GLOBALS REQUIRED

MICROBOX III
SYSTEM MANUAL

Written by Garey T. Mills 1986
for
Micro Concepts

TABLE OF CONTENTS

INTRODUCTION

The Microbox III is a low cost single board computer using the 68000/68010 processor. It has memory-mapped colour graphics, 512k bytes of ram, up to 192k bytes of EPROM, serial, parallel, and floppy disc interfaces, stereo sound, a battery-backed real-time clock, and an eprom programmer; all on a double-Eurocard sized board.

The board's features, besides making it an extremely powerful personal and business computer, suit it for image processing, networking, games, process control and other real-time applications.

The Microbox III can be purchased with a number of different operating systems, giving it versatility to run applications the user wants. A software monitor,MON_K, containing debugging and service routines, is included on the board.

This manual will describe the hardware functioning of the board, how to mount it and how to configure it. It will also describe the memory map, commands and system calls provided by the software monitor.

Programming the various chips will not be discussed in detail. This has already been done well by the various manufacturers. A bibliography of the various technical data sheets available is included in Appendix B.

An attempt has been made to give a general idea of what can be expected of the various chips, and some representative code from the monitor has been included, but this is no substitute for the detail available from the manufacturers.

Section 1 contains specifications and a complete description of the circuit. Section 2 discusses mounting and configuring the board. Section 3 is given over to discussing the software monitor.

Microbox III is delivered with TRIPOS III, a multi-tasking, real-time operating system, and VROOMS, a graphics driver and user interface. These are described in a separate manual.

LIST OF ILLUSTRATIONS

1.0 HARDWARE DESCRIPTION

1.1 SPECIFICATIONS

SIZE: 235mm x 220mm x 20mm (double Eurocard)
POWER REQUIREMENTS: +12V (max.100mA, typical 50mA)
+ 5V (max 2 amps, typical 1.5 amps)

1.2 DESCRIPTION OF CIRCUIT

Below, the circuitry of the board is broken into eleven logical parts. (Note: in this description active low signals are prefixed with an asterisk, for example *INT)

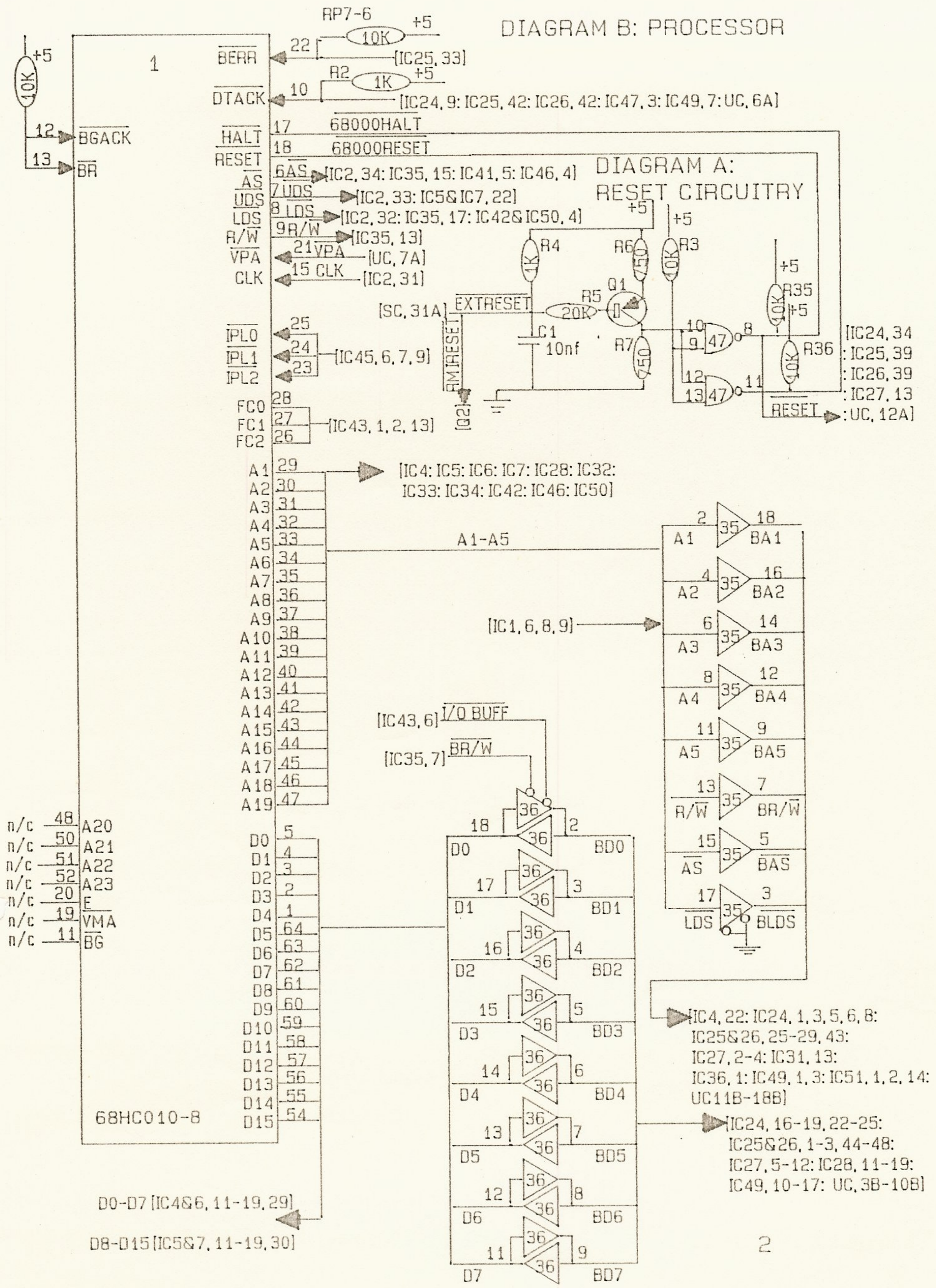
1.2.1 The reset circuitry. (Diagram A)

The reset circuitry consists of Q1 and two gates of IC47. The reset signal is generated by SW1, which pulls the *EXTRESET signal low.

1.2.2 The processor. (Diagram B)

The signals *BG, *BGACK, *BR, *VMA, and E are not required in this design. The address lines A20-23 will be used in a memory expansion board. Two buffers, IC35 and IC36, provide buffering for the bottom seven data lines, the bottom five address lines and the strobes *R/W, *AS and *LDS.

DIAGRAM B: PROCESSOR



1.2.3 Decoding circuitry. (Diagram C)

All of the signals generated by IC41 are enabled or disabled by *AS from the central processor. IC41 decodes the three-bit 'S-Bus' lines from IC2 (the RMI) to seven address blocks. Lines zero, one and two are anded to form the system eprom enable signal. Line three is the user eprom enable signal. Line four is the synchronous input/output enable. Line five is the asynchronous input/output enable. Line six enables input/output peripheral chips which are connected external to the board. Lines zero through six are active low. The three function code lines from the central processor (FC0,FC1,FC2) are anded to give *INTACK. When they are all high, IC2&3 (the RMS chips) and IC41 are disabled. *INTACK is also inverted and becomes one of the enabling signals for IC46, which generates the interrupt acknowledge signals. *SYNCI/O, *ASYNCI/O , *INTACK and the *EXTI/O signals are 'ored' to form the *I/OBUFF signal. This has the effect of enabling the input/output buffer (IC36) whenever either any of the I/O enabling signals (or *INTACK) are asserted or there is an interrupt cycle.

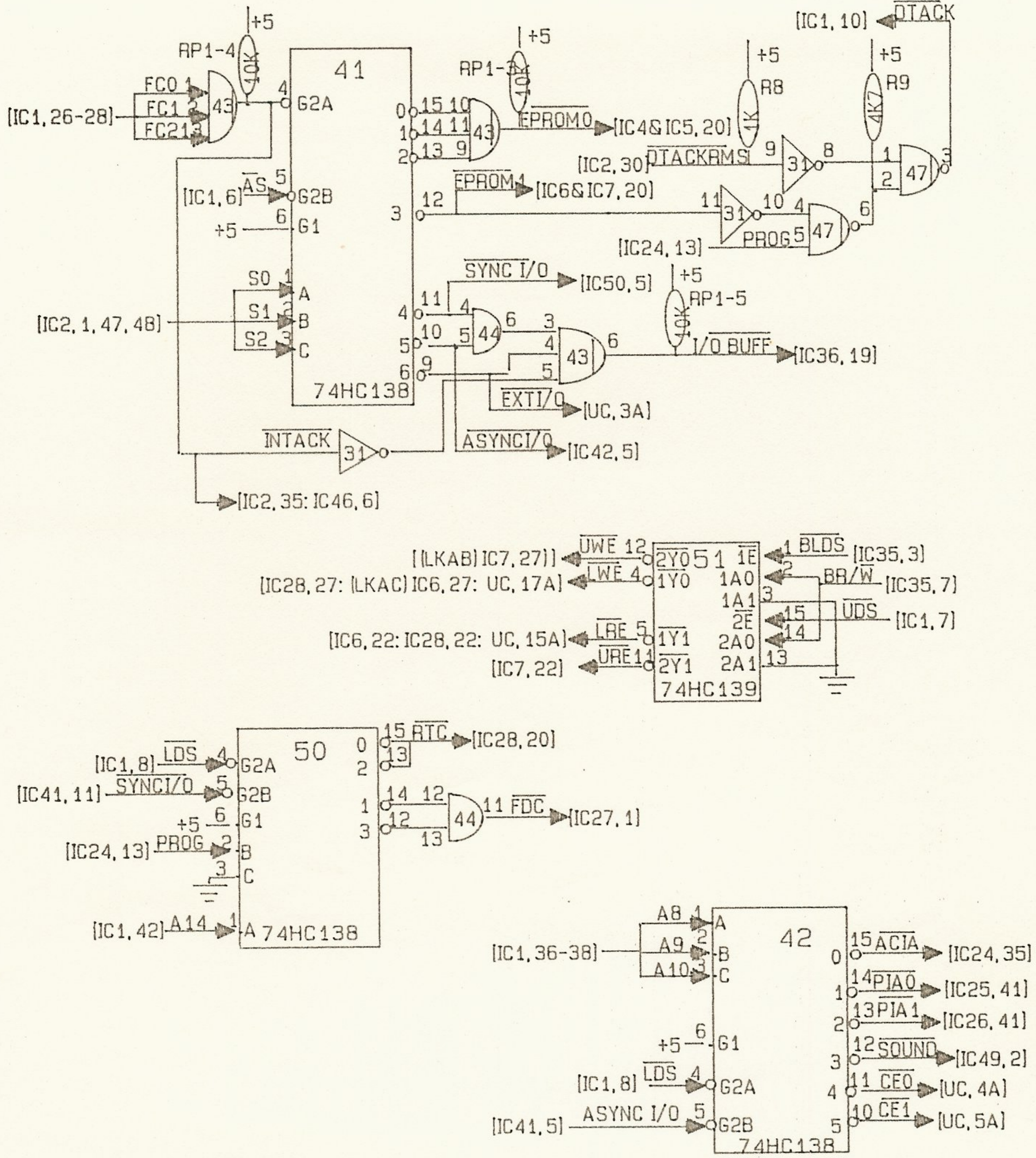
IC42 decodes the asynchronous I/O space, enabled by *LDS (Lower Data Strobe) and *ASYNCI/O. Six asynchronous devices are provided for. Four are on board; the ACIA (IC24), the two PIAs (IC25&26) and the sound chip (IC49). Two may be added by the user. Signals for those are taken out to the user connector as *CE0 and *CE1.

IC50 decodes the synchronous I/O space. It is enabled by *LDS and *SYNCI/O. The two synchronous devices are the RTC (IC29) and the FDC (IC27). PROG is used as one of the decoded signals so that the RTC (and its 8k bytes of battery-backed ram) can be withdrawn from the memory map. The RTC can only be accessed when PROG is high. Since PROG is a dual purpose signal, and ,when high, is also used to signal that the user eproms should be programmed (if they are accessed), it is important to note:

The RTC must not be accessed from within the user eprom space.

IC51 decodes the lower data strobe, the read/write signal and the upper data strobe into the read and write enable signals for the user and system eproms. Note that the write signals will only be needed if the user has installed static ram in the user eprom sockets.

DIAGRAM C: DECODING CIRCUITRY



1.2.4 Floppy Disk Controller. (Diagram D)

The floppy disk controller is a standard WD1772. It is clocked by a separate 8Mhz oscillator to free it from variations in the processor clock rate. (The processor clock rate will vary if the board is used for gen-locking.) Signals to and from the disk are buffered by open collector gates in IC39 and IC40.

The IRQ (Interrupt Request) line goes through an inverter to interrupt link G. DRQ (Data Service Request) and WPRT (Write Protect) go to edge sensitive inputs in the ACIA where they are used to generate further interrupts. The WPRT line gives an indication of whether the disk has been changed.

1.2.5 Interrupts and Interrupt Acknowledge (Diagram E)

IC 45 encodes interrupts and passes the information to the processor via the three bit IPL(Interrupt Priority Level) bus. IC46 decodes interrupt acknowledge signals from the processor. IC44 provides for assertion of VPA (Valid Peripheral Address) for auto-vectoring peripherals. On this board 2 peripherals are auto-vectoring; the floppy disk controller and the RMS chip set.

DIAGRAM D: FLOPPY DISK CONTROLLER

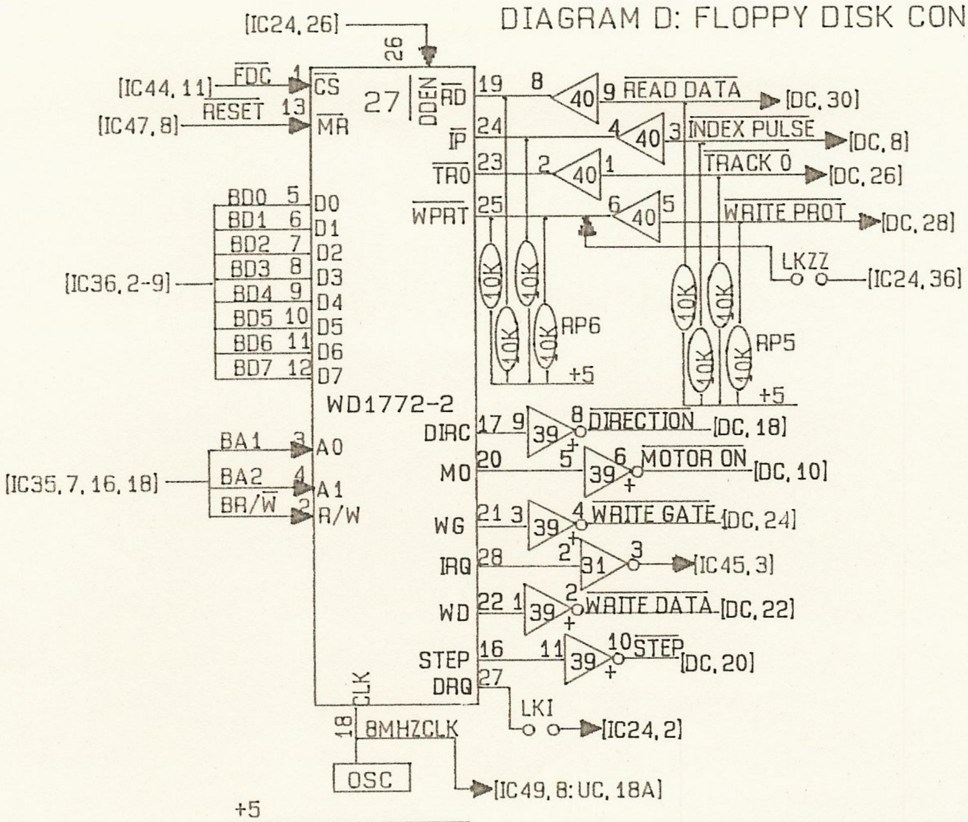
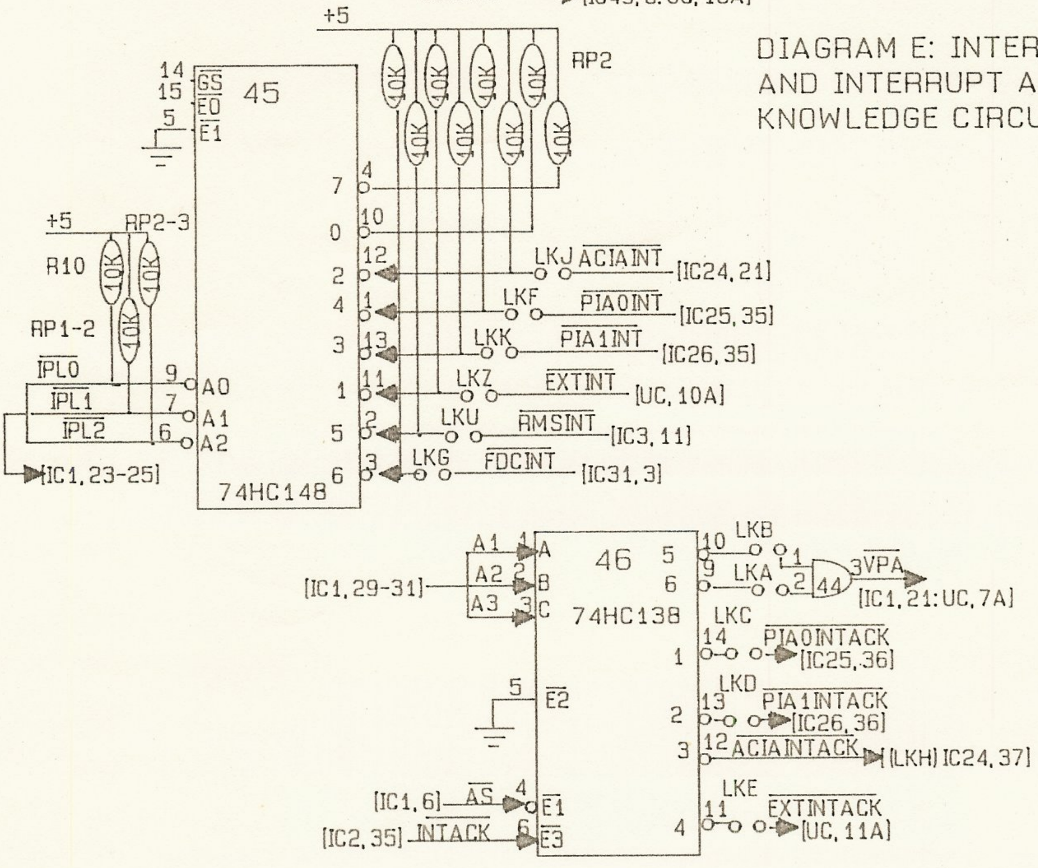


DIAGRAM E: INTERRUPT AND INTERRUPT ACKNOWLEDGE CIRCUITRY



1.2.6 RMS and memory (Diagram F)

The RMS(Raster Management System) is composed of two chips. IC2 is a memory controller and IC3 is a graphics and video generator. The two chips communicate via the 'X-bus', pins 37-46 on IC2 and pins 21-30 on IC3. A processor address is multiplexed to the chips by ICs 32-4. IC2 turns this into a multiplexed ram address for the dynamic rams (ICs 8-23). Data is gated between the rams and the processor through IC29 and IC30.

The RMS is reset by pulling either X2 (for PAL) or X3 (for NTSC) low via Q2.

The devices normally optain their master clock from X1. Optionally, for gen-locking, an external clock can be applied to OSCIN (IC2,pin 28), with OSCOUT and X1 disconnected. In either case, a signal to clock the MPU is generated by IC2.

Video from IC3 goes directly to the output buffers. The video signal is buffered to 1 volt peak to peak (75 ohm) via the discrete transistor buffers Q4-Q9. The signal is standard analogue RGB with syncs in green for monochrome monitors.

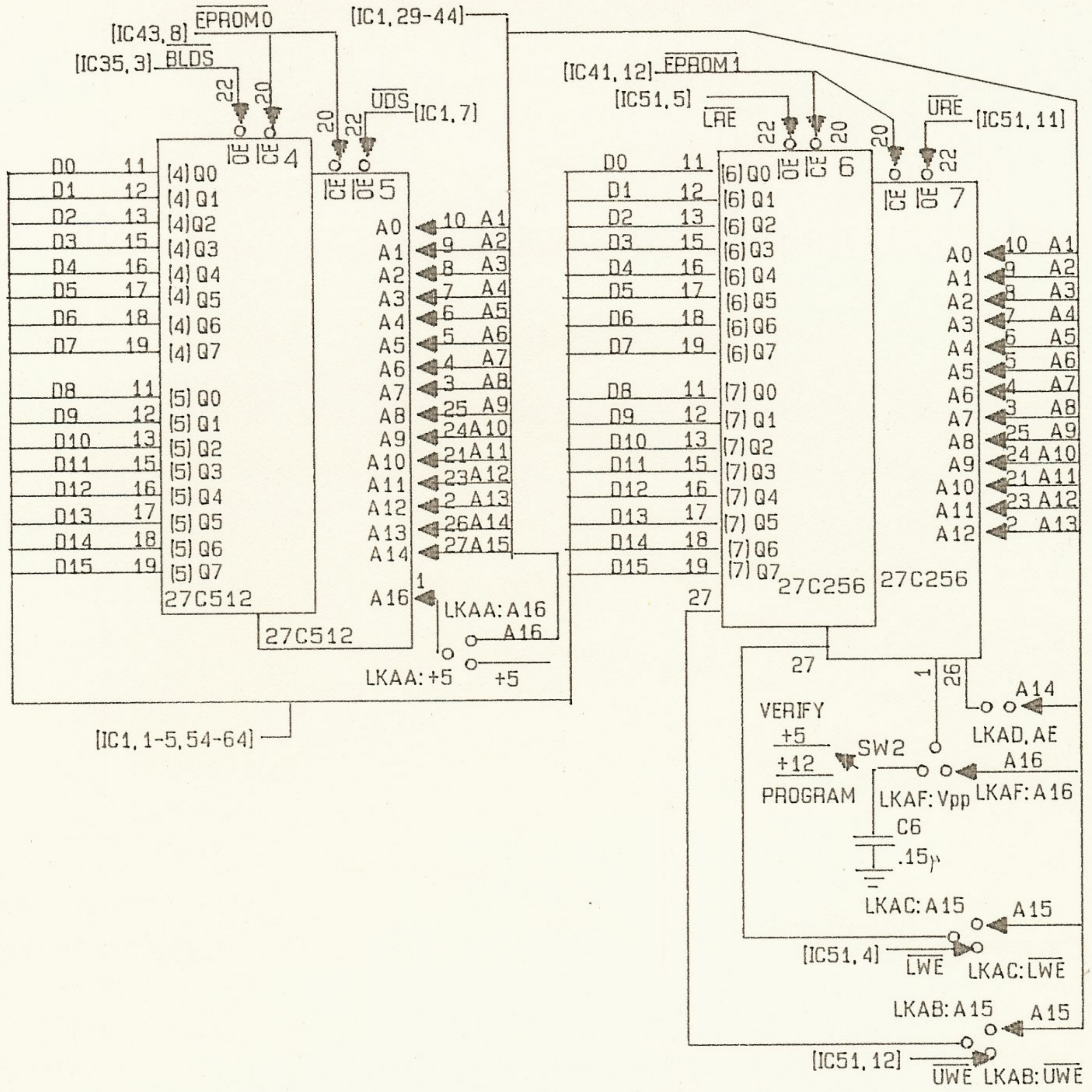
1.2.7 Eproms (Diagram G)

There are two separate eprom areas. Both are connected to the processor address and data buses.

The system eprom area consists of two 27512 eproms for a total of 128K bytes of memory. Provision is made through Link AA for 27256 eproms to be installed, for a memory of 64k bytes.

The user eprom sockets may be configured through Links AB - AF. The options are: 2 27256 eproms, 16k bytes of battery-backed ram (using "smart sockets" and 8k by 8 static rams), or 64k bytes of battery-backed ram (using smart sockets and 32k by 8 static rams).

DIAGRAM G: EPROMS



1.2.8 Parallel Input/Output (Diagram H)

The board's parallel I/O consists of two 68230 PIAs. Together these give a total of 52 independently programmable I/O lines and two timers. One of the timers is used as a bus watchdog leaving the other one free for user applications.

In the conception of the designers, and in the monitor software, the 52 lines have been divided up into one uncommitted and 4 committed ports as follows:

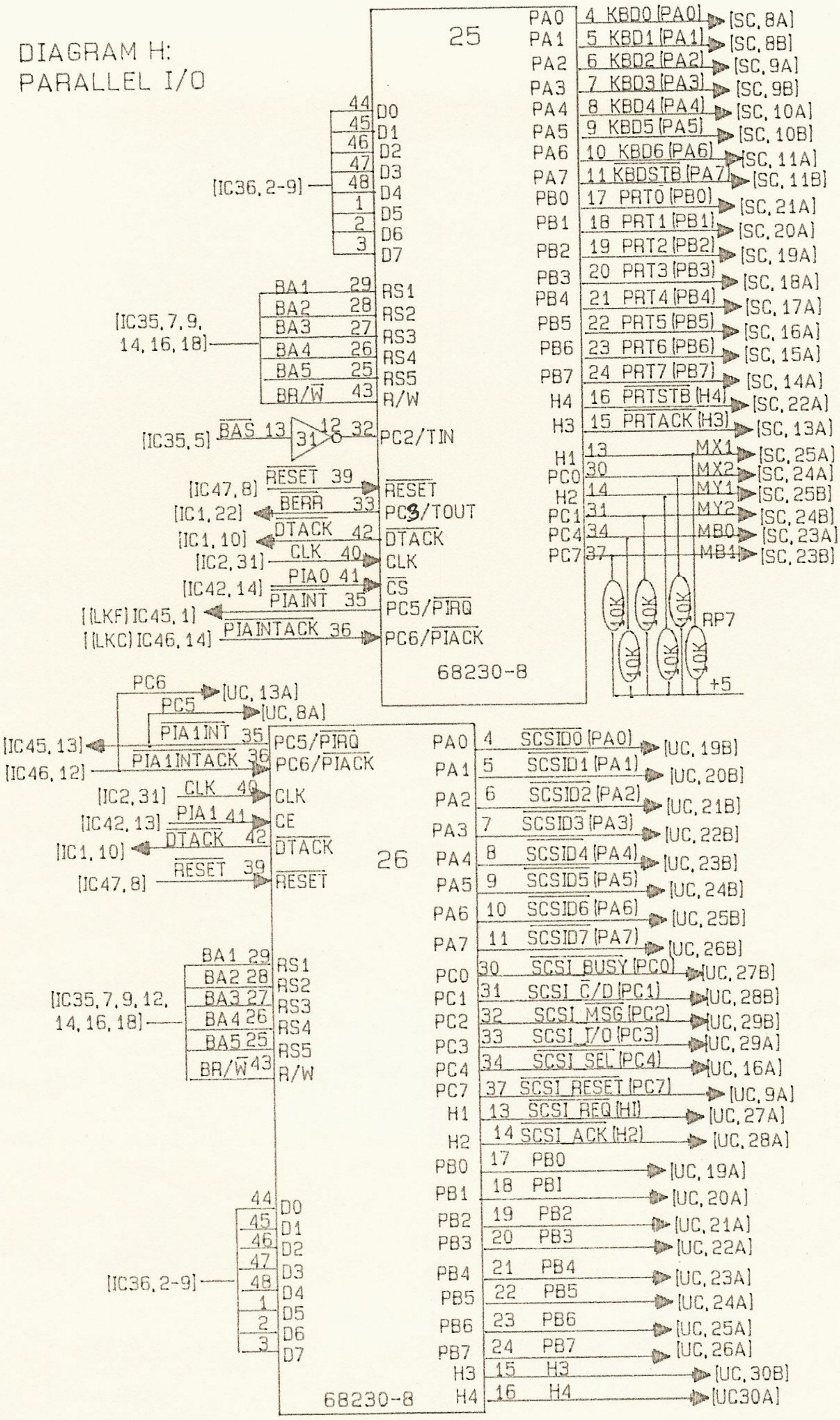
IC25

PA0-PA7: parallel keyboard port. This interface is 7 bits wide with a negative going strobe on PA7. The strobe should be approximately 1 millisecond in length.
PB0-PB7,H3,H4: Centronics printer port.
PC0,1,4,7,H1,H2: mouse interface.
The timer in IC25 is the bus watchdog.

IC26

PA0-7,PC0-4,H1,H2: SCSI data bus and control signals. This interface corresponds to the SCSI standard and will interface to any hard disk with an on-board controller that also conforms to the standard.
PB0-7,H3,H4: an uncommitted parallel port, with enough handshaking signals to form another centronics port.

DIAGRAM H:
PARALLEL I/O



1.2.9 ACIA (Diagram I)

The 68681 ACIA performs three separate functions for the board. First, with IC37 and IC38, it provides two serial channels. These channels may be configured through links L and M to be either RS232 or TTL (TTL on receive only). A +/- 12 volt tie-up point (Link N) is provided on board to tie RS232 lines high or low if needed.

Second, the ACIA provides a set of outputs which control the RTS lines for the serial ports, sound a piezo-electric sounder, control the PROG signal line (see 2.2.3), switch a remote/local signal which switches video sources for gen-locking purposes, and select single/double density, disk side and disk drive for the floppy disk controller.

Third, the ACIA has a 6 bit input port which monitors CTS lines for the serial ports and Switch 3, four switches which set system parameters.

1.2.10 Sound Synthesizer (Diagram J)

The sound synthesizer is an SAA1099. R32 sets the bias current for the chips analog to digital converters. RP7-7&8, R33,R34 and capacitors C8-11 form a low pass filter rolling off to about 10Khz. This filters out the 60Khz switching frequency of the synthesizer. The output signal is approximately 2 volts into 10k ohms.

1.2.11 Clock (Diagram K)

The clock is both a clock and a battery-backed socket for an 8k byte CMOS static ram.

The clock and memory are interfaced to the peripheral data area of the processor. They can be removed from the memory map and returned by setting and clearing a bit in the output port of the ACIA (2.2.9). This makes the data in the ram completely secure from corruption or tampering.

Reading the clock corrupts the bottom byte of the 8k byte ram, therefore this position should not be used for any storage.

Since the signal which enables the clock has the additional function of enabling the programming sequence in the user eprom sockets(if the eproms are enabled), the clock <u>cannot</u> be access from routines located in the user eprom. Accessing the clock from the user eprom area will cause a bus error and may corrupt data in the user eproms. Routines are provided in the system service eproms for reading data into and out of the security ram.

13210

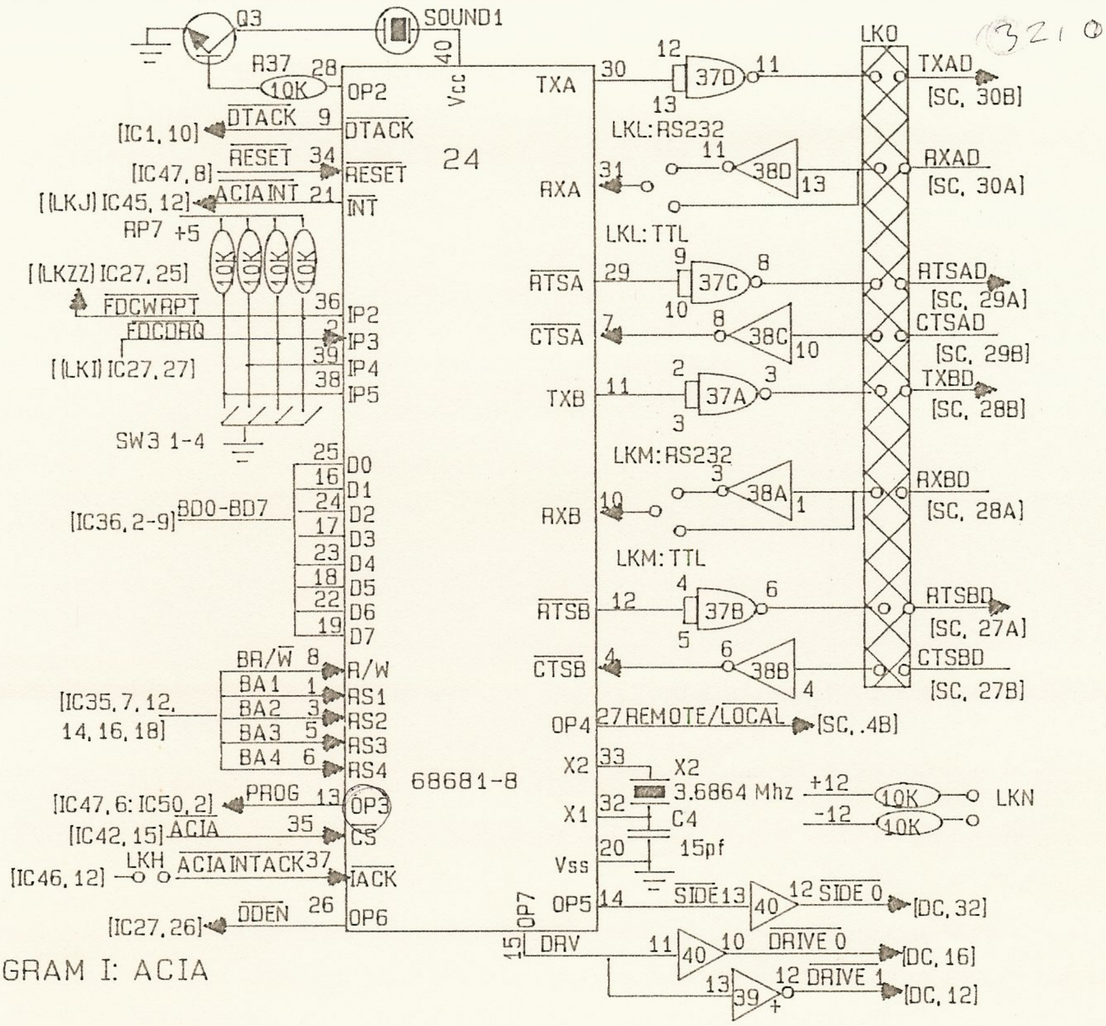


DIAGRAM I: ACIA

DIAGRAM J: SOUND SYNTHESIZER

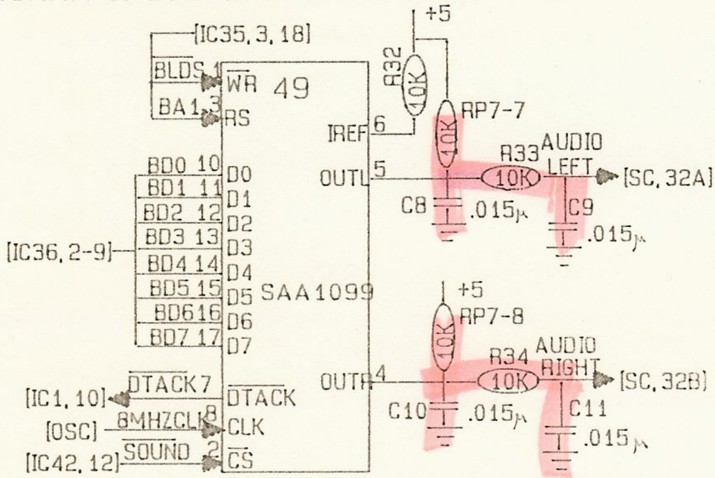
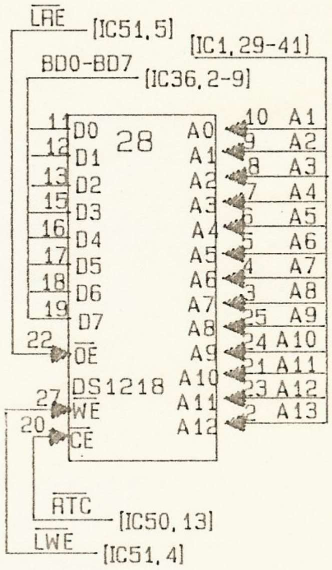


DIAGRAM K: CLOCK



ARE THESE NOW CORRECT?

1.3 CHIP CAPABILITIES

In this section, the capabilities of the major chips on the board, and any limitations the design puts upon them, are discussed.

1.3.1 Processor

The processor is the Motorola 68000 (or 68010). When the board is configured for PAL operation (i.e. throughout Great Britain and Europe), the processor runs at 7.886542Mhz, being driven by a clock derived from the graphics chips. On the board, address lines 20-23 are not connected. It is intended they be used on a piggy-backed memory expansion board.

1.3.2 Display

Based on the Motorola Raster Management System (RMS). This chip set is a versatile and powerful colour graphics and text display system. Its capabilities form a superset of the capabilites of most of the commercially available graphics computers.

The chips give a choice of 50 software programmable resolutions, from a low of 256h x 192v to a high of 640h x 500v.

Virtual screens can be software defined to be up to 512k bytes in area, and smooth-scrolled in both vertical and horizontal directions.

Look up tables allow access to 32 colours out of a range of 4096 in all but the highest horizontal resolutions. Once the 32 colours are set, any one of them is available as a border colour. 16 of them are directly available to objects and characters on the screen. The other 16 are available in some list modes through offsets into the look up table. In horizontal resolutions of 512 and 640, a maximum of 4 colours at any one time are available to objects and characters on the screen. In all resolutions in all modes true objects (sprites) can use up to 24 colours.

Eight hardware true objects (sprites) may be used on the screen at once. The RMS chip set will store up to 256 different sprite patterns.

The chips have two basic modes of operation, list mode and bit-plane mode. In list mode, which is designed for word-processing, games and character graphics, the user may have screens of up to 80 x 50 characters or objects. Each of these characters can have a number of different attributes, including flashing, underlining, reversed color, etc. The user may define characters in software, using pel grids of up to 16 x 10. List screens take up less storage than bit-plane screens, for example, the user could store approximately 160 screens of 80 x 25 text in one large virtual screen on the Microbox III. Sprites can be used in bit-plane and list mode, giving quick and easy implementation of cursors and animation sequences.

Other features of the chips include collision reporting between sprites and between sprites and fixed objects, color collision and shading.

The chips also provide sophisticated circuitry for gen-locking, allowing images derived from other video sources (including video tapes, other RMS sets, and "difficult" sources) to be overlaid on the screen and written to video recorders. An off-board gen-locking hardware module is required to take advantage of the gen-locking capabilities of the chips.

The software monitor provides several preset screen definitions; an 80x25 text list mode screens and two bit-plane mode screens, one at 640x500 resolution (monochrome) and one at 320x250 resolution (16 colours).

The RMS addressable memory on the Microbox III is fixed at 512k, one half of its maximum 1 Mbyte adressing range.

1.3.3 Ram

There is 512k bytes of dynamic ram using 16 256k bit d-rams. The ram is shared between the processor and display using an interleaved scheme. The processor can always access ram without disturbing the display.

There is also a separate 8k byte of battery-backed CMOS ram, located in the the peripheral address space. This 8k byte ram can be both read and write protected under software control. Uses for it include holding sensitive system parameters, character sets that need to be quickly accessed, etc.

A further 16-64k of CMOS battery-backed ram can be installed on the board in lieu of installing 64k bytes of user eprom.

1.3.4 Eprom

Up to 192k bytes using two 512k bit and two 256k bit eproms. The board is supplied with two 27512's programmed with MON_K, together with an operating system. The monitor commands and calls are detailed in Section 3. The operating system supplied with the board, TRIPOS III, and VROOMS, a set of graphics management subroutines, are described in a separate manual.

1.3.5 Mass Storage

There is a dual 5-1/4" or 3-1/2" floppy drive interface using the WD1772. It supports 40/80 track, single/double density, single/double sided drives. A SCSI bus interface is provided, designed to connect to a 3-1/2" 20 Mbyte Winchester disc drive.

1.3.6 Communications

Two RS-232 or TTL serial ports are provided using an MC68681. The ports are software baud rate, stop-bit and parity setable, with input baud rates set independently of output baud rates. The 68681 can also be programmed to a user defined baud rate through an on-chip timer.

The MC68681 can be run in multidrop mode. This means that the ACIA will "listen" to a serial line. When it is addressed, it will issue an interrupt to the processor. This is useful for network applications.

The parallel interface is two MC68230 PI/Ts (Peripheral Interface/ Timer). These chips have port modes which include bit I/O, single direction 8 or 16 bit, or bi-directional 8 or 16 bit. Each chip can generate 5 different interrupt vectors. Each chip has 4 programmable handshake lines, which can also be used as I/O lines. Each 68230 also contains a 24 bit timer. In the extreme case the two chips could give a total of 52 individually programmable I/O lines. On the board, they are configured as discussed in Section 1.2.8

1.3.7 Real Time Clock

A "Smart Watch" real-time clock (DS1216) with 8k bytes of battery-backed ram. The clock counts hundredths and tenths of a second, minutes, hours (12 or 24 hour clock), day of the week, month, year and ten year (for 99 year total) figures. The ram is used to store system parameters, baud rates etc.

1.3.8 Sound

The SAA1099 furnishes stereo sound with six frequency generators, two noise sources, twelve amplitude/envelope controllers and two six channel mixers.

2.0 HARDWARE CONFIGURATION

2.1 MOUNTING THE BOARD

Connections for power, video, comms, and expansion bus (a reduced bus for I/O expansion) are on two 64/64 way DIN indirect connectors at one end of the board. This, and the board's standard size, make rack mounting it relatively easy.

Alternatively, the board can be mounted on standoffs. There are six .3cm holes in the board, 4 on the connector side and two on the side opposite.

The board should have ventilation, but it is not necessary to employ forced air cooling.

Diagram M shows detail for the two 64/64 way DIN connectors. The A and B row names refer to the A and B pinout rows on a standard 64/64 way connector. The name 'USER CONNECTOR' refers to the connector closer to the CPU. Diagram L shows the floppy disk connector pin outs. This is a standard pin out for both 31/2" and 51/4" drives.

DIAGRAM L: FLOPPY DISK CONNECTOR DETAIL

1 2	
3 4	
5 6	
7 8	INDEX PULSE
9 10	MOTOR ON
11 12	DRIVE 1
13 14	
15 16	DRIVE 0
17 18	DIRECTION
19 20	STEP
21 22	WRITE DATA
23 24	WRITE GATE
25 26	TRACK 0
27 28	WRITE PROTECT
29 30	READ DATA
31 32	SIDE 0
33 34	

ALL ODD PINS ARE GROUND

DIAGRAM M: CONNECTOR DETAIL

USER CONNECTOR

B		A
+5	1	+5
0V	2	0V
BD0	3	EXT I/O
BD1	4	CE0
BD2	5	CE1
BD3	6	DTACK
BD4	7	VPA
BD5	8	PIA1 INT (PC5)
BD6	9	SCSI_RESET (PC7)
BD7	10	EXTINT
BA1	11	EXTINTACK
BA2	12	RESET
BA3	13	PIA1 INTACK (PC6)
BA4	14	RMSCE
BA5	15	LRE
BR/W	16	SCSI_SEL (PC4)
BAS	17	LWE
BLDS	18	8 MHZ CLOCK
SCSID0 (PA0)	19	PB0
SCSID1 (PA1)	20	PB1
SCSID2 (PA2)	21	PB2
SCSID3 (PA3)	22	PB3
SCSID4 (PA4)	23	PB4
SCSID5 (PA5)	24	PB5
SCSID6 (PA6)	25	PB6
SCSID7 (PA7)	26	PB7
SCSI_BUSY (PC0)	27	SCSI_REQ (H1)
SCSI_C/D (PC1)	28	SCSI_ACK (H2)
SCSI_MSG (PC2)	29	SCSI_I/O (PC3)
H3	30	H4
0V	31	0V
-12V OUT	32	+12V IN

SYSTEM CONNECTOR

B		A
RED GROUND	1	RED
GREEN GROUND	2	GREEN
BLUE GROUND	3	BLUE
REMOTE/LOCAL	4	OVERLAY
SYNC	5	HSYNC
CLOCK GROUND	6	LPEN
GENLOCK CLOCK	7	CSC
KBD1 (PA1)	8	KBD0 (PA0)
KBD3 (PA3)	9	KBD2 (PA2)
KBD5 (PA5)	10	KBD4 (PA4)
KBDSTB (PA7)	11	KBD6 (PA6)
0V	12	+5
	13	PRTACK (H3)
	14	PRT7 (PB7)
	15	PRT6 (PB6)
	16	PRT5 (PB5)
PRINTER GROUND	17	PRT4 (PB4)
	18	PRT3 (PB3)
	19	PRT2 (PB2)
	20	PRT1 (PB1)
	21	PRT0 (PB0)
	22	PRTSTB (H4)
MB1 (PC7)	23	MB0 (PC4)
MY2 (PC1)	24	MX2 (PC0)
MY1 (H2)	25	MX1 (H1)
0V	26	+5
CTSB	27	RTSB
TXB	28	RXB
CTSA	29	RTSA
TXA	30	RXA
SIGNAL GROUND	31	RESET SWITCH
SOUND-R	32	SOUND-L

ABBREVIATIONS

BD = BUFFERED DATA
BA = BUFFERED ADDRESS
BR/W = BUFFERED READ/WRITE
BAS = BUFFERED ADDRESS STROBE
BLDS = BUFFERED LOWER ADDRESS STROBE
CE = CHIP ENABLE
DTACK = DATA ACKNOWLEDGE
EXT = EXTERNAL
H2, H3 = UNCOMMITTED STROBE LINES
INT = INTERRUPT
INTACK = INTERRUPT ACKNOWLEDGE
LRE = LOWER READ ENABLE
LWE = LOWER WRITE ENABLE
PB = PORT B
RMSCE = RASTER MANAGEMENT SYSTEM CHIP ENABLE
SCSI = SHUGART CONSULTANTS STANDARD INTERFACE
VPA = VALID PERIPHERAL ADDRESS

CSC = COLOR SUB CARRIER
CTS = SERIAL CLEAR TO SEND
KBD = KEYBOARD
KBDSTB = KEYBOARD STROBE
LPEN = LIGHT PEN
M = MOUSE
PRT = PRINTER
RTS = SERIAL REQUEST TO SEND
RX = SERIAL RECEIVE DATA
TX = SERIAL TRANSMIT DATA

= POWER
= EXP. BUS
= SCSI BUS
= PAR. PORT
= VIDEO
= KEYBOARD
= PRINTER
= MOUSE
= SERIAL
= RESET
= SOUND

2.2 LINKS

See the board overlay in the diagrams appendix for the location of the links. This section will discuss their function. Appendix D contains a table of the link set up used by TRIPOS III and CP/M 68K.

2.2.1 Link Functions.

The links on the board fall into seven functional groups. They are:

1) Interrupt links. There are six wire-wrap interrupt links, which allow setting the interrupt priority of peripheral devices. Each interrupt link has two signals associated with it; an interrupt from a specific peripheral, and an interrupt priority level. Specifics for each link are shown in TABLE 1.

TABLE 1
INTERRUPT LINKS

LINK	PRIORITY LEVEL	SIGNAL
Z	1	*EXTINT
J	2	*ACIAINT
K	3	*PIA1INT
F	4	*PIA0INT
U	5	*RMSINT
G	6	*FDCINT

(1) cont.)

There are four wire wrap interrupt acknowledge links and two wire wrap VPA links. These should be set to correspond to the interrupt link setup. The VPA (Valid Peripheral Address) links are for enabling or disabling of the VPA signal generated for auto-vectoring peripherals.The links, link priorities and signals are shown in TABLE 2.

TABLE 2
INTERRUPT ACKNOWLEDGE LINKS

LINK	PRIORITY LEVEL	SIGNAL
	1	*PIA0INTACK
D	2	*PIA1INTACK
H	3	*ACIAINTACK
E	4	*EXTINTACK
A	5	*VPA (for RMS)
B	6	*VPA (for FDC)

2) Floppy Disk Data Transfer Links. Link I is the DRQ (Data Service Request) signal to the ACIA. Link ZZ is the WPRT (Write Protect) signal from the ACIA to the FDC.

3) Serial Links. These links allow each serial port to be set to RS232 or TTL level functioning, and serial lines to be crossed or linked. This group also includes a +/- 12 tie up point for RS232 configuration. TABLE 3 summarizes their function.

TABLE 3
SERIAL LINKS

LINK	FUNCTION
L	This link sets serial input to port A to RS232 or TTL functioning. Connecting IC24,31 and IC38,11 sets to RS232.
M	This link functions as link L for port B. Linking IC24,10 to IC38,3 sets to RS232.
N	RS232 tie point.
O	This is actually a set of links. They function to configure serial signals. Specifics are detailed in Section 2.3

4) Video links. RGB signals are carried across the board on twisted pairs to reduce noise. The twisted pairs are tied to these links. The links have one signal pin and one ground pin apiece. Links V,X and Y are red, blue and green video out. Links S,Q and R are red, blue and green buffer input.

5) Gen-lock Clock links. The RMS video chips, plus some addition external hardware allow mixing of on-board and remote video signals. These links configure the board if the gen-locking option is purchased. Link P and part of link T are another set of twisted pair anchors. Another part of Link T is actually two traces between the crystal X1 and the oscillator pins of IC2. These traces are cut if gen-locking is installed.

6) NTSC or PAL link. This board can be operated in the U.S. or anywhere the NTSC video standard is in effect by replacing crystal X1 and shorting this link to the correct setting. For PAL operation crystal X1 = 35.46895Mhz, for NTSC X1 = 35.79545Mhz. Link W.

7) Eprom configuration option links. The system eprom sockets can be configured to accept 27256s (64k total) or 27512s (128k total). The user eprom sockets can be configured to accept 27256s (64k total) or 6164 static rams (16k total). There is also capacity for 32k x 8 static rams when they become available. TABLE 4 details the link functions.

TABLE 4
EPROM CONFIGURATION LINKS

LINK	FUNCTION
AA	Sets the system eprom size. When set to +5 selects 256s.
AB	Connects pin 27 of the upper user eprom socket to write enable or A15. Set to A15 if using eproms or 32k static rams, write enable if using 8k static ram.
AC	Same function as AB, except for lower user eprom socket.
AD	Connects pin 26 of the lower user eprom socket to A14. For 27256 eproms or 32k static rams , not needed for 8k static rams.
AE	Same function as AD, except for upper user eprom socket.
AF	Multiple function in conjunction with Switch 2. It has two configurations, V_{pp} and A16. At present A16 is not used. Switch 2 should be set to Verify (+5) for normal operation. By setting Switch 2 to Program (+12), a programming voltage can be applied to 27256 eproms. A15 from link AB
or	AC should be tied to this link if 32K static rams are used.

2.3 SERIAL SIGNAL LINK CONFIGURATION

Link O is a set of links used for configuring the serial ports. TABLE 5 gives the signal through each link. The order of the links reflects their relative position on the board. The first link, CTSBD, is identified on the board overlay under the ACIA (IC24). The rest of the links are just to the connector side of CTSBD and run away from the processor. If the serial devices you are connecting do not use the RTS and CTS signals these can be shorted on the chip (inboard) side of the links. A +/- 12 volt tie up point (Link N) is provided for RS232. This can be used, for example, to pull DSR (Data Set Ready) high for a serial printer.

Note that RS232 or TTL operation for each port are set by the links L and M.

TABLE 5
SERIAL SIGNAL LINKS

LINK
CTSBD
RTSBD
RXBD
TXBD
RTSAD
CTSAD
TXAD
RXAD

2.4 SWITCH SETTINGS

Two switches must be set on the board. Please refer to the Board Overlay in the appendix for switch locations.

Switch 2 should be set in the VER(ify) position for normal operation. Setting it to PROG(ram) will gate +12 volts to pin 1 of the user eprom sockets.

Switch 3 selects input port, output port and autoboot at start up.

Switch 3-1 is spare, but must be set open (off) for correct operation under TRIPOS III and CP/M 68K.

Switch 3-2 sets the input port. Closed (on) sets input to the parallel keyboard. Open (off) sets input to ACIA channel A. For correct operation under TRIPOS III it must be set open.

Switch 3-3 sets the output port. Closed (on) sets the output to the board's video output. Open (off) sets output to ACIA channel A.

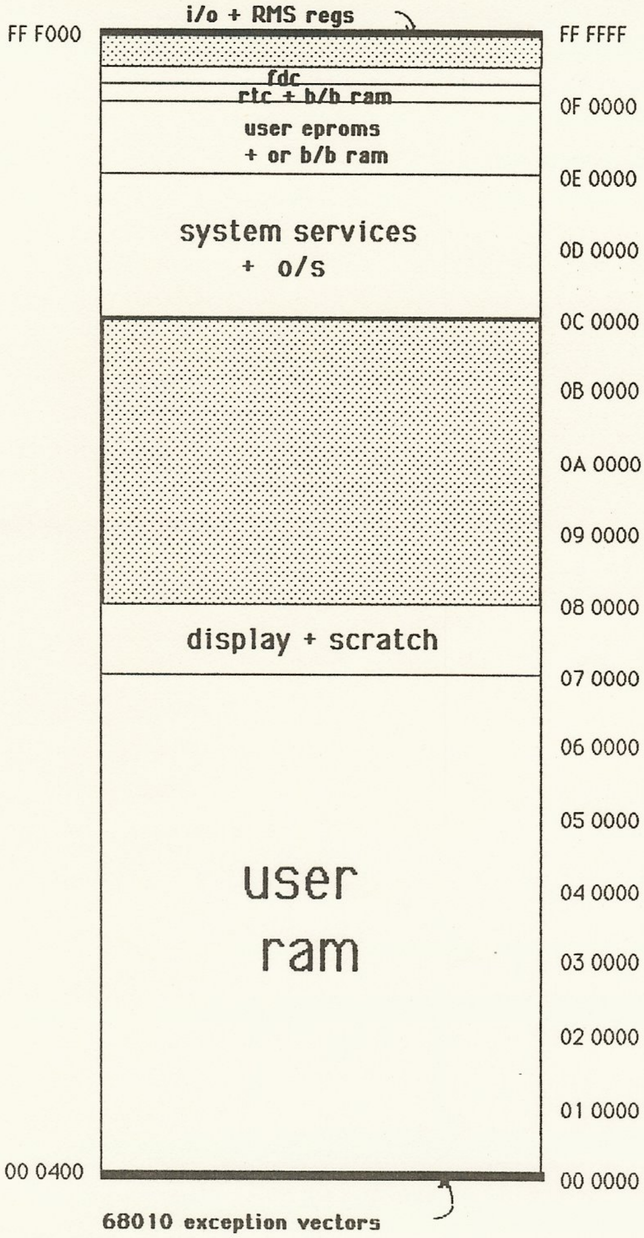
Switch 3-4 sets autoboot on or off. If the switch is closed (on), the board will attempt to boot the system from the system rom. If the switch is open (off) the board will come up in the system monitor.

3.0 SYSTEM MONITOR SOFTWARE

In this section the board's software monitor is discussed. The board will be delivered with a header file of equates and monitor variables to link into assembly language programs. The monitor header file will always take precedence over the account given here.

3.1 The memory map

The memory map for the Microbox III is shown in DIAGRAM N.

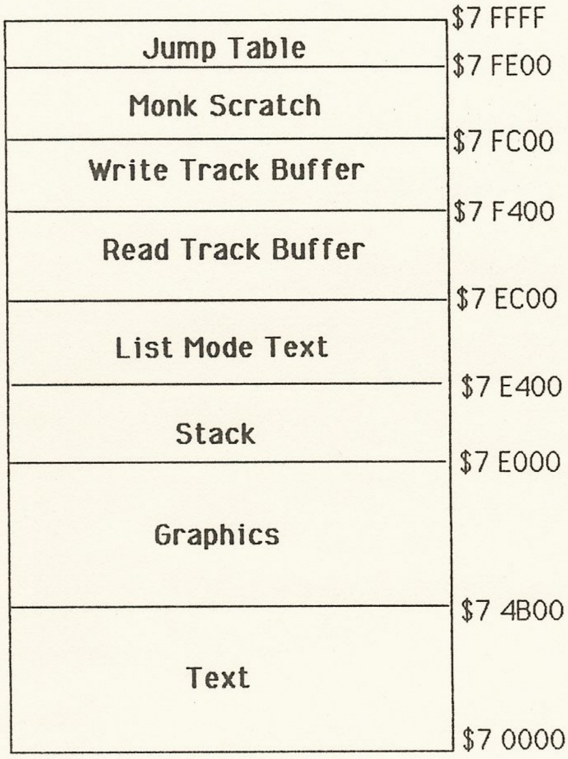


3.1.1 Ram

512k bytes of ram are in the basic memory map running from $00 0000 to $07 FFFF. Two areas of this ram are not available to the programmer, they are the bottom 1k, which is used by the 68010 exception vectors, and the top 64k ($07 0000 to $07 FFFF) which is used for the text and graphics displays, and for I/O buffers. Note that while the 1k used for the exception vectors is determined by the graphic chips' memory map, the 64k is reserved only when the monitor's graphics and I/O routines are used. In other words, it is only reserved by a convention in the monitor. Diagram 0 shows detail of the graphics and scratch area.

DIAGRAM 0: DETAIL OF DISPLAY AND SCRATCH MEMORY MAP

The text area running from $7 0000 is a graphics mode text screen. The list mode text area is used for a small list mode terminal emulator. The stack grows toward low memory.



User Ram

A 320k byte ramdisk can be defined by routines in the monitor. This runs in the memory map from $02 0000 to $07 0000.

3.1.2 Eprom

There are two areas of eprom in the memory map. The system eproms run from $0C 0000 to $0E FFFF (128k), and contain the system service routines and operating system. The user eprom space runs from $0E 0000 to $0E FFFF and may contain either 64kbytes of eprom, or 16kbytes of battery backed CMOS ram.

3.1.3 I/O

The devices in the I/O address space all have their registers mapped to the low bytes of consecutive words (i.e. consecutive registers appear at odd addresses). They are in two groups, the PIAs, ACIA and sound are based at $0F F000, whilst the real-time clock and the floppy disc controller are based at $0F 0000.

The real time clock is at the first byte of an 8k byte battery-backed security ram located from $0F 0001 to $0F 3FFF. This ram is also in the low bytes, this time of a 16k byte address space.

The floppy disk controller has four external byte wide registers, running from $0F 4001 to $0F 4005.

The ACIA has 16 external byte wide registers, running from $0F F001 to $0F F01F.

PIA0 and PIA1 both have 32 external byte wide registers. PIA0 runs from $0F F101 to $0F F13F, and PIA1 runs from $0F F201 to $0F F23F.

The sound chip has two external byte wide registers, at locations $0F F301 and $0F F303.

Table 6 gives the peripheral device base addresses.

TABLE 6
PERIPHERAL BASE ADDRESSES

Peripheral	Type	Base address
ACIA0,1	68681	$0F F001
PIA0	68230	$0F F101
PIA1	68230	$0F F201
SOUND	SAA1099	$0F F301
RTC	DS1216	$0F 0001
FDC	1772	$0F 4001

3.1.4 RMS

The RMS registers are at the top of the memory map. They take up 192 contiguous bytes from $0F FE00 to $0F FEBF. Not all of them have a use in this design, please consult the RMS user's manual for details.

3.2 MONITOR COMMANDS

The monitor commands include memory commands, for testing memory, and viewing, changing and moving data in memory; program commands for running and debugging programs; disk commands for drive testing, formatting, and reading and writing sectors; and a miscellaneous category including setting input and output ports, setting baud rates, reading from and writing to the clock, loading text from the keyboard, and terminal emulation.

Hex numbers can be input in free format, i.e. they can start with any number of zeros and contain any number of digits up to 8. They are terminated by a space character.

Both upper and lower case may be used. Table 7 gives an overview of the commands.

TABLE 7: MONITOR COMMANDS

MEMORY COMMANDS	
HD	Hexadecimal dump of memory
AD	ASCII dump of memory
ME	Memory examine and alter
PM	Poke memory
TM	Test memory
FM	Fill memory
SM	Shift memory
FI	Find byte string

PROGRAM COMMANDS	
DR	Display registers
SD	Set data register
SA	Set address register
SS	Set status register
JU	Jump to program
RP	Run program
TR	Trace
DB	Define breakpoints
BR	List current breakpoints
CP	Continue after breakpoint
JC	Jump to warmstart

DISK COMMANDS	
TD	Test drive
TS	Test stepping
DF	Format disk
WD	Format winchester
RS	Read sector
WS	Write sector

MISCELLANEOUS	
SI	Set input port
SO	Set output port
SB	Set baud rate
DC	Display clock
MC	Modify clock
DP	Display peripheral data
CO	Communication
LK	Load from keyboard
S0, S1...	S record load

3.2.1 Memory Commands

COMMAND DESCRIPTION
SUBCOMMANDS NOTES

==

HD HEXADECIMAL DUMP OF MEMORY
CR = forward one page.
- = back one page.

HD prompts for a hexadecimal starting address. It will then display 16 lines of memory any other character terminates command

locations, showing the starting address of each line, and the hexadecimal number of each column. Each line contains 16 hexadecimal characters, with the corresponding 16 ASCII characters to the left.

AD ASCII DUMP OF MEMORY

CR = forward one page.
- = back one page
any other character terminates command.

AD prompts for a hexadecimal starting address. It will then display 16 lines of 64 characters in ASCII format, showing the starting address of each line, and the hexadecimal number of each column. Non-printable characters will show as a full stop (.).

ME MEMORY EXAMINE AND ALTER

CR = forward one unit.
- = back one unit.
BS = read same location again.
B = set stepping unit to byte.
W = set stepping unit to word.
L = set stepping unit to long word.
any other character terminates command

ME prompts for a starting address. When the command is entered the contents of the byte at that location are shown. Entering a carriage return will show the next byte. Enter a W to display the next even-boundaried word and set the stepping increment to word, or L for long word. Enter B to return to byte increment and display. BS, to read a location again, is useful for examining the input from port registers.

PM POKE MEMORY

PM prompts for an address and a byte value.

TM TEST MEMORY

TM prompts for a beginning and ending address. It performs a simple test, writing and reading suc-

	cessive values over the specified range.
FM FILL MEMORY	FM prompts for a beginning and ending address and a byte fill value.
SM SHIFT MEMORY	SM prompts for a beginning and ending address to shift memory from, and a beginning address to shift memory to. Shifting between odd and even addresses is supported.
FI FIND BYTE STRING	FI asks for the beginning and ending address of an area to look in, the number of bytes of the string and the string. If the number of bytes and the length of the string do not match, no match will be found.

3.2.2 Program Commands

COMMAND DESCRIPTION SUBCOMMANDS	NOTES
DR DISPLAY REGISTERS	DR simply displays registers.
SD SET DATA REGISTER	SD prompts for a register and a value to load. It accepts values from 0 to 7
SA SET ADDRESS REGISTER	SA prompts for a register and a value to load. It accepts values from 0 to 7.

SS SET STATUS REGISTER	Will set the status register to the given value.
JU JUMP TO PROGRAM	Jump to a program. Address will be prompted for. Programs written to run under the monitor should end by calling TRAP 15.
RP RUN PROGRAM	~~Similar to JU, but first displays registers and allows user to change individual registers.~~
TR TRACE **SPACE** = next step. any other character terminates command.	Prompt for location to jump to. Runs a program in trace mode. Shows the register set and program counter at each step.
DB DEFINE BREAKPOINTS	Allows the user to set up to five breakpoints. Breakpoints are defined as an absolute address.
BR LIST CURRENT BREAKPOINTS	Shows current breakpoints.
CP CONTINUE AFTER BREAKPOINT	To exit trace and continue running program.
JC JUMP TO WARM START	Jumps to warmstart location. If the operating system is CP/M68k, control C must be pressed after entering or re-entering the operating system from MON_K.

3.2.3 Disk Commands

The monitor supports five logical disk drives. They are mapped to physical devices as shown in table 8.

TABLE 8
LOGICAL DRIVE/PHYSICAL DRIVE

DRIVE 0 = FLOPPY DISK 0
DRIVE 1 = FLOPPY DISK 1
DRIVE 2 = WINCHESTER
DRIVE 3 = RAM DISK
DRIVE 4 = EPROM (OR STATIC RAM) DISK

The monitor ram disk has a capacity of 320k bytes and is located in the system memory from 20000h to 70000h. The eprom or static ram disk is located in the user eprom sockets. It starts at $E 0000.

COMMAND DESCRIPTION SUBCOMMANDS	NOTES
TD TEST DRIVE any character or CR to terminate command.	TD tests the drive by reading random sectors.
TS TEST STEPPING	Tests the stepping on the floppy drives by stepping in and out. Can be used on any drive.
DF FORMAT DISK	Formats disks to 5 1024 byte sectors per track. Sectors are not interleaved as track buffering is implemented. Not to be used on the winchester disk, logical drive 2.
WD FORMAT WINCHESTER	Issues a command to the on-drive

winchester controller.

THIS COULD GO ON
PREVIOUS PAGE

RS READ SECTOR

Prompts for sector number (1-5) and track number (0-32,676), then for location in memory to read into. Note that there are only 64 tracks on the ram disk, only 12 tracks in 64k bytes of eprom, and only 3 tracks in 16k of static ram. A 20M byte winchester will contain approx. 4096 tracks.

WS WRITE SECTOR

Prompts for sector and track number, then location in memory of sector to write. Writing to the eprom is not flagged as an error.

3.2.4 Miscellaneous Commands

Among these routines are those for setting the input and output ports. Four ports are supported. 0,1 and 2 can be either input or output, 3 output only. The following table shows how they are specified.

TABLE 9
INPUT AND OUTPUT PORT NUMBERS

Port Number	Input	Output
0	parallel keyboard	screen
1	acia0	acia0
2	acia1	acia1
3	-----	parallel printer

COMMAND DESCRIPTION SUBCOMMANDS	NOTES
SI SET INPUT PORT	SI sets the input port. It will prompt for a port number.
SO SET OUTPUT PORT	SO sets the output. It will prompt for a port number.
SB SET BAUD RATE	Sets the baud rate of either serial channel. The supported rates are: 75,110,134.5,150 300,600,1200,2000 2400,4800,1800 9600,19.2k The same rate is set for both the input and output of each channel.The monitor sets the serial ports to 8 data bits, 1 stop bit, and no parity.
DC DISPLAY CLOCK	Display time and date.
MC MODIFY CLOCK	Set the time and date. This command will prompt for entry of the information in the correct format.
DP DISPLAY PERIPHERAL DATA	Since the 8K bytes of ram and the registers of the various peripherals are byte wide devices, the processor sees them as the low bytes of consecutive even addresses. A simple display of memory will show these as alternating bytes. This command shows registers, and also data in the 8K as contiguous values.

CO COMMUNICATION

Clears the screen output by the RMS and turns the computer into a dumb terminal connected to serial port B.

LK LOAD FROM KEYBOARD

Loads ASCII text from keyboard. Prompts for a memory location to load to. Will load from the set input port.

S0,S1...S9 S RECORD LOAD

On the receipt of the command Sn, where n equals 0 through 9, the Microbox III will load an S record with the same number. This is a standard Motorola s record loader which facilitates transfer of hex data.

3.3 SYSTEM SERVICE MONITOR CALLS

3.3.1 Introduction to the system services

All of the system services are entered as subroutines via a jump table located between $07 FF00 and $07 FFFD. A jump table is used, rather than the more common TRAP exception for reasons of speed. Also for speed, all parameters are passed in and out of the routines in registers.

The routines are well-behaved; except for the graphics routines they only affect the registers used to call them and in which they pass values back. Also, in general, the values passed will be returned by the routines unless the register explicitly contains a new returned value.

Seven classes of routine are supported; General, Character I/O, String I/O, Hex I/O, Misc, Disk I/O and Graphics. Table 10 gives an overview of the routines, the rest of this section details their use.

TABLE 10: SYSTEM SERVICE CALLS

CLASS ONE: GENERAL	
_mcold	Power up cold start, all peripherals reset.
_mwarm	Warm start point.

CLASS TWO: CHARACTER I/O	
_status	Return active port status.
_inchne	Input character, no echo to output.
_inch	Input character with echo.
_outch	Output character.

CLASS THREE: STRING I/O	
_pdata	Print string until zero byte encountered.
_pcrlf	Print cr/lf.
_pstrng	Print cr/lf followed by string.
_outs	Print a space.
_outns	Print 'n' spaces.

CLASS FOUR: HEX I/O	
_inhex	Input hex number.
_prompt	Print a string, then call _inhex.
_outh	Print hex nibble.
_out2h	Print hex byte.
_out4h	Print hex word.
_out8h	Print hex longword.

CLASS FIVE: MISCELLANEOUS	
_delay	Wait for 'n' milli-seconds.
_beep	Beep for 'n' milli-seconds.
_random	Return random number.
_getrtc	Read data from rtc.
_putrtc	Write data to rtc.
_in	Write to security ram.
_out	Read from security ram.

CLASS SIX: MASS STORAGE I/O	
_select	Select drive.
_restore	Recalibrate to track 0.
_seek	Seek to track.
_read	Read sector.
_write	Write & verify sector.
_flush	Flush buffers

CLASS SEVEN: GRAPHICS	
_dvs	Define virtual screen.
_sync	Wait for vert. blanking.
_loadcmr	Load colour map ram.
_text	Display text screen.
_higraph1	640x500 mono screen.
_lograph	320x256 16 colour.
_clearg	Clear graphics screen.
_border	Set border colour.
_setpen	Defines drawing style.
_move	Move logical cursor.
_query	Return position logical colour value.
_point	Plot point.
_line	Plot line.
_rect	Plot rectangle.
_circle	Plot circle.
_fill	Plot filled rectangle.
_patdef	Define fill pattern.
_flood	Arbitrary area fill.
_scroll	Scroll 'n' lines.
_pan	Pan 'n' pixels.
_locate	Scroll and pan to centre cursor.
_mouse	Send curs. coords to mouse, return mouse coords and status.
_wordblt	Word block move.
_bitblt	Bit block move.
_init_csr	Initialize cursor.

3.3.2 Class One Routines - General

Entry point	Name	Function	Registers
$07 FE00	_mcold	Monk cold start point, entered on power up, all peripherals reset.	ENTRY: no registers set. EXIT: no registers set.
$07 FE06	_mwarm	Monk warm start point, for debug use.	

3.3.3 Class 2 routines - Character I/O

In general, d0.b contains the character code (7 bits/zero parity), and no other registers are altered. Four ports are supported. The active port is defined by the value of _oport and _iport as shown in Table 9. _oport and _iport can be set by monitor commands (see section 3.2.10).

Entry point	Name	Function	Registers
$07 FE30	_status	return active port status.	ENTRY: no registers set. EXIT: **Status Register.** Status register Z bit-0 if a character is queued.
$07 FE36	_inchne	Input char, no echo to outch.	ENTRY: no registers set. EXIT: **d0.b** Returns with zero parity byte in d0.b.
$07 FE3C	_inch	Input char with echo.	ENTRY: no registers set. EXIT: **d0.b** Returns with zero parity byte in d0.b. Character is echoed to current output port.
$07 FE42	_outch	Output character. If a character is output to the printer port, the routine waits for an acknowledge before returning.	ENTRY: **d0.b** Byte to be output must be placed in d0.b. EXIT: no registers set.

3.3.4 Class 3 routines - String I/O

These routines print to the current output port.

Entry point	Name	Function	Registers
$07 FE60	_pdata	Print string. Characters are output until a zero byte is reached.	ENTRY: **a0.l** a0.l points to the first character of the string. EXIT: no registers set.
$07 FE66	_pcrlf	Print cr/lf.	ENTRY: no registers set. EXIT: no registers set.
$07 FE6C	_pstrng	Print cr/lf followed by string. Characters are output until a zero byte is reached.	ENTRY: **a0.l** a0.l points to the beginning of the string. EXIT: no registers set.
$07 FE72	_outs	Print a space.	ENTRY: no registers set. EXIT: no registers set.
$07 FE78	_outns	Print 'n' spaces.	ENTRY: **d1.b** d1.b contains 'n', the number of spaces to print. EXIT: no registers set.

3.3.5 Class 4 routines - Hex I/O

In general, d1.l contains the hex number. None of the printing routines in this section print a CR/LF.

Entry point	Name	Function	Registers
$07 FE90	_inhex	Input free-format hex number.* This routine waits for a space before returning.	ENTRY: no registers set. EXIT: **d1.l** Upon return, d1.l holds the input number.
$07 FE96	_prompt	Print a string, then _inhex. This routine is used to prompt for, e.g. address input. Waits for a space before returning.	ENTRY: **a0.l** The first character of the string is pointed to by a0.l. EXIT: **d1.l** Upon return, d1.l holds the input number.
$07 FE9C	_outh	Print hex nibble.	ENTRY: **d1.b** The hex nibble is output in d1.b. EXIT: no registers set.
$07 FEA2	_out2h	Print hex byte.	ENTRY: **d1.b** EXIT: no registers set.
$07 FEA8	_out4h	Print hex word.	ENTRY: **d1.w** EXIT: no registers set.
$07 FEAE	_out8h	Print hex longword.	ENTRY: **d1.l** EXIT: no registers set.

*Hex numbers are input in free format, i.e. they can start with any number of leading zeros and contain any number of digits up to 8. They are terminated by a space character.

3.3.6 Class 5 routines - Miscellaneous

Entry point	Name	Function	Registers
$07 FEC0	_delay	Wait for 'n' milliseconds.	ENTRY: **d1.l** d1.l contains 'n'. EXIT: no registers.
$07 FEC6	_beep	Beep for 'n' milliseconds.	ENTRY: **d1.l** d1.l contains 'n'.
$07 FECC	_random	Return random number. If d0 = 0, then the random number will be between 0 and 255. If 0 < d0 <= 255, then the random number will be less than or equal to d0. If d0 > 255, it will become the new random number seed.	ENTRY: **d0.l** EXIT: **d0.b**

$07 FED2	_getrtc	Read rtc data block to (a0). On entry, a0.1 points to an 8 byte block of memory. On return the block will contain the following data in BCD format. First byte bits 0 - 3 - hundredth seconds (0-9) bits 4 - 7 - tenth seconds (0-9) Second byte bits 0 - 3 - seconds (0-9) bits 4 - 6 - tens of seconds (0-5) bit 7 - 0 (not used) Third byte bits 0 - 3 - minutes (0-9) bits 4 - 6 - tens of minutes (0-5) bit 7 - 0 (not used) Fourth byte bits 0 - 3 - hours (0-9) bit 4 - tens of hours (0-1) bit 5 - a.m.(0) or p.m.(1) 12 hr. clock second tens of hours 24 hr. clock bit 6 - 0 (not used) bit 7 - 12 hr. clock when 1 24 hr. clock when 0 Fifth byte bits 0 - 3 - day (1-7) bit 4 - reset. If 1 reset will be ignored. If 0, reset will abort data transfer without changing clock data. bit 5 - oscillator on/off. Off is 1. bits 6 - 7 - 0 (not used.) Sixth byte bits 0 - 3 - date (low numeral) bits 4 - 5 - date (high numeral) bits 6 - 7 - 0 (not used) Seventh byte bits 0 - 3 - month (low numeral) bit 4 - month (high numeral) bits 5 - 7 - 0 (not used) Eighth byte bits 0 - 3 - year (0-9) bits 4 - 7 - tens of years (0-9)	ENTRY: **a0.1** EXIT: no registers set.
$07 FED8	_putrtc	Write rtc data block from (a0). The eight byte block format is shown above.	ENTRY: **a0.1** a0.1 points to an eight byte block. EXIT: no registers set.

$07 FEDE	_in	Write data to the 8k byte security ram. a0.l points to the data in main memory, a1.l contains contains the offset in the 8k. Note that the bottom byte location is corrupted by reading and writing the clock and should not be used. d0.w contains the number of bytes to write to the clock.	ENTRY: **d0.w,a0.l,a1.l** d0.w - no# of bytes to write. a0.l - pointer to data in memory. a1.l - offset in 8k bytes. EXIT - no registers set.
$07 FEE4	_out	Read from 8k byte security ram. The registers should be set as above.	ENTRY: **d0.w,a0.l,a1.l** d0.w - number of bytes to read. a0.l - pointer to location in main memory to which to read. a1.l - offset into 8k bytes. EXIT: no registers set.

3.3.7 Class 6 routines - Mass storage I/O

This class of routines transfer data between memory and floppy, hard or ramdisk. All transfers are controlled by a Disk Control Block (DCB) pointed to by a0. Table 11 has the DCB format. An error code is returned in d0.l, zero is no error, any other value is an error code from the hardware (the ramdisk drivers simulate some errors). If the drive returning the error is the floppy, then these error codes obtain: 1 = busy; 4 = data lost 8 = data field error; 16 = track, sector or side not found; 24 = CRC error in ID field(s); 128 = write protected disk. Other disks are only guaranteed to return a non-zero value if there is an error.

The logical to physical drive mapping is supplied by the _select routine. Five types of physical drive are supported by Monk. Table 8 (in Section 3.2) gives the current types and additional information. The types are repeated here for convenience.

LOGICAL DRIVE/PHYSICAL DRIVE:

DRIVE 0 = FLOPPY DISK 0
DRIVE 1 = FLOPPY DISK 1
DRIVE 2 = WINCHESTER
DRIVE 3 = RAM DISK
DRIVE 4 = EPROM (OR STATIC RAM) DISK

The recommended floppy disk format is 80 track double sided/double density (800k bytes). Even tracks should be on side 0, and odd tracks on side 1. Each track should contain five sectors of 1024 bytes. The sectors should be numbered 1 <= sector <= 5. **NO** interleaving should be used as Monk uses track buffering to speed disk transfers. If another type of drive is to be supported, the ram vectors should be overlaid after calling _select.

TABLE 11
DISK CONTROL BLOCK

First byte = Logical drive 1,2,3,4 or 5
Second byte = Physical drive (supplied by _select)
Third and fourth bytes = Track number (as word)
Fifth and sixth bytes= Sector number (as word)
Seventh through tenth bytes = Data move address(as long word)

Entry point	**Name**	**Function**	**Registers**
$07 FEF0	_select	Sets physical drive type from the passed logical drive type. Copies vectors and if you are switching between floppy drives, swaps track registers. Finally, it selects the drive.	ENTRY: **a0.l** a0.l points to dcb EXIT: **d0.l** d0.l contains 0 if select was sucessful, if not it returns a hardware error code.
$07 FEF6	_restore	Re-calibrates drive to track 0.	ENTRY: **a0.l** a0.l points to dcb. EXIT: **d0.l** d0.l contains 0 if recalibration was sucessful, a hardware error code otherwise.
$07 FEFC	_seek	Seeks to track. Track is specified in the third and fourth bytes of the dcb. The first track is 0.	ENTRY: **a0.l** a0.l points to dcb. EXIT: **d0.l** d0.l contains 0 if seek was sucessful, a hardware error code otherwise.

$07 FF02	_read	Read sector. The sector number is contained in the fifth and sixth byte of the dcb. The first sector is 1. The data will be read to the address contained in seventh through tenth bytes of the dcb.	ENTRY: **a0.l** a0.l points to the dcb. EXIT: **d0.l** d0.l contains 0 if sucessful, a hardware error code other-wise.
$07 FF08	_write	Immediate write of sector, and verify. The sector number is held in the fifth and sixth bytes of the dcb. The data to be written is pointed to by the seventh through tenth bytes of the dcb.	ENTRY: **a0.l** a0.l points to the dcb. EXIT: **d0.l** d0.l contains 0 if sucessful, a hardware error code other-wise.
$07 FF0E	_flush	Flushes the write buffer to disk and clears the read buffer.	ENTRY: no registers set. EXIT: no registers set.

3.3.8 Class 7 routines - Graphics

There are two graphics modes supported by Monk, 640 x 500 monochrome and 320 x 250 sixteen colours. For reasons of speed, each mode has it's own routines. The selection of a mode places the correct vectors in Monk's jump table. The origin of the coord system is always in the bottom left hand corner of the screen. Support for a mouse and cursor is provided by these routines.

In general, for values passed by the user, a0.l points to any data structures, and d0.w and d1.w hold the X & Y coordinates (0 relative).

The first call to a graphics routine setting up a screen returns some of the CPU registers set to values that are used by any further graphics routines. See table 12.

TABLE 12: CPU REGISTERS BETWEEN PROCEDURES

Graphics routines hold values in registers between procedure calls. These values, with the single exception of the display base address, should only be altered by calling the appropriate routines. For example, the X and Y coordinates should be changed through _move. The registers not filled with a value below can be used.

Data register	Contents
D0	
D1	
D2	
D3	
D4	Mode / Pattern
D5	Bit Pointer
D6	Cursor X position
D7	Cursor Y position

Address register	Contents
A0	
A1	
A2	
A3	
A4	Current Pointer
A5	Display Base
A6	TRIPOS III reserved
A7	Stack

Mode = the drawing style set by _setpen. That is whether the pattern is used to replace, complement, etc.
Pattern = 16 bit pattern used for drawing. Also set by _setpen.
Bit Pointer = a value from 0 to 7 indicating which bit of the byte located by the current pointer the cursor is on.

Current Pointer = Address offset of cursor from display base.
Display base = a pointer to the base address of the screen. If you wish to locate the screen elsewhere in memory, for example lower to allow more room for large virtual screens, this value should be changed after the first screen set up call.

Yuk!

Entry point	Name	Function	Registers
$07 FF20	_dvs	Define virtual screen.	
$07 FF26	_sync	Wait for vertical blanking.	ENTRY: no registers set. EXIT: no registers set except if called after screen set up. See Table 12.
$07 FF2C	_loadcmr	Load colour mapping ram. The colour look-up table in the monitor is 32 bytes long, or 16 colours. The user can specify 32 colours (64 bytes), but see RMS documentation on how to access them. Also see RMS documentation on the correct format for the table.	ENTRY: **a0.1** a0.1 points to a colour look up table. EXIT: no registers set except if called after screen set up. See Table 12.
$07 FF32	_text	Display text screen. This routine blanks screen, then sets an 80 column by 25 line screen.	ENTRY: no registers set. EXIT: see Table 12.
$07 FF38	_higraph1	Display 640 x 500 monochrome graphics screen. This routine blanks screen, then sets up graphics screen with origin in lower left. Displayed points are in range 0-639h, 0-499v.	ENTRY: no registers set. EXIT: see Table 12.
$07 FF44	_lograph	Display 320 x 250 16 colour graphics screen. Action same as above screen. Displayed points are in range 0-319h, 0-249v.	ENTRY: no registers set. EXIT: see Table 12.
$07 FF4A	_clearg	Clear graphics area. Sets all the screen area to background colour.	ENTRY: no registers set. EXIT: no registers set except if called after screen set up. See Table 12.
$07 FF50	_border	Define border colour. The border can be set to one of 32 logical colours.	ENTRY: **d0.b** d0.b - logical colour (0-31). If called after screen set up See Table 12.

$07 FF56	_setpen	Defines drawing style. There four operations on pixels; REPLACE, which replaces any pattern on the screen with the one sent in the command;SET, which sets pels on the screen if they are set in the pattern sent;CLEAR, which clears any pixel on the screen which is set in the pattern sent; and COMPLEMENT, which clears any set pel on the screen and sets any clear pel. These modes are set by the value sent in do.b: REPLACE - 0 SET - 01b CLEAR - 10b COMPLEMENT - 11b d1.w is the 16 bit pattern sent, d2.b contains a 4 bit logical drawing colour, and d3.b contains a 4 bit logical background colour, to which pels are cleared.	ENTRY: **d0.b,d1.w,d2.b,d3.b** d0.b - drawing style d1.w - pen pattern d2.b - drawing colour d3.b - background colour EXIT: no registers set except if called after screen set up. See Table 12.
$07 FF5C	_move	Move logical cursor. X and Y values are held in d0.w and d1.w in all routines which need them.	ENTRY: **d0.w,d1.w** d0.w - x coordinate d1.w - y coordinate EXIT: see Table 12.
$07 FF62	_query	Return pixel value at x and y in the bottom 4 bits of d2.b. If the screen is in monochrome mode this routine will just return.	ENTRY: **d0.w,d1.w** d0.w - x coordinate d1.w - y coordinate EXIT: **d2.b, Tab. 13** d2.b - pixel colour. Also see Table 12.
$07 FF68	_point	Plot point. Colour and drawing style are defined in _setpen.	ENTRY: **d0.w,d1.w** d0.w - x coordinate d1.w - y coordinate EXIT: see Table 12.
$07 FF6E	_line	Plot line. Colour, pattern and drawing style are defined in _setpen. The line is drawn from the logical cursor to the x and y coordinates.	ENTRY: **d0.w,d1.w** d0.w - x coordinate d1.w - y coordinate. EXIT: see Table 12.

$07 FF74	_rect	Plot rectangle. Colour, pattern and drawing style are defined in _setpen. The x and y coordinates are the of the opposite corner of the rectangle. The cursor position will be the bottom left of the figure.	ENTRY: **d0.w,d1.w** d0.w - x coordinate d1.w - y coordinate EXIT: see Table 12.
$07 FF7A	_circle	Plot circle. Colour, pattern and pen type set in _setpen. The radius is held in d0.w.	ENTRY: **d0.w** d0.w - radius EXIT: see Table 12.
$07 FF80	_fill	Plot filled rectangle. Colour, pattern and drawing style are defined in _setpen. The x and y coordinates are the of the opposite corner of the rectangle. The cursor position will be the bottom left of the figure.	ENTRY: **d0.w,d1.w** d0.w - x coordinate d1.w - y coordinate EXIT: see Table 12.
$07 FF86	_patdef	Define fill pattern. a0.l should point to an eight byte fill pattern.	ENTRY: **a0.l** a0.l points to 8 byte pattern. EXIT: no registers set except if called after screen set up. See Table 12.
$07 FF8C	_flood	Arbitrary area fill. This routine will fill a closed area. If the area is not closed the screen and ramdisk will be filled. The x and y coordinates passed must be inside the area.	ENTRY: **d0.w,d1.w** d0.w - x coordinate d1.w - y coordinate EXIT: see Table 12.
$07 FF92	_scroll	Scroll 'n' lines. This routine will scroll to the border colour. A negative number (in twos complement form) scrolls up, a positive number down.	ENTRY: **d0.w** d0.w - number of lines to scroll. EXIT: see Table 12.
$07 FF98	_pan	Pan 'n' pixels. This routine will pan to the border colour. A negative number (in twos complement form) pans to the right, a positive to the left.	ENTRY: **d0.w** d0.w - number of pixels to pan. EXIT: see Table 12.
$07 FF9E	_locate	Scroll and pan so that cursor is in the centre of the screen. Areas originally off the screen will be represented by the border colour.	ENTRY: no registers set. EXIT: see Table 12.

$07 FFA4 _mouse

Send cursor coordinates to the mouse, and return mouse co-ordinates and status. The first two bits of d2.b will be set
- NO BUTTON - 00b
- RIGHT BUTTON - 01b
- LEFT BUTTON - 10b
- CENTRE BUTTON - 11b

ENTRY: **d0.w,d1.w**
d0.w - x coordinate (screen)
d1.w - y coordinate (screen)
EXIT: **d0.w,d1.w,d2.b,Tab. 13**
d0.w - x coordinate (mouse)
d1.w - y coordinate (mouse)
d2.b - mouse status
see also Table 12.

$07 FFAA _wordblt

Word aligned block move with logical op. d0.l contains the top right x coordinate of the source block. It also contains the mode in the top two bits (see below). d1.l contains the top right y coordinate. The cursor position (from d6 & d7) is the bottom left origin of the source. d2.l and d3.l hold the bottom left X and Y coordinates for the destination. The source will be rounded down to word size.
The mode bits work as follows:
- REPLACE Dest. w/source -00b
- OR Dest. w/ source -01b
- AND Dest. w/ source -10b
- EOR Dest. w/ source -11b

ENTRY: **d0.l,d1.l,d2.l,d3.l d4.b**
d0.l - top right x coord.(source)
d0.l (bits 30&31)- mode.
d1.l - top right y coord. (source)
d2.l - bottom left x coord. (dest.)
d3.l - bottom left y coord. (dest.)
EXIT: see Table 12.

$07 FFB0 _bitblt

Bit aligned block move with logical op. d0.l contains the top right x coordinate of the source block, with bits 30 and 31 set to the mode. d1.l has the top right y coordinate of the source. The cursor (d6 & d7) is the bottom left origin. d2.l and d3.l hold the x and y bottom left origin of the destination respectively. The mode bits work as follows:
- REPLACE Dest. w/ source -00b
- AND Dest. w/ source -01b
- OR Dest. w/ source -10b
- EOR Dest. w/ source -11b

ENTRY: **d0.l,d1.l,d2.l,d3.l d4.b**
d0.l - top right x coord. (source)
d0.l (bits 30&31)- mode
d1.l - top right y coord. (source)
d2.l - bottom left x coord. (dest.)
d3.l - bottom left y coord. (dest.)
EXIT: see Table 12.

$07FFB6 _init_csr

Initializes the cursor to the location and pattern passed. a0.l should point to an eight byte pattern. The cursor is true object 0.

ENTRY: **d0.w,d1.w,a0.l**
d0.w - x coordinate
d1.l - y coordinate
a0.l points to pattern.
EXIT: see Table 12.

3.4 The system register.

The ACIA (68681) contains a register which is used for system control functions. It has 6 inputs and 8 outputs. This register can be read directly, but is written to by setting or clearing bits using a mask. Thus writing $80 to _acia + _setreg will set bit 7 of the register without altering any other bits. The format of the register is shown in DIAGRAM P:

DIAGRAM P: SYSTEM REGISTER

bit	7	6	5	4	3	2	1	0
read	X	X	auto boot	out port	in port	spare	$\overline{\text{CTSA}}$	$\overline{\text{CTSB}}$
write	DRV	$\overline{\text{DDEN}}$	SIDE	REM/ LOCAL	PROG	BELL	$\overline{\text{RTSA}}$	$\overline{\text{RTSB}}$

DRV is 0 for drive 0 and 1 for drive 1, DDEN is 0 for double density and 1 for single density, side is 0 for side 1 and 1 for side 0, bell is 1 to enable the on-board sounder. Bits 2-5 on read are the state of the four set-up switches (0 for on). Note that bit 2 read can be linked to the 1772 WRPT line and bit 3 read can be linked to the 1772 IRQ line. These two lines have transition detectors which can generate an ACIA interrupt. The switches connected to these lines should be off if this option is selected (which it is for TRIPOS III and CP/M 68k). The program line enables the 8k byte CMOS ram with the RTC. If this line is not set, then the ram does not appear in the memory map, and can not be altered. The REM/LOCAL line switches the video output between the local and remote sources if the gen-lock option is fitted.

APPENDIX A: PROGRAMMING EXAMPLES

Here are some of the initialization routines in the software monitor. These are put here to give you some feel for programming the chips. They are not intended as recommendations, nor guaranteed the best way to use the chips.

```
* Device base addresses and equates.
rms     equ     $000ffe00     RMS registers base address
acia    equ     $000ff001     ACIA base address
rtc     equ     $000f0001     Real Time Clock base address
pia0    equ     $000ff101     PIA0 base address
pia1    equ     $000ff201     PIA1 base address
fdc     equ     $000f4001     Floppy disk controller base address
*
* RMS (68486 & 68487)
mem_map equu                $00     RMS memory map register
display_mode  equ           $01     List mode + lines/char row + bits/pel
int_stat  equ               $02     Interrupt status
border_col  equ             $03     Wrap mode + mapa + video on + border
colour
to_free  equ                $04     True object free register
vector_map  equ             $06     Vector mapping register
vert_scroll  equ $07        Vertical scroll register
hori_scroll equ             $08     Horizontal scroll register
drcs_tsa  equ               $0a     DRCS definition table start
to_tsa     equ              $0c     True object table start
fo_tsa  equ                 $0e     Fixed object table start
collision  equ              $18     Collision reports
rt_output  equ              $1c     Real time output
rt_input  equ               $20     Real time input
mem_type  equ               $24     Memory type and banks
video_op  equ               $25     Video interlace + format
sync_mode  equ                      $26
    Sync mode and output
screen_base  equ            $28     Virtual screen base address
vert_off  equ               $2c     Vertical offset register
hori_off  equ               $30     Horizontal offset register
screen_size  equ            $34     Virtual screen size
screen_width  equ           $38     Virtual screen width
clut  equ                   $40     Colour look up tables
```

MIXING HEX/DECIMAL

```
*
* ACIA
*
mode0  equu          00    Mode register A
stat0   equ          02    Status register A + clock select register
com0    equ          04    Command register A
data0   equ          06    Transmit and receive register
ipcr    equ          08    Input change register + aux. control register
isr     equ          10    Interrupt mask + interrupt status
ctur0   equ          12    Counter timer upper register
ctlr0equ             14    Counter timer lower register
*
mode1   equ          16    Mode register B
stat1   equ          18    Status register B + clock select register
com1    equ          20    Command register B
data1   equ          22    Transmit and receive register
ivr     equ          24    Interrupt vector register
sysreg  equ          26    Input port + output port config. register
clrreg  equ          28    Set (reset) output port bits + start counter
setreg  equ          30    Reset (set) output port bits + stop counter
*
* SYSREG (see section 3.5)
*
drv     equ         $80    bit 7 = drive select bit
dden          equ          $40
    bit 6 = drive density bit
side    equ         $20    bit 5 = drive side select bit
remote equ          $10    bit 4 = remote/local video source switch bit
program  equ        $08    bit 3 = program signal enable bit
bell    equ         $04    bit 2 = bell enable bit
init_spare  equ      2     Disk change flag  (Switch 0)
init_ip  equ         3     Initial input port (Switch 1)
init_op  equ         4     Initial output port (Switch 2)
a_boot  equ          5     Auto boot O/S  (Switch 3)
```

```
*
* PIA
*
pgcr    equ        $00   Port general control register
pacr    equ        $0c   Port A control register
pbcr    equ        $0e   Port B control register
paddr   equ        $04   Port A data direction register
pbddr   equ        $06   Port B data direction register
padr    equ        $10   Port A data register
pbdr         equ         $12
     Port B data register
psr     equ        $1a   Port status register
*
*Initialize RMS
*
 move.l #rms,a0            set up rms base address
 move.b #$20,mem_map(a0)   set control register set to unfolded
 move.b #$80,display_mode(a0)  set to bit plane mode
 move.b #$20,border_col(a0)  this bit is video enable
 move.b #$00,vert_scroll       initialize vertical scrolling register
 move.b #$00,hori_scroll(a0)  initialize horizontal scrolling register
 move.b #$c4,mem_type(a0)   tell the chips two banks 16 bit wide mem
 move.b #$2f,video_op(a0)   set screen to non-interlace 480x640
 move.b #$05,sync_mode(a0)  select composite sync
*next routine sets virtual screen base,vert. & horiz. offset, screen size
 move.l #rms + screen_base,a0
 lea.lrmstab(pc),a1
 moveq #19,d0
rmsl1 move.b (a1)+,(a0)+
 dfb     d0,rmsl1
*next routine loads the colour look up table
 move.l #rms + clut,a0
 move.b #0,(a0)+
 move.b #0,(a0)+           each entry in the table is two bytes long
*                          with lower 13 bits used. 13 = enable then 4 r,g,b
 moveq #61,d0              the table is 64 bytes long
rmsl2 move.b #$ff,(a0)+  and is now being loaded for monochrome
 dbf     d0,rmsl2
 bra.s   clt
```

```
* rmstab
rmstab dc.l      $00070000
 dc.l            $00000000
 dc.l            $00000000
 dc.l            $00004b00
 dc.l            $00000050
*
* Initialize screen to text
clt     bsr     cleart            subroutine not listed here
*Initialize floppy disk controller
 move.b #$ff,rtrk                 sets the read track to FF
 move.b #$ff,wtrk                 sets the write track to FF
 move.b  $0,fdc                   restores the disk drive 0 to track 0
*
*Initialize ACIA to 1200 baud
 move.l #acia,a0                  set up acia table in a0
 move.b #$13,mode0(a0)            sets A to 8 data bits, no parity
 move.b #$0f,mode0(a0)
      sets A to 2 stop bits
 move.b #$bb,stat0(a0)            set transmit and receive to 9600 baud
 move.b #$05,com0(a0)             enable receiver and transmitter
 move.b #$13,mode1(a0)
 move.b #$0f,mode1(a0)
 move.b #$bb,stat1(a0)
 move.b #$05,com1(a0)             same for B
 move.b #$04,ipcr(a0)
 move.b #$2,isr
*
*Initialize RTC
*
 lea.ltime_st,a0         time_st is an eight byte time space
 jsr     getrtc
*
*Initialize PIA0. On reset, PIA sets to two separate latching input ports
 move.b #$f0,pbcr+piao            set port b to bit I/O. H3=status pin,
*                                 H4= negated output pin
 move.b #$ff,pbddr+pia0           set port b to output
 move.b #$30,pgcr+pia0            set sense of handshake pins
```

Here also are the routines for reading from and writing to the clock.

```
*Get a time string from the real time clock
*Entry: (a0) pointer to 8 byte time space
*Exit: no registers altered
getrtc  movem.l d0/a0,-(sp)  save registers
        move.b   rtc,d0          the accessing sequence is set up by a read
        move.l   #$5ca33ac5,d0   accessing code
        bsr.s    putclk          move it to the rtc twice
        bsr.s    putclk
        bsr.s    getclk          get and store 8 byte string
        move.l   d0,(a0)+
        bsr.s    getclk
        move.l   d0,(a0)+
        movem.l  (sp)+,d0/a0     restore registers
        rts
putclk  movem.l d0-d1,-(sp)  save registers
        move.l   #31,d1          set up loop count
pcl     move.b   d0,rtc          move the bottom byte of code to rtc
        lsr.l    #1,d0           shift the code right
        dbf      d1,pcl          until finished
        movem.l  (sp)+,d0-d1     restore
        rts                      return
getclk  movem.l d1-d2,-(sp)  save registers
        move.l   #31,d1          set up loop count
        clr.l    d0
gcl     move.b   rtc,d2          get first bit in one byte
        and.l    #1,d2           clear garbage
        or.l     d2,d0           transfer bit to d0
        ror.l    #1,d0           make room for the next bit
        dbf      d1,gcl          until done
        movem.l  (sp)+,d1-d2     restore
        rts                      return
```

Appendix B: Chip Bibliography

Dallas Semiconductor Publications

Smart Watch DS1216

Motorola Publications

MC68486 Raster Memory Interface (RMI)
MC68681 Dual Asynchronous Receiver/Transmitter (DUART)
MC68230 Parallel Interface/Timer (PI/T)
Raster Memory System User's Manual*

Mullard Publications

SAA1099 Microprocessor Controlled Stereo Sound Generator for Sound Effects and Music Synthesis

Western Digital Corporation Publications

WD1770/1772 Floppy Disk Controller/Formatter

*Please note that at time of publication Motorola were at work on a new version of this manual. Also, no advance information for the 68486 companion chip, the 68487, was available.

APPENDIX C: MICROBOX III PARTS LIST

I.C's

IC Num	IC Type	Function	+5v	0v
IC1	MC68000-8	Processor	14,49	16,53
IC2	MC68646	RMI	36	13
IC3	MC68647	RMC	12,36	13,37
IC4-5	27CXXX	System eprom	28	14
IC6-7	27CXXX	User eprom	28	14
IC8-23	HM50256-15	Main memory	8	16
IC24	MC68681-8	Serial ports	40	20
IC25-26	MC68230-8	Parallel ports	12	38
IC27	WD1772-02	Floppy controller	15	14
IC28*	DS1216	Smart watch	28	14
IC29-30	74LS646	Data buffers	24	12
IC31	74HCT04	Misc logic	14	7
IC32-34	74HCT257	Address mpx	16	8
IC35	74HCT244	Bus buffer	20	10
IC36	74HCT245	Bus buffer	20	10
IC37	1488	RS-232 driver	-	7
IC38	1489	RS-232 receiver	14	7
IC39	7406	Floppy drive buffer	14	7
IC40	7407	Floppy drive buffer	14	7
IC41	74HCT138	Master decode	16	8
IC42	74HCT138	Async decode	16	8
IC43	74HCT11	Decode	14	7
IC44	74HCT08	Decode	14	7
IC45	74HCT148	Interrupt encode	16	8
IC46	74HCT138	Interrupt decode	16	8
IC47	74HCT03	Misc logic	14	7
IC48	Si7661	-12v gen	-	3
IC49	SAA1099	Sound	18	8
IC50	74HCT138	Sync decode	16	7
IC51	74HCT139	Misc logic	16	8

* IC28 includes a 5564 (8k static ram).

Discrete Components

Resistors all 1/8w 5%

R15,20,25	3 x 22R	R17,22,27	3 x 100R
R18,23,28	3 x 200R	R6,7	2 x 750R
R12	820R	R14,19,24	3 x 910R
R2,4,8,11	8 x 1K0	R9	4K7
R16,21,26,32			
R1,3,10,13	12 x 10K	R5	20K
R29,30,31,33			
R34,35,36,37			

Resistor packs

RP3,4	68R x 7 14 pin dip		
RP5	330R x 4	RP8	820R x 9
RP1	10K x 5	RP7	10K x 8
RP2,6	10K x 9		

Capacitors

C7	15p ceramic	C2	33p ceramic
C65	68p ceramic	C1,3,5	3 x10n ceramic
C6	100n ceramic	C8,9,10,11	4 x 1n5 poly
C12,13	2 x 22u 25v Tant		
C40-64,66,67	22 x 100n ceramic decoup		

Transistors

Q1	BC179	Q2	2N2222A
Q3	BC109C	Q4,6,8	3 x 2N3904
Q5,7,9	3 x 2N3905		

Misc

X1	35.46895Mhz	X2	3.6864Mhz
OSC1	8Mhz 'can'		
ZD1	6v8 'Transorb' RS 283-225		
L1	330uH		
SOUND1	5V sounder RS 249-794		
SW1	Reset switch RS 337-598		
SW2	Program switch RS 334-224		
SW3	dil switch RS 332-981		

Connectors

LINKS	83 x 'Berg' type pins + links
CONN1,2	2 x 64/64 way Euro
CONN3	34 way vertical idc

IC Sockets

1 x 64 way	4 x 48 way	1 x 40 way
6 x 28 way	2 x 24 way .3" pitch	
2 x 20 way	1 x 18 way	25 x 16 way
8 x 14 way	1 x 8 way	

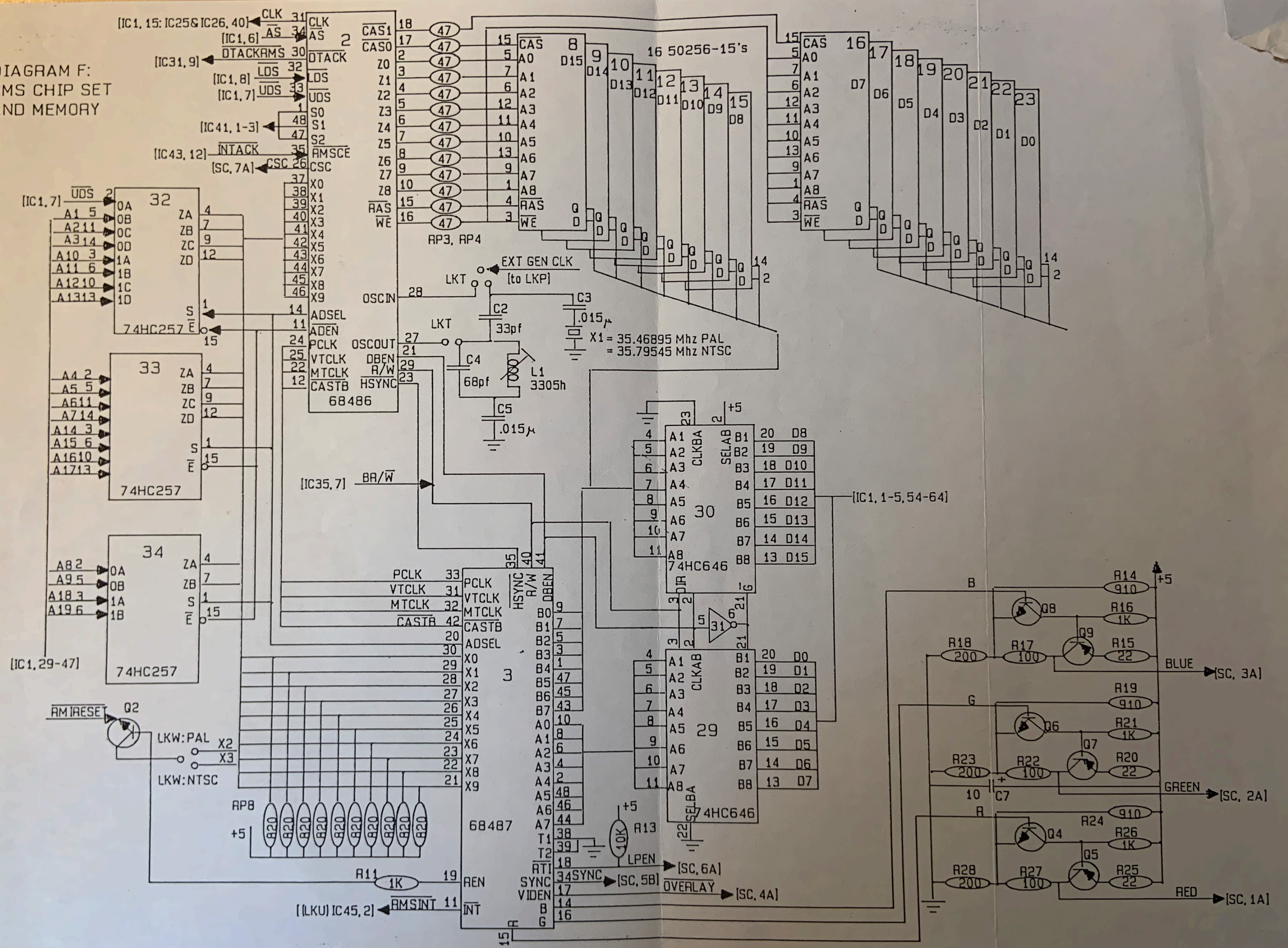
APPENDIX D: TRIPOS III and CP/M 68k Link Configuration

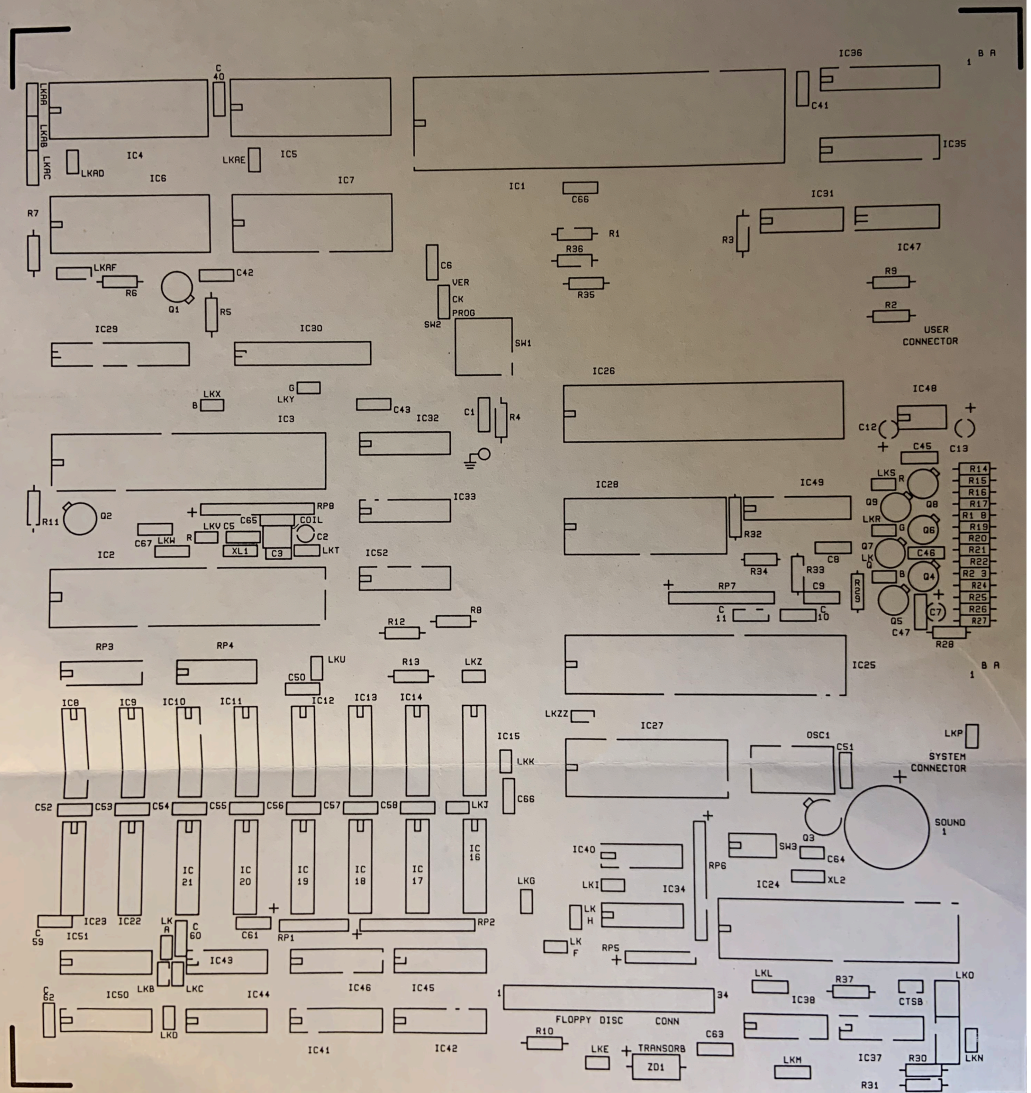
Here are the link and the configuration for TRIPOS III and CP/M 68k. In this configuration the system eproms are 27512s, and the user eprom sockets accept 8k byte static rams.

LINK	CONFIGURATION
A	shorted
B	shorted
C	shorted
D	Interrupt acknowledge shorted to link H, *ACIAINTACK *PIA1INTACK shorted to link H, interrupt acknowledge
E	shorted
F	open
G	shorted
H	*ACIAINTACK shorted to link D, interrupt acknowledge Interrupt acknowledge shorted to link D, *PIAINTACK
I	shorted
J	shorted
K	open
L	TTL
M	RS232
N	open
O	see diagram in section 2.3
P	open
Q	coaxial connection to link X
R	coaxial connection to link Y
S	coaxial connection to link V
T	open (traces intact)
U	shorted
V	to link S
W	PAL
X	to link Q
Y	to link R
Z	open
AA	+5
AB	Upper Write Enable
AC	Lower Write Enable
AD	open
AE	open

AF	V_{pp} (Switch SW2 set to +5)
ZZ	shorted

DIAGRAM F:
RMS CHIP SET
AND MEMORY




USER CONNECTOR
SYSTEM CONNECTOR
SOUND 1
FLOPPY DISC
CONN
TRANSORB
IC1
IC2
IC3
IC4
IC5
IC6
IC7
IC8
IC9
IC10
IC11
IC12
IC13
IC14
IC15
IC16
IC17
IC18
IC19
IC20
IC21
IC22
IC23
IC24
IC25
IC26
IC27
IC28
IC29
IC30
IC31
IC32
IC33
IC34
IC35
IC36
IC37
IC38
IC40
IC41
IC42
IC43
IC44
IC45
IC46
IC47
IC48
IC49
IC50
IC51
IC52
SW1
SW2
SW3
VER
CK
PROG
OSC1
XL1
XL2
ZD1
COIL
LKAA
LKAB
LKAC
LKAD
LKAE
LKAF
LKP
LKO
LKN
LKM
LKL
LKK
LKZ
LKZZ
LKU
LKX
LKY
LKE
LKG
LKI
RP1
RP2
RP3
RP4
RP5
RP6
RP7
RP8