC/	I/O Write Control. The upper 64K of physical address space maps into bus I/O commands. Reading a byte (or word) generates IORC/; writing data generates IOWC/. These signals are outputs only.
P1-35 to P1-42		Bus Interrupt lines INT0/ to INT7/. (8 lines) The bus supports eight interrupts. The HK68 uses port B of the CIO to monitor these lines and interrupt the MPU when one is active. The CIO can mask particular lines to set priority levels. As outputs, these lines may be used to generate bus interrupts, again under control of the CIO. They may also be used as a general purpose parallel I/O port, if desired. Refer to section 10.6 for details.
P1-33	INTA/	Interrupt Acknowledge. Not used.
P1-43 to P1-58		Bus Address lines ADR0/ to ADRF/. (16 lines)
P1-28,30,32,34		Bus Address lines ADR10/ to ADR13/. (4 lines)
P2-55 to P2-56		Bus Address lines ADR13/ to ADR17/. (4 lines)
P1-59 to P1-74		Bus Data lines DAT0/ to DATF/. (16 lines) Note: DATF/ is the most significant data bit, as it should be, to allow communication with a 16 bit I/O device.

10.2 On-card going off (TO the bus)

When the MPU or DMAC makes a request for bus facilities, the arbitration logic takes over. If necessary, the requesting board enters a wait state until the bus is available (but only for the maximum time allowed by the Watchdog timer). When the requested operation is completed, the bus will be released according to the state of the two control signals, BC1 and BC0. These signals are under software control via the Bus Control Latch.

BC1	BC0	Bus release status
0	0	Release bus after every operation
0	1	Release bus if any other board has a request for the bus. (Uses CBRQ/)
1	0	Release the bus only if a higher priority board has a request for the bus. (Uses BPRN/)
1	1	Never release the bus, once acquired. This state can be used to capture the bus.

Logical addresses aimed at the bus are mapped by the MMU in the same manner as on-card memory.

Although I/O requests from the bus are ignored, it is possible to generate an I/O command TO the bus. The upper 64K physical addresses are mapped to the bus as I/O commands. For example, to do an "OUTPUT" to a bus device with an I/O address of 48 (hex), do a 'move byte' instruction specifiying a physical destination of FF0048. Since a 64K-byte space is reserved for this function, eight or 16-bit device addressing is supported. In order to guarantee that all bytes are transferred over the lower 8 data lines, use byte mode instructions or turn SPLIT on, as discussed in section 10.8.

10.3 Off-card coming on (FROM the bus)

The conventional method of board assignment in the Multibus address space is to utilize a group of DIP switches or jumpers to specify a base address for each board. The Heurikon HK68 uses a special bus mapping PAL which monitors the Multibus for a particular combination of the upper 12 bus address lines. The PAL may be programmed with up to 16 different address space areas; and, the MPU can select the particular space to be used. Since the upper 12 address lines are used, the board may be mapped into any 4K address block. By ignoring the state of some of the lower address lines, the size of the address space may be enlarged (up to 16 megabytes). Refer to the "Bus Map," section 10.4, below.

Once a valid bus request has been detected, an on-card bus request is generated to the MPU and DMAC. When the current cycle is completed, the MPU/DMAC will release the on-card bus. The Multibus address and data is then gated on. The bus address lines are utilized as follows:

Bus Adrs Line	Usage
ADR17 (MSB)	Input to bus mapping PAL only. Otherwise ignored.
ADR16	" " " " " " "
ADR15	" " " " " " "
ADR14	" " " " " " "
ADR13	Input to bus mapping PAL and LA19 input to MMU
ADR12	" " " " LA18 " "
ADR11	" " " " LA17 " "
ADR10	" " " " LA16 " "
ADRF	" " " " LA15 " "
ADRE	" " " " LA14 " "
ADRD	" " " " LA13 " "
ADRC	" " " " LA12 " "
ADRB	On-card physical address A11
ADRA	" " " A10
~	~ ~
~	~ ~
ADR1	" " " A1
ADR0 (LSB)	" " " A0 (UDS, LDS)

The MMU can logically map the bus access into any 4K block of phsyical RAM memory. The bus cannot access the on-card I/O devices or ROM. The MMU usually has control of the upper 16 physical address lines; however, during an access from the bus, the MMU controls only A12 through A19 (eight lines). A20 through A23 are forced to point to the RAM memory bank, and A0 through A11 are controlled directly by the Multibus address lines. Refer to the MMU section (6.0) for more details.

I/O commands FROM the bus are ignored. The HK68 does not monitor IORC/ or IOWC/.

Lengthy memory cycles (hundreds of microseconds), originating from the bus, should be avoided. RAM refreshing is suspended during an access by the bus. When XACK is received from the HK68 board, the master processor must terminate the bus request. If the Watchdog Timer is enabled, any long access from the bus will automatically be aborted. Refer to section 8.3, "Watchdog Timer." Memory parity is not checked when doing a read from the bus.

10.4 Bus Map

The MPU selects the address space by setting the four Bus Map Control lines, via the Bus Control Latch, as follows:

BMAP- 3 2 1 0	Space	Standard Configuration
0 0 0 0	0	Off. Accesses FROM the bus are not allowed.
0 0 0 1	1	One Megabyte of memory starting at address 000000
0 0 1 0	2	" " " " " " " C00000
0 0 1 1	3	" " " " " " " D00000
0 1 0 0	4	" " " " " " " E00000
0 1 0 1	5	512 Kilobytes of memory starting at address 000000
0 1 1 0	6	" " " " " " " E00000
0 1 1 1	7	" " " " " " " E80000
1 0 0 0	8	256 Kilobytes of memory starting at address 000000
1 0 0 1	9	" " " " " " " E00000
1 0 1 0	10	" " " " " " " E40000
1 0 1 1	11	" " " " " " " E80000
1 1 0 0	12	128 Kilobytes of memory starting at address 000000
1 1 0 1	13	" " " " " " " E00000
1 1 1 0	14	" " " " " " " E80000
1 1 1 1	15	All access occuring on the bus will be acknowledged by this card. i.e., the HK68 board fills the entire bus space.

Note that the actual configuaration may be changed by custom programming the bus mapping PAL (chip U68).

10.5 Bus Control Latch (physical address FEC000 hex, upper, write only)

D15	D14	D13	D12	D11	D10	D9	D8
MISC	SPLIT	BC1	BC0	BMAP3	BMAP2	BMAP1	BMAP0

Bit	Function	Meaning
7	Miscellanous Control	See jumper J16 description
6	16 bit word split function	See "Bus Data Conventions" section 10.8
5	Bus Control 1	See section 10.2
4	Bus Control 0	
3	Bus Map Select, bit 3	See section 10.4
2	Bus Map Select, bit 2	
1	Bus Map Select, bit 1	
0	Bus Map Select, bit 0	

All control bits are set to zero ("0") at power on, after a system reset, as a result of an MPU 'RESET' instruction being executed, or if the Watchdog timer expires.

10.6 Bus Interrupts

The eight Multibus interrupts are monitored and controlled by the CIO chip. By programming the CIO, certain combinations of the bus interrupt lines may be monitored, and a vectored interrupt to the MPU can be generated when the desired state is realized. Also, by selectively programming the CIO lines as outputs, bus interrupts may be generated under software control.

The "B" CIO input port (physical address FEF603) is used for this function. The bit assignments are:

D7	D6	D5	D4	D3	D2	D1	D0
INT 7	INT 6	INT 5	INT 4	INT 3	INT 2	INT 1	INT 0
P1-36	P1-35	P1-38	P1-37	P1-40	P1-39	P1-42	P1-41

A "0" indicates an active (true) interrupt condition if the CIO is not inverting the inputs. (May be changed to "1" by initializing the CIO to invert these inputs.)

Refer to section 11 for information on programming the CIO.

10.7 Write Protection

There are two ways to protect the on-card RAM from being modified by the bus.

a. When the MMU is used for address conversions, it may be programmed to provide write protection for the segments assigned to the bus (function code 15). If the MMU detects a write violation, memory contents will not be altered. The MMU will not complete the requested operation and no XACK will be issued to the bus. The Watchdog timer will terminate the request. The MMU can be programmed to generate an interrupt following the aborted access.

b. It is also possible to disable the MWTC signal signal on the Multibus to prevent write requests from even activating the on-card bus arbitration logic. In this case, if a write is attempted by the bus, the write command is ignored and no XACK is issued to the bus. The on-card facilities will not be disturbed by the request. See section 10.5 ("MISC" bit) and jumper J16 in section 10.9.

Note that with either method, the Multibus request is NOT acknowledged. The processor board making the request must eventually timeout in order to free the bus facilities for another access.

10.8 Bus Data Conventions

The Motorola data convention specifies that higher order bytes of a word are stored in lower address cells. This is opposite to the Intel/Zilog convention. The following chart showes how the two methods differ:

Data to be stored (two 16-bit words): 1234 5678 (hex)

Memory Address		Motorola Contents		Intel/ Zilog Contents	
(data bit)		15 8	7 0	15 8	7 0
000000		12	34	34	12
000002		56	78	78	56

This can create some problems in systems using both types of processors. Also, most 16 bit processor boards require all memory boards to be 16 bits wide if executable code is stored there. The Heurikon HK68 has two bus control bits which may be used to deal with these problems. (See "Bus Control Latch," section 10.5 above, for programming information.)

Control Bit	Function
SPLIT	If True ("1") all 16 bit accesses to the bus are split into two BYTE operations. A 16 bit word is divided into two bytes (for a write) or assembled into a word from two bytes (for a read) using an on-card byte swap buffer. This allows the HK68 board to function with 8-bit memory boards without using the byte mode for the instructions. The HK68 can even EXECUTE out of an eight bit memory board.
CONVENTION	If low, the Motorola convention is used for all byte operations on the bus. Higher order bytes are stored in lower addresses. If high, the Zilog convention is used for byte transfers. This signal also controls the translation of bus address bit ADR0 into Upper and Lower Data Strobes (UDS, LDS) during a byte access FROM the bus. If low, a byte access from the bus with ADR0 true operates on the lower half of a word (Motorola convention).

Multibus Signal			Motorola Convention		Zilog Convention	
BHEN/	ADR0/	Byte	UDS/	LDS/	UDS/	LDS/
true	false	both	true	true	true	true
false	false	even	true	false	false	true
false	true	odd	false	true	true	false

The SPLIT bit effectively turns all 16-bit requests to the bus into a pair of byte operations. CONVENTION operates to invert address line ADR0.

Signal	Access FROM the bus	Access TO the bus
SPLIT	no effect	Causes all requests to be byte-sized and on the lower bus data lines.
CONVENTION	Inverts ADR0 if BHEN false (byte mode)	Inverts ADR0 if SPLIT on or if byte mode instruction

BHEN (Byte High Enable) is a Multibus signal which, if true, indicates that the operation is for 16-bits and will use the upper eight data lines.

All MPU/DMAC generated Multibus byte operations (as well as word accesses while SPLIT is on) are conducted on the lower eight Multibus data lines, with DAT7/ being the most significant bit. For word transfers, DATF/ is the most significant bit, as per the Multibus specification. Caution: Some of our competition always puts the MSB on DAT7/, even for 16-bit transfers!

10.9 Relevant Jumpers (Bus Control)

Jumper	Function	Pos	Notes
J15	RS-232 Status Default	J15-A	True
		J15-B	False
J16	Miscellaneous Control	J16-A	Convention = Motorola
		J16-B	Convention = latched D15 ("0" = Motorola)
		J16-(not A or B)	Conv = Zilog
		J16-C	Bus Wr Prot = latched D15 ("0" = disable)
		J16-D	Bus Write Protect disabled
		J16-(not C or D)	Bus WProt enabled
J18	Bus Clock Rate Select	J18-A	8 MHz
		J18-B	10 MHz (option)
J19	Bus CCLK Enable	Install on highest priority board	
J20	Bus BPRN Enable	Install on highest priority board	
J21	Bus BCLK Enable	Install on highest priority board	
J24	Bus INIT/ Select (P1-14)	J24-A	INIT/ = on-card RESET/ (out)
		J24-B	INIT/ = AuxRESET/ (Input)
J26	Bus LOCK/ Enable	Install to enable Bus LOCK fuction	

10.10 Multibus Compliance Levels

```
Master: D16 M24 I16 V0 L
Slave:  D16 M24     V0 L
```

For P1 and P2 connector pinouts, refer to section 19, "Multibus Interface."

11.0 CIO USAGE

The on-card CIO device performs a variety of functions. In addition to the three 16-bit timers which may be used to generate interrupts or count events, the CIO has numerous parallel I/O bits which control functions such as, SBX module options and Winchester port interrupts.

The following tables show some of the particulars concerning the CIO. Refer to later pages for more details and for programming information.

Port	Initialize as	Function
A	Bit Mode	SBX, Tape, Winc status (inputs)
B	Bit Mode	Multibus Interrupts (input and output)
C	-	SBX option controls

11.1 Port A Bit Definition (physical address FEF605 hex, lower data, read)

All bits should be programmed as non-inverting inputs. The one's catcher should be used for the Watchdog Timer input (bit 0) because the timer is automatically reset following a time-out.

D7	D6	D5	D4	D3	D2	D1	D0
XINT0	XINT1	ARDY	AEXC	SA-IO	SA-CD	WINTR	DOG
P7-8	P8-8	P3-28	P3-30	P4-50	P4-46	P4-36	

Bit	Function	Meaning (all bits are inputs)
7	SBX Module Interrupt, P7-8	SBX module dependent (See J8)
6	SBX Module Interrupt, P8-8	SBX module dependent (See J8)
5	P3-28 (Archive Tape Ready)	"0" means true*
4	P3-30 (Archive Exception)	"0" means true*
3	P4-50 (SASI Winc I/O)	Depends on controller
2	P4-46 (SASI Winc C/D)	Depends on controller
1	P4-36 (Winc Interrupt)	Depends on controller
0	Watchdog Timer	"1" means the timer expired* (uses the CIO 1's catcher)

*assumes the CIO has NOT been programmed to invert inputs.

11.2 Port B Bit Definition (physical adrs FEF603 hex, lower data, read/write)

The eight Multibus interrupts are monitored and controlled by port B of the CIO chip. By programming the CIO, certain combinations of the bus interrupt lines may be monitored, and a vectored interrupt to the MPU can be generated when the desired state is realized. Also, by selectively programming the CIO lines as outputs, bus interrupts may be generated under software control.

D7	D6	D5	D4	D3	D2	D1	D0
INT 7	INT 6	INT 5	INT 4	INT 3	INT 2	INT 1	INT 0
P1-36	P1-35	P1-38	P1-37	P1-40	P1-39	P1-42	P1-41

A "0" indicates an active (true) interrupt condition if the CIO is not inverting the inputs. (May be changed to "1" by initializing the CIO to invert these inputs.) Port B bits may also be programmed as external I/O lines for counter/timer channels 1 & 2. This feature could be used to count rapid external events or indicate count completion.

11.3 Port C Bit Definition (physical adrs FEF601 hex, lower data, read/write)

This bi-directional port is used to control the SBX module option lines. The exact function of these bits is determined by the particular SBX module used. The CIO allows bit addressable writes of each of these control bits by using a mask nibble in the upper four data bits; and they may be independently programmed as inputs or outputs. Refer to the CIO manual for details (appendix E).

D7	D6	D5	D4	D3	D2	D1	D0
				OPT11	OPT10	OPT01	OPT00
				P8-28	P8-30	P7-28	P7-30

Bit	Fucntion
3	Controls SBX connector P8 pin 28, Option 1
2	Controls SBX connector P8 pin 30, Option 0
1	Controls SBX connector P7 pin 28, Option 1
0	Controls SBX connector P7 pin 30, Option 0

Refer to section 14 for more details on the SBX interface.

11.4 Counter/Timers

There are three independent, 16-bit counter/timers in the CIO. Each may be used as a counter in conjunction with the port B and C lines, which are connected to the Multibus interrupt and the SBX option lines, respectively; or as timers to implement real-time clocks for programmed delays. For long delays, timers 1 and 2 may be internally linked together to form a 32-bit counter chain. When programmed as timers, the following table may be used to determine the time constant value for a particular interrupt rate.

--Desired Frequency/Rate--		------Time Constant Values------			
Frequency (Hz)	Interrupt Rate	Timer Value		--Linked Timer-- Timer1	Timer 2
2,000	0.50 msec	1,000			
1,000	1.00 msec	2,000			
500	2.00 msec	4,000			
200	5.00 msec	10,000			
100	10.00 msec	20,000			
60	16.67 msec	33,333	(approx)		
50	20.00 msec	40,000	or	2,000	20
20	50.00 msec			2,000	50
10	100.0 msec			2,000	100
5	200.0 msec			2,000	200
2	500.0 msec			2,000	500
1.0	1.0 second			2,000	1,000
0.5	2.0 second			2,000	2,000
0.2	5.0 second			2,000	5,000
0.1	10.0 sec			2,000	10,000
	30.0 sec			2,000	30,000
	1.0 minute			20,000	6,000

When the timer is clocked internally, the count rate is 2 Mhz (500 nanoseconds) per count. The HK68 board uses a Motorola 16.000 MHz clock oscillator as the system time base. The frequency stability specification is +/- 0.05% (includes calibration tolerance, stability versus input voltage, stability versus load, aging and normal environment). If you are using the 16 MHz clock as the CIO timebase, the maximum accumulative timing error will be 43 seconds per day. However, since the day-to-day stability will be high, it is possible for the operating system to correct for the frequecny error in the software time-of-day routines to yield an accurate time clock.

11.5 Register Summary (CIO)

Register	Physical Address	Function
Port C, Data	FEF601	SBX options (4)
Port B, Data	FEF603	Multibus Interrupts (8)
Port A, Data	FEF605	SBX Ints, Winc & Tape Status
Control Regs	FEF607	CIO Configuration & Control

11.6 CIO Initialization sequence

The following table shows a typical initialization sequence for the CIO:

Data	Register Address		Function
x	FEF607	(read)	Read from CIO control reg to insure we are pointing to reg 0
00	"	(write)	Cmd Un-reset, internal state unknown
x	"	(read)	CIO now un-reset and in state 0
00	"	(write)	CIO ready to write reg 0
01	"	"	Reset command
00	"	"	Force CIO to state 0
00,02	"	"	Turn off reset, right justify
20,00	"	"	Port A, Bit control mode
22,00	"	"	Port A, All paths non-inverting
23,FF	"	"	Port A, All inputs
24,01	"	"	Port A, 1's catcher on D0 (Watchdog)
28,00	"	"	Port B, Bit control mode
2A,00	"	"	Port B, All paths non-inverting
2B,FF	"	"	Port B, All inputs
2C,01	"	"	Port B, normal inputs (bus ints)
05,00	"	"	Port C, non-inverting
06,FF	"	"	Port C, all inputs (SBX options)
07,00	"	"	Port C, normal inputs
1C,80	"	"	Timer 1, continuous, internal control
16,9C	"	"	Timer 1, Time constant, H (20.0 msec)
17,40	"	"	Timer 1, Time constant, L (20.0 msec)
01,94	"	"	Reg 1, enable ports A, B, C

The notation "00,02" (etc) means the values 00 (hex) and 02 should be sent to the specified CIO port. The first byte selects the internal CIO register; the second byte is the control data. The above sequence only initializes ports A, B, C and one timer. The specific directions of some of the PIO lines (e.g. SBX options), and interrupts need to be set by the user. An active low signal can be inverted, so that a "1" is read from the data port when the signal is true, by initializing the port to invert that particular bit. Refer to section 4.2 concerning CIO interrupt vectors.

11.7 Relevant Jumpers (CIO Usage)

Jumper	Function	Pos	Notes
J8	SBX Interrupt Config	J8-A	SBX P7 pin 12 (INT1) to XINT0
		J8-B	SBX P7 pin 12 (INT1) to XINT1
		J8-C	SBX P8 pin 12 (INT1) to XINT1
		J8-D	SBX P8 pin 12 (INT1) to XINT0

Note: P7-14 and P8-14 connect directly to CIO port A bits 7 & 6, respectively, regardless of J8 settings. See section 14.3.

11.8 CIO Programming Example

The two routines that make up a CIO interrupt test are shown below. They set up counter/timer channel 3 to generate 60 interrupts a second. The interrupt service routine divides this by 60 and generates one blink on the user LEDs every second. Once started, the process is kept alive by interrupts, while the main body of the routine returns to the monitor.

The routines come in two parts: a 'C' section and a machine code section. First, the 'C' code:

```
/*
 *      CIO test program has channel 3 generate interrupts and report them on
 *      the leds. It can be used to time the cio ticks.
 */

#define CIOPORT 0xFEF601
#define CIOCTRL CIOPORT + 0x06


#define LEDS 0xFEC001

#define CIO_VECTOR 0x60

#define out(port, data) *(char *) (port) = data
#define in(port) *(char *) (port)

static unsigned short int tickcnt = 0;
static unsigned short int seccnt = 0;
static unsigned short int mincnt = 0;
static unsigned char ledc = 0;

main()
{
    tickcnt = seccnt = mincnt = ledc = 0;
    initvec();
    initcio();
}

/*
 *      initvec()
 *
 *      Initializes the interrupt vector to point to the assembly language
 *      routine interrupt().
 */

initvec()
{
    int interrupt();

    *(int (**)()) (CIO_VECTOR << 2) = interrupt;
}
```

```
/*
 *      initcio()
 *
 *      Initializes the CIO such that channel 3 will generate 60 interrupts
 *      per second.
 */

initcio()
{
    register int cnt;
    register char c;
    register char *p;
    static unsigned char ciotable[] = {
        0x00,   0x86
        0x1E,   0x80,
        0x1A,   0x82,           /* Time Constant, H */
        0x1B,   0x35,           /* Time Constant, L */
        0x04,   0x60,
        0x0C,   0x20,
        0x0C,   0xC6,
        0x01,   0x10,
    };

    p = CIOCTRL;
    c = in(p);
    out(p, 0);
    c = in(p);
    out(p, 0x00);
    out(p, 0x01);
    out(p, 0x00);
    for (cnt = 0; cnt < sizeof(ciotable); cnt++)
        out(p, ciotable[cnt]);
}

/*
 *      ciointr()
 *
 *      This is the routine that is invoked during the interrupt service
 *      routine.
 */

ciointr()
{
    tickcnt++;
    if (tickcnt >= 60) {
        tickcnt = 0;
        ledc ^= 0x03;
        seccnt++;
        if (seccnt >= 60) {
            seccnt = 0;
            mincnt++;
        }
        ledc = (ledc & 0x0F) + (mincnt << 4);
        out(LEDS, ~ledc);
    }
    out(CIOCTRL, 0x0A);
    out(CIOCTRL, 0x24);
}
```

Now, here is the machine code section:

```
                .text
                .even

_interrupt:     movem.l         d0-d7/a0-a6, -(a7)
                jsr             _ciointr
                movem.l         (a7)+, d0-d7/a0-a6
                rte

                .globl          _interrupt
```

12.0 SERIAL I/O

There are four RS-232C serial I/O ports on the HK68 board. One of the ports may optionally be configured for RS-422 operation. Each port has a separate baud rate generator and can operate in asynchronous or synchronous modes.

12.1 RS-232 Pinouts

Data transmission conventions are with respect to the device. The HK68 board appears as a "Data Set." The connector pinouts are as follows:

	Pin	"D" Pin	RS-232 Function	(direction)
Port A	x	1	Protective ground. Not connected on the HK68	
	P6- 1	14	x	
	P6- 2	2	Tx Data	(from device)
	P6- 3	15	Tx Clock	(from device)
	P6- 4	3	Rcv Data	(to device)
	P6- 5	16	x	
	P6- 6	4	*Request To Send	(from device)
	P6- 7	17	Rcv Clock	(to device)
	P6- 8	5	Clear To Send	(to device)
	P6- 9	18	x	
	P6-10	6	Data Set Ready	(to device)
	P6-11	19	x	
	P6-12	7	Signal Ground	
	P6-13	20	*Data Terminal Ready	(from device)
	P6-14	8	x	
	P6-15	21	x	
	P6-16	9	x	
	P6-17	22	*Ring Detect	(from device)
	x	10-13	x	
	x	23-25	x	
Port B	x	1	Protective ground. Not connected on the HK68	
	P6-18	14	x	
	P6-19	2	Tx Data	(from device)
	P6-20	15	Tx Clock	(from device)
	P6-21	3	Rcv Data	(to device)
	P6-22	16	x	
	P6-23	4	*Request To Send	(from device)
	P6-24	17	Rcv Clock	(to device)
	P6-25	5	Clear To Send	(to device)
	P6-26	18	(+5 via J3)	
	P6-27	6	Data Set Ready	(to device)
	P6-28	19	x	
	P6-29	7	Signal Ground	
	P6-30	20	*Data Terminal Ready	(from device)
	P6-31	8	x	
	P6-32	21	(+12 via J1)	
	P6-33	9	(-12 via J2)	
	P6-34	22	*Ring Detect	(from device)
	x	10-13	x	
	x	23-25	x	

	Pin	"D" Pin	RS-232 Function	(direction)
Port C	x	1	Protective ground. Not connected on the HK68	
	P5- 1	14	x	
	P5- 2	2	Tx Data	(from device)
	P5- 3	15	Tx Clock	(from device)
	P5- 4	3	Rcv Data	(to device)
	P5- 5	16	x	
	P5- 6	4	*Request To Send	(from device)
	P5- 7	17	Rcv Clock	(to device)
	P5- 8	5	Clear To Send	(to device)
	P5- 9	18	x	
	P5-10	6	Data Set Ready	(to device)
	P5-11	19	x	
	P5-12	7	Signal Ground	
	P5-13	20	*Data Terminal Ready	(from device)
	P5-14	8	x	
	P5-15	21	x	
	P5-16	9	x	
	P5-17	22	*Ring Detect	(from device)
	x	10-13	x	
	x	23-25	x	
Port D	x	1	Protective ground. Not connected on the HK68	
	P5-18	14	x	
	P5-19	2	Tx Data	(from device)
	P5-20	15	Tx Clock	(from device)
	P5-21	3	Rcv Data	(to device)
	P5-22	16	x	
	P5-23	4	*Request To Send	(from device)
	P5-24	17	Rcv Clock	(to device)
	P5-25	5	Clear To Send	(to device)
	P5-26	18	(+5 via J7)	
	P5-27	6	Data Set Ready	(to device)
	P5-28	19	x	
	P5-29	7	Signal Ground	
	P5-30	20	*Data Terminal Ready	(from device)
	P5-31	8	x	
	P5-32	21	(+12 via J6)	
	P5-33	9	(-12 via J5)	
	P5-34	22	*Ring Detect	(from device)
	x	10-13	x	
	x	23-25	x	

Note that the interconnect cables from P5 and P6 are split in such a manner that the "D" connector pinouts are correct for RS-232C conventions. Not all pins on the "D" connectors are used.

Signals indicated with "*" have a default pullup resistor, controlled by J15.

Refer to "Miscellaneous On-card Devices," section 9.1, for details on the Ring Detect input port.

12.2 Signal Naming Conventions (RS-232)

Since the RS-232 ports are configured as "data sets," the naming convention for the interface signals may be confusing. The interface signal names are with respect to the terminal device attached to the port while the SCC pins are with respect to the SCC as if it, too, is a terminal device. Thus all signal pairs, e.g., "RTS" & "CTS," get "reversed" between the I/F connector and the SCC chip. For example, "Transmit Data," P6-2, is the data transmitted from the device to the HK68 board; the data appears at the SCC receiver as "Received Data." For the same reason, the "RTS" and "DTR" interface signals appear as the "CTS" and "DSR" bits in the SCC, respectively. If you weren't confused before, any normal person should be by now. Study the chart below and see if that helps.

SCC Signal Name	I/F Signal (P5 & P6)	Direction
Tx Data	Rcv Data	to device
Rcv Data	Tx Data	from device
Tx Clock	Rcv Clock	to device
Rcv Clock	Tx Clock	from device
RTS	CTS	to device
CTS	RTS	from device
DTR	DSR	to device
DCD	DTR	from device

The SCC was designed to look like a "data terminal" device. Using it as a "data set" creates this nomenclature problem. Of couse, if you connect the HK68 board to a modem ("data set"), then the SCC signal names are correct, however, a cable adapter is needed to properly connect to the modem. (Three pairs of signals must be reversed.)

SCC Signal	P5 & P6 Pin #s	"D" Pin # at HK68	"D" Pin # at modem	
x	x	1	1	
Rcv Data	2 (or 19)	2	3	\
Tx Data	4 (or 21)	3	2	|
CTS	6 (or 23)	4	5	|
RTS	8 (or 25)	5	4	> Reversals
DTR	10 (or 27)	6	20	|
DSR	13 (or 30)	20	6	/
Ring Det	17 (or 34)	22	22	
Sig Gnd	12 (or 29)	7	7	

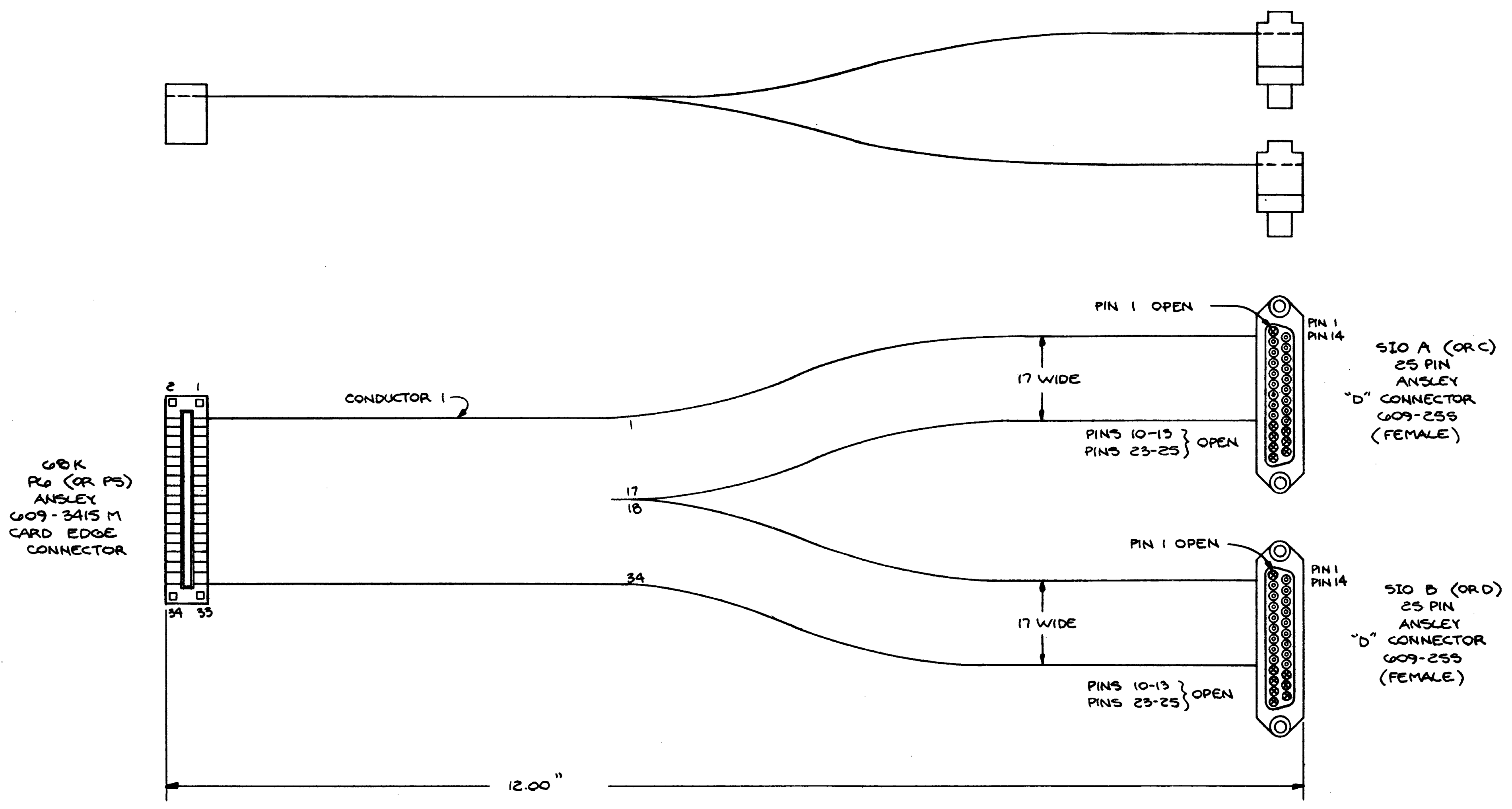
12.3 Connector Conventions

The EIA RS-232-C standard says the following concerning the mechanical interface between data communications equipment (section 3.1):

> "The female connector shall be associated with...the data communications equipment... An extension cable with a male connector shall be provided with the data terminal equipment... When additional functions are provided in a separate unit inserted between the data terminal equipment and the data communications equipment, the female connector...shall be associated with the side of this unit which interfaces with the data terminal equipment while the extension cable with the male connector shall be provided on the side which interfaces with the data communications equipment."

Substituting "modem" for "data communications equipment" and "terminal" for "data terminal equipment" leaves us with the impression that the modem should have a FEMALE connector and the terminal should have a MALE. The Heurikon HK68 microcomputer interface cables are designed with female "D" connectors, because the serial I/O ports are configured as data sets (modems). Terminal manufacturers typically have a female connector also, despite the fact that they are terminals, not modems. Thus, the extension cable used to run between a terminal and the HK68 (or a modem) will have male connectors at both ends.

If you do any work with RS-232 communications, you'll end up with zillions of cable adapters. Double males, double females, double males and females with reversal, cables with males and females at both ends, you name it! We'll be happy to help make special cables to fit your needs.



HEURIKON CORPORATION MADISON, WISCONSIN	RS232 SERIAL INTERFACE CABLE	DRWN: RK	DATE: 7-20-82
		SCALE: 1"=1"	CKD: JM

12.4 RS-422 Option

Serial port A may be optionally configured as an RS-422/449 port for reliable data communications at high speeds over long distances. The interface uses differential data transmission to permit data rates of up to 1 megabit over 400 feet or 100,000 bits per second over 4000 feet.

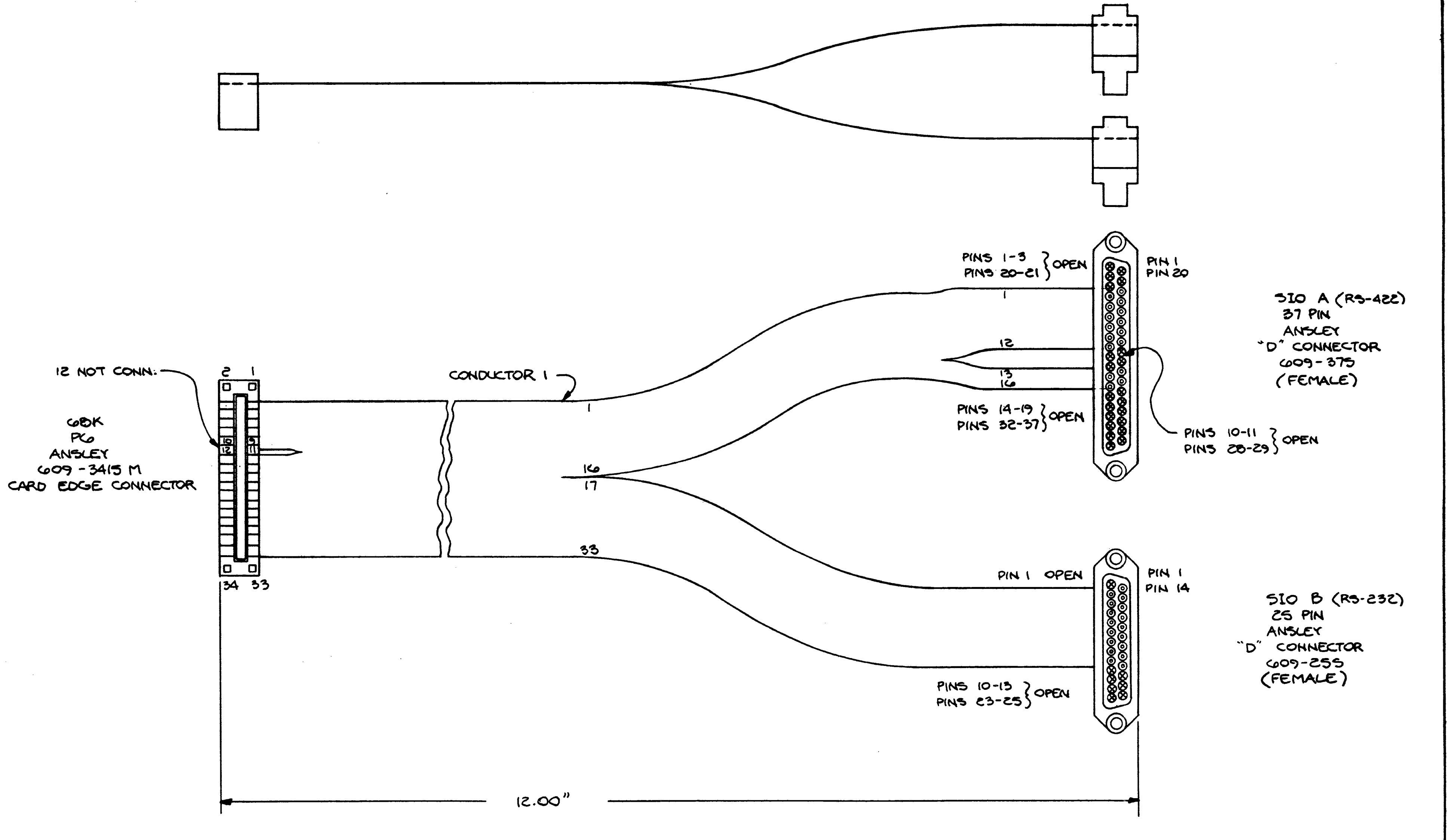
The following chips and components should be installed according to the desired configuration:

Component	RS-232C	RS-422/449
R1 & R2	(remove)	Terminator for Data/Control to device
R3 & R4	"	Terminator for Data/Control from device
U2	"	75174 Data/Control driver
U3	"	75173 Data/Control receiver (use U9)
U4	9636A	(remove)
U7	9636A	"
U9	75173	"

	Pin	"D" Pin	RS-422/449 Function	(direction)
Port A	x	1-3		
	x	20,21		
	P6- 1	22	SD+ Send Data	(to device)
	P6- 2	4	SD- Send Data	(to deivce)
	P6- 3	23	ST+ Send Timing	(to device)
	P6- 4	5	ST- Send Timing	(to device)
	P6- 5	24	RD+ Rcv Data	(from device)
	P6- 6	6	RD- Rcv Data	(from device)
	P6- 7	25	RS+ Request to Send	(to device)
	P6- 8	7	RS- Request to Send	(to device)
	P6- 9	26	RT+ Rcv Timing	(from device)
	P6-10	8	RT- Rcv Timing	(from device)
	P6-11	27	CS+ Clear to Send	(from device)
	P6-12	x	Ground	
	P6-13	9	CS- Clear to Send	(from device)
	x	28		
	x	10		
	x	29		
	x	11		
	P6-14	30	TR+ Terminal Ready	(to device)
	P6-15	12	TR- Terminal Ready	(to device)
	P6-16	31	RR+ Receiver Ready	(from device)
	P6-17	13	RR- Receiver Ready	(from device)
	x	32-37		
	x	14-19		

Port B: P6-18 through P6-34, same as RS-232, see section 12.1, above.

All signals to the device are enabled by jumper J4, which may be set to either allow control via "DTR" from the SCC or to be always enabled. The signal names for the RS-422 port are consistant with the SCC I/O bits, unlike the RS-232 interface. The RS-422 interface is wired as a "data terminal."



HEURIKON CORPORATION MADISON, WISCONSIN	RS422 SERIAL INTERFACE CABLE	DRAWN: RK	DATE: 7-20-82
		SCALE: 1"=1"	CKD: JM

12.5 SCC Initialization Sequence

The following table shows a typical initialization sequence for the SCC:

	Data	Register Address		Function
Port A	00	FEF403	(write)	Reset SCC register counter
	09,C0	"	"	Force reset (do for ports A & C only)
	04,4C	"	"	Async mode, x16 clock, 2 stop bits tx
	05,EA	"	"	Tx: RTS, Enable, 8 data bits
	03,E1	"	"	Rcv: Enable, 8 data bits
	01,00	"	"	No Int, Update status
	0B,56	"	"	No Xtal, Tx & Rcv clk internal,BR out
	0C,baudL	"	"	Set Low half of baud rate constant
	0D,baudH	"	"	Set high half of baud rate constant
	0E,03	"	"	Null, BR enable
Port B	00	FEF401	(write)	Reset...
	~	~	~	~ (repeat above sequence)
Port C	00	FEF503	(write)	Reset...
	~	~	~	~ (repeat above sequence)
Port D	00	FEF501	(write)	Reset..
	~	~	~	~ (repeat above sequence)

Note: the notation "09,C0" (etc) means the values 09 (hex) and C0 should be sent to the specified SCC port. The first byte selects the internal SCC register; the second byte is the control data. The above sequence only initializes the ports for standard asynchronous I/O without interrupts. The 'baudL' and 'baudH' values refer to the low and high halves of the baud rate constant which may be determined from the "Baud Rate Table," section 12.7, below.

For information concerning SCC interrupt vectors, refer to section 4.2.

12.6 Port Address Summary

	Physical Address Value			
Register	Port A	Port B	Port C	Port D
Control	FEF403	FEF401	FEF503	FEF501
Data	FEF407	FEF405	FEF507	FEF505

12.7 Baud Rate Constants

If the internal SCC baud rate generator logic has been selected, the actual baud rate must be specified during the SCC initialization sequence by loading a 16-bit time constant value into each generator. The following table gives the values to use for some common baud rates. Other rates may be generated by applying the formula given below.

	Baud Rate Gen Time Constant (Decimal)	
Baud Rate	x1 clock rate	x16 clock rate
50	39,998	2,498
75	26,665	1,665
110	18,180	1,134
134.5	14,868	927
150	13,331	831
300	6,665	415
600	3,331	206
1200	1,665	102
1800	1,109	67
2400	831	50
3600	554	33
4800	415	24
7200	276	15
9600	206	11
19200	102	5

The time constant values listed above are computed as follows:
(The SCC clock is 4.0 MHz.)

```
TC = Integer { [ 4,000,000 / ( 2 * BaudRate * Factor ) ] - 1.5 }
        where "Factor" is either 1 or 16.
```

There is a tradeoff to consider when selecting the factor (x1 or x16). When using the x1 mode, the baud rate generator is using a larger time constant value and thus the resolution of the counter is greater. This results in a smaller error between the desired baud rate frequency and the actual frequency. (The error occurs because the digital counter cannot be set with a fractional value, only the nearest integer.) This problem is most prevalent for the higher baud rates. On the other hand, the x16 mode will obtain better results with asynchronous protocols because the receiver can search for the middle of the start bit. (In fact, the x1 mode will probably produce frequent receiver errors.)

The maximum SCC data speed is 1 megabit per second, using the x1 clock and synchronous mode. For asynchronous transmission, the maximum practical rate using the x16 clock is 62,500 baud.

12.8 Sample I/O Routines

These two subroutines show a typical sequence for character input and output, using machine code, for character I/O over serial port A. Nothing fancy, such as interrupts, is handled here.

```
        Lable    Opcode   Operand                  Comments
        -------  -------  ----------------------   ----------------------------
        SACNTRL   =       0xFEF403                 *Control port address
        SADATA    =       0xFEF407                 *Data port address

        INPUT:   btst     #0,SACNTRL               *Test Data Ready bit of SCC
                 beq      INPUT                    *Wait for ready

                 move.b   SADATA,d0                *Read byte (port A) to reg D0
                 rts                               *Return


        (D0 has byte to output)
        OUTPUT:  btst     #2,SACNTRL               *Test Transmitter Ready
                 bne      OUTPUT                   *Wait for ready

                 move.b   d0,SADATA                *Output Byte to Port A
                 rts                               *Return
```

The next software example is written in 'C' and shows how the SCC ports may be initialized and used for asynchronous character I/O.

```
#define NOTREADY 0                  /* for returns to BDOS */
#define READY ~NOTREADY

#define PORTAC  0xFEF403            /* define I/O port addresses */
#define PORTAD  0xFEF407
#define PORTBC  0xFEF401
#define PORTBD  0xFEF405
#define PORTCC  0xFEF503
#define PORTCD  0xFEF507
#define PORTDC  0xFEF501
#define PORTDD  0xFEF505

#define RcvREADY 0x01
#define TxREADY  0x04
#define BAUDRATE 9600               /* default baudrate for all SCC ports */

char *pc[] ={
        PORTAC, PORTBC, PORTCC, PORTDC
};

char *pd[] ={
        PORTAD, PORTBD, PORTCD, PORTDD
```

(continued...)

```
sccinit()
{
        char x, i;
        static char scctab[] ={
                0x00,   0x04, 0x4c,     0x05, 0xea,
                        0x03, 0xe1,     0x09, 0x00,
                        0x01, 0x00,     0x0b, 0x56,
                        0x0e, 0x03
        };

        for (x=0; x<4; x++) {
                for(i=0; i<sizeof(scctab); i++)
                        *pc[x]  = scctab[i];
                setbaud(x,BAUDRATE);
        }
}

setbaud(port,rate)
char port;
int rate;
{
        int value;
        *pc[port] = 0x0c;
        *pc[port] = value = ((4000000/(rate<<4))-3)/2;
        *pc[port] = 0x0d;
        *pc[port] = value>>8;
        return;
}


instat(device)
char device;
{
        return ( (*pc[device] & RcvREADY) ? READY : NOTREADY );
}

char portin(device)
char device;
{
        while (instat(device) == NOTREADY) ;
        return (*pd[device] & 0x7f);
}

outstat(device)
char device;
{
        return( (*pc[device] & TxREADY) ? READY : NOTREADY );
}

portout(device, ch)
char device, ch;
{
        while(outstat(device) == NOTREADY);
        return(*pd[device] = ch);
}
```

12.9 Relevant Jumpers (Serial I/O)

Jumper	Function	Pos	Notes
J1	+12 power to P6		
J2	-12 power to P6		
J3	+5 power to P6		
J4	RS-422 Control	J4-A	Enable RS-422 outputs
		J4-B	Control via Port A DTR signal
J5	-12 power to P5		
J6	+12 power to P5		
J7	+5 power to P5		
J15	RS-232 Status Default	J15-A	True
		J15-B	False

13.0 PARALLEL I/O PORTS

The HK68 has two 8-bit parallel I/O ports (P3 and P4), which may be used as general purpose ports. They are intended to be used for Winchester and Streamer Tape I/O. Alternate functions are: use as printer ports or, if both ports are combined, as a 16-bit I/O port for inter-processor communications.

Each port consists of a data latch/buffer and a control PAL. The PALs are programmed for the functions decribed below. (Custom PALs may be created for special applications.)

13.1 Streamer Tape I/O Port (P3)

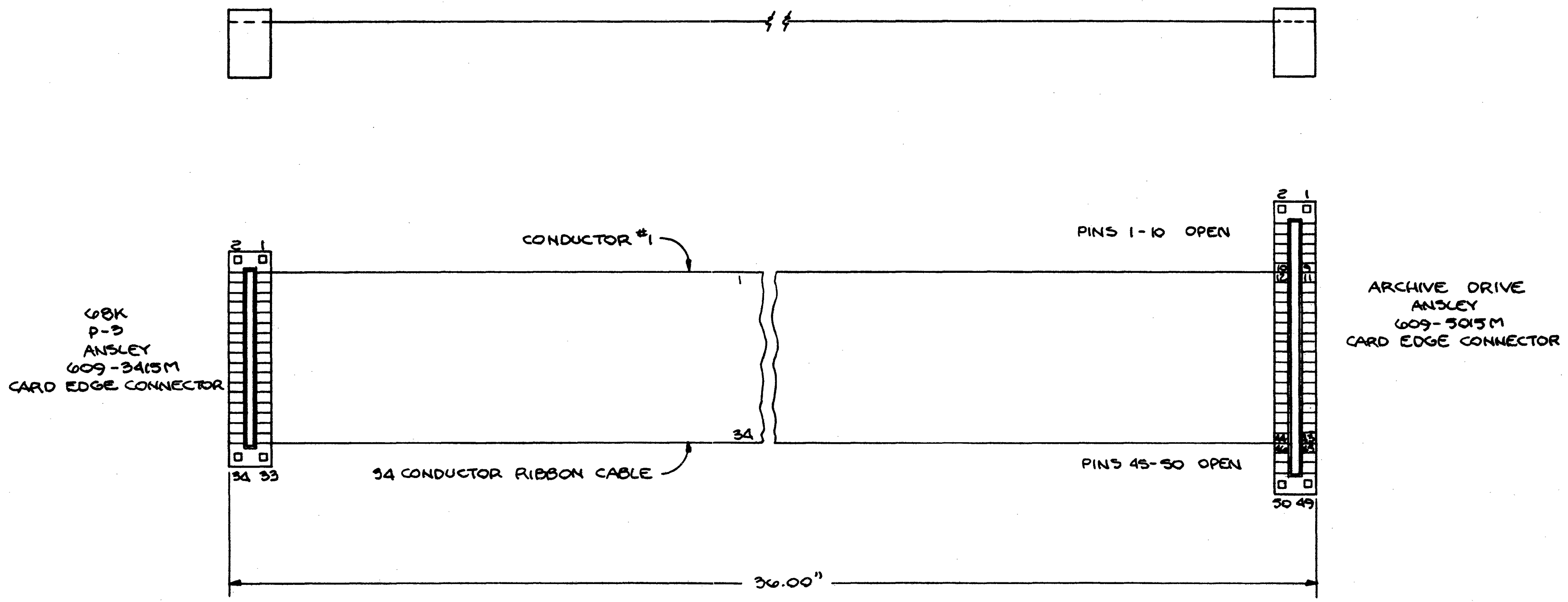
This 8-bit parallel port is designed for direct connection to an Archive Corporation streaming tape drive. This port is also suitable for use as a Centronics compatible printer interface.

	Archive	Streamer	Centronics	Printer
P3 Pin	Signal	Arch Pin	Signal	Cent Pin ("D")
P3- 2	HB7/	12	D7	9
P3- 4	HB6/	14	D6	8
P3- 6	HB5/	16	D5	7
P3- 8	HB4/	18	D4	6
P3-10	HB3/	20	D3	5
P3-12	HB2/	22	D2	4
P3-14	HB1/	24	D1	3
P3-16	HB0/	26	D0	2
P3-18	ONL/	28	x	
P3-20	REQ/	30	x	
P3-22	RESET/	32	RESET/	31
P3-24	XFER/	34	STROBE/	1
P3-26	ACK/	36	x	
P3-28*	READY/	38	BUSY	11
P3-30*	EXC/	40	SLCT	13
P3-32	DIRC/	42	x	
P3-34	n/c	44	x	
P3-odd	Gnd	odd	Gnd	19-30

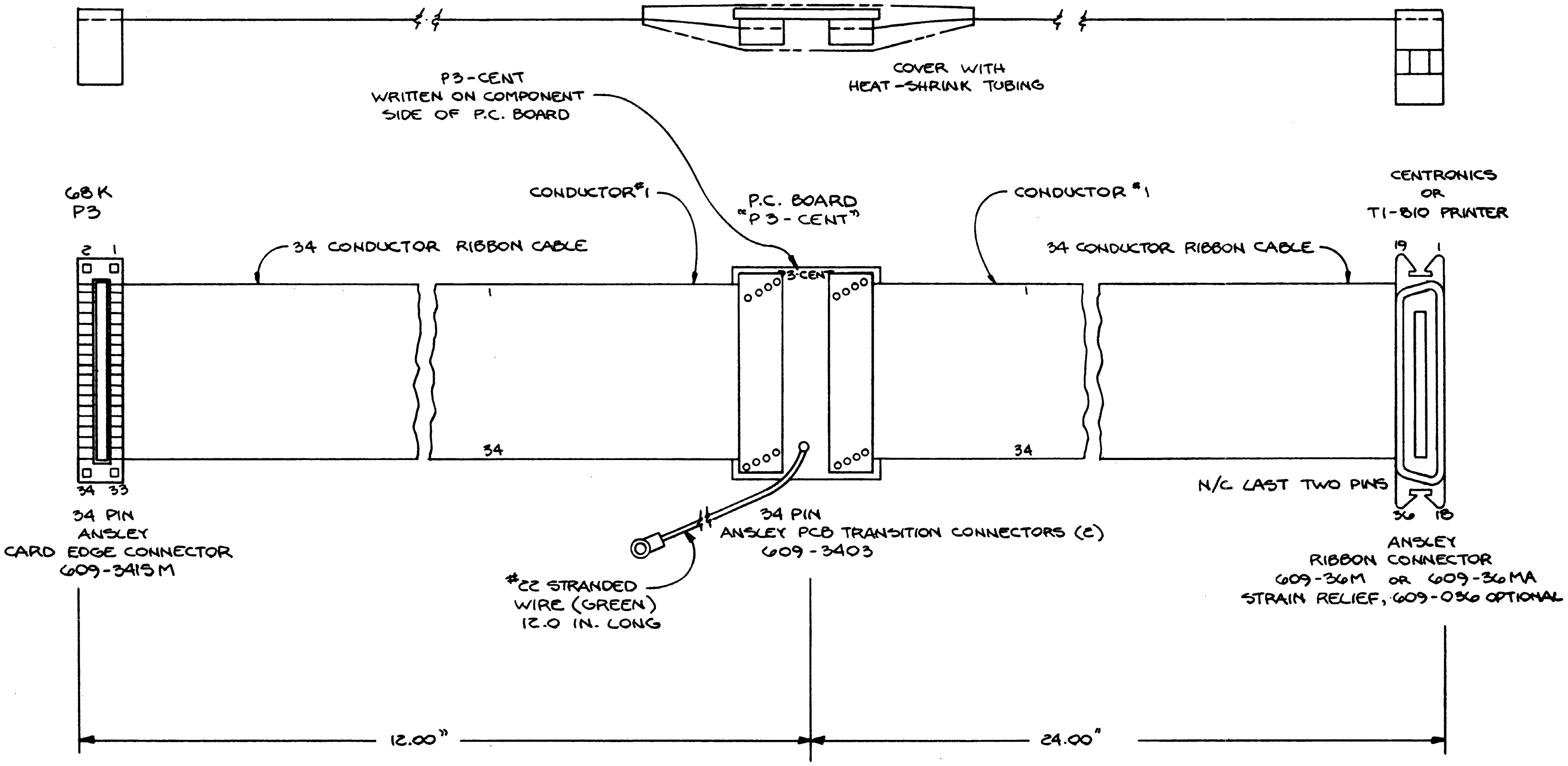
*P3-28 & 30 signals appear on CIO port A D5 & D4, respectively. (See "CIO Usage", section 11.)

The connection to a Centronics device is made via a "crazy cable" which contains an in-line p.c.b to correctly adjust the pinouts.

The data bus interface is inverting. A logical "1" from the MPU translates to a low level on the interface data lines. Data sent to a Centronics-type device should be inverted prior to being output (see software example, below.)



HEURIKON CORPORATION MADISON, WISCONSIN	ARCHIVE STREAMER TAPE INTERFACE	DRAWN: RK	DATE: 7-21-82
		SCALE: 1" = 1"	CKD: JM


COVER WITH HEAT-SHRINK TUBING
P3-CENT WRITTEN ON COMPONENT SIDE OF P.C. BOARD
68K P3
CONDUCTOR #1
P.C. BOARD "P3-CENT"
CONDUCTOR #1
CENTRONICS OR TI-810 PRINTER
34 CONDUCTOR RIBBON CABLE
34 CONDUCTOR RIBBON CABLE
2
1
P3-CENT
1
1
19
1
34
34
34
33
N/C LAST TWO PINS
36
18
34 PIN ANSLEY CARD EDGE CONNECTOR 609-3415M
34 PIN ANSLEY PCB TRANSITION CONNECTORS (2) 609-3403
#22 STRANDED WIRE (GREEN) 12.0 IN. LONG
ANSLEY RIBBON CONNECTOR 609-36M OR 609-36MA STRAIN RELIEF, 609-036 OPTIONAL
12.00"
24.00"

HEURIKON CORPORATION MADISON, WISCONSIN	CENTRONICS ADAPTOR CABLE ASSEMBLY	DRAWN: RK	DATE: 7-21-82
		SCALE: 1"=1"	CRD: JM

This page has been intentionally left blank.

13.2 Control Port Addresses (P3)

The Streamer tape I/F logic uses the following physical memory addresses for data and control functions:

Device: Archive Steamer Tape

Physical Address	Function (read)	Function (write)
FEE000	Read data (8 bits) Set XFER	Write to data latch (8 bits) Set XFER
FEE002	Read data (8 bits) Clear XFER	Write to data latch (8 bits) Clear XFER
FEE004	Read data	Set REQ (P3-20)
FEE006	Read data Clear XFER	Clear REQ Clear XFER
FEE008	Read data	Set ONL (P3-18)
FEE00A	Read data Clear XFER	Clear ONL Clear XFER
FEE00C	D0 = ACK ("1" = true) (P3-26)	Set REQ, Set ONL
FEE00E	D0 = DIRC ("1" true) (P3-32) Clear XFER	Clear REQ, Clear ONL Clear XFER

Device: Centronics Printer

Physical Address	Function (write)	
FEE000	Write to data latch (8 bits)	Set STROBE
FEE002	Write to data latch (8 bits)	Clear STROBE

The printer status bit is read in from the CIO port. Follow this procedure when using the tape port for a Centronics printer: (See the software example, below.)

```
1.  Wait for the printer READY signal. (Test bit D5 in CIO port A.)
2.  Write the character (inverted) to port 0xFEE002.
3.    "    "      "          "      "  "   0xFEE000. (This turns
                                                the STROBE on.)
4.    "    "      "          "      "  "   0xFEE002. (This turns
                                                the STROBE off.)
```

13.3 Software Example (P3)

Centronics printer routines: This is a routine that sends a test message to the streamer tape port configured as a printer interface.

```
/*
 *      Centronics printer test program
 *      Outputs a couple lines to the streamer tape port being used as a
 *      Centronics interface
 */

#define SET_CENTRONICS  (char *) 0xFEE000
#define RES_CENTRONICS  (char *) 0xFEE002
#define STAT_CENTRONICS (char *) 0xFEF605
#define CIO_COMMAND     (char *) 0xFEF607

main()
{
    initcio();          /* Init the CIO */
    print();            /* Print a message */
}

print()
{
    register int cnt;

    putsprn("This is a test of the Centronics printer interface...\n\r");
    putsprn("If this test fails you will know it because you won't be ");
    putsprn("able to read this.\n\r\n\r");
    putsprn("This is a Heurikon HK68 Microcomputer talking.\n\r\n\r");

    for (cnt = ' '; cnt < 0x7F; cnt++)
        putprinter(cnt);
    putsprn("\n\r\n\r");
}


putsprn(p)
register char *p;
{
    while (*p != '\0')
        putprinter(*p++);
}

putprinter(c)                             /* handshake with the printer */
register char c;
{
    while (*STAT_CENTRONICS & 0x20);      /* wait for printer ready */
    *RES_CENTRONICS = ~c;                 /* send data, leave STROBE off */
    *SET_CENTRONICS = ~c;                 /* turn STROBE on */
    *RES_CENTRONICS = ~c;                 /* turn STROBE off */
}
```

```
initcio()
{
    register char c, *p;
    register int cnt;
    static char ciotable[] = {
        0x00,   0x02,                           /* Turn off reset          */

        0x20,   0x00,                           /* Port A Bit control mode */
        0x22,   0x00,                           /* Port A Non inverting    */
        0x23,   0xFF,                           /* Port A All inputs       */
        0x24,   0x01,                           /* Port A Normal exept DOG */

        0x01,   0x04,                           /* Enable channel A        */
    };

    c = *CIO_COMMAND;                           /* Reset register pointer  */
    *CIO_COMMAND = 0x00;                        /* Un-reset command        */
    c = *CIO_COMMAND;                           /* CIO now in state 0      */
    *CIO_COMMAND = 0x00;                        /* Pointer to register 0   */
    *CIO_COMMAND = 0x01;                        /* Reset the CIO           */
    *CIO_COMMAND = 0x00;                        /* Force CIO to state 0    */

    p = ciotable;
    for (cnt = 0; cnt < 12; cnt++)
        *CIO_COMMAND = *p++;
}
```

13.4 Winchester I/O Port (P4)

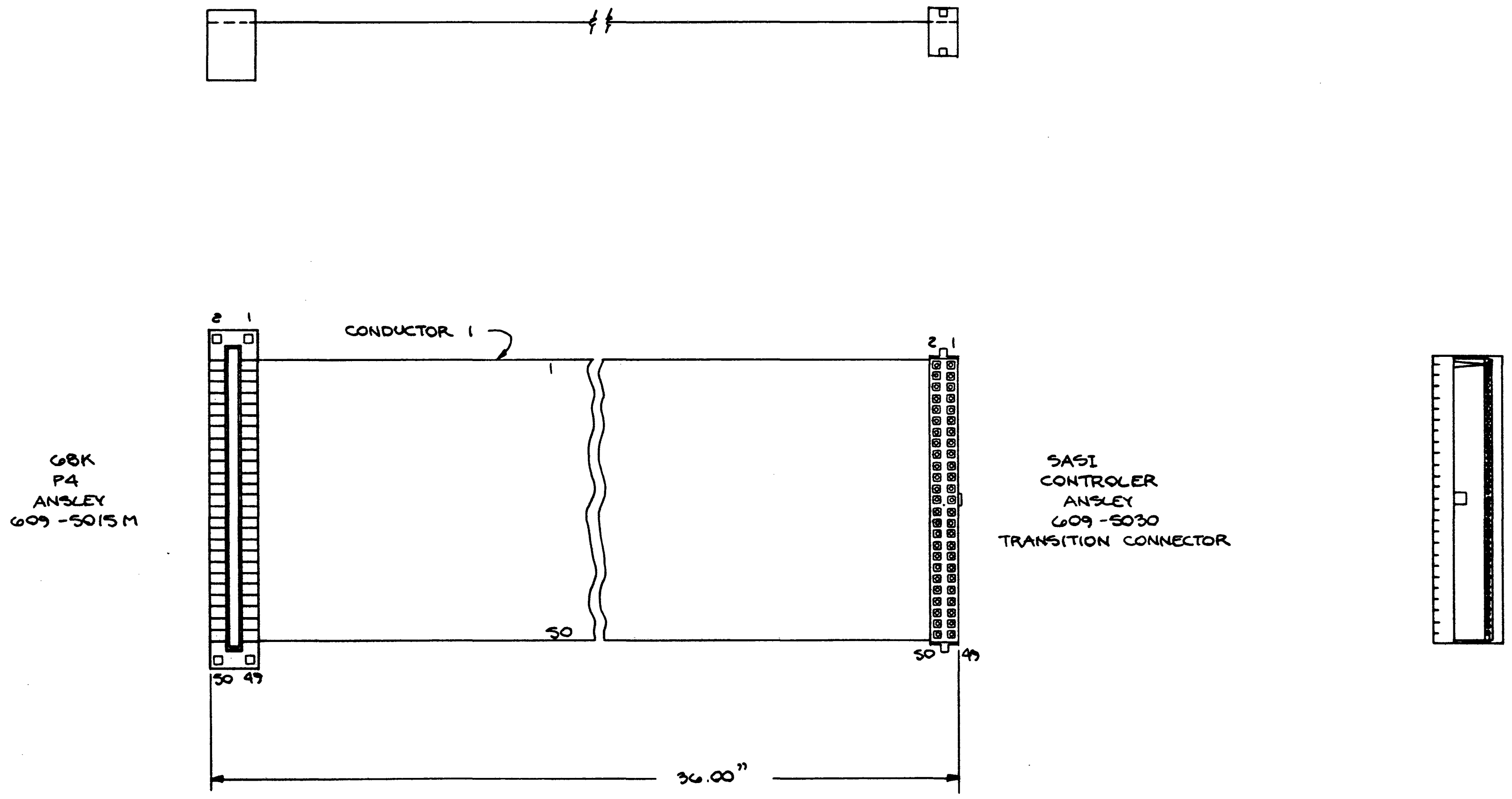
The Winchester port is designed to operate with three controller variations: a SASI (Shugart standard, type 1403D), a Western Digital WD1001 or a Priam "Smart" controller. The cable attached to P4 determines which controller is used. The interface control PAL is programmed to operate with all three controllers. The proper set of PAL equations are automatically selected by the I/F logic through the use of P4 pins 1 and 41. The application program can read the controller type by examining two bits in the DIP Switch port and branch to the appropriate controller routine. (See section 9.2.)

	---Controller	Type----		
P4 Pin	SASI	WestDig	Priam	Notes
P4- 1*	Gnd	open	Gnd	To DIP Switch port, D6
P4- 2	D0/	D0	D0	
P4- 4	D1/	D1	D1	
P4- 6	D2/	D2	D2	
P4- 8	D3/	D3	D3	
P4-10	D4/	D4	D4	
P4-12	D5/	D5	D5	
P4-14	D6/	D6	D6	
P4-16	D7/	D7	D7	
P4-18	x	A0	HAD0	MPU A1
P4-20	x	A1	HAD1	MPU A2
P4-22	x	A2	HAD2	MPU A3
P4-24	x	CS/	x	
P4-26	x	WE/	HWR/	
P4-28	x	RE/	HRD/	
P4-30	x	WAIT/	x	
P4-32	x	x	x	
P4-34	x	x	x	
P4-36*	BUSY/	INTRQ	HIR/	To CIO port A, D1
P4-38	ACK/	DRQ	x	
P4-40	RST/	MR/	RESET/	
P4-41*	Gnd	Gnd	open	To DIP Switch port, D7
P4-42	MSG/	x	x	
P4-44	SEL/	x	x	
P4-46*	C/D	x	DBUSENA/	To CIO port A, D2
P4-48	REQ/	x	DTREQ/	
P4-50*	I/O	x	HREAD	To CIO port A, D3

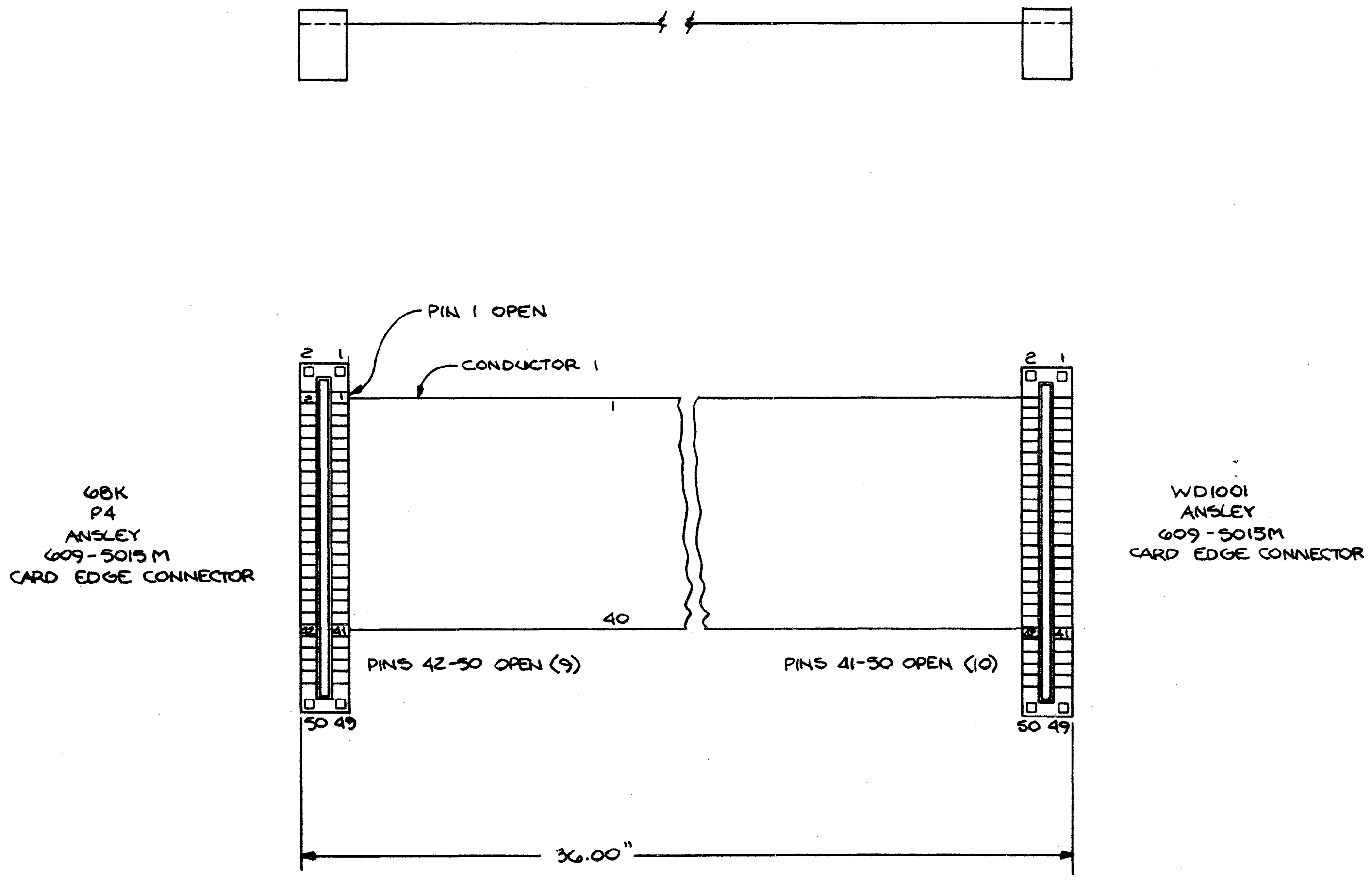
P4-odd (except P4-1 and P4-41) are connected to ground (23 pins)

*P4-36, 46 & 50 appear on CIO port A. (See "CIO Usage.")
P4-1 & P4-41 connect to the DIP Switch port. (See section 9.2.)

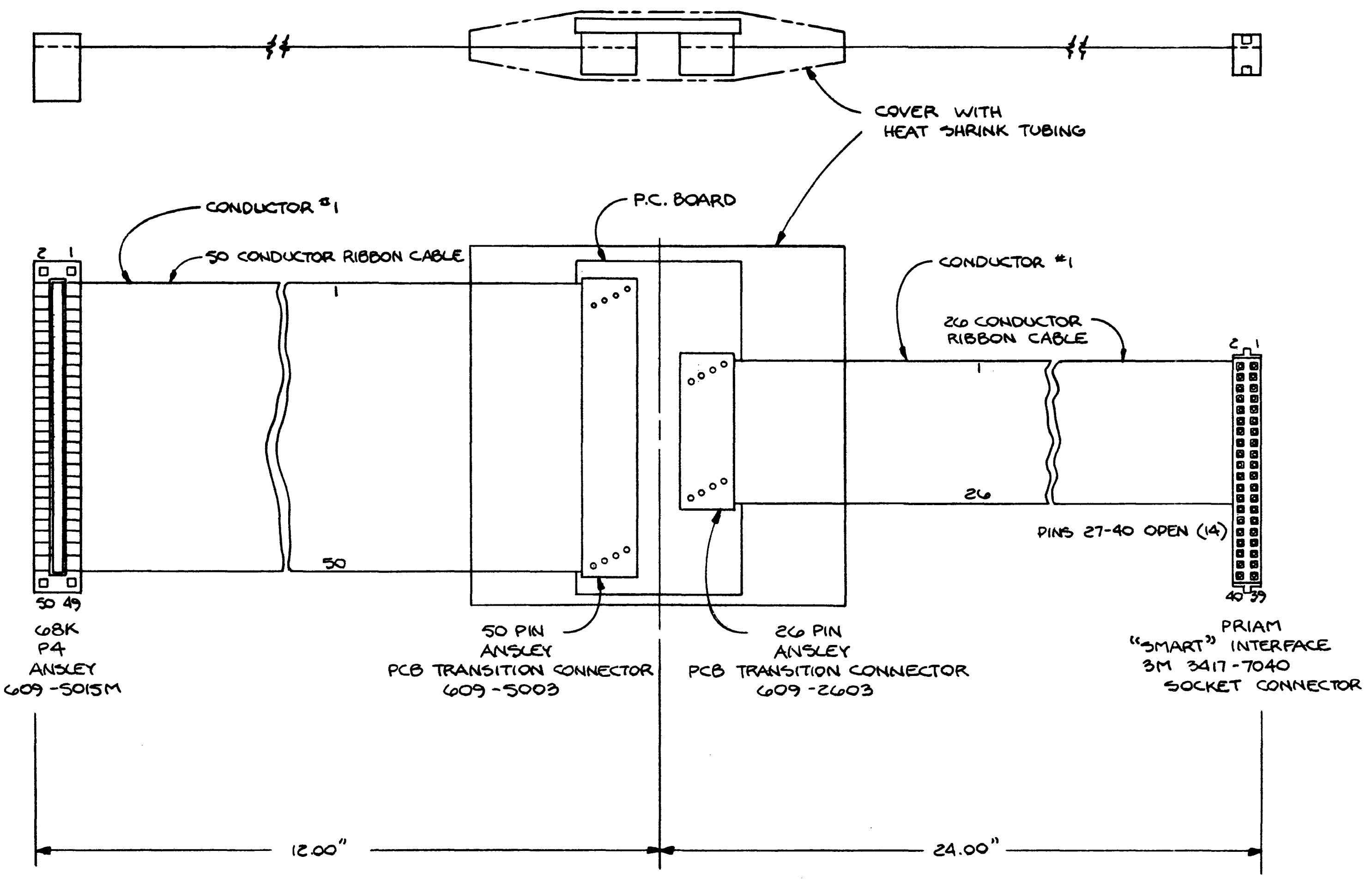
The data bus I/F is inverting. Thus, a high level on an MPU data line translates to (or from) a low level on the P4 data pins (P4-2 through P4-16). Logically, the SASI type controllers expect the I/F data to be low true. The Western Digital and Priam controllers, however, expect a high true data bus. Thus, for those controllers, the MPU and DMAC should invert data, commands and status bytes prior to, or following, transfers over the Winchester interfaces. Note that it should not be necessary to actually invert the data, because any data inverted by the I/F while writing to the disk will be re-inverted when reading back. Also, the commands and status bytes need not be inverted at run time if they are appropriately defined in the program at compile time as inverted constants.



HEURIKON CORPORATION MADISON, WISCONSIN	SHUGART ASSOC. STANDARD I/F SASI I/F CABLE	DRAWN: RK	DATE: 7-20-82
		SCALE: 1"=1"	CKD: JM



HEURIKON CORPORATION MADISON, WISCONSIN	WESTERN DIGITAL INTERFACE CABLE	DRAWN: RK	DATE: 7-20-82
		SCALE: 1"=1"	CKD: JM



COVER WITH HEAT SHRINK TUBING
CONDUCTOR #1
50 CONDUCTOR RIBBON CABLE
P.C. BOARD
CONDUCTOR #1
26 CONDUCTOR RIBBON CABLE
2 1
1
50
26
PINS 27-40 OPEN (14)
50 49
40 39
68K P4 ANSLEY 609-5015M
50 PIN ANSLEY PCB TRANSITION CONNECTOR 609-5003
26 PIN ANSLEY PCB TRANSITION CONNECTOR 609-2603
PRIAM "SMART" INTERFACE 3M 3417-7040 SOCKET CONNECTOR
12.00"
24.00"

HEURIKON CORPORATION MADISON, WISCONSIN	PRIAM INTERFACE CABLE	DRAWN: RK	DATE: 7-21-82
		SCALE: 1" = 1"	CKD: JM

13.5 Control Port Addresses (P4)

The Winchester I/F logic uses the following physical memory addresses for data and control functions:

Device: Winchester (P4) with WD1000/1001 Controller

Physical Address	Function (read)	Function (write)
FEE001	Read Data	Write Data
FEE003	Read Error Register	Set Write Prcomp Register
FEE005	Read Sector Count Reg	Write Sector Count Reg
FEE007	Read Sector Number Reg	Write Sector Number Reg
FEE009	Read Cylinder Low Reg	Write Cylinder Low Reg
FEE00B	Read Cylinder High Reg	Write Cylinder High Reg
FEE00D	Read Size/Drive/Head	Write Size/Drive/Head Reg
FEE00F	Read Status Register	Write Command Register

Device: Winchester (P4) with SASI Controller

Physical Address	Function (read)	Function (write)
FEE001	Read data (8 bits)	Write data (8 bits)
FEE003	Read data	Write data
FEE005	Turn on I/F D0 Set SEL (P4-44)	Turn on I/F D0 Set SEL (P4-44)
FEE007	Clear SEL Release I/F D0	Clear SEL Release I/F D0

Device: Winchester (P4) with Priam SMART Controller

Physical Address	Function (read)	Function (write)
FEE001	Read Interface Status	Write Command
FEE003	Read Data	Write Data
FEE005	Read Result Reg 0	Write Parameter 0
FEE007	Read Result Reg 1	Write Parameter 1
FEE009	Read Result Reg 2	Write Parameter 2
FEE00B	Read Result Reg 3	Write Parameter 3
FEE00D	Read Result Reg 4	Write Parameter 4
FEE00F	Read Result Reg 5	Write Parameter 5

13.6 16-Bit Parallel I/O Port (P3 & P4)

P3 and P4 may be combined into a single 16-bit port for high speed data transfers. This could be used for inter-processor communications or a 16-bit external device. When used in this mode, P3 carries the upper eight data bits, and P4 carries the lower eight bits.

The handshake scheme is similar to the IEEE-488 3-wire handshake.

P3 Pin	Signal	P4 Pin	Signal
P3- 2	D15/	P4- 1	To DIP Sw port, D6
P3- 4	D14/	P4- 2	D0/
P3- 6	D13/	P4- 4	D1/
P3- 8	D12/	P4- 6	D2/
P3-10	D11/	P4- 8	D3/
P3-12	D10/	P4-10	D4/
P3-14	D9/	P4-12	D5/
P3-16	D8/	P4-14	D6/
		P4-16	D7/
P3-18	ONL/		
P3-20	REQ/	P4-18	A1
P3-22	RESET/	P4-20	A2
P3-24	XFER/	P4-22	A3
P3-26	ACK/		
P3-28	To CIO port A, D5	P4-36	To CIO port A, D1
P3-30	To CIO port A, D4	P4-41	To DIP Sw port, D7
P3-32	DIRC/	P4-46	To CIO port A, D2
P3-34	n/c	P4-50	To CIO port A, D3
P3-odd	Gnd	P4-odd (except 1 & 41)	Gnd

Device: 16-bit Parallel Port

Physical Address	Function (read)	Function (write)
FEE000	Read Data (16 bits) Set XFER	Write to data latch (16 bits) Set XFER
FEE002	Read data (16 bits) Clear XFER	Write to data latch (16 bits) Clear XFER
FEE00C	D0 = ACK ("1" = true) (P3-26)	Set REQ, Set ONL
FEE00E	D0 = DIRC ("1" true) (P3-32) Clear XFER	Clear REQ, Clear ONL Clear XFER

More details of this mode will be provided with a later manual release.

13.7 Use with the DMAC

Channel 0 of the DMAC is dedicated for Winchester I/O operations in that the Winchester ready signal is wired directly to the DMAC channel 0 ready input and the DMAC channel 0 acknowledge line returns to the Winchester logic. This makes the single addressing mode possible for high speed data transfers. Refer to section 5 for DMAC details.

The Streamer Tape port may be used with channel 1 of the DMAC by setting jumper J9-1,2 and J9-6,7.

Set J11 according to the desired DMA mode (see section 5.)

13.8 Relevant Jumpers (Parallel I/O)

Jumper	Function	Pos	Notes
J9	DMA Request/Ack	J9-1,2	DMA Ack1 to Tape Acknowledge
		J9-3,4	SIO A Ready to DMA Req 2
		J9-3,7	SIO A Ready to DMA Req 1
	See section 5 for	J9-4,5	SBX P8 Ready to DMA Req 2
	more info on the	J9-4,8	SBX P7 Ready to DMA Req 2
	DMAC logic.	J9-5,9	SBX P8 Ready to DMA Req 3
		J9-6,7	Tape Ready to DMA Req 1
		J9-7,8	SBX P7 Ready to DMA Req 1
		J9-8,9	SBX P7 Ready to DMA Req 3
		J9-9,10	SIO B Ready to DMA Req 3
J11	Winc/Tape DMA Mode	Install for Single Address transfers to the Winchester or Tape ports. Remove for Dual Address transfers	

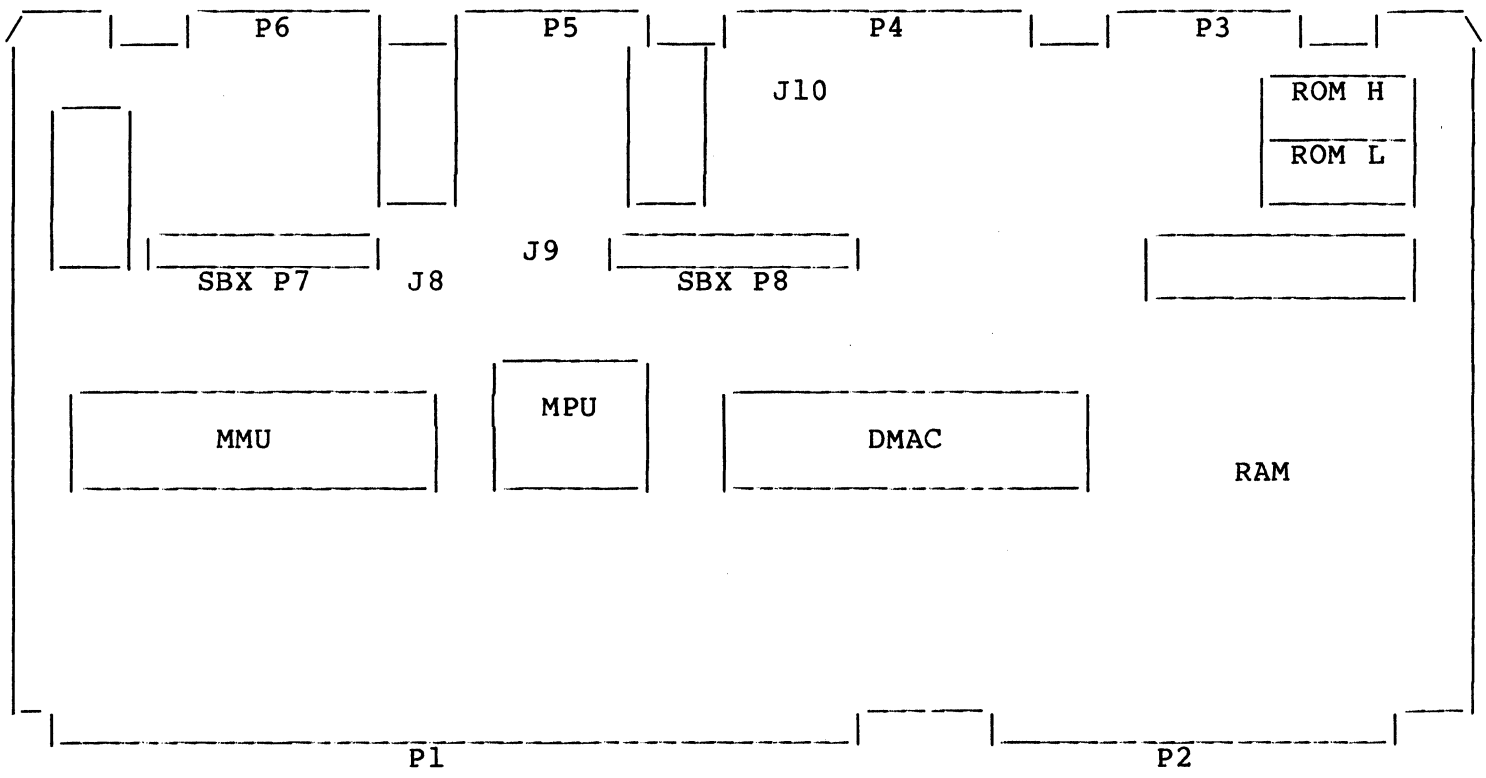
14.0 SBX Expansion I/O Interface

The HK68 board has provisions for two SBX modules. These modules allow users to expand the I/O capabilities of the board by adding appropriate modules. The following list shows some of the modules available:

Function	Part Nmbr	Manufacturer	Module ID
Floppy Disk		Heurikon	xxx00001
Quad Serial I/O		Heurikon	xx000010
*Math Processor (APU)		Heurikon	xxx00011
*GPIB Bus I/F		Heurikon	xxx00100
*Real Time Clock		Heurikon	xxx00101
*Modem		Heurikon	xxx00110
*Ethernet I/F		Heurikon	xxx00111
Parallel I/O	iSBX 350	Intel	(n/a)
Analog Input	iSBX 311	Intel	(n/a)
Analog Output	iSBX 328	Intel	(n/a)
Video Display	iSBX 270	Intel	(n/a)

* future product

The two connectors are arranged so that one standard module and one double width module may be used. The connectors on the HK68 board are designated "P7" and "P8.) Both can accommodate either a single or double-width module. (P7's module, however, must be single-width in order to use P8.)



We advise that you do not rely on convection cooling when using an SBX module. The serial interface logic, located above P7 and P8, runs warm.

14.1 SBX Connector Pin Assignments

Both connectors are effectively wired in "parallel," except for the control signals which appear on the even numbered pins.

Pin	P7	P8	Pin	P7	P8	
1	+12	+12	2	-12	-12	
3	Gnd	Gnd	4	Vcc	Vcc	
5	RESET	RESET	6	XCLK	XCLK	(from J10)
7	A3	A3 (MPU)	8	MPS-0	MPS-1	(DIP Sw Port)
9	A2	A2 (MPU)	10	n/c	n/c	
11	A1	A1 (MPU)	12	J8-A,B	J8-C,D	(see J8)
13	WR/	WR/	14	XINT0	XINT1	(to CIO)
15	RD/	RD/	16	WAIT/	WAIT/	
17	Gnd	Gnd	18	Vcc	Vcc	
19	D7	D7	20	CE1/	CE3/	(see below)
21	D6	D6	22	CE0/	CE2/	(see below)
23	D5	D5	24	n/c	n/c	
25	D4	D4	26	n/c	n/c	
27	D3	D3	28	SOPT01	SOPT11	(to CIO)
29	D2	D2	30	SOPT00	SOPT10	(to CIO)
31	D1	D1	32	n/c	n/c	
33	D0	D0	34	DMARDY	DMARDY	(to J9)
35	Gnd	Gnd	36	Vcc	Vcc	

14.2 Device Address Summary (SBX)

The functions assigned to each port address are determined by the particular module attached to the port.

	SBX pins			P7 Address		P8 Address	
	7	9	11	CE0	CE1	CE2	CE3
	0	0	0	FEF001	FEF101	FEF201	FEF301
	0	0	1	FEF003	FEF103	FEF203	FEF303
	0	1	0	FEF005	FEF105	FEF205	FEF305
	0	1	1	FEF007	FEF107	FEF207	FEF307
	1	0	0	FEF009	FEF109	FEF209	FEF309
	1	0	1	FEF00B	FEF10B	FEF20B	FEF30B
	1	1	0	FEF00D	FEF10D	FEF20D	FEF30D
	1	1	1	FEF00F*	FEF10F	FEF20F*	FEF30F
MPU:	"A3	A2	A1"				
SBX:	"A2	A1	A0"	P7-22	P7-20	P8-22	P8-20

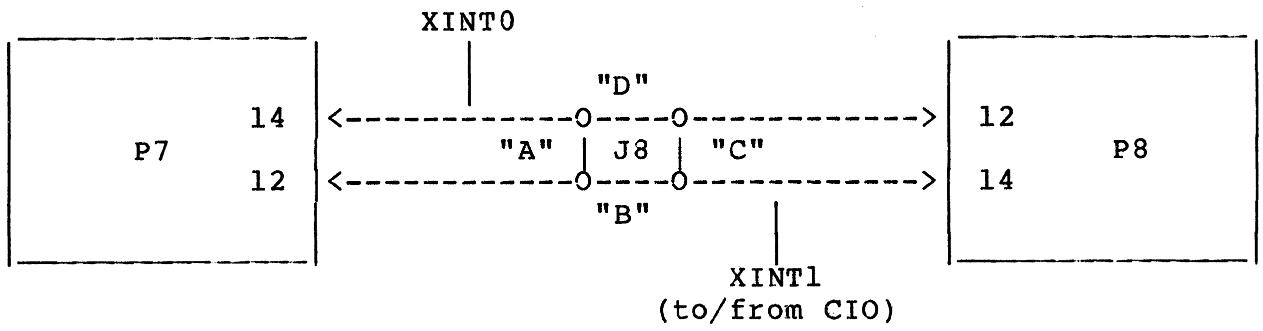
*Addresses FEF00F and FEF20F are used by the Heurikon SBX modules to read a "Module ID" code number, which uniquely identifies the type of module plugged into the respective SBX connector. (In the event that these addresses are required for other SBX functions, only the first access following a reset will provide the ID code.) This feature allows a program to dynamically configure itself according to the function and placement of the modules installed on the board.

Before accessing the SBX modules, the program should test the module present bits (MPS-0 and MPS-1) via the DIP switch port to be sure the modules are plugged in. See section 9.2 for details.

14.3 Relevant Jumpers (SBX Expansion)

Jumper	Function	Pos	Notes
J8	SBX Interrupt Config (See diagram, below)	J8-A	SBX P7 pin 12 (INT1) to XINT0
		J8-B	SBX P7 pin 12 (INT1) to XINT1
		J8-C	SBX P8 pin 12 (INT1) to XINT1
		J8-D	SBX P8 pin 12 (INT1) to XINT0
J9	DMA Request/Ack	J9-1,2	DMA Ack1 to Tape Acknowledge
		J9-3,4	SIO A Ready to DMA Req 2
		J9-3,7	SIO A Ready to DMA Req 1
		J9-4,5	SBX P8 Ready to DMA Req 2
		J9-4,8	SBX P7 Ready to DMA Req 2
		J9-5,9	SBX P8 Ready to DMA Req 3
		J9-6,7	Tape Ready to DMA Req 1
		J9-7,8	SBX P7 Ready to DMA Req 1
		J9-8,9	SBX P7 Ready to DMA Req 3
		J9-9,10	SIO B Ready to DMA Req 3
J10	SBX Clock Select	J10-A	8 MHz
		J10-B	10 MHz (option)

Jumper J8 diagram:



15.0 PHYSICAL ADDRESS MAPPING

```
          Hex        Function       Notes

          FFFFFF  ----------------- (Top of 16 Meg adrs space)
                  |               |
                  |  Multibus I/O |
                  |     (64k)     |
                  |               |
          FF0000  -----------------
                  |  CIO,SCC,SBX  |       Ref sections 11, 12, 14
          FEF000  -----------------
                  |  WINC, TAPE   |       Ref section 13
          FEE000  -----------------
                  |       x       |
                  -----------------
                  | DIP,LED,CNTRL |       Ref section 9
          FEC000  -----------------
                  |       x       |
                  -----------------
                  |       x       |
          FEA000  -----------------
                  |     DMAC      |       Ref section 5
          FE9000  -----------------
                  |     MMU       |       Ref section 6
          FE8000  -----------------
                  |     ROM       |
                  |    (32k)      |  ^^^ Upper 128k bytes ^^^^
          FE0000  -----------------  -------------------------
                  |               |
                  |   Multibus    |
                  |    Memory     |       Ref section 7
            ~     ~       ~       ~
            ~     ~       ~       ~
                  |               |
          100000  ---------       | (Top of 1 Meg adrs space)
                  | RAM   |       |
                  | 512k  |       |      If Deck 2 is not present,
                  | Deck2 |       |      the deck 2 space is released
                  |       |       |      to the Multibus.
          080000  ---------       |
                  | RAM   |       |
                  | 512k  |_______|
                  | Deck1 | 128-2 |
                  |       | 128-1 |
          000000  ----------------- (Bottom of memory)
                   256kx1  64k x 1
```

Not shown to scale.

After an external reset (Power-up or Reset P.B.), RAM is turned off and ROM appears at address 000000 until any address above FE0000 is accessed. Refer to section 7.4 for a power-up memory map. If jumper J23 is removed, no on-card RAM exists. (All addresses below FE0000 will be on the Multibus.)

16.0 I/O PORT ADDRESSES

All I/O devices are memory mapped into the top of the 16 megabyte physical address space. The top most 64K memory locations (FF0000 hex and above) are translated into I/O commands on the Multibus with the lower 16 address lines providing the 16-bit Multibus I/O device number. The next lower addresses are assiqned to the on-card devices according to the table below.

The logical addresses (those used by the MPU and DMAC) could differ from the physical addresses as a result of action by the MMU. This table lists the PHYSICAL address values.

	Physical Address Lines																
DEVICE	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	HEX (base)
Multibus	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	FFxxxx
(spare)	1	1	1	1	x	1	1	1	x	x	x	x	x	x	x	x	FEF7xx
CIO	1	1	1	1	x	1	1	0	x	x	x	x	x	D	D	1	FEF6x1
SCC(1)	1	1	1	1	x	1	0	1	x	x	x	x	x	D	D	1	FEF5x1
SCC(0)	1	1	1	1	x	1	0	0	x	x	x	x	x	D	D	1	FEF4x1
SBX(3)	1	1	1	1	x	0	1	1	x	x	x	x	D	D	D	1	FEF3x1
SBX(2)	1	1	1	1	x	0	1	0	x	x	x	x	D	D	D	1	FEF2x1
SBX(1)	1	1	1	1	x	0	0	1	x	x	x	x	D	D	D	1	FEF1x1
SBX(0)	1	1	1	1	x	0	0	0	x	x	x	x	D	D	D	1	FEF0x1
TAPE	1	1	1	0	x	x	x	x	x	x	x	x	D	D	D	0	FEExx0
WINC	1	1	1	0	x	x	x	x	x	x	x	D	D	D	D	1	FEExx1
CONTROL	1	1	0	0	x	x	x	x	x	x	x	x	x	x	x	0	FECxx0 (write)
LED	1	1	0	0	x	x	x	x	x	x	x	x	x	x	x	1	FECxx1 (write)
RING	1	1	0	0	x	x	x	x	x	x	x	x	x	x	x	0	FECxx0 (read)
DIPSW	1	1	0	0	x	x	x	x	x	x	x	x	x	x	x	1	FECxx1 (read)
DMAC	1	0	0	1	x	x	x	x	D	D	D	D	D	D	D	D	FE9x00
MMU	1	0	0	0	x	x	x	x	x	x	D	D	D	D	D	D	FE8x00

NOTES: "D" means that address bit function is defined by the device.
"x" means don't care

A0 = "D" means the device can be used with 8 or 16 bit data
A0 = "0" means the device uses the upper 8 data bits only
A0 = "1" means the device uses the lower 8 data bits only.

"write" means the device is write only
"read" means the device is read only

Refer to individual device descriptions for details on the structure of the I/O devices.

17.0 HARDWARE JUMPERS

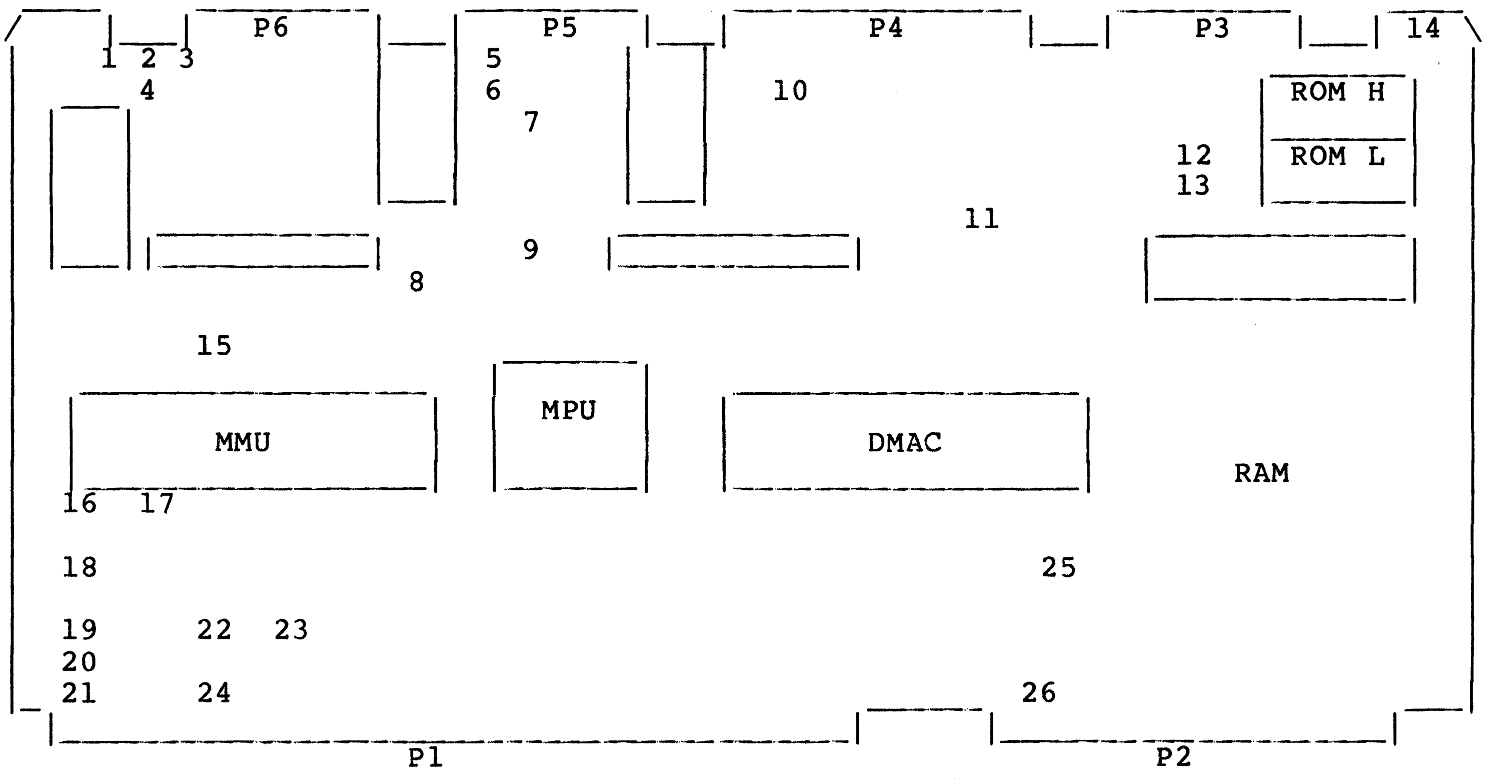
17.1 Jumper Functions and Settings

Jumper	Function	Pos	Notes
J1	+12 power to P6		
J2	-12 power to P6		
J3	+5 power to P6		
J4	RS-422 Control	*J4-A	Enable RS-422 outputs
		J4-B	Control via Port A DTR signal
J5	-12 power to P5		
J6	+12 power to P5		
J7	+5 power to P5		
J8	SBX Interrupt Config	J8-A	SBX P7 pin 12 (INT1) to XINT0
		J8-B	SBX P7 pin 12 (INT1) to XINT1
		J8-C	SBX P8 pin 12 (INT1) to XINT1
		J8-D	SBX P8 pin 12 (INT1) to XINT0
J9	DMA Request/Ack	*J9-1,2	DMA Ack1 to Tape Acknowledge
		J9-3,4	SIO A Ready to DMA Req 2
		J9-3,7	SIO A Ready to DMA Req 1
		J9-4,5	SBX P8 Ready to DMA Req 2
		*J9-4,8	SBX P7 Ready to DMA Req 2
		*J9-5,9	SBX P8 Ready to DMA Req 3
		*J9-6,7	Tape Ready to DMA Req 1
		J9-7,8	SBX P7 Ready to DMA Req 1
		J9-8,9	SBX P7 Ready to DMA Req 3
		J9-9,10	SIO B Ready to DMA Req 3
J10	SBX Clock Select	*J10-A	8 MHz
		J10-B	10 MHz (option)
J11	Winc/Tape DMA Mode	Install for Single Address transfers to the Winchester or Tape ports.	
		*Remove for Dual Address transfers	
J12	RAM Speed/Parity	*J12-A	Fast (no parity)
		J12-B	Slow, Parity Enabled
J13	ROM Type Select	J13-A, J14-A	2716 type
J14	ROM Type Select	*J13-A, J14-B	2732, 2764 type
		J13-B, J14-B	27128 type
J15	RS-232 Status Default	*J15-A	True
		J15-B	False
J16	Miscellaneous Control	*J16-A	Convention = Motorola
		J16-B	Convention = latched D15 ("0" = Motorola)
		J16-(not A or B)	Conv = Zilog
		J16-C	Bus Wr Prot = latched D15 ("0" = disable)
		*J16-D	Bus Write Protect disabled
		J16-(not C or D)	Bus WProt enabled

J17	MMU Mode Select	*J17-A	Mode S1 (factory adjustment)
		J17-B	Mode S2
J18	Bus Clock Rate Select	*J18-A	8 MHz
		J18-B	10 MHz (option)
J19	Bus CCLK Enable	*Install on highest priority board	
J20	Bus BPRN Enable	*Install on highest priority board	
J21	Bus BCLK Enable	*Install on highest priority board	
J22	RAM Device Size Select	*J22-A	64K x 1
		J22-B	256K x 1
J23	RAM Deck Select	J23-A	Single Deck
		*J23-B	Double-Decker
		(none)	on-card RAM disabled
J24	Bus INIT/ Select (P1-14)	J24-A	INIT/ = on-card RESET/ (out)
		*J24-B	INIT/ = AuxRESET/ (Input)
J25	Watchdog Disable	*Install to disable Watchdog Timer	
J26	Bus LOCK/ Enable	Install to enable Bus LOCK fuction	

* indicates standard jumper configuration

17.2 Jumper Locations



Selected portions of this listing appear in other parts of this manual - within the sections describing the particular features.

18.0 SOFTWARE INITIALIZATION SUMMARY

This section outlines the steps for initializing the facilities on the HK68 board. Certain steps must be performed in sequence, while others may be rearranged or omitted entirely, depending on your application.

Item	Step
1.	Fetch the reset vector. This is done automatically by the the 68000 following a system reset to load the supervisor stack pointer and program counter. The reset vector is in the first 8 bytes of the ROM.
2.	Determine RAM configuration. (Reference: section 7.6)
3.	Turn on the RAM by branching to the ROM (at base FE0000). (Reference: section 7)
4.	Clear RAM to prevent parity errors due to uninitialized memory reads. (Reference: section 8.2)
5.	Setup the exception vector table in RAM (at base 000000.) This step links the various exception and interrupt sources with the appropriate service routine. (Reference: sections 4.2 and 4.3)
6.	Initialize the MMU. (Reference: section 6)
7.	Initialize the CIO. (See sample program in "CIO Usage," section 11.8.)
8.	Initialize the Bus Control Latch. (Reference: section 10.5)
9.	Initialize the Serial ports. (Reference: section 12.5)
10.	Initialize the User LED port. (Reference: section 9.4)
11.	Read the SBX module ID codes (Heurikon only), and initialize the modules, as required. (Reference: section 14)
12.	Initialize off-card memory and I/O devices, as necessary.
13.	Enable system interrupts, as desired. (Reference: section 4.1)

19.0 MULTIBUS INTERFACE

(See also "Memory Configuration," section 7, and "Bus Control," section 10.)

The Multibus consists of Pl and P2 address, data, and control signals. The following tables indicate which signals are used on each portion of the bus.

19.1 Connector Pl Pin Assignments

Pl Pin	Signal	Pl Pin	Signal
Pl- 1	Gnd	Pl- 2	Gnd
Pl- 3	Vcc	Pl- 4	Vcc
Pl- 5	Vcc	Pl- 6	Vcc
Pl- 7	+12	Pl- 8	+12
Pl- 9	(reserved)	Pl-10	(reserved)
Pl-11	Gnd	Pl-12	Gnd
Pl-13	BCLK/	Pl-14	INIT/
Pl-15	BPRN/	Pl-16	BPRO/
Pl-17	BUSY/	Pl-18	BREQ/
Pl-19	MRDC/	Pl-20	MWTC/
Pl-21	IORC/	Pl-22	IOWC/
Pl-23	XACK/	Pl-24	(reserved)
Pl-25	LOCK/	Pl-26	(reserved)
Pl-27	BHEN/	Pl-28	ADR10/
Pl-29	CBRQ/	Pl-30	ADR11/
Pl-31	CCLK/	Pl-32	ADR12/
Pl-33	(reserved)	Pl-34	ADR13/
Pl-35	INT6/	Pl-36	INT7/
Pl-37	INT4/	Pl-38	INT5/
Pl-39	INT2/	Pl-40	INT3/
Pl-41	INT0/	Pl-42	INT1/
Pl-43	ADRE/	Pl-44	ADRF/
Pl-45	ADRC/	Pl-46	ADRD/
Pl-47	ADRA/	Pl-48	ADRB/
Pl-49	ADR8/	Pl-50	ADR9/
Pl-51	ADR6/	Pl-52	ADR7/
Pl-53	ADR4/	Pl-54	ADR5/
Pl-55	ADR2/	Pl-56	ADR3/
Pl-57	ADR0/	Pl-58	ADR1/
Pl-59	DATE/	Pl-60	DATF/
Pl-61	DATC/	Pl-62	DATD/
Pl-63	DATA/	Pl-64	DATB/
Pl-65	DAT8/	Pl-66	DAT9/
Pl-67	DAT6/	Pl-68	DAT7/
Pl-69	DAT4/	Pl-70	DAT5/
Pl-71	DAT2/	Pl-72	DAT3/
Pl-73	DAT0/	Pl-74	DAT1/
Pl-75	Gnd	Pl-76	Gnd
Pl-77	(reserved)	Pl-78	(reserved)
Pl-79	-12	Pl-80	-12
Pl-81	Vcc	Pl-82	Vcc
Pl-83	Vcc	Pl-84	Vcc
Pl-85	Gnd	Pl-86	Gnd

Refer to section 10.1 for signal descriptions.

19.2 Connector P2 Pin Assignments

P2 Pin	Signal		P2 Pin	Signal
P2- 1	Gnd		P2- 2	Gnd
P2- 2	(reserved)	thru	P2-18	(reserved)
P2-19	PFIN/		P2-20	
P2-21	Gnd		P2-22	Gnd
			P2-38	AuxRESET/
P2-23	(reserved)	thru	P2-54	(reserved)
P2-55	ADR16/		P2-56	ADR17/
P2-57	ADR14/		P2-58	ADR15/
P2-59			P2-60	

19.3 Multibus Compliance Levels

```
Master: D16 M24 I16 V0 L
Slave:  D16 M24     V0 L
```

19.4 Power Requirements

Voltage	Current	Usage
+5	4.0A, max	All logic
+12	1.0A, max	Timing logic, RS-232 I/F
-12	1.0A, max	RS-232 I/F

NOTICE: Power dissipation is 20 watts, nominal. The SCC, CIO serial I/F and some bus logic runs warm to hot. Fan cooling should be used if the 68K board is placed in an enclosure, or if other boards are used in the card cage.

19.5 Mechanical

Length	Width
12.00 in.	6.75 in.

--------Maximun-Thickness-------- (height above board)

RAM decks:	1 deck	2 decks	
w/o SBX:	0.400 in.	0.500 in.	
with SBX:	0.810 in.	0.810 in.	(Includes SBX module maximum thickness.)

HK68 boards with double decker RAM sockets (but without SBX modules) will fit side-by-side in card racks with 0.6 inch slot spacing.

20.0 TIDBITS

These things don't seem to fit anywhere else...

20.1 PAL Usage

The Heurikon HK68 makes substantial use of Programmable Array Logic to implement logical functions. This technique reduces the chip count for the random logic sections of the board, and allows additional LSI chips to be used to improve the features on the board. It also makes bug correction easier, since circuit changes can be made simply by adjusting the PAL logic equations and programming a new chip. There are 17 PALs on the HK68 and only a few SSI chips, such as 74LS04, 7408, etc.

PAL	Function Summary	Type
A	Bus arbitration control	16L8
B	Bus Control signal interface	16L8
C	Multibus I/F timing state machine	16R4
D	Multibus Data gate control	16L8
E	Bus arbitration state machine	16R4
F	Bus Map	16C1
G	Miscellaneous control	16L8
H	Memory and Bus Error control	16R6
I	Internal control signal generator	16L8
J	Interrupt Acknowledge decoder	16L8
K	Physical memory map	16L8
M	MPU and MMU control	16L8
N	I/O Map	16L8
P	Leftovers	16R4
Q	More Leftovers	16L8
S	Streamer Tape I/F control	16L8
W	Winchester I/F control	16L8

As an example, here is the equation used to monitor the Multibus and Bus Map for an access request from the bus:

```
                                                         Key:
             Term 1         Term 2          Term 3       / means "NOT"
           ----------  --------------  --------------    * means "AND"
    RQST = MRDC*MAPOK + MWTC*MAPOK*/WP + LOCK*RQST*/DOG  + means "OR"
```

RQST, when true, indicates that a valid request from the Multibus is pending. This is one of the output signals from PAL "A." RQST will become true if one or more of the "terms" in the equation are true.

Term 1 causes RQST to go true when the Memory Read Control signal (MRDC on the bus) goes true, AND the Bus Map indicates that the bus address lines are correct. ("MAPOK" is generated in another PAL via an equation whose terms include the Multibus address signals.)

Term 2 causes RQST to go true when the Memory Write Control signal (MWTC) goes true, AND the Bus Map indicates a valid address AND the HK68 board is NOT Write Protected (by jumper J16).

Term 3 causes RQST to STAY true during a "locked" bus cycle as long as the bus Watchdog Timer has NOT expired.

20.2 Configuration Options

Certain features of the HK68 board may not be useful in your application. Standard configuration includes most of the functions described in this manual. "Stripped" versions are available. Consult factory for details.

Standard Configuration:

All features described in this manual (but without the 10 Mhz clock) may be required for some SBX modules or Multibus slave boards instead of our 8 Mhz clock.) The RAM memory capacity may be specified on your order to be 128K, 256K, 512K or 1 megabyte (assuming 256K x 1 chip availability). 128K and 512K configurations use standard IC sockets. The 256K and 1 megabyte RAM arrays use double-decker sockets. Serial port A is configured for RS-232.

Additional Options:

1. 10 MHz oscillator for use by certain SBX modules or the Multibus
2. RS-422 serial interface kit for SIO port A
3. Software monitor program in ROM
4. Various SBX modules
5. Custom Bus Mapping PAL
6. Board Serialization

Features which could be deleted: (special order)

1. MMU and/or DMAC
2. Connectors for the two SBX modules
3. RAM parity bits and associated logic
4. RAM itself (uses all off-card RAM)
5. For extremely cost sensitive systems, certain other features such as DIP switches, LED's, parallel I/F, etc., may be removed.

20.3 Accessories

Cables
Enclosures
Peripherals
Software
  Operating Systems: UniPlus+, CP/M-68K
  Monitor, Utilities
Custom Interfaces
Development Systems

21.0 APPENDICES

The appendices offer detailed technical information on the various chips used to implement the Heurikon HK68. Their pages are actually copies of the manufacturer's data sheets for the devices. Certain information has been deleted to improve clarity (specifically, that which concerns hardware design or unnecessary features), and reduce weight. Where a variance exists between the manufacturer's data and text earlier in this manual, the earlier information should prevail.

Section	Device	Document	Manual Ref
A	MPU (MC68000)	Motorola MC68000 Spec	Section 4
B	DMAC (MC68450)	Hitachi HD68450 Spec	Section 5
C	MMU (MC68451)	Motorola MC68451 Spec	Section 6
D	SCC (Z8530)	Zilog SCC Technical Manual	Section 12
E	CIO (Z8536)	Zilog CIO Technical Manual	Section 11

Note: The appendices are available as a separate booklets from Heurikon.

This page has been intentionally left blank.

READER COMMENT FORM

We would appreciate any comments you have concerning this manual. Please let us know if you have found any errors or feel certain sections should be expanded. Thank you.

Name:____________________________________ Tilte:______________ Date:________

Company:_________________________________

Address:_________________________________

City: ______________________State: ________ ZIP:_________

Telephone: (   )-_______-______________

HK-68-b

Section	Comments

Would you like us to contact you? _______

Mail To: Heurikon Corporation
3001 Lathan Drive
Madison, WI 53713