LSI

OCTOPUS

TECHNICAL MANUAL

THE LSI OCTOPUS COMPUTER SYSTEM

Technical Manual

Part No. 901133 - Iss 1

1 DECEMBER 1984

COPYRIGHT NOTICE

DISCLAIMER

L.S.I. Computers Ltd makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, L.S.I. Computers Ltd reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of L.S.I. Computers Ltd to notify any person of such revision or changes.

OCTOPUS TECHNICAL MANUAL

PREFACE

This Octopus technical manual is aimed at the system builder, programmer, hardware designer or the, justifiably, interested user who wishes to know more about the Octopus computer system.

It is intended to be two documents in one - a hopefully fairly readable overview of the Octopus computer system and a highly detailed technical reference manual. In order to achieve this, the reader is urged to cover the main sections, skipping the Appendices at a first reading, to gain a basic feel for the overall system. This should only take one to two hours. The location of the technical details should then be clear and the manual's second role as a reference manual will become more relevant.

In general, detailed technical parts, schematics, connection details, listings, etc. are kept in the Appendix in order to ease the reader through the general sections without becoming bogged down in detail.

It is assumed that the reader has a basic working knowledge of computer systems and is familiar with the jargon used in the manual. At the end of this preface is a list of reference books and, if the reader finds himself somewhat out of his depth, he is urged to turn to this section and to read some of the excellent primers mentioned there.

To save describing the large scale integrated circuits, which is best done by their manufacturers, a list of relevant data sheets and where they may be obtained is also included.

This manual is broken into three main sections. The first section is the basic Octopus system. It gives an overview of the system philosophy, a description of the hardware and software and their interaction, and contains (in the Appendices) the detailed technical data.

The second section is split into many smaller sections, one for each of the Option Boards contained in your system. You may, of course, not have any. If option boards or add-on units are installed at a later date, the documents enclosed with them should be added to this section in order to keep all the data together.

The third section is a set of Application Notes which demonstrate a number of ways the Octopus may be used. Application notes are being written on a continuous basis as more and more uses are devised. A list of the currently available set is available from your dealer and any obtained should be added to this section.

The last pages of this manual are some of the most important - the reader reply cards. This manual was written solely for you, the reader, and, like the Octopus itself, is intended to grow and adapt to the changing requirements of computing. It is through these cards that we obtain the feedback necessary to provide you with your next Octopus add-ons and to give you the best technical support - so please use them.

Whilst every effort has been made at technical accuracy, LSI Computers Ltd is not responsible for any errors or malfunction arising out of using this manual.

LSI

OCTOPUS

TECHNICAL MANUAL

SECTION 1

THE OCTOPUS COMPUTER

CONTENTS

INDEX OF ILLUSTRATIONS

INTRODUCTION TO SECTION 1

This section deals with the basic Octopus system and does not cover the options and add-on units which, if your system has any, will be found later in the manual in Section 2.

There are three chapters to be read through and these cross refer to the Appendices which are primarily the reference section and contain all the detailed information.

Chapter 1 is an overview and discusses the Octopus system philosophy by introducing the various components that can go into a complete Octopus system. This identifies where system functions are performed and shows the way in which your system can be expanded if required at a later date.

Chapter 2 describes the Octopus hardware covering first the main logic board, occasionally abbreviated to MLB, the power supply unit (PSU), the disc systems, the display monitors and the keyboard.

Chapter 3 describes the system firmware contained in the on-board EPROM. A list of PROM calls and their functions and parameters are covered in section 1. Section 2 covers some of the more sophisticated features such as the interaction of the two processors and handling of the serial interfaces (an area of some difficulty under CP/M). Section 3 discusses certain MSDOS features and utilities, the equivalent CP/M utilities are covered in the Octopus User's manual.

Chapter 1- System Overview

1.0.0 SYSTEM PHILOSOPHY

There is one major fact that separates the Octopus system from most micro-computer systems and that is its great capacity for change and growth. It is therefore important to stress that your Octopus system is not just today's offering of computer technology but one of the first of a vast range of system configurations - the majority of which have not yet even been considered! This effectively destroys the 'built-in obsolescence' argument that is quite rightly levelled at many of the currently available systems in the market.

In reading the subsequent sections, the reader familiar with mini and mainframe computers will see many design features taken from these admirable historical examples of computer architecture. It was their flexibility that gave them their system building power. Hence, this has been a prime influence on the Octopus system design. (It is interesting to note at this juncture how many mini and mainframe manufacturers are now basing their future products upon micros.)

Having designed such a flexible system, one is unfortunately at somewhat of a loss as to where to start to describe it as if any particular configuration is explained it is more than likely not going to be the one you, the reader, has just purchased! At this point then, an apology: the manual will describe all the bits and pieces that go to make up the Octopus system and probably some that you do not have. A number of 'typical' system configurations will be covered and it is asked of your goodself, the reader, to read their relevance into your own particular system.

1.1.0 THE BUILDING BLOCKS

The heart of the Octopus is its main logic board containing all the basic system hardware functions and the firmware. This is mounted in the base of the Octopus box and with the power supply unit represents in theory the simplest functional Octopus system.

This is, however, not a terribly useful Octopus as it cannot communicate without peripherals and has nowhere to store its data.

Communication with the main logic board is usually 'in' via the keyboard interface and 'out' via the integral video control circuits. Some multi-user systems may only have remote VDUs attached, in which case the firmware will endeavour to communicate via these. The presence of the keyboard is detected so, if you unplug it and power up, you will notice the lower part of the power up screen (the log on menu) will not be there - it will have gone to the VDUs if you have them (Note that further dialogue is operating system dependant however).

A number of keyboards and monitors are intended to be used with the Octopus as new software becomes available; these are covered in detail in Section 2. It is worth noting that the user can have either composite video or TTL monitors attached simultaneously and these may be 'daisy-chained' dependent upon the monitor in use. This is particularly useful for training purposes or CAD applications where graphics can be displayed separately to the text.

Most Octopuses (not Octopi as they are Greek not Latin) have some form of integral disk storage. This will either be one or two 400K or 800K floppy disk drives or one floppy drive and a Winchester hard disk drive of one of a variety of capacities. See Appendix 3 for all the current configurations and an explanation of how the model numbering system works.

It is also possible to have a 'driveless' Octopus which can be used as a rather extravagant VDU or terminal emulator or, more usually, as a specialised node in a network, e.g. a graphics workstation, communications controller, network gateway, etc.

Refer to figs.(ia) & (ib) to see how these parts fit together for the two main system configurations. You should be familiar with the external connections from your user's manual - for completeness, these are covered in the Appendices. A few points of note are mentioned here. All Octopuses have identically rated power supplies allowing expansion right through the range up to the most power hungry configuration.

The fan is buried inside to reduce noise and to give a high velocity airflow over the disk drives, which can get alarmingly hot if not cooled. In order to function properly, it is important to have adequate room on each side of the chassis to allow the air to circulate freely outside the unit.

The loudspeaker is located against the left-hand side at the front, its sound escaping through the left side air vents.

In the centre, towards the rear of the main logic board, is the secret of the Octopus's expansion capability - the Option Bus Connector. It is through this socket that most of the expansion features are added.

Most micro-computer buses are of the type that allows the user to attach extra resources, e.g. memory, I/O, controllers, etc., and require a fairly high level of control as these tend to be somewhat 'loosely coupled'. The more sophisticated buses will allow the addition of extra 'bus masters' which are allowed to gain control of the system and its resources as required.

It was felt that neither of these methods of attachment was suitable for the sort of system expansion envisaged for the Octopus. The bus chosen is in effect an extension of the fundamental architecture of the machine and provides in addition to all the features of the above bus types a level of connection to the basic workings of the machine that is more

usually found on a mini-computer backplane. Four independent direct memory access control signals and seven levels of interrupt are provided to give the fastest system response to added resources, resulting in a true integrated system rather than a box full of boards. Although the main logic board data bus is 8 bits wide (for compatibility with the Z80), the expansion bus is 16 bits wide so that new processors may be attached at a later date. Baud clocks, video sync signals and access to the loudspeaker are also provided for easy expansion.

In contrast to the complexity of the electrical connection, the mechanical aspects are elegantly simple. To save supplying a more often than not unused backplane and connectors, forcing the less demanding user to pay for something he may not use and to overcome the cost of a racking system, the Octopus expansion bus makes use of its novel 'stacking' system (see fig.(ii)).

The MLB has its 96 way bus connector socket pointing up. The option boards have 96 pins pointing down with corresponding sockets above. These pins plug into the sockets one above the other. To mechanically 'lock' the arrangement, the rear panels have 'pips' on the upperside and 'dents' underneath; these register with each other to give a rigid box sectional structure. As the MLB has a taller rear than the option boards, a link piece jumps over the gap and provides a 'star point' for earthing the option boards.

Apart from the cost advantage of only buying your bus in incremental units as you buy your boards, the electrical path is via large square sectional area pins and not small copper tracks, hence reducing bus impedance and associated noise problems.

This kit of building blocks gives the system designer enormous scope to tailor his particular configuration to best match the immediate requirements, still allowing the option of changing the system as needs dictate.

Typical configurations may be :-

- A small twin floppy 'starter system'
- A more sophisticated 10Mb winchester system
- A VDU added for two user operation
- A communications board, more memory and more VDUs added to give a multiuser system sharing one processing unit.
- A graphics workstation with 3/4 Mb memory
- A single drive 'personal computer'
- The above three systems networked to give one large computing system with, say, a 40Mb winchester system added for mass storage
- A shared 'gateway' to a mainframe added
- A similar (or even the same node) gateway to various telecommunication services
- More multi-terminal 'cluster' controllers added
- More mass storage and streaming tape back-up
- High speed telecommunications links to other regional or Octopus networks
- Etc., etc., etc.,

It should now be apparent that this is just a fraction of possible system configurations, the limit being the extent of system builder's imagination !

The rest of this manual is intended to give sufficient insight into the Octopus system to stimulate this imagination in order to provoke new uses for Octopus computer systems.

WINCHESTER OCTOPUS

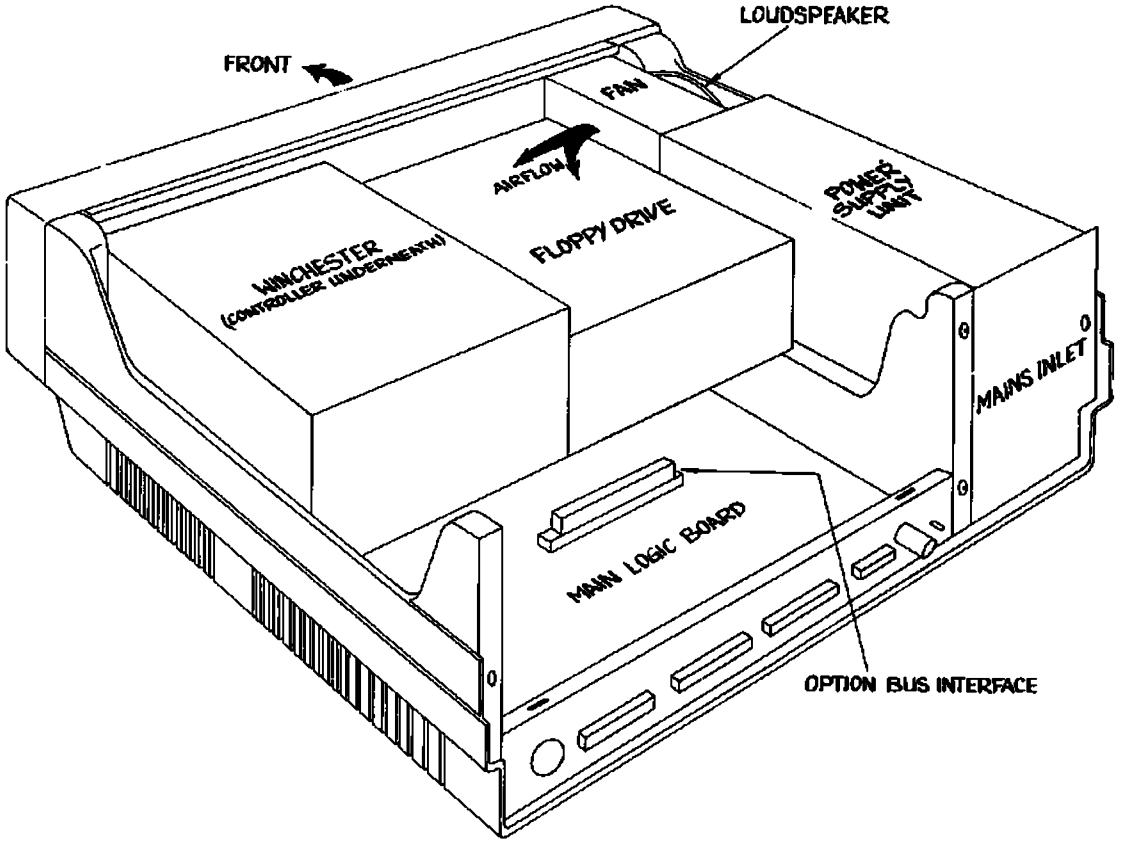


fig (ia)

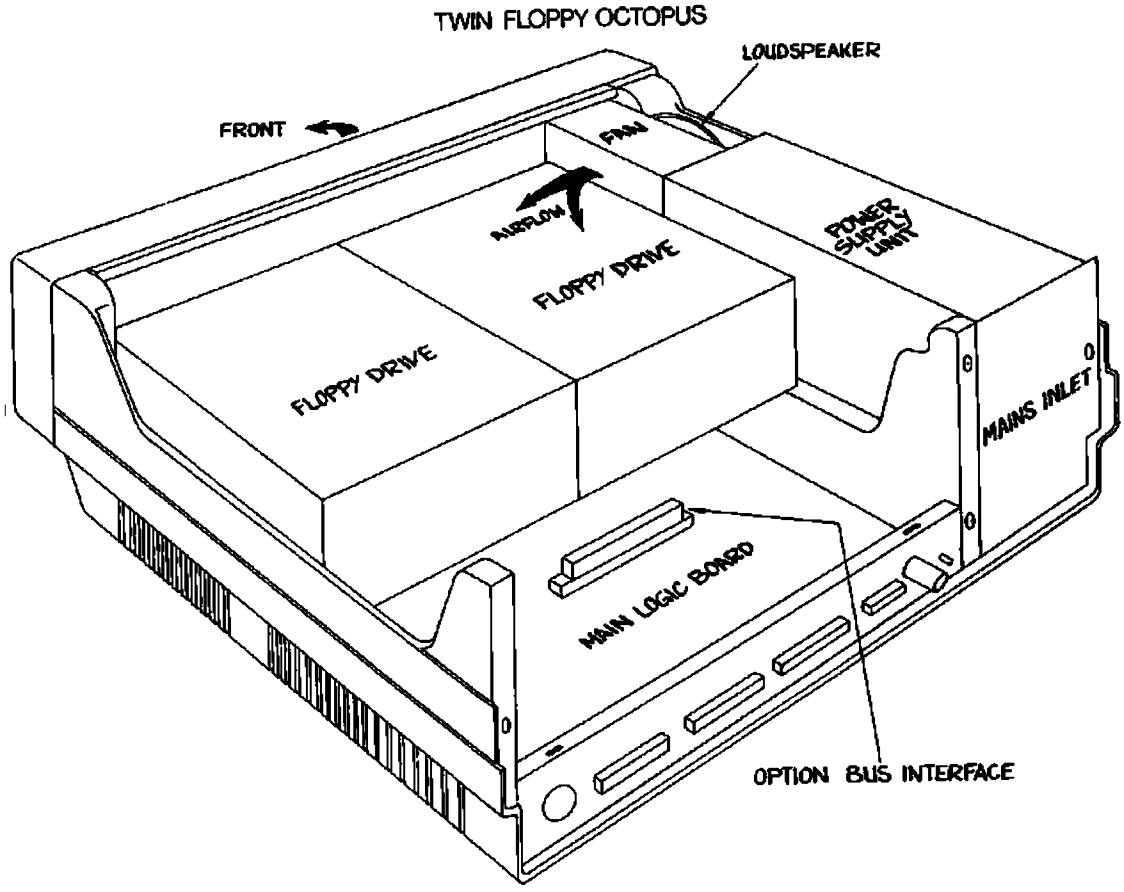

TWIN FLOPPY OCTOPUS
LOUDSPEAKER
FRONT
FAN
AIRFLOW
FLOPPY DRIVE
FLOPPY DRIVE
POWER SUPPLY UNIT
MAINS INLET
MAIN LOGIC BOARD
OPTION BUS INTERFACE

fig (ib)

STACK SYSTEM FOR OPTION BOARDS

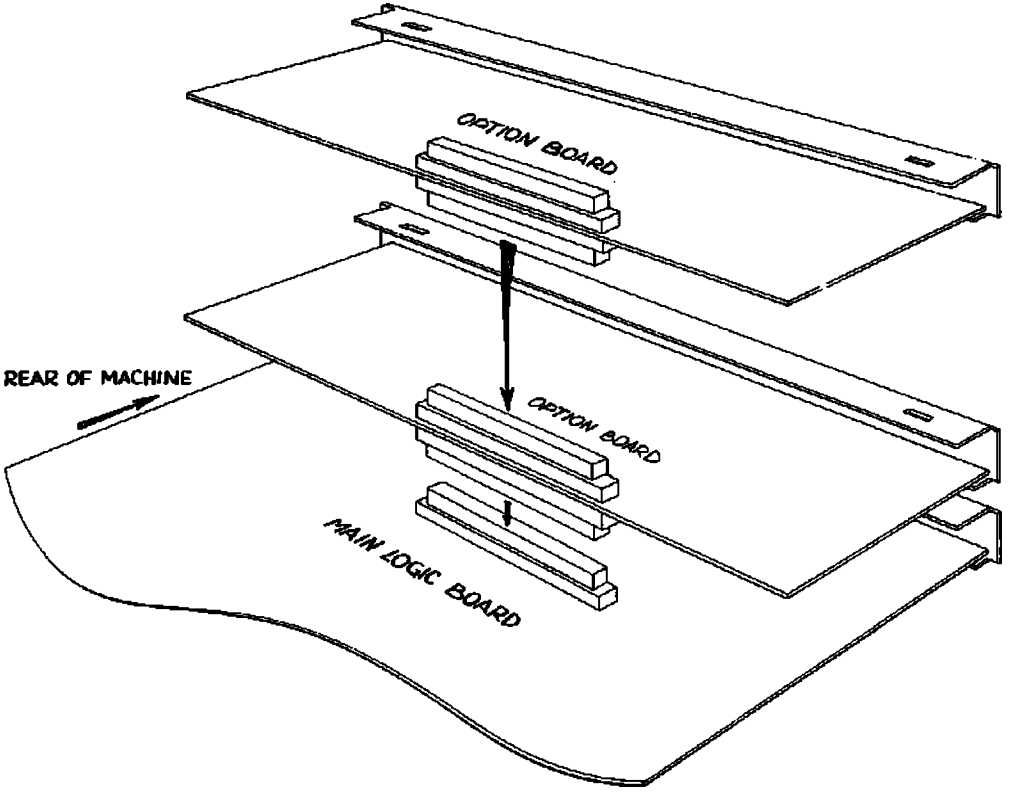


fig (ii)

Chapter 2 - Hardware Description

2.0 INTRODUCTION

This chapter deals with the major functional hardware units and divides into six sections.

Section 1 - covers the main logic board and its various sub-divisions.

Section 2 - deals with aspects of power distribution.

Section 3 - splits into two parts, one for the floppy discs and one for the winchester discs and their controller card.

Section 4 - again splits into two parts, one for the monochrome monitor and one for the colour monitor.

Section 5 - covers the keyboard.

Section 6 - discusses miscellaneous mechanical details.

TOWN PLAN BLOCK DIAGRAM

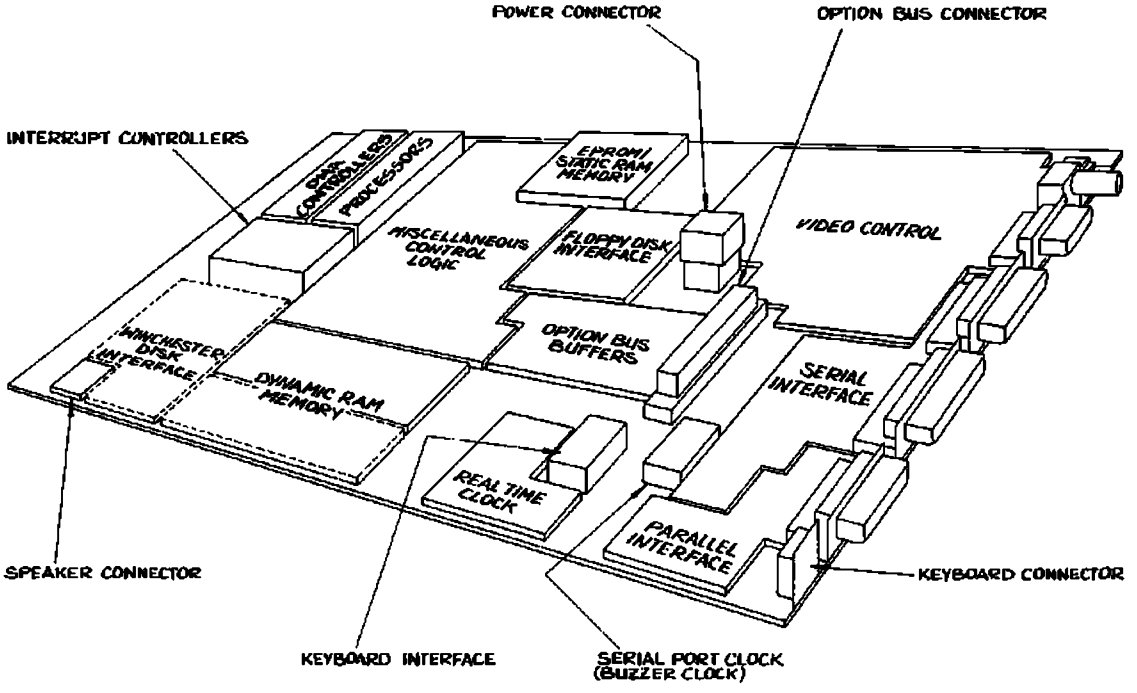


fig. 2.1

2.1 THE MAIN LOGIC BOARD

This is the heart of the Octopus computer, it is an extremely dense four-layer printed circuit board and contains a number of very large scale integrated circuits (VLSI) that form various functional sub-sections of the board (fig. 2.1). The system buses that connect all these devices together pass through a number of bus buffers which control the flow of data and addresses around the system.

The control of these buffers and the interconnecting control signals from the VLSI circuits are co-ordinated by four Field Programmable Logic Arrays designed by LSI. These dramatically reduce the number of small 'random logic' devices needed by a factor of over five to one, with a corresponding reduction in board space. Due to the fewer number of chips the board is inherently more reliable.

Each of the above circuit device types are referred to in the appropriate sections in this Chapter. The addresses of the devices and the system memory map are covered in Appendix 7.

2.1.1 THE PROCESSORS

Synopsis - Two processors provide the capability to run both eight and sixteen bit software.

Description - Refer to fig. 2.1.1, the main logic board includes a Z80-B processor (Ref 7) to support eight bit software, and an 8088-2 processor (Ref 1) to support sixteen bit software via an eight bit data bus interface. Provision is also made for off -board processors to communicate with main board resources by temporarily owning the main board bus, giving a route for expanding the processing power of the basic system as and when required. Two 8237 devices (Ref 1) support normal DMA transfers, and also arbitrate the bus ownership requests from off-board masters.

The system address and data buses permit access to memory and I/O, and may be driven by any of the 8088-2, Z80-B, DMA and off-board masters. All potential users contend for ownership of the main board facilities, but only one is allowed to be active at any one time while all the others are held dormant. Arbitration is handled by an FPLA (Field Programmable Logic Array). The arbitor gives priority to DMA requests, and the buses will always be released for DMA handling at the earliest opportunity. This applies to normal DMA transfers and also to off-board master requests via the DMA controllers. Following the DMA cycle the buses are returned to the previous owner, either the Z80-B or the 8088-2.

Either processor may intentionally relinquish bus control to the other processor. The 8088-2 hands over to the Z80-B by executing an I/O cycle to a pseudo device called Z80SEL. The Z80-B hands over to the 8088-2 by performing any I/O cycle. Additionally, an interrupt request during Z80-B ownership automatically hands control to the 8088-2 to service the interrupt.

The various READ and WRITE lines from potential bus owners are melded into one consistent set of control signals with optimum timing by another FPLA. This device also produces a signal called BSMC (Bus Start Memory Cycle) in advance of bus READ or

PROCESSORS BLOCK DIAGRAM

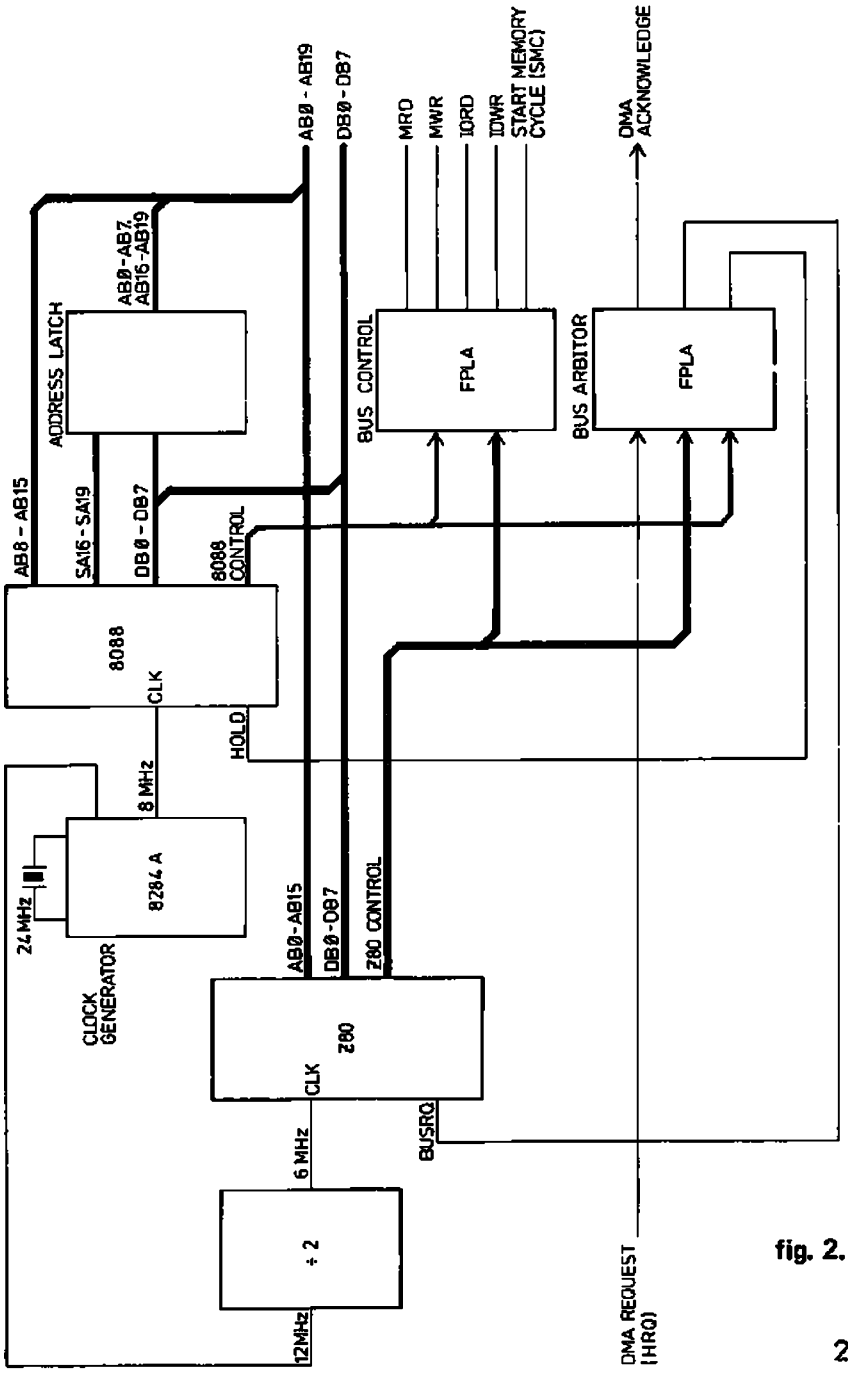


fig. 2.1.1

WRITE for a memory access cycle, in order to initiate a dynamic RAM cycle as early as possible and obviate the need for wait state insertion. The Z80-B is not permitted to access I/O devices directly, hence all Z80-B I/O is handled indirectly by a temporary processor switch to the 8088-2. Similarly, off-board bus masters are NOT permitted to access on-board I/O devices.

The 8088-2 supports a one megabyte address range directly via twenty address lines. The Z80-B and DMA controllers have a restricted address range of sixtyfour kBytes, both of which are expanded to the full one megabyte via programmable four bit bank select registers. (See section 2.1.2, the DMA structure).

An 8284A clock generator and driver device (Ref 2) gives an 8 MHz clock for the 8088-2, a 6MHz clock for the Z80-B via a divider and a 4 MHz clock for the DMA controllers. All these clocks originate from a 24 MHz crystal reference connected to the 8284A. Additionally the 8088-2 and DMA controllers are synchronised to slow I/O devices with the help of the 8284A, which asserts not READY in order to implant wait states into bus cycles. See section 2.1.4.

2.1.2 THE DMA STRUCTURE

Synopsis - Two DMA controllers support data transfer between disc and memory, memory refresh and bus arbitration for four levels of off-board master requests.

Description - Direct Memory Access (DMA) allows fast and efficient access to memory via dedicated hardware rather than via the control of a processor such as the 8088-2 or Z80B. This means that data is transferred at a higher rate, as memory accesses are not diluted with processor op-code fetches.

Two 8237A-5 DMA devices (Ref 3) are connected in a master/slave configuration, illustrated in fig. 2.1.2. There are two modes of DMA operation possible as a result of this hardware structure for the pre-allocated channels, each is described in the following text.

Mode 1. This is the simplest mode, where a request from an on-board device enters the MASTER controller. Hard disc and Memory Refresh are examples of this type of cycle. Resulting from this request the following sequence of events happen:-

1) The master DMA controller asserts a hold request, asking for the system buses.

2) The bus arbitor FPLA recognises the request, clears the processors from the bus and then asserts a hold acknowledge back to the master DMA controller, declaring the system buses free to use.

3) The 8237A-5 master DMA controller drives the system address and control buses to carry out the desired data transfer. Address bits eight to fifteen are output on a multiplexed address/data bus from the DMA controller, and have to be latched in a 74LS373 by the 8237A-5 address strobe signal, ADSTB. Address bits zero to seven are output on dedicated lines. This sixteen bit address field output by the 8237A-5 is augmented by a four bit field from a register file, giving the full one megabyte address range. For the Memory Refresh a memory cycle is performed but no data transfer takes place.

4) When the transfer is complete, the hold request is removed and control is subsequently handed back to an on-board processor.

Mode 2. This mode supports on-board DMA requests into the SLAVE controller, and is used only for floppy disc data transfers. Following a request of this type, the following sequence of events ensues:-

1) The slave DMA controller asserts a hold request, which in turn asserts a DMA request into the cascaded channel of the master DMA controller.

2) Events then follow the course described above for the Mode 1 DMA transfer, except the master controller does not actively drive any of the system buses. This task it delegates to the slave DMA controller via the master DMA acknowledge line. Thus the master DMA controller only assumes a role of bus arbitor.

3) When the transfer is complete, the hold request into the slave controller is negated. This ripples through the master controller and subsequently control is handed back to an on-board processor.

Due to the inherently slow response of the floppy disc controller device, DMA cycles involving the floppy have to be slowed down by the insertion of wait states. The 8237-A has a 'READY' input for just this purpose, and this is activated by a three state counter to lengthen the DMA cycle by 2 DMA clock periods during floppy cycles.

Un-allocated channels. For the un-allocated DMA channels, either the active mode 1 or the passive mode 2 described above may be implemented. This applies to the single channel of the master device and all three channels of the slave device.

If the active mode is used, the on-board controllers will assume the responsibility of driving address and control buses. If the passive mode is used, it is the responsibility of the requesting option board hardware to seize and drive the address and control buses once they are freed by the main board. The option hardware would typically contain its own 8237A-5 DMA controller programmed into active mode to support this requirement.

The DMA cycle length may be extended as and when necessary by implanting wait states after the T2 or T3 DMA state, by activating the bus wait line, BWAIT.

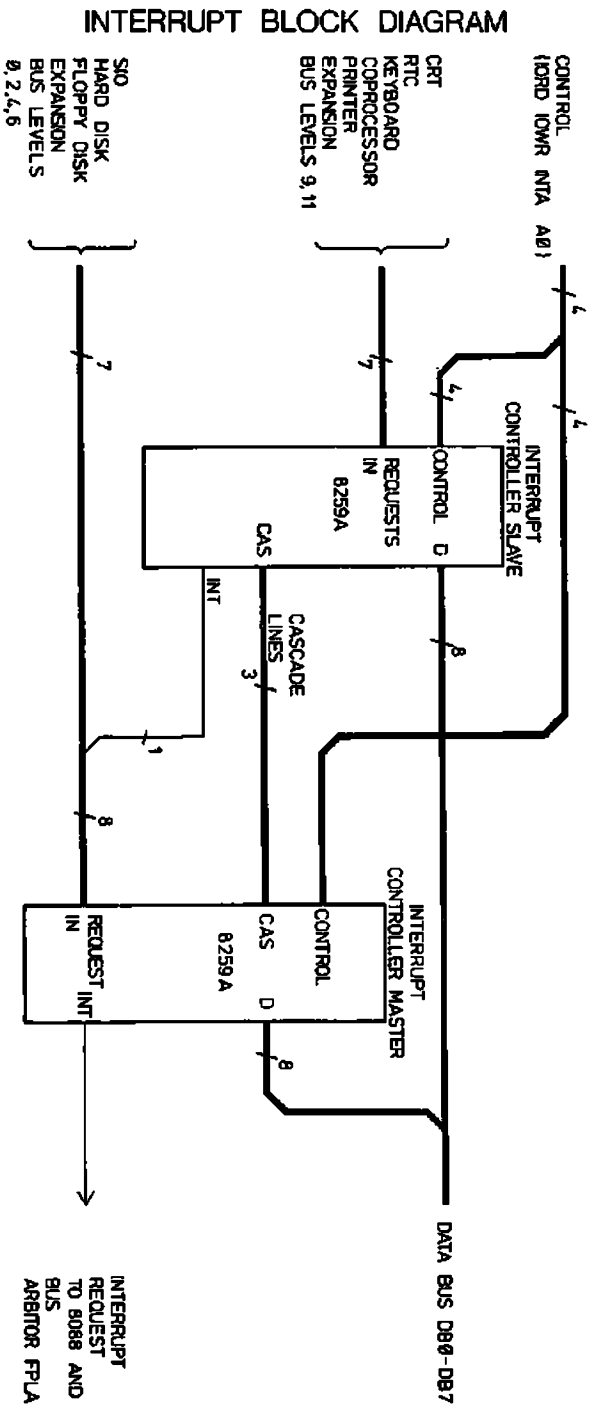

INTERRUPT BLOCK DIAGRAM
CONTROL
IORD IOWR INTA A0
CRT
RTC
KEYBOARD
COPROCESSOR
PRINTER
EXPANSION
BUS LEVELS 9,11
SIO
HARD DISK
FLOPPY DISK
EXPANSION
BUS LEVELS
0,2,4,6
INTERRUPT CONTROLLER SLAVE
CONTROL
REQUESTS IN
8259A
CAS
D
INT
CASCADE LINES
INTERRUPT CONTROLLER MASTER
CONTROL
CAS
8259A
REQUEST IN
INT
D
DATA BUS D0-D7
INTERRUPT REQUEST TO 8088 AND BUS ARBITOR FPLA

fig. 2.1.3

2.1.3 THE INTERRUPT STRUCTURE

Synopsis - Two interrupt controllers connected in a master-slave configuration support main board and expansion bus interrupt requests.

Description - Refer to fig. 2.1.3 the means of temporarily suspending execution of one part of a program to service the urgent needs of an asynchronous demand from an input or output device is most elegantly handled by interrupts.

With this technique there is a dedicated signal for each device, which is directly connected to the interrupt controller to request attention. The controller acts a manager by accepting individual requests from the peripheral devices, resolving differences in priority between concurrent requests, and also considers the importance or level of any interrupt routines currently in service. Based on these factors the controller may issue a demand to the processor for attention, whereupon the processor will acknowledge the request and accept a data byte, or vector, from the interrupt controller to indicate which input or output device is to be serviced. The existing state of the computer is saved before the interrupt service routine is started, allowing eventual return to the original program when the input or output is complete.

This contrasts with the alternative technique of polling, where every input or output device must be interrogated at regular intervals to determine its state of readyness for attention. This is extremely wasteful of processing power.

Two 8259A interrupt controller devices (ref 4) connected in a master/slave arrangement control the interrupt handling of the main and option boards. Together they give seven levels of on-board interrupt support, and seven levels of option board support.

The action of the master device is to consider all the factors outlined above and then to assert the interrupt request line to the 8088-2 if applicable. If the 8088-2 has control of the bus and software has enabled interrupts, the processor will respond with two interrupt acknowledge (INTA) pulses to the 8259-A. The first pulse is used to freeze the state of the 8259-A, the second to output a pointer onto the data bus which is used by the 8088-2 to vector to the appropriate interrupt service routine. Interrupt requests into the slave device are resolved internally to produce a cascaded interrupt request into the master controller, level-7, when applicable. This level-7 request is in turn processed by the master controller to produce an interrupt request to the 8088-2. When the 8088-2 responds, the first INTA pulse freezes both the master and slave interrupt controllers, and also the master controller defers control to the slave via the cascade lines, CAS 0 to CAS 2. The second INTA pulse then causes the slave to output the interrupt service routine pointer onto the data bus.

If an interrupt request occurs during Z80B bus ownership, a processor swap is invoked by the bus arbitor FPLA in order to handle the interrupt situation via the 8088-2. Following this a swap back to the Z80B may be made.

ADDRESS DECODING AND WAIT STATE GENERATION

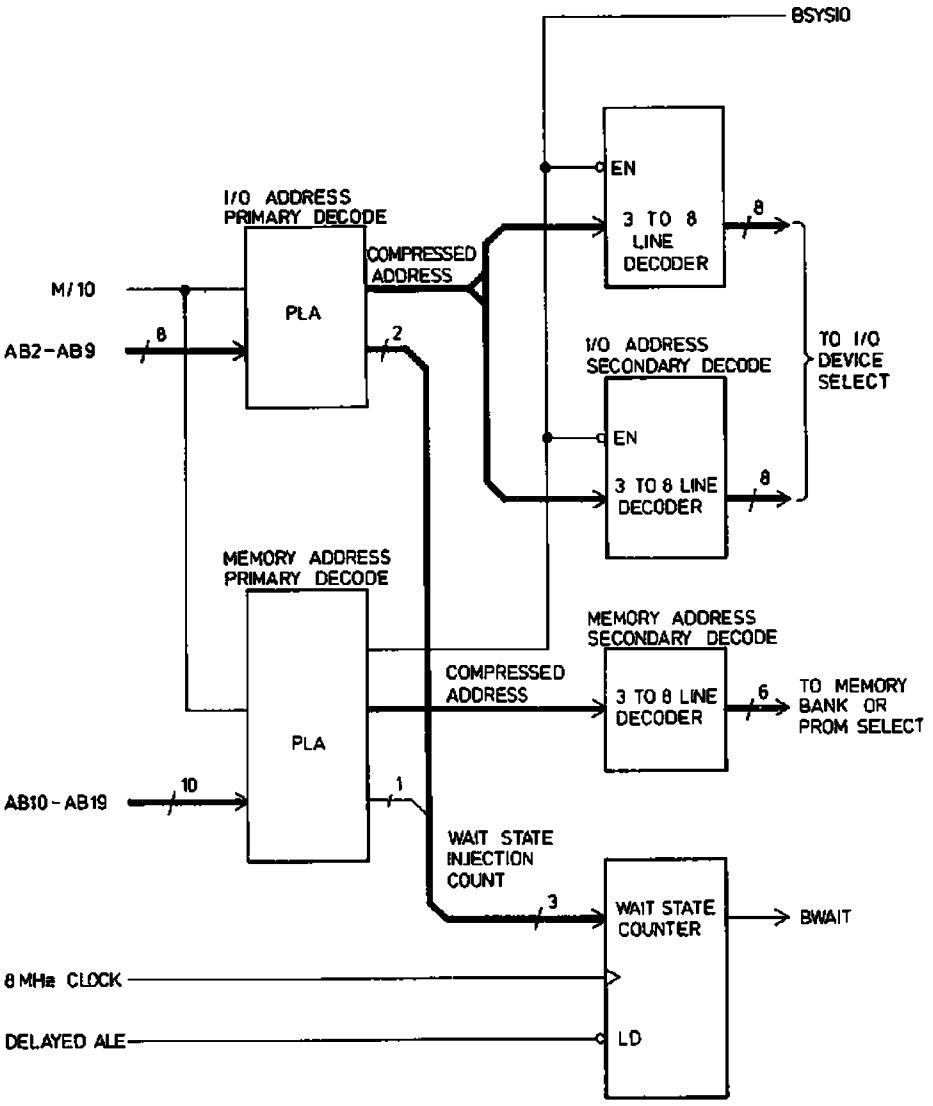


fig. 2.1.4

2.1.4 ADDRESS DECODING AND WAIT STATE GENERATION

Synopsis - Provides selection for 16 on board I/O mapped devices and 6 on-board memory mapped devices (App. 6). Programmable wait states can be inserted for on and off-board devices by this logic, allowing slower devices to be interfaced very efficiently.

Description - The address decoding logic, fig. 2.1.4, is focussed around two FPLAs (Field Programmable Logic Array) and three high speed decoders. The primary function of one FPLA is to decode memory addresses by monitoring the high order address bits (AB10 to AB19) during memory cycles and the other FPLA is to decode I/O addresses by monitoring the low order address bits (AB2 to AB9) during I/O cycles. On detection of a device address the FPLA generates a unique compressed address (4 or 5 bits wide) for that device. This is used to control the secondary decoders which provide the selection lines to each device on the board. The FPLA's jointly control the bus interface data transceiver and provide the option bus signal BSYSIO (L) (see 2.1.14). Two versions of the memory FPLA are currently installed - one for the 128Kbyte model, the other for the 256Kbyte model. By the use of two versions of the memory FPLA offboard memory can be made to begin after 128Kbytes or 256Kbytes respectively.

The FPLA's also provide information relating to the length of the current cycle detected. Memory cycles can have up to one wait state inserted, I/O cycles can have up to six wait states inserted. Each decoded address has an associated wait state field of one or two bits. These have been selectively programmed into the FPLA. If a cycle requiring extension is detected (e.g. VECSEL requires 6 wait states) the two bits are loaded into a down counter early in the cycle. The down counter holds up the processor for the programmed number of processor clock cycles allowing the processor to subsequently complete the cycle in the normal way.

Programming Details - The FPLA's are factory programmed and installed. User access to these components is not provided.

DYNAMIC MEMORY AND PROM FUNCTIONAL DIAGRAM

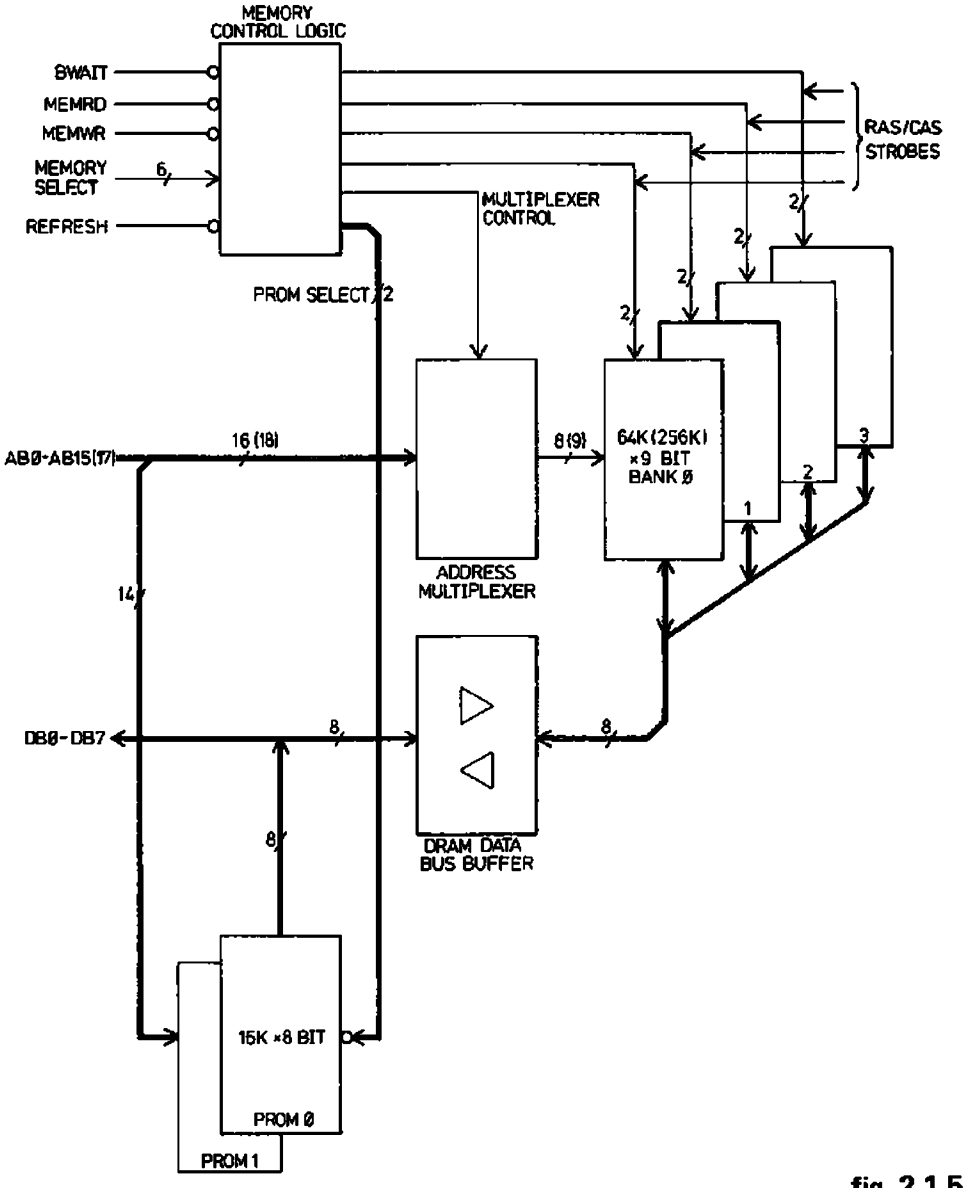


fig. 2.1.5

2.1.5 MEMORY

Synopsis - Upto 256Kbytes of dynamic memory (DRAM) and 32Kbytes of programmable read only memory (PROM) is provided on the main board. Dynamic memory is used for running transient software such as operating systems, applications programs etc., PROM is used for system diagnostics, bootstrapping (i.e loading a system loader into dynamic memory) and I/O primitives (see Chapter 3).

Description - The main logic board dynamic memory, fig. 2.1.5, consists of a maximum of four banks of 64K by 9 bits. The ninth bit is called a 'parity' bit and is used to protect the other 8 bits at that memory location. Should a memory location gain or drop an odd number of bits the parity detection logic will raise a non maskable interrupt (NMI) to the 8088 processor. The software running at that time will be responsible for the action taken in the case of such an error, LSI provided operating systems will indicate the error condition on the status line of the primary display device as a flashing high intensity warning message. Provision has been made on the logic board for the use of 256K dynamic memories instead of the currently used 64K. This will allow over 768Kbytes of memory to be accessed without resort to an extension memory option board when the devices become commercially viable.

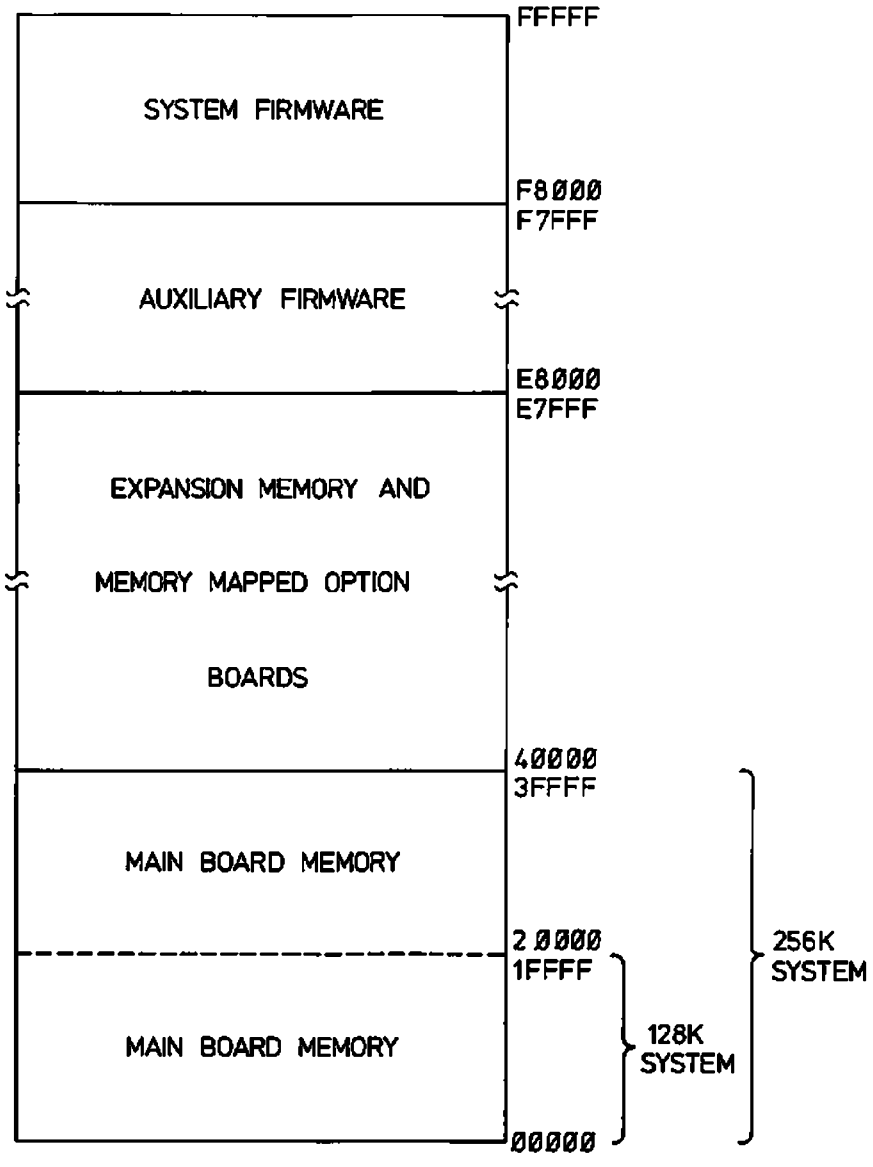
Refresh of the dynamic memories is accomplished by devoting one direct memory access (DMA) channel for this task (see 2.1.2). The DMA controllers used on the OCTOPUS - 8237 (ref 3) lend themselves ideally for this application. The signals provided by the DMAC during refresh, benefit both on- and off-board memory simultaneously (BREFRESH (L)).

The main logic board programmable read only memory consists of two 28 pin sockets configured to accept 2,4,8,16 Kbyte EPROMs. Link adjustments must be made for certain configurations. 24 Pin EPROMS (2 and 4 Kbyte) require correct positioning in the 28 pin socket (see App. 3b). The first socket (U23) is reserved for the system primitives. The second socket is used by LSI for in house diagnostics but otherwise can be used for general applications. 32Kbyte EPROM's can be accomodated but requires an FPLA exchange.

Issue 5 and later revisions of the OCTOPUS main logic board provide a linkable feature to permit the second socket (U24) to be used for bytewide static RAM.

Address decoding for both DRAM and EPROM is provided by a programmable logic array as described earlier. This device is also capable of inserting one wait state into any 1Kbyte range in the 1Mbyte address space. On or off-board slow memory can benefit from this feature. No wait states are currently introduced during any 8088 or Z80 memory cycles.

MEMORY ADDRESS MAP



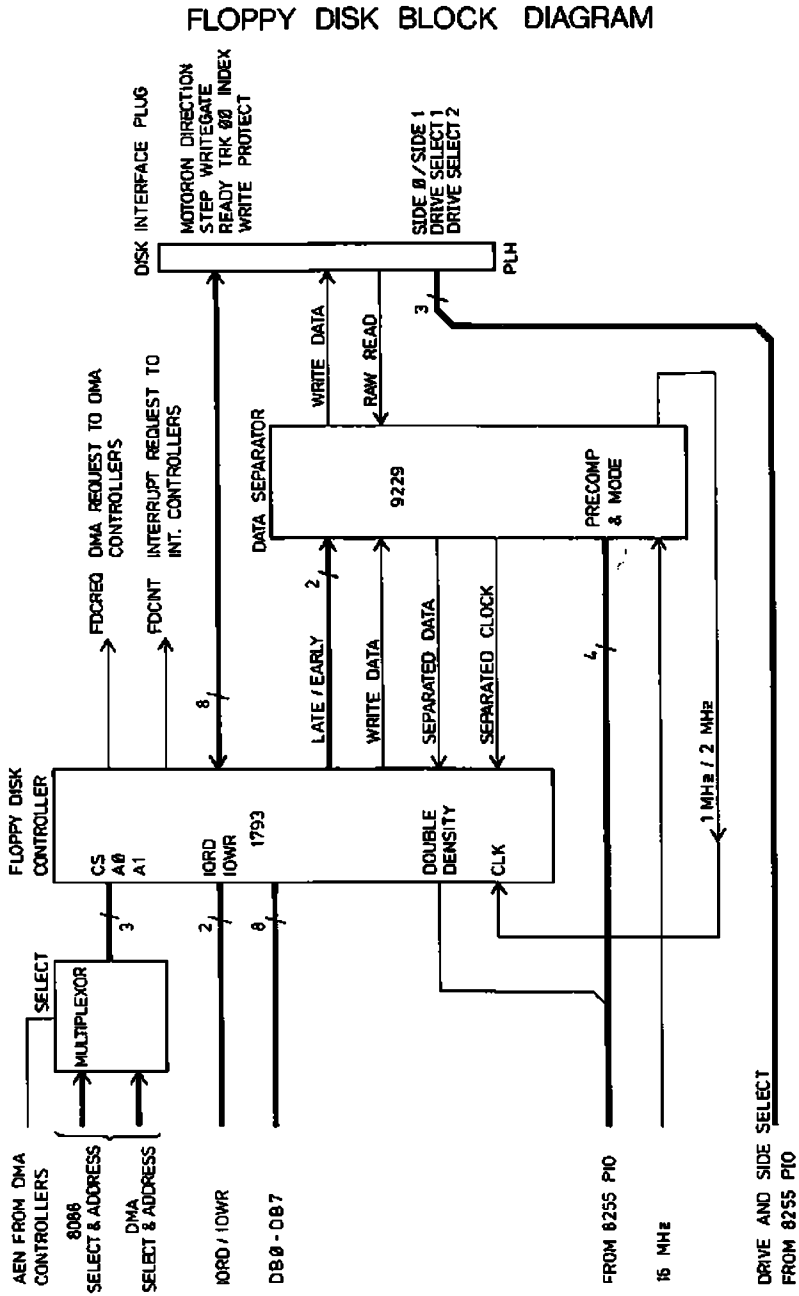

FLOPPY DISK BLOCK DIAGRAM
DISK INTERFACE PLUG
MOTORON DIRECTION
STEP WRITEGATE
READY TRK 00 INDEX
WRITE PROTECT
SIDE 0/SIDE 1
DRIVE SELECT 1
DRIVE SELECT 2
PLH
3
WRITE DATA
RAW READ
DATA SEPARATOR
9229
PRECOMP
& MODE
FDCREQ DMA REQUEST TO DMA CONTROLLERS
FDCINT INTERRUPT REQUEST TO INT. CONTROLLERS
8
2
LATE / EARLY
WRITE DATA
SEPARATED DATA
SEPARATED CLOCK
4
FLOPPY DISK CONTROLLER
CS
A0
A1
IORD
IOWR
1793
DOUBLE DENSITY
CLK
1 MHz / 2 MHz
SELECT
MULTIPLEXOR
3
2
8
AEN FROM DMA CONTROLLERS
8086 SELECT & ADDRESS
DMA SELECT & ADDRESS
IORD / IOWR
DB0 - DB7
FROM 8255 PIO
15 MHz
DRIVE AND SIDE SELECT FROM 8255 PIO

fig. 2.1.6

2.1.6 THE FLOPPY DISC INTERFACE

Synopsis - The floppy interface provides for the control of one or two 5.25 inch drives.

Description - The interface, fig 2.1.6, consists of the WD or SMC 1793 device (ref 9) and the SMC 9229 (ref 10).

The 1793 performs all the functions necessary to read or write data to a 5.25 inch single or double density drive. The device generates or monitors the following standard interface signals:-

- motor on
- direction
- step
- write gate
- ready
- trk00
- index
- write protect
- raw data

The 8255A controls the following lines:-

- side select
- drive select 0
- drive select 1

The task of data conditioning is handled by the 9229 device. When writing data this device handles write precompensation by repositioning the data according to the EARLY and LATE control lines from the 1793. The amount of precompensation applied is dictated by the P0 and P1 lines, which are driven by two lines from an 8255A (ref 5) programmable peripheral interface device.

When reading data the 9229 separates and regenerates the clock and data from the composite data stream, presenting optimally positioned pulses for the 1793.

Data is normally exchanged between the 1793 and memory under DMA control. A 74LS157 multiplexer serves to control the chip select and address inputs of the 1793 during a DMA cycle, which is stretched to accomodate the slow speed of the 1793 as explained in section 2.1.2.

The 16MHz clock supplied to the 9229 is internally divided to provide a 1MHz clock to the 1793. This is increased to a 2MHz rate during head slewing for quad drives to satisfy the 3 mS period between step pulses by manipulating the MINI input to the 9229. The 9229 also accepts a single or double density input signal, DENS. Both MINI and DENS are derived from two spare 74LS374 buffer lines in the video circuitry.

Programming Details - These are not provided as low level access to the resource is unnecessay. Access to the floppy drive(s) should be made at the very lowest level by calling the disc primitives in firmware (Chapter 3 - calls 5-7). Normally access to these resources would be at a higher level still - by making sequential or random access to the resource through the operating system.

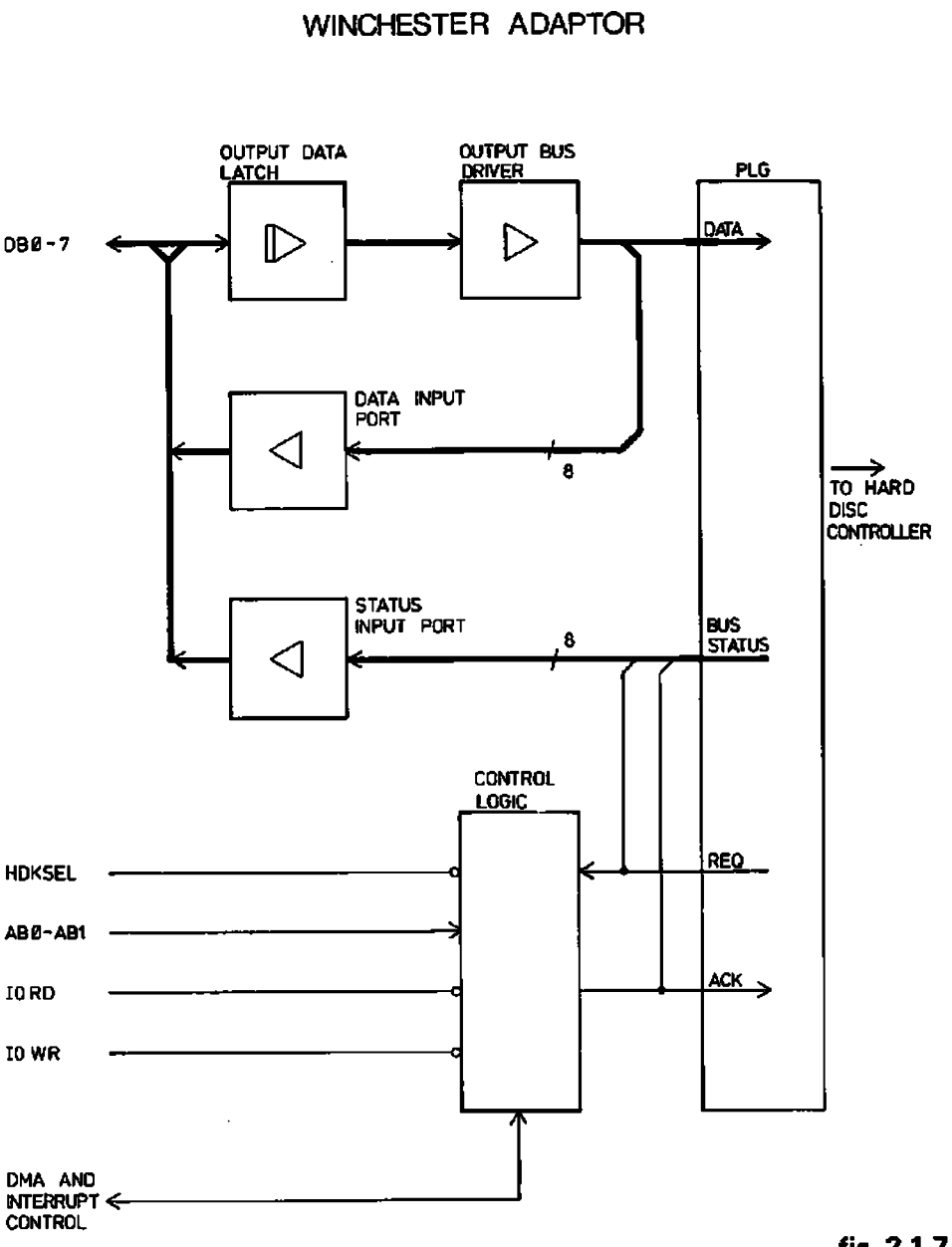

WINCHESTER ADAPTOR
OUTPUT DATA LATCH
OUTPUT BUS DRIVER
PLG
DB0-7
DATA
DATA INPUT PORT
8
TO HARD DISC CONTROLLER
STATUS INPUT PORT
8
BUS STATUS
CONTROL LOGIC
HDKSEL
AB0-AB1
IO RD
IO WR
REQ
ACK
DMA AND INTERRUPT CONTROL

fig. 2.1.7

2.1.7 THE WINCHESTER DISC INTERFACE

Synopsis - Provides a reduced Small Computers Standard Interface (SCSI) interface for the control of one winchester controller only. The controller may however control up to 2 winchester drives.

Description - The interface design is implemented entirely in bipolar logic (TTL). It consists of two main sections - data transfer and control.

With reference to fig. 2.1.7 the data transfer logic is made up of an ouput latch and SCSI bus driver for sending data to the controller and input buffer for receiving data from the controller.

The control logic consists of one status port for monitoring the SCSI bus status during data flow, a controller SELECT flip flop, the SCSI standard handshaking logic used during both programmed I/O and DMA transfers and a 'soft' controller RESET facility. The remainder of the logic provides the DMA handshake pair and an end of transfer interrupt.

Programming Details - These are not provided as low level access to the resource is unnecessary. Access to the winchester drive(s) should be made at the very lowest level by calling the disc primitives provided in firmware (Chapter 3 - calls 8-10). Normally access to these resources would be at a higher level still - by making sequential or random access to the resource through the operating system.

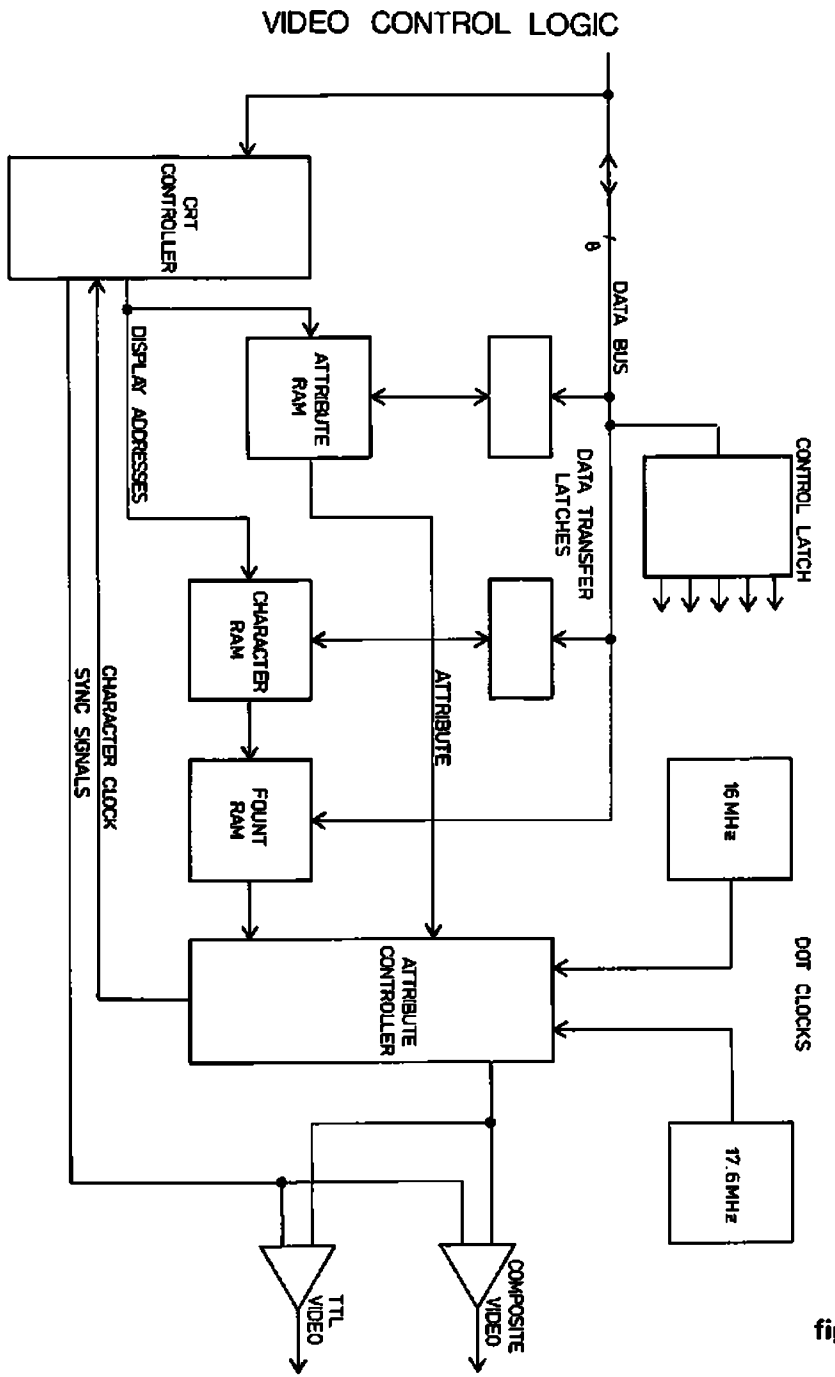

VIDEO CONTROL LOGIC
CRT CONTROLLER
8
DATA BUS
ATTRIBUTE RAM
DISPLAY ADDRESSES
DATA TRANSFER LATCHES
CONTROL LATCH
CHARACTER RAM
ATTRIBUTE
FOUNT RAM
16 MHz
SYNC SIGNALS
CHARACTER CLOCK
ATTRIBUTE CONTROLLER
DOT CLOCKS
17.6MHz
TTL VIDEO
COMPOSITE VIDEO

fig. 2.1.8

2.1.8 VIDEO CONTROL LOGIC

Synopsis - A highly versatile CRT controller provides many different display modes in both monochrome and colour. A user modifiable 'soft fount' allows tailoring of individual characters.

Description - Refer to fig 2.1.8. The heart of the video control logic is the Signetics Advanced CRT Controller type SCN2674 (Ref 12). This device provides the display refresh addressing information during the active display portion and allows CPU access to the display memory during blanked portions, controlling this action via status flags.

It is possible to modify not only the number of displayed lines per character row but also the displayed width in dots. The total number of characters per row and rows per screen are also modifiable as in conventional CRT controllers. This versatility gives rise to a number of display modes only limited by the user's imagination. The firmware (later versions) however, currently has code to initialise and maintain the video logic in the following popular operating modes (further modes are planned as applications requirements stabilise).

a) 25 rows of 80 columns 7x9 in 9x13
* b) 25 rows of 40 columns 7x9 in 9x11 double width
c) 29 rows of 132 columns 5x7 in 6x11
d) 27 rows of 80 columns 7x9 in 9x12
e) 27 rows of 40 columns 7x9 in 9x12 double width

All the modes have a stationary half intensity reverse video status line at the bottom of the screen with the rest of the screen scrollable at the rate selected. (This is a simple example of the split screen capability of the controller.)

* This mode should only be used with a UHF modulator.

In order to provide the range of display formats and still maintain the synchronism of the monitors it is necessary to have a choice of master dot (or pixel) frequencies. The appropriate dot clock is selected via the control register bit 0. Bit 0 cleared selects 17.6 MHz and gives up to a total of 800 pixels across the screen, 792 of these are used in the 29x132 mode. Bit 0 set selects 16 MHz and is used in normal width for 25x80 mode and in double width (each pixel twice the horizontal size) for 25x40 mode giving 720 pixels and 360 pixels respectively.

The selected clock drives the Attribute Controller, Signetics SCB2675 (Ref 13), where it clocks the internal video shift registers. The clock frequency is divided selectively to provide a character clock signal for the CRT controller (see App. 4a) which forms the main timing control signal. A fuller description of the CRT and Attribute controllers can be found in the references.

The CRT controller generates video refresh addresses that are passed to two pairs of 2Kbyte static RAMs, one pair for the displayed character's fount addresses and one pair for their attributes giving a maximum of 4096 displayed characters. The 8 bit character address bits combine with 4 further video line number bits to form a 12 bit address into the 4Kbyte fount memory. This is mapped as an array of 256 groups of 16 bytes, one group per character, so character 0 line 0 (top line) is at address 000h, character 1 line 0 is at address 010h etc. For hardware reasons the bits are displayed starting with the leftmost displayed bit as bit 0, then bit 1 and so on to bit 7 which is duplicated for greater than 8 bit wide characters. This 'reversal' permits 'joined up writing' and is used extensively in pseudo graphics displays. Note that the bit reversal is performed automatically in the font load utilities and hence is normally transparent to the user.

The 8 attribute bits from the attribute RAM pass directly to the attribute controller where they are delayed and brought into synchronism with their corresponding fount dots, cursor and blanking signals. The effect of the attribute bits depends on whether the logic is in monochrome or colour mode (bit 8 of the system switches or 'ESC m' control), the details of the two modes are in App. 4b. Note the two general purpose bits in monochrome mode only are free to the user for field protection etc. and have no visual effect. The are, however, output on the 9-way TTL connector as GP1=GREEN (pin 4) and GP2=LUM (pin 6) if a user supplied monitor or other display device can make use of them.

The resultant stream of video dots drives both the 9-way TTL connector (App. 1c) and also provides a level shifted composite signal on the phono connector (App. 5d).

Programming Access to the display memories is via the data transfer latches, these are read/write ports, one for the character addresses and one for the attribute data. When writing the ready flag in the 2674 is polled until ready, access is then granted, whereupon the data may be written and the 2674 instructed to write it to the appropriate address. Conversely when reading a read command is given and when the flag is ready the data may be read at the CPU's leisure. The user is however urged to access the display only by using the primitives provided in the firmware (Chapter 3 - calls 11-24), these allow both single character and block transfers of video data whilst still maintaining the integrity of the display pointers so that screen switching will not be violated. See Chapter 3 for full details.

KEYBOARD INTERFACE AND SPEAKER

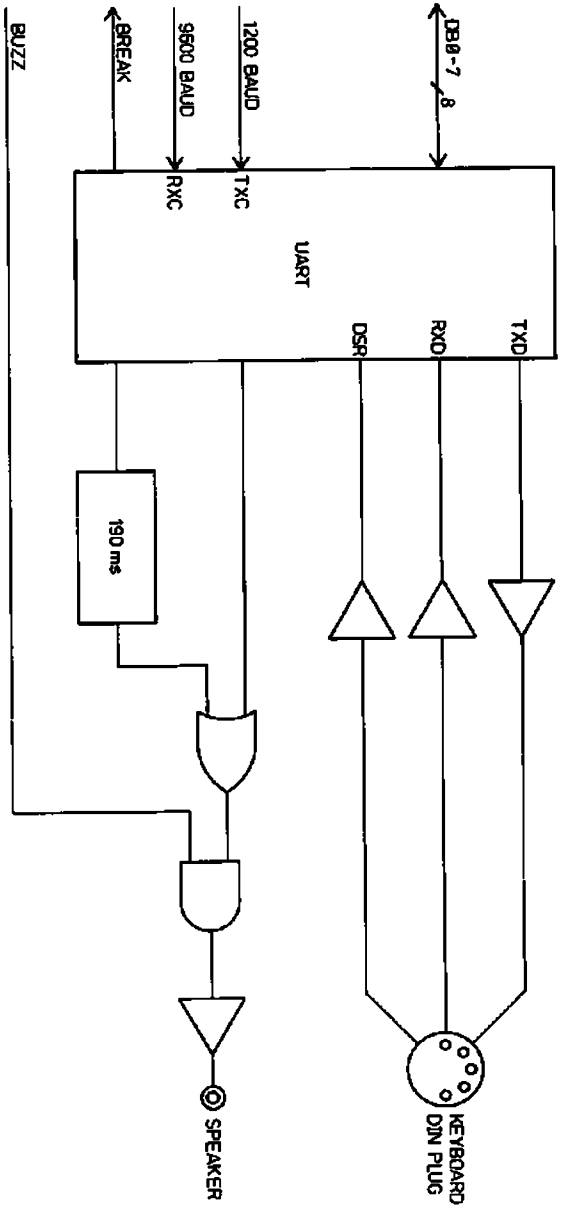


fig. 2.1.9

2.1.9 THE KEYBOARD INTERFACE AND SPEAKER

KEYBOARD - Synopsis - The serial keyboard communicates bi-directioinally with the main logic board via up/down codes.

Description - The keyboard physically connects to the main logic board via a 5-way circular DIN connector. The DIN pinouts are defined in appendix 1d.

The keyboard generates up/down codes, each code representing a particular key position within the keyboard matrix. These codes are serialised by the keyboard, and sent to the main logic board at 9600 bits/s. Serial data is also sent to the keyboard, at 1200 bits/s, to control the LEDs in the Shift and Caps Lock keys. The serial bit streams are interfaced to the main logic board via an 8251A UART. Data is transferred at TTL levels, and TTL buffering is provided on the main board.

Programming Details Due to the complexities involved in accessing the keyboard directly, access should only be attempted through the system firmware routine (see Chapter 3 - calls 25-27).

SPEAKER - Synopsis - Generates either continuous tone or beep.

Description - The Octopus speaker is driven from the main logic board via an open-collector TTL driver, and also connects to the bus line BAUDIO, to allow option boards access to audio output if desired. When driven by the main logic board the audio pitch is governed by the output of an 8253 device, which is programmed to output a square wave at 1200Hz. The note duration is governed either by triggering a 100ms monostable, or by a spare control output on the keyboard UART to give variable periods.

Programming Details Due to the complexities involved in accessing the speakerd directly, access should only be attempted through the system firmware routine (see Chapter 3 - calls 32-33).

CALENDAR CLOCK FUNCTIONAL DIAGRAM

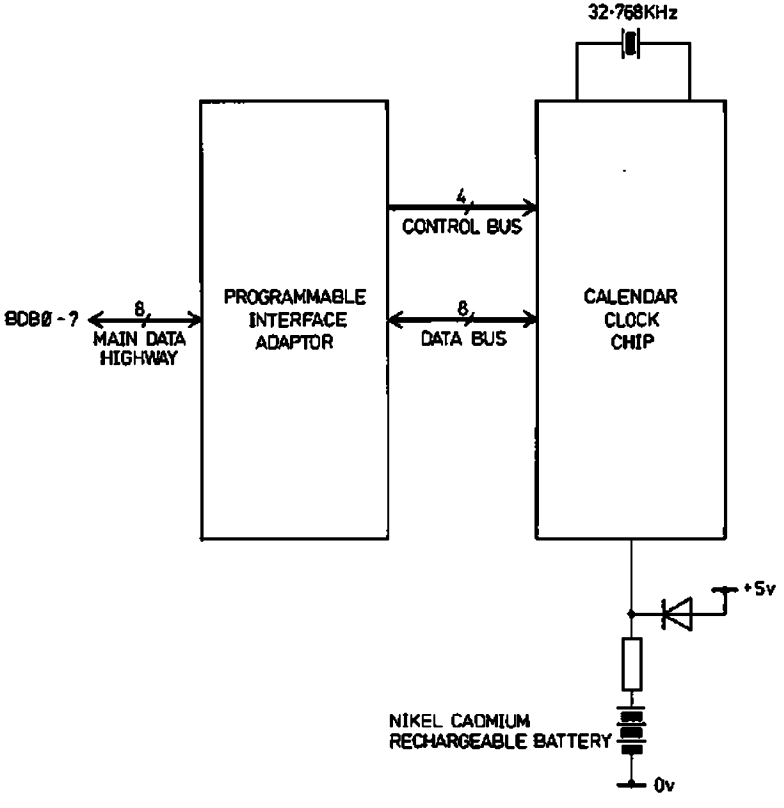


fig. 2.1.10

2.1.10 THE CALENDAR CLOCK

Synopsis - A continuously running subsystem provided as an operating system resource for maintaining time and date. System ticks are also provided by this device for time-slicing in a multitasking environment.

Description - The calendar clock device, fig. 2.1.10, is a very low power CMOS device - MC146818 (Ref 11) - capable of being sustained in a power down condition by an on-board Nickel Cadmium cell. This cell is automatically recharged during power up periods making replacement of the cell unnecessary for the lifetime of the computer. On being programmed with the year, month, day, hour, and second, (CP/M 'TIME ...') the device will continue to maintain this to crystal accuracy - 20 ppm/degC. The other duty this device performs is sourcing system 'ticks' required to reschedule tasks running on a time-slicing basis.

The MC146818 also provides 50 bytes of non volatile storage.

Due to the lower speed A.C characteristics of the CMOS device, access to the clock chip is provided by a programmable interface adaptor (8255A - Ref 5). The 8255 and the clock chip firmware therefore jointly emulate the timing of a suitable slow host processor. In addition the firmware ensures access to the clock registers at the permitted times.

Programming Details Due to the complexities involved in accessing the clock registers and non-volatile ram directly, access should only be attempted through the system firmware routine (see Chapter 3 - calls 28-29).

PARALLEL INTERFACE FUNCTIONAL DIAGRAM

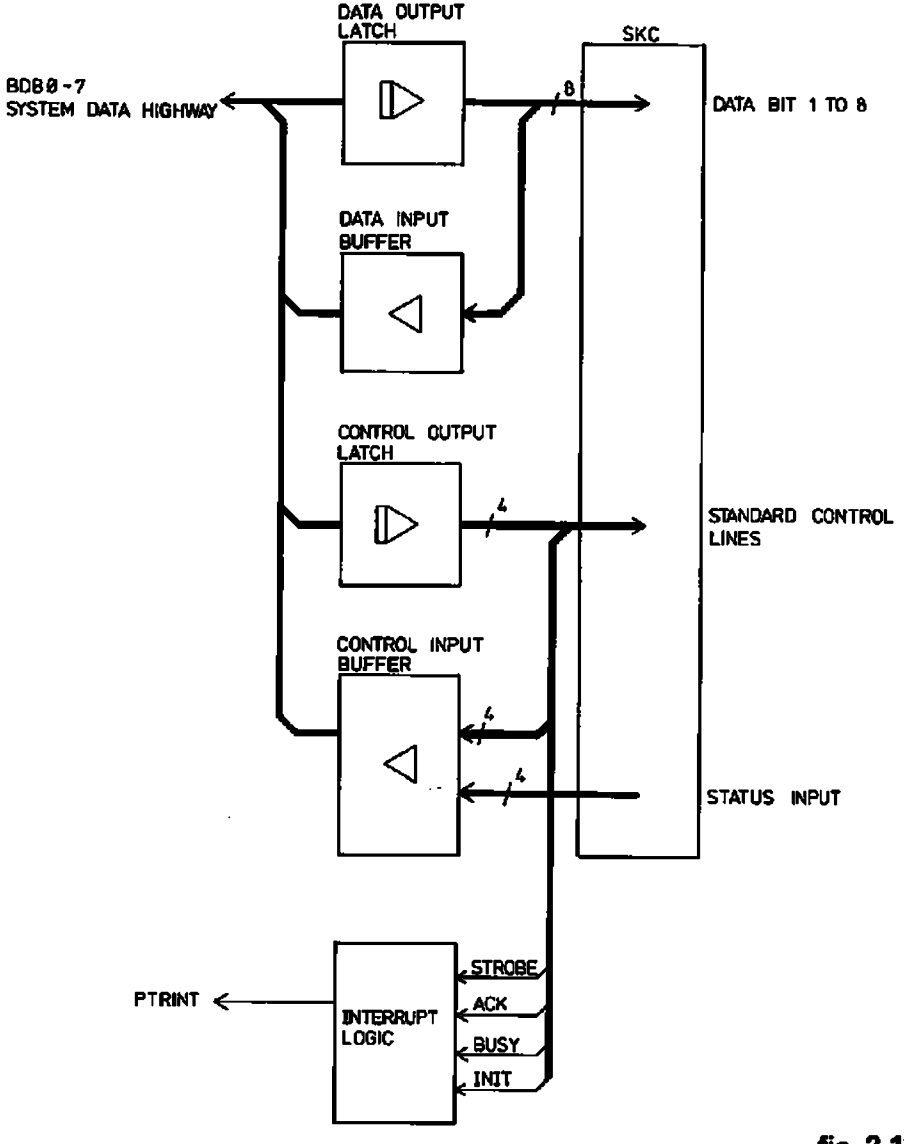


fig. 2.1.11

2.1.11 THE PARALLEL INTERFACE

Synopsis - An interface provided to transfer data to or from the OCTOPUS in an 8 bit parallel fashion at speeds of up to 160Kbytes/sec. System software automatically configures and drives this port as a standard 'CENTRONICS' compatible output port, primarily for use by parallel printers.

Description - The port, fig. 2.1.11, consists of an 8 bit data output latch, an 8 bit data input buffer, a four bit control output latch and an 8 bit control input buffer. In addition the port is provided with a hardware interrupt facility (see Chapter 2 - 2.1.3 Interrupt Controllers). The bidirectional data and control bits have two system control bits used for defining their direction.

The following describes the signals presented to the 25 way connector at the rear of the OCTOPUS labeled 'PARALLEL':-

- The Data Group

MNEMONIC: DATA BIT 1(H) - DATA BIT 8(H)

These eight data output lines are used for transferring 8 bit data (normally ASCII) to a parallel printer port. If so configured they can also act as eight data input lines. Only half duplex transmission can take place using this port in a bi-directional mode.

- The Control Group (Centronics Driver)

MNEMONICS:			
	STROBE(L)	INIT(L)	AUTO FD XT(L)
	ACK(L)	SLCTIN(L)	ERROR(L)
	BUSY(H)	SLCTOUT(H)	PE(H)

Data Strobe - STROBE(L)

This signal is used to inform recipient that data is available and stable on the Data Lines DATA BIT 1(H) to DATA BIT 8(H). The leading (negative going) edge is the reference.

Data Acknowlege - ACK(L)

This is a 'handshake' from the recipient that data has been received and this current cycle can now be concluded.

Printer Busy - BUSY(H)

This informs the OCTOPUS that the printer is currently indisposed and cannot receive data. It may also indicate that no printer is connected as this signal defaults to BUSY in this event.

Initialize Printer - INIT(L) also known as INPUT PRIME(L)

This signal is sent out once during the system power up sequence to reset the printer. Printers generally will read their switch settings on power up or on receipt of this signal.

Select Printer - SLCTIN(L)

This signal is used to bring a printer 'on-line'. It has a similar function to the 'READY' or 'ON-LINE' buttons found on printers. It is not normally used. The SLCTIN(L) input to the printer is normally grounded.

Printer Selected - SLCTOUT(H)

This signal is used to detect whether a printer is 'ON-LINE' i.e. whether the 'ON-LINE' button has been depressed and the SLCTIN(H) line at the printer has been asserted.

Auto Line Feed - AUTO FD XT(L)

This signal informs the printer when asserted to insert the Line Feed character (0Ah) after receiving a Carriage Return character (0Dh).

Printer Error Condition - ERROR(L)

Whenever a printer fault condition arises the printer will assert this line. Other fault lines may be asserted simutaneously.

Paper Empty - PE(H)

Some printers can detect when there is no paper left. This line is asserted to inform the OCTOPUS as well. The ERROR(L) line will also be asserted with this condition.

Note: On Issue 4 boards and earlier, input data is 'mirrored' - DATA BIT 1 becomes processor bit 7, DATA BIT 2 becomes bit 6 etc. An LSI designed device, Part No. 900492, is available to remedy this situation if the input mode is used - contact your sales representative for ordering details.

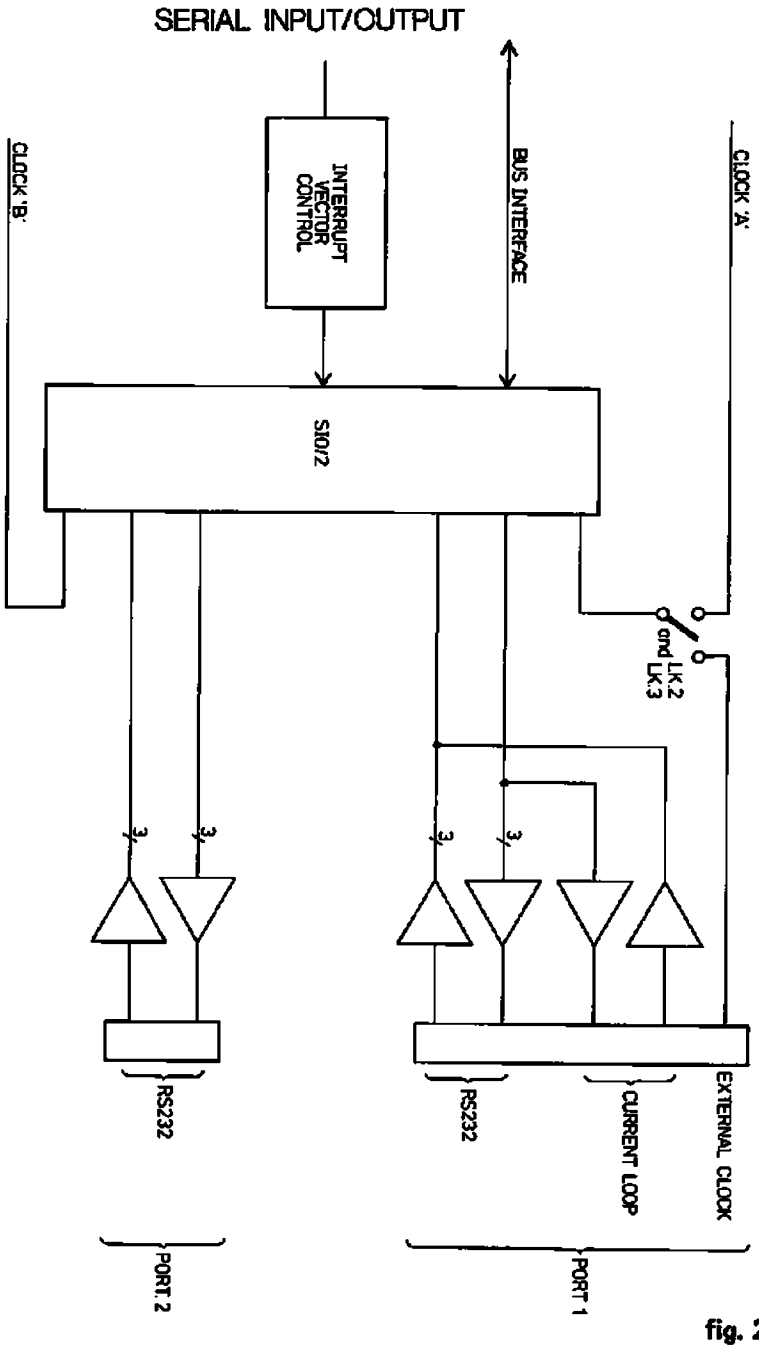

SERIAL INPUT/OUTPUT
INTERRUPT VECTOR CONTROL
BUS INTERFACE
CLOCK 'A'
CLOCK 'B'
SIO/2
and LK.2 LK.3
3
RS232
RS232
CURRENT LOOP
EXTERNAL CLOCK
PORT 2
PORT 1

fig. 2.1.12

2.1.12 THE SERIAL INTERFACE

Synopsis - Provides RS232C asynchronous communication at two ports - PORT1 and PORT2 and provides RS232C synchronous and optically isolated 20mA current loop communication at one port - PORT1. Both ports provide full modem control signals.

Description - The interface design, fig. 2.1.12, is based around the Zilog SIO/2 (ref 8), currently one of the most advanced communications devices available. This device enables a variety of asynchronous and synchronous protocols to be used (Bisync, SDLC, HDLC etc) at up to asynchronous baud rates of 19.2Kbaud, and synchronous baudrates of 880Kbaud. At high baud rates however the RS232C interface bandwidth/distance limitation would limit transmission to only tens of feet. The primary function of the SIO is to serialize data from the processor prior to transmission and deserialize received data into a processor readable form.

The remainder of the design can be divided into three separate sections - the processor interface, the baud rate generators and the line interface.

The SIO, a Zilog design, cannot be directly interfaced to the 8088. Additional logic is provided to convert the 8088 I/O signals to a Z80 set. This not only involves the generation of signals for carrying out simple I/O functions but also for emulating the interrupt acknowledge cycle through the generation of VECSEL0.

Baud rate clocks required by the SIO are provided by the Intel Programmable Interval Timer 8253-5 (Ref 6). Each channel has one independantly selectable baud rate clock. However the receive and transmit clocks for each channel are the same and not independantly selectable. One exception to this is when external clocks are provided for PORT1 and the links (LK2 and LK3) appropriately connected.

To acheive the RS232C electrical specification, the line interface uses the industry standard line drivers and receivers - 75188 and 75189 respectively (Ref 14). The receivers are slugged to give a high frequency roll off with a 3dB point at around 1.5MHz to ensure adequate rejection of R.F picked up by long peripheral leads.

PORT1 has in addition an isolated 20mA current loop interface. This is provided for longer distance transmission than is safely acheivable using RS232. Isolation is achieved by the opto-coupler. Low optocoupler frequency response limits transmission speeds to around 9600 Baud. Higher speeds can be achieved but on a more selective basis.

Programming Details Baud rate settings and asynchronous serial port data structures (frame length, stop bits, parity etc) are best altered using PARMGEN and LOADPARM (see the User Manual). These parameters should, wherever possible be application program independant and hence the need to control these facilities directly should be very limited. However for these limited instance the following details are now provided.

1. To setup the baudrate (bitrate if synchronous) generator:-

Mnemonic	Register Description	Port No.	Frequency Generated
BAUDSEL0_CT0	Counter 0	1 (Ch A)	2.457/(CCCCh + 1)
BAUDSEL0_CT1	Counter 1	2 (Ch B)	2.457/(CCCCh + 1)

where CCCCh is the 16 bit count value <> 0000h else 2exp16.

Example :-

Sending a count of 1 to BAUDSEL0_CT0 sets SIO Ch A receive and transmit clocks to 1.228MHz.

2. To setup the serial ports :-

Mnemonic	Register Description
DARTSEL_ADATA	Channel A I/O Data Port
DARTSEL_ACNTL or DARTSEL_ASTAT	Channel A Control Port
DARTSEL_BDATA	Channel B I/O Data Port
DARTSEL_BCNTL or DARTSEL_BSTAT	Channel B Control Port

(Please ignore the mnemonic reference to DART this dates back to an earlier specification - the device is truly an SIO !).

Armed with these references the reader is directed at this stage to the full programming description of the SIO (Ref 8).

BUS INTERFACE

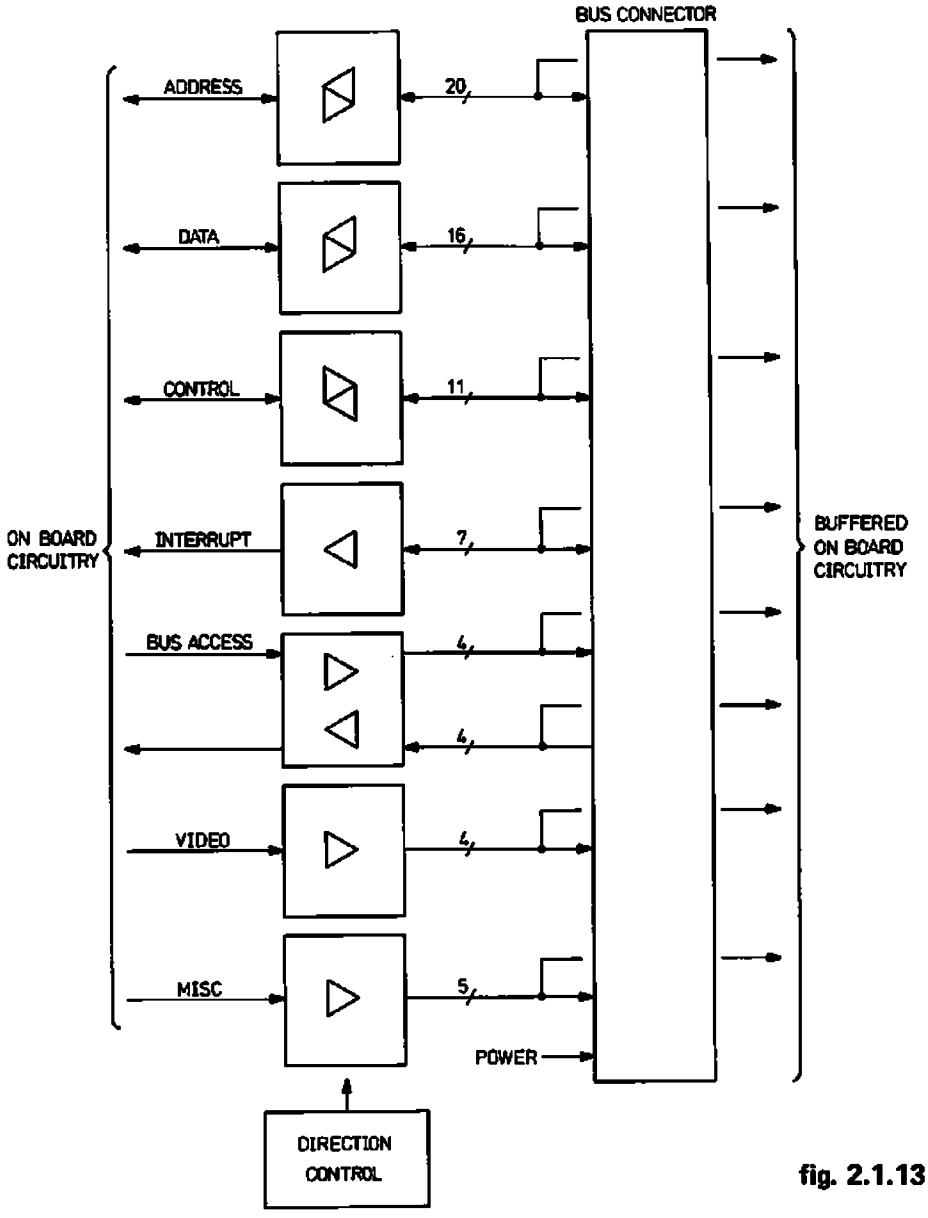


fig. 2.1.13

2.1.13 THE BUS INTERFACE

Synopsis - Maintains overall system dynamic A.C. characteristics (operating speed) and enhances the D.C. characteristics of the more commonly used system signals.

Description - Both the data and address buses are controlled by a set of transceivers. Data clearly requires dual direction for the read and write processes. Not so obviously however is the need for the address bus to be dual direction. With reference to the positioning of the address transceivers in the functional diagram Fig. 2.1.13, if an option board bus master wishes to communicate with on-board memory, the processors relinquish the bus allowing both address and data busses to be controlled by the option board. In such a case the address is sourced from the option bus side of the transceivers and therefore they need to become receivers to the option and drivers to the main system address busses. Logic within this section is also provided to reverse the sense of direction of the data transceivers. This depends on whether off- or on-board control of the transceiver prevails and whether the current cycle is read or write.

The remaining hardware within this section deals with the conditioning of signals for correct presentation to the main system components - processors, DMA controllers, and priority interrupt controller.

2.1.14 SWITCHES AND LINKS

Synopsis - Switches and links are provided on the main logic board for configuration purposes.

Description - Twelve individual switches are provided on the main logic board in the form of two DIL switch packs, one bank of 8 and a second bank of 4. They are positioned such that they can be easily seen and adjusted once the main processor lid is removed. The bank of eight switches are either used or reserved for system configuration purposes (see Appendix 3a). The smaller bank of 4 switches are used for language selection purposes (see Appendix 3b). LSI reserves the right to use unallocated switches for system purposes without prior publication of the fact. The switch setting is read into memory on power up of the unit by the initialization firmware. Access to the power up switch status is provided by the system firmware (see Chapter 3 - call 30). If this call is used, subsequent changes to the switch setting whilst the system remains powered will have no effect. Only a power down or system reset condition will update the new state .

A number of links are provided on the board for various functions. These would normally be factory set. Alteration of the 'default' links can only be done by removal of the main logic board and should NOT be attempted by unauthorized personnel. Link alterations generally involve cutting the 'default' link and soldering the alternative link to the board (see Appendix 3d & 3e).

2.1.15 THE OPTION BUS

Synopsis - One 96 way connector on the main logic board provides the power and signals necessary for the main board processors to control four horizontally stacked option boards and provides additional signals enabling any of these option boards to control the main board memory.

Description - The OCTOPUS option bus is a proprietary bus based on a novel interconnect approach using state-of-the-art connector technology. The DIN 41612 96 Way connector has been adopted as the company option bus standard connector due to the two part and highly reliable design of the connector. The bus provides both signals and power. The main features of the bus are :-

- o 8 and 16 bit data bus.
- o Multi-master (8/16bit) access up to 4 bus masters.
- o Upto 7 vectored interrupt levels.
- o Access to system loudspeaker.
- o Main processor and baud rate clocks provided.

The option bus is a proprietary small systems synchronous bus. The prime objective during its conception was to fulfill the following requirements :-

- o Low main board overhead.
- o Reliability.
- o Simplicity.
- o Support 8 and 16 bit.
- o Support multiprocessors.
- o Fast.
- o Simple to interface.

By highlighting the results of these objectives a good appreciation of the bus can be acheived - see over.

Low main board overhead - Conventional single board computers have assigned valuable board space to provide option facilities. The Octopus option bus system has only ONE connector on the main logic board but nevertheless allows option expansion upto four further option boards. This is acheived by a novel stacking arrangement which in conjunction with the interlocking rear panels make an intrinsically stable and rigid design.

Reliability - Past experience has repeatedly demonstrated the edge connector to be notoriously unreliable in all but the most expensive of racking arrangements. To maintain reliability but to keep costs down, the well proven DIN 41612 Type C 96 Way connector was adopted. The option boards use the 'press-fit' (solderless) versions of these connectors enabling one connector assembly to provide both a plug, to connect to the board below and a socket, to connect to the board above. In addition should any option board connector contact be damaged, 'press-fit' connector technology provides the means to replace that contact. In contrast a board with a damaged edge connector 'finger' contact would be rendered unusable.

Simplicity - To interface to the bus is no more complex than interfacing to the 8088 or Z80 directly. Signals to assist in simplifying option board design are provided on the bus (see below). Thought has been given not only to the ease of design but also to the ease of implementation . A standard 'quick-connect' prototyping board has been developed to enable a design to be rapidly manufactured using the latest in prototyping techniques. The board is fully compatible with the bus electrical and mechanical standards and can be accomodated by the Octopus as if it were a production option board. These reuseable boards can be bought from LSI or Dage Eurosem, Aylesbury. LSI PN 900500.

8 and 16 Bit Support - The bus was designed from the outset to support both 8 and 16 bit data transfers. All accesses by the main board are 8 bit, however option board memory has both 16 and 8 bit modes of operation. This will enable option board memory to become a shared memory resource for the main board processors and future 16 bit processors operating from option boards.

Multiprocessor Capability - The bus has three prioritized bus access channels. These channels allow option boards to become masters of the bus. Once granted mastership of the bus an option board can control the entire memory resource of the system. Only byte access can be made to main board memory, however 16 bit access can be made to option board memory. This enables 16 bit bus masters to transfer data at twice the main processor data rate.

Fast - No wait states are introduced during memory access across the option bus by the main board processors.

The following describes the option bus signals.

The bus is best described by grouping the signals into the following catagories:-

- o Address group.
- o Data group.
- o Control group.
- o Interrupt group.
- o Bus access group.
- o Miscellaneous group.
- o Power group.

A description of each group in turn will now be given. Please refer to Appendix 2a for the connector pin assignments and Appendix 5 for timing diagrams.

The Address Group - BAB0(H) -BAB19(H)

These signals are all active HIGH signals and represent the 1 MByte addressable memory space which can be accessed directly by the main board 8088 processor or through a bank switching mechanism by the main board Z80B. Using this address bus the 64KByte addressable I/O space may be accessed by the on-board 8088 but NOT however by the on-board Z80B.

The Data Group - BAD0(H) - BAD15(H)

The main logic board uses the lower 8 bits ONLY (BAD0(H) to BAD7(H)) of the active high group of signals. The high byte is used during 16 bit data transfer across the option bus (see BHE(L) later).

The Control Group - BMWR(L) BIOWR(L) BM(H)/IO(L) BALE(H)
BWAIT(L) BMRD(L) BIORD(L) BSMC(L)
BCLK(H) BHE(L) BSYSIO(L)

These signals are used to control the transfer of data between memory, I/O devices and processors (on or off board) in a precisely defined manner. The read and write strobe signals are active low to provide high quiescent state noise immunity as these are particularly sensitive signals. The signals are decribed in turn below.

Bus Memory Write - BMWR(L)

Asserted during memory write access cycles. Normally asserted by the 8088 or Z80 main board processor. A bus master may take control of this line but must ensure that the bus specifications are met.

Bus Memory Read - BMRD(L)

Asserted during memory read access cycles. Normally asserted by the 8088 or Z80 main board processor. A bus master may take control of this line but must ensure that the bus specifications are met.

Bus I/O Write - BIOWR(L)

Asserted during I/O write access cycles. Normally asserted by the 8088 main board processor. Main board Z80 does not have control of this signal. A bus master may not take control of this line. I/O duties should be passed back to the 8088.

Bus I/O Read - BIORD(L)

Asserted during I/O read access cycles. Normally asserted by the 8088 main board processor. Main board Z80 does not have control of this signal. A bus master may not take control of this line. I/O duties should be passed back to the 8088.

Bus Memory or I/O - BM(H)/IO(L)

This signal is high during memory cycles and low during I/O cycles. It is useful where advanced cycle recognition is desired as this signal is stable before the trailing edge of BALE(H). It can be used therefore in conjunction with BALE(H) and an address decode to assert the BWAIT(L) line for slow memory or I/O.

Bus Start Memory Cycle - BSMC(L)

This signal is controlled excusively by the main logic board to provide memory devices with an advanced trigger to start the time consuming address multiplexing cycle required by dynamic RAM devices prior to the memory read or write strobes. Option board bus masters requiring access to memory must only assert BMRD(L) or BMWR(L) as described above. BSMC(L) will be generated automatically by the main logic board.

Bus Address Latch Enable - BALE(H)

This is generated by the main logic board and is only present at the beginning of 8088 bus access cycles. It must not be used where main board Z80 bus access is required. During Z80 cycles it will remain inactive.

Bus Clock - BCLK(L)

This clock is the 8 MHz processor clock. In conjunction with BALE(H) cycle synchronization can take place. Dividing the clock by 2 the peripheral clock PCLK can be generated.

Bus High Enable - BHE(L)

This signal is used only when sixteen bit data transfers are desired. Bus High Enable is asserted low by the calling device (a bus master) to instruct a called device that a 16 bit data transfer cycle is expected. It is used currently by option board memory to switch memory into 16 bit mode as follows:

BHE(H)	BABO(H)	ACCESS DESCRIPTION
0	0	- 8 bit even byte transfer
0	1	- 8 bit odd byte transfer
1	0	- 16 bit even byte transfer
1	1	- reserved state

It can be seen that true 16 bit transfers must occur on even byte boundaries. Some devices (8086 family) make odd byte 16bit transfers possible by a double bus byte transfer. Without this automatic double bus access mechanism 16 bit odd byte access will result in errors.

This 16 bit operation of option board memory and 16 bit data transfer provision on the bus enables true 16 bit processors to share option board memory with the 8 bit main board processors for such tasks as communication, synchronization, data transfer, code upload etc.

Bus Wait - BWAIT(L)

This line is routed directly to the 8088 clock driver. All on board and off-board devices requiring wait states in either memory or I/O cycles must pull this line sufficiently early in the I/O cycle for it to be detected. It is the responsibility of the asserting device to ensure that the BWAIT(L) line is released as soon as possible as system performance can easily be severely reduced.

> Note: I/O addresses 300 to 3FF have 2 wait states automatically injected into their I/O read and write cycles.

Bus System I/O - BSYSIO(L)

This active low signal is asserted when I/O access is being made to a device address in the range 0 - 3FF (i.e. in the first 4 pages of the I/O map). It means effectively that the current cycle is an I/O cycle and BAB10(H) to BAB15(H) are all zero. Therefore to fully decode an I/O address only BSYSIO(L) and normally 7 or 8 address bits (BAB2(H) to BAB9(H)) are sufficient. For examples of correct decoding techniques using this signal refer to Application Notes section.

> Note: The address block 380h to 3FFh has been made available to third party manufacturers of special interfaces for the Octopus and can therfore make use of BSYSIO(L). LSI guarantees not to use these address in order to avoid option board address conflicts. If addresses other than those recommended within the first four pages are use, system malfunction could arise.

The Interrupt Group - BINT0(L) - reserved for RS422
BINT1(L) - reserved for RS232
BINT2(L) - reserved for network
BINT4(L) - reserved for streamer
BINT6(L) - reserved for 127 Board
BINT9(L)
BINT11(L)

These active low signals are normally masked by the system default initializing software. Should an option board require the use of an interrupt level the associated application software must be responsible for providing the correct vectoring information and enabling the appropriate level. Check with LSI for current system assignments. The on board Priority Interrupt Controller is intialized to provide the following access to the interrupt vector table:-

Interrupt Level	PIC Level	Interrupt Type Number (Hex)
BINT0	M0	60
BINT1	M1	61
BINT2	M2	62
BINT4	M4	64
BINT6	M6	66
BINT9	S1	69
BINT11	S3	6B

M = Master PIC , S = Slave PIC

The Bus Access Group

BREQ0(L) BACK0(L) - Reservedfor Streamer.
BREQ3(L) BACK3(L) - Reserved for RS422.
BREQ5(L) BACK5(L)
BREQ6(L) BACK6(L)

This group of four pairs of signals are provided to enable option boards to claim the option and main board busses to allow control of data flow between itself and memory resources. This can be viewed as Direct Memory Access type

transfers where non intelligent option board bus masters take control of the bus on a cycle stealing basis or as dual porting memory where option board processors take control of the bus. Non intelligent option board bus master would generally have an 8237 DMA controller and bank register file (74LS670) to provide the necessary control signals and addressing information (see Application Notes for example interface). Access to the bus is made by asserting the request line. The main board will respond by asserting the acknowldege line once :-

a) The current bus cycle has completed and the current 8088 instruction is not prefixed by a LOCK instruction or

b) All higher DMA requests have been serviced.

Once the acknowlege has been received the option board has control of the system memory space.

Video Group - VSYNC(L) DCLK(L) HSYNC(L) CCLK(L)

These signals are generated by the CRT control interface on the main logic board and are provided for synchronization purposes for option boards wishing to overlay, video information. (see Appendix 5 for precise timing).

Miscellaneous Group - BREFRESH(L) BAUDIO BRESET(L)
BPFAIL(L) BAUDCLK

Bus Refresh - BREFRESH(L)

This signal is asserted during a refresh cycle and is provided on the bus to refresh off-board dynamic memory. In this way ALL dynamic memory is refreshed during this short (DMA) read cycle. The main board also provides the rotating address (modulo 256) required by standard dynamic memories during this cycle.

Parity Fail - BPFAIL(L)

In order to detect option board memory malfunction, byte parity generation and checking is provided on LSI's standard dynamic memory option board. BPFAIL(L) is asserted as soon as such a malfunction is detected and an NMI (non maskable interrupt) is generated on the main logic board. The main logic can distinguish between on- and off-board parity failures making the task of isolating such faults very much easier.

Bus Audio - BAUDIO

This signal is linked to the system's loudspeaker and can be used by option boards for more sophisticated sound generation, speech etc. The speaker is a 1/2 watt 8 ohm impedance speaker.

Baud Clock - BAUDCK

This signal is a general purpose 2.458 MHz clock with a 50% duty cycle used primarily as a frequency source from which to generate baud rate clocks for such devices as UARTS, SIO's etc.

Bus Reset - BRESET(L)

This signal is a global system reset signal. It can be asserted in three ways:-

a) For a short period to initialize the system after the system power supply has achieved maximum power.

b) By depressing the system RESET button at the rear of the unit.

c) By simultaneously pressing the keyboard keys CTRL, SHIFT and DEL.

Power Group

The main logic board provides, for expansion, the following voltages:-

VOLTAGE	RIPPLE	CURRENT RATING
+5V	20mV PP	8A
+12V	20mV PP	0.8A
-12V	20mV PP	0.4A

2.2 POWER DISTRIBUTION

2.2.0 MAINS INLET PANEL

Mains enters the Octopus via a 3-pin IEC plug located on a panel to the left hand side of the rear of the machine looking from the front. A socket is mounted just above this to provide a mains outlet for the monitor. Warning - power is always present on this socket even when the Octopus is turned off, the maximum current taken from this socket must not exceed 2 Amps.

Located above these two connectors is the main on/off switch, when reaching over the top to turn the Octopus on 'blind', the power is on when the side of the rocker nearest the centre of the rear is depressed.

The mains inlet fuse is in a small drawer located under the mains inlet plug and for safety reasons cannot be removed with the mains plug in. It is a 2 Amp rating.

Mains voltage from the inlet panel is routed to the power supply unit (below) and also supplies the low noise fan mounted at the front end of the power supply assembly. Note that although the PSU may be selected to run at 110 Volts the fan may not and an alternative must be fitted for operation in the USA, contact LSI for details.

2.2.1 POWER SUPPLY UNIT

The power supply unit is standard regardless of Octopus model type and is capable of providing sufficient power for the largest possible Octopus configuration permitting retro-fittable units to be added.

It is a 130 Watt switching regulator using flyback techniques. Four power rails are provided :-

+5 Volts
+12 Volts (1)
+12 Volts (2)
-12 Volts

Full ratings and regulation figures are in App. 8.

!! IMPORTANT WARNING !!

This power supply operates with internal voltages exceeding 400 Volts with sufficient energy to kill !

Under NO circumstances is the Octopus to be used with the lid off by non-authorised personnel.

!! IMPORTANT WARNING !!

2.2.2 POWER DISTRIBUTION

The methods of low tension power distribution within the Octopus vary depending upon model type. Both methods, however, supply +5v, +12v, -12v and 0v to the main logic board and hence to the option bus.

The central earth star point is the 6 way spade connector mounted on the left hand side of the mains inlet panel. A secondary 6 way star point is created by the option bus link when option boards are added. It is most important that the option board installation procedures are followed in order to maintain the integrity of the system earthing.

You will note, when fitting the first option board, that the spade connector on the 4 height blanking panel is 4mm and not 3mm like the 1, 2 & 3 height panels. This is to ensure the earth from the power loom is correctly fitted to the bus star point.

The two methods mentioned above are as follows :-

1. Systems with (i) two floppy drives
   (ii) one floppy drive only
   (iii) no drives

   - Two connectors with +12v, +5v & 0v daisy chain to the drives where fitted, they are ty-wrapped back for safety in reduced drive configurations.

2. Systems with winchester drives

   - Two connectors with +12v, +5v & 0v daisy chain to the winchester drive and the host adaptor.
   - One connector off a separate +12v supply feeds +12v, +5v & 0v to the floppy drive.

A small 6 pin connector on the front right hand side of the main logic board provides a current limited +5v to drive the power on LED on the front panel. This loom also connects the 8 Ohm loudspeaker to its driver on the main logic board.

2.3 DISCS

The hierarchy of Octopus models contain

a) No drives
b) One floppy drive
c) Two Floppy drives
d) One floppy plus one Winchester drive

The add-on storage units, where fitted, contain

a) An additional winchester drive
b) A streaming tape drive
c) Both

The streaming tape drive is covered, if your system has one,. in the add-on units section 2.

2.3.1 FLOPPY DISC DRIVES

Sometimes known as diskette drives.

There are two capacities of half height floppy drive that may be configured within the Octopus computer.

a) Double sided, double density - 48 tracks per inch.

b) Double sided, quadruple density - 96 tracks per inch.

The Octopus uses the following formatting :-

```
 40 cylinders per drive type (a)
 80 cylinders per drive type (b)
  2 tracks (sides) per cylinder
  5 sectors per track
1024 bytes per sector
```

This gives a total formatted capacity of 409kbytes for (a)
and 818kbytes for (b)

The LSI format sector numbering is with no interleave and no skew running side 0 cylinder 0, side 1 cylinder 0,side 0 cylinder 1..etc.

2.3.2 WINCHESTER DISC DRIVES

Sometimes called hard disc drives.

The winchester disc is available in the following approximate formatted sizes (in practice they are about 7% larger)

a) 5 Mbytes
b) 10 Mbytes
c) 20 Mbytes
d) 40 Mbytes

Drives (a),(b) & (c) contain 1,2 & 4 platters respectively, drive (d) contains 4 platters at twice the track density.

The actual capacities are obtained thus :-

320 cylinders per drive for (a),(b) & (c)
640 cylinders per drive for (d)
2 heads (tracks per cylinder) for (a)
4 heads (tracks per cylinder) for (b)
8 heads (tracks per cylinder) for (c) & (d)
17 sectors per track
512 bytes per sector

Giving
a) 5.57 Mbytes
b) 11.14 Mbytes
c) 22.28 Mbytes
d) 44.56 Mbytes

The control of the winchester is achieved via the SCSI bus which interfaces to a host adaptor similar to the Xebec S1410.

2.4 MONITORS

There are two types of Octopus monitor supplied by LSI. A high resolution 12 inch (30 cm) monochrome monitor driven from the composite video connector and a medium (0.4mm stripes) 14 inch (35 cm) colour monitor driven from the TTL connector and capable of displaying 7 colours and black.

2.4.1 MONOCHROME MONITOR

This monitor is designed specifically for use with the Octopus and factory calibrated to synchronise with the Octopus timing details, App 5, no user adjustment is necessary.

2.4.2 COLOUR MONITOR

This monitor is designed specifically for use with the Octopus and factory calibrated to synchronise with the Octopus timing details, App 5, no user adjustment is necessary.

2.5 KEYBOARD

The Octopus keyboard is a 109 station capacitive keyboard with profiled 'sculptured' caps. It is mounted in an injection moulded low profile plastic box and conforms to the proposed DIN 30mm centre row height requirements. There is a tilt bar mounted under the rear which can be used to raise the height of the rear 20mm to increase the rake for operator comfort.

Refer to App.9 , there are 4 groups of keys :-

a) The QWERTY group
b) The Numeric group
c) The cursor group
d) The function group

Note that there are a total of 32 function keys distributed over the above groups.

It is connected to the Octopus via a 600mm (3000mm extended) screened coiled cable terminating in a 5 pin 180 degree DIN plug.

The keyboard connections are in App. 1d. The keyboard present signal is grounded when the keyboard is plugged in and is readable by the software.

The keyboard contains a microprocessor that is resposible for scanning the key matrix, debouncing any depressed keys and transmitting their matrix positions to the Octopus. The matrix codes are given in App. 9 and encoded within the 7 least significant bits of the byte. The most significant bit is zero when a key is pressed and a one when released.

The scanned code is transmitted as a serial stream with TTL levels at 9600 baud with one start and one stop bit no parity. The Octopus illuminates the 'Shift Lock' and 'Caps Lock' keys by transmitting to the keyboard their matrix positions with the most significant bit set to illuminate the LED and cleared to extinguish it. The data from Octopus to keyboard is at 1200 baud with the same format as above.

There is no interpretation of the key positions by the keyboard microprocessor. This permits keyboard encoding to be done by the firmware allowing language changes to be achieved with only one EPROM as this also contains power up messages and logon prompts.

The effect of this partial encoding and the user definable keys is covered in detail in Section 3 and the User Manual under the KEYGEN and LOADKEY.

One exception to the above is the simultaneous depression of 'Ctrl', 'Shift' & 'Del' which causes the serial data line from the keyboard to the Octopus to 'space' for 300ms this is interpreted by the UART on the main logic board as a break which will then force a system reset.

2.6 MECHANICAL

The Octopus computer concept is designed to be capable of producing a wide range of model types with upgradable capabilities. In order to achieve this, considerable effort has gone into the design of the case and mechanical assemblies.

The basic assembly is made from flame retardant injection moulded polyurathane and consists of a base tray and a bonded facia. Virtually all the internal mounting positions are controlled from this assembly which is correspondingly complex but due to the manufacturing method reasonably inexpensive.

The major sub-assemblies are attached to this as follows.

The power supply assembly and distribution loom drop into a moulded recess and are secured by 'hooks' at the front and one single large washer screwed down at the rear, the strength being obtained from the plasic, not from the threaded inserts.

The mains inlet panel is attached by two screws and the looms plugged on.

The drive shelf assembly slides along two runners and engages into the facia, two screws hold it in place.

The main logic board assembly complete with any option boards slides along moulded guides until home. It is retained on the left by a small clamping plate that also restrains the mains inlet panel and blanking panels.

The lid slides along the sides and engages its hooks on the facia moulding. It is retained by two screws and locks the whole assembly together.

The essence of the above is that the strength of the unit is achieved by the interlocking components with the screws being only necessary for retention. This gives an exceedingly robust construction but may still be dismantled for service or upgrade with the minimum of fuss (there are only 9 screws !!).

There are four M5 inserts in the base for vertical mounting or affixing to a desk for security purposes. Details of these and all the mechanical and environmental specifications are in App. 10.

Chapter 3 - Software Description

3.0 INTRODUCTION

This chapter deals with the software aspects of the Octopus computer system. It generally does not describe the various Operating Systems available with the Octopus, these are covered by either the User's Manual or the manuals available from the Operating System's vendor. The only departure to the above are descriptions of enhancements added by LSI, certain ambiguities in some of the above mentioned documentation and clarification of areas where LSI has experienced some user difficulties.

The chapter divides into five sections :-

Section 1 - lists all the firmware primitives contained in the EPROM and how to call them.

Section 2 - explains the interrupt handling

Section 3 - shows how to call 8088 routines from the Z80

Section 4 - discusses serial port access and status handling under CP/M and Concurrent CP/M

Section 5 - contains general notes and hints on MSDOS

3.1 THE FIRMWARE (PROM) PRIMITIVES

3.1.1. GENERAL

All Prom routines are entered through the common entry point at F800:7FF5 via a Far Call (CALLF) intruction. This call is implemented by a Codemacro called PROM_ENTRY, defined as :-

```
CodeMacro PROM_ENTRY
     DB   9Ah
     DW   7FF5h
     DW   0F800h
EndM
```

this is eqivalent to CALLF F800:7FF5 (ie. absolute FFFF5h) and provides a way of specifying the call without confusing the CP/M relocatable assembler. Prior to the call, the parameters required by the PROM must be pushed onto the stack and register AL set up with the required command. Different commands require different amounts of data on the stack. The parameters will be removed from the stack by the Prom routine prior to return (final return in the case of the disc primitives). All registers except AX and those returning results will be preserved.

IMPORTANT

The user is urged to handle all I/O control primarily through the operating system and only if absolutely necessary via the EPROM primitives.

LSI will prserve the integity of the primitive calls but stress that they are under no obligation to retain the existing machine level I/O structure and reserve the right to enhance and modify the architecture as desired with no prior notice.

IMPORTANT

3.1.2 - PROM CALLS

The following is a list of the PROM calls, their call numbers and their actions.

Reset		
	hwinit	0 - Hardware initialization
Character I/O		
	char_in	1 - I/O device input - wait if chr not ready
	char_out	2 - I/O device output - wait if tx buf not empty
	char_in_stat	3 - I/O device input status
	char_out_stat	4 - I/O device output status
Floppy disc		
	flop_set	5 - Floppy parameter initialization
	flop_act	6 - Floppy activity
	flop_abort	7 - Floppy abort
Winchester disc		
	hdsk_act	8 - Hard disk activity
	hdsk_abort	9 - Hard disk abort
	hdsk_size	10 - Hard disk: number of sectors/track
Screen control		
	scrn_mode	11 - Set screen mode
	scrn_config	12 - Set screen format
	scrn_blank	13 - Blank/unblank screen
	scrn_scroll	14 - Scroll screen
	scrn_attrib	15 - Set attributes for scrn_write
	scrn_write	16 - Write to screen
	scrn_wr_block	17 - Write block to screen
	scrn_rd_block	18 - Read block from screen
	scrn_curs_pos	19 - Set cursor position
	scrn_curs_type	20 - Set cursor personality
	scrn_wr_stat	21 - Write status line
	scrn_rd_stat	22 - Read status line
	scrn_wr_font	23 - Write fount

Keyboard control

keybd_map	25 - Keyboard to ASCII translation
keybd_check	26 - Check for key repeat
keybd_type	27 - Keyboard type

Clock control

clock_read	28 - Read clock registers
clock_write	29 - Write clock registers

Configuration data

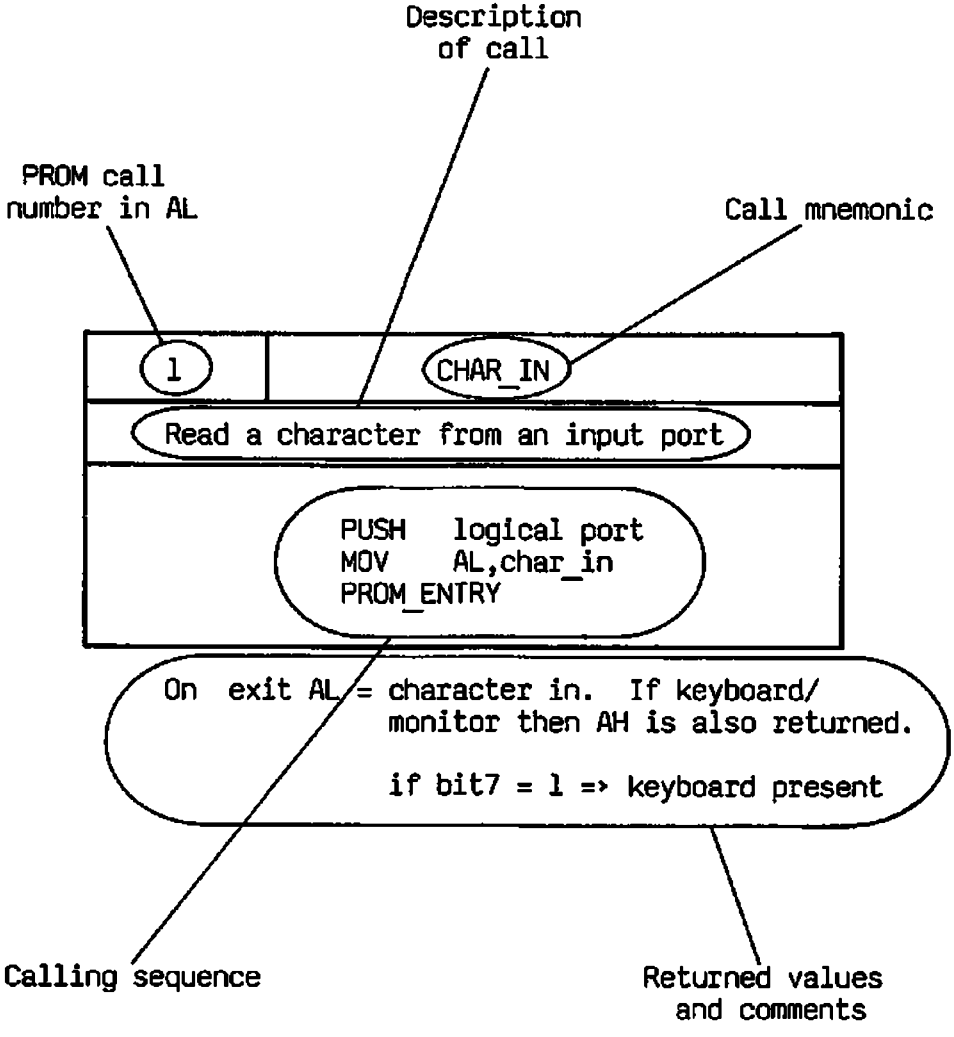
get_switch	30 - System type switch
mc_version	31 - Get hardware firmware & OEM versions

Sonund generation

sound_cont	32 - Continuous sound generation
sound_beep	33 - Beep generation

3.1.2 - FORMAT OF CALLS

The calls are presented in the following example :-



0	HWINIT
Full system initialise (cold boot)	
MOV AL,hwinit PROM_ENTRY	

This is the Prom restart. All hardware except the Z80 is re-initialised and the system boots from disc. The boot sector is entered with DL:AX = read sector number (always zero, but similar to the action of the Winchester disc loader) and DH = flop_act or hdsk_act depending on whether floppy or Winchester boot. This call does not return.

3.1.2.2 CHARACTER I/O ROUTINES

char_in	1 - I/O device input - wait if chr not ready
char_out	2 - I/O device output - wait if tx buf not empty
char_in_stat	3 - I/O device input status
char_out_stat	4 - I/O device output status

Definition of parameters:

- CHARACTER (byte) is the 8-bit character to output to port device.

- LOGICAL PORT assignments (byte) are as follows:

  0 = keyboard/internal screen
  1 = on board DART channel A
  2 = on board DART channel B
  3 = Comms board DART channel A
  4 = Comms board DART channel B
  5 = Comms board SIO 1 channel A
  6 = Comms board SIO 1 channel B
  7 = Comms board SIO 2 (RS-422)
  64 = Parallel interface

1	CHAR_IN
Read a character from an input port	
PUSH logical port MOV AL,char_in PROM_ENTRY	

On exit AL = character in.

If keyboard/monitor (port 0) then AH is also returned.

AH bit 7 = 0 => no keyboard
AH bit 7 = 1 => keyboard present

2	CHAR_OUT
Write a character to an output port	

```
PUSH    character
PUSH    logical port
MOV     AL,char_out
PROM_ENTRY
```

If keyboard/monitor (port 0) then AH is returned

AH bit 7 = 0 => no keyboard
AH bit 7 = 1 => keyboard present.

3	CHAR_IN_STAT
Read the status of an input port	
PUSH logical port MOV AL,char_in_stat PROM_ENTRY	

On exit AL = 00 (false), Z for not ready
AL = 01 (true), NZ for ready

If keyboard/monitor (port 0) then AH is also returned

AH bit 7 = 0 => no keyboard
AH bit 7 = 1 => keyboard present

4	CHAR_OUT_STAT
Read the status of an output port	
PUSH logical port MOV AL,char_out_stat PROM_ENTRY	

On exit AL = 00 (false), Z for not ready
AL = 01 (true), NZ for ready

If keyboard/monitor (port 0) then AH is returned

AH bit 7 = 0 => no keyboard
AH bit 7 = 1 => keyboard present.

3.1.2.3 FLOPPY ROUTINES

flop_set	5 - Floppy parameter initialization
flop_act	6 - Floppy activity
flop_abort	7 - Floppy abort

Definition of parameters:

- CONTROLLER/DRIVE (byte):

```
bits 7  6  5  4  3  2  1  0

     (cntl. no)  (drive no)
```

  - Valid combinations:

```
0  0  0  0  0  0  0  0   controller 0, drive 0
0  0  0  0  0  0  0  1   controller 0, drive 1
0  0  0  1  0  0  0  0   controller 1, drive 0
0  0  0  1  0  0  0  1   controller 1, drive 1
```

- FLAGS (byte):

```
0    x  x  -  -  dd x  x  ss
1   wdq x  x  dq sd x  x  ds
```

x = don't care
dd = double density
sd = single density
dq = 48 tpi disc in 96 tpi drive
ss = single sided
ds = double sided
wdq = allow writing to 48 tpi disc in 96 tpi drives. (NOTE: discs written in this way may not be readable in 48 tpi drives!)

- SECTOR LENGTH (word)
  1024 bytes/sector for all LSI formats

- NUMBER SECTORS ON DRIVE (word)
  No. of sectors/track * no. tracks/side * no. sides

- SECTORS/TRACK (byte)
  Number of physical sectors per track

- NUMBER OF RETRIES (byte)
  Usually 10

- BUFFER SEGMENT (word)
  Segment to read/write from/to

- BUFFER OFFSET (word)
  Offset within above segment

- START SECTOR (word)
  Start sector number

- NUMBER OF SECTORS TO TRANSFER (word)
  Reads/writes required number of sectors

- COMMAND (lsb of word)

  recal wr rd vfy (unused)

  Valid combinations:

0	0	0	0	0	0	0	0	status only
1	1	1	1	0	0	0	0	uncommitted entry point
1	0	0	0	0	0	0	0	recal only
x	0	0	1	0	0	0	0	verify
x	0	1	0	0	0	0	0	read
x	1	0	0	0	0	0	0	write
x	1	0	1	0	0	0	0	write + verify
x	1	1	0	0	0	0	0	seek

x = does recal first if set

If uncommitted command: msb of word = command, lsb = F0H

- SUPPLEMENTARY INFORMATION FOR UNCOMMITTED COMMAND (word)

```
bits 0 - 14  = length of DMA
bit 15       = read (0), write (1)
```

For uncommitted commands, the START SECTOR parameter and the current parameters as set up by FLOP_SET will be used to determine the side selected, precompensation value and sector register. The computed track number will be placed in the data register.

5	FLOP_SET
Set floppy command parameters	

```
PUSH    sector length
PUSH    number of sectors on drive
PUSH    sectors/track
PUSH    number of retries
PUSH    flags
PUSH    floppy controller/drive
MOV     AL,flop_set
PROM_ENTRY
```

Sets up the command parameters for the next floppy call

6	FLOP_ACT

Activate the floppy command

```
PUSH    buffer segment
PUSH    buffer offset within segment
PUSH    dummy (for HARD DISK compatability)
PUSH    start sector
PUSH    command
PUSH    extra info for uncommitted command
PUSH    number of sectors to transfer
PUSH    floppy controller/drive
MOV     AL,flop_act
PROM_ENTRY
JMP     NOT_FINISHED_CODE
        (must end with "retf")
JMP     FINISHED_CODE
```

The routine will return to the return address + 3 when finished. Also at this point AH = FDC status, AL = PROM error. All registers are saved, except AX, from start to finish. The not finished return saves only the segment registers. The Z flag is set if no error occurred or if only re-coverable errors occurred (see over).

Error codes (in AL):

10	seek error
20	CRC error in ID
30	write protected (from FDC)
40	record not found
50	lost data (from FDC)
60	CRC error in data
70	DMA not reached TC
80	prom busy
90	sector out of range
A0	bank crossing
B0	write to protected disc
C0	no track 00
D0	controller/drive combination does not exist
E0	wrong no. sides on disc (8" drives only)

The retry count (0-7) is added to this code.

7	FLOP_ABORT
Abort a currently active floppy command	

```
PUSH    floppy controller/drive
MOV     AL,flop_abort
PROM_ENTRY
```

3.1.2.4 HARD DISC ROUTINES

hdsk_act	8 - Hard disk activity
hdsk_abort	9 - Hard disk abort
hdsk_size	10 - Hard disk: number of sectors/track

The hard disc sector size is 512 bytes. The number of sectors/track depends on the controller type.

Definition of parameters:

- o CONTROLLER/DRIVE (byte)

```
bits 7  6  5  4  3  2  1  0
     (cont. no.)  (drive no.)
```

- o COMMAND (lsb of word)

recal wr rd vfy (see below)

Valid combinations:

```
0  0  0  0  -  -  -  -   uncommitted entry point
1  0  0  0  -  -  -  -   recal only
x  0  0  1  -  -  -  -   verify
x  0  1  0  -  -  -  -   read
x  1  0  0  -  -  -  -   write
x  1  0  1  -  -  -  -   write + verify
```

x = do recal first

If uncommitted command: msb of word = command, lsb = 0

- o SUPPLEMENTARY INFORMATION FOR UNCOMMITTED COMMAND (word)

```
bits 0 - 14  = length of DMA - 1
bit 15       = read (0), write (1)
```

- BUFFER SEGMENT (word)
  Segment to read/write from/to

- BUFFER OFFSET (word)
  Offset within above segment

- START SECTOR - MSW (word)
  Start sector number - most significant word
- START SECTOR - LSW (word)
  Start sector number - least significant word

- NUMBER OF SECTORS TO TRANSFER (byte)
  Reads/writes above number of sectors

8	HDSK_ACT

Activate a hard disc command

```
PUSH    buffer segment
PUSH    buffer offset within segment
PUSH    start sector (msw)
PUSH    start sector (lsw)
PUSH    command
PUSH    supplementary info for uncommitted
          command
PUSH    number of sectors to transfer
          (lower 8 bits only)
PUSH    hard disk controller/drive
MOV     AL,hdsk_act
PROM_ENTRY
JMP     NOT_FINISHED_CODE
        (must end with "retf")
JMP     FINISHED_CODE
```

The routine will return to the return address + 3 when finished. Also at this point AH = 14xx error sense byte, AL = prom error or retries. All registers are saved, except AX, from start to finish. The not finished return does saves only the segment registers. The Z flag is set if no error occurred or if only recoverable errors occurred, see over.

If error:

CH:DX = bad sector address
AL = error code + retry count
AH = 14xx status (error sense)

AL:

7	6	5	4	3	2	1	0
< error code from prom >					-	Error Corrected	Did Retry

Bits 3 thru 7 of AL are zero if there were no errors or if all errors were fixed by error correction or retries.

Errors:

00-07	no error (correctable data error if AH = 98h)
08-0F	select error
10-17	error sending cdb (ie timeout)
18-1F	timeout waiting for ~ACK
20-27	wrong status after e-o-c-interrupt
28-2F	no msg byte
30-37	
38-3F	
40-47	
48-4F	error in reading error
50-57	
58-5F	
60-67	
68-6F	command or RAM error
70-77	DMA not reached TC
78-7F	Error has been successfully read
84	Prom controller busy error
88	Controller bit busy
A0	Bank crossing error
D0	disk controller does not exist

9	HDSK_ABORT
Abort hard disc activity	

```
PUSH    hard disk controller/drive
MOV     AL,hdsk_abort
PROM_ENTRY
```

10	HDSK_SIZE
Read the size of the disc	
PUSH hard disk controller/drive MOV AL,hdsk_size PROM_ENTRY	

On return:
BL = number of heads
CX = 512b sectors/track
SI = number of cylinders
DXAX = total number of sectors
BH = controller type: 0 = Xebec S1410

3.1.2.5 - SCREEN DRIVER ROUTINES

scrn_mode	11 - Set screen mode
scrn_config	12 - Set screen format
scrn_blank	13 - Blank/unblank screen
scrn_scroll	14 - Scroll screen
scrn_attrib	15 - Set attributes for scrn_write
scrn_write	16 - Write to screen
scrn_wr_block	17 - Write block to screen
scrn_rd_block	18 - Read block from screen
scrn_curs_pos	19 - Set cursor position
scrn_curs_type	20 - Set cursor personality
scrn_wr_stat	21 - Write status line
scrn_rd_stat	22 - Read status line
scrn_wr_font	23 - Write fount

The Octopus screen is implemented by a Signetics 2674/2675 set. It is not memory mapped, but the screen primitives allow for fast block read and write operations. For programs which require to read screen data frequently, it is recommended that a copy of the screen image is kept in RAM, since reading from the copy will be faster than reading from the video RAM via the 2674.
Definition of parameters:

- MODE (byte) is defined as follows:

  bits 0,1: 00 = 24 x 80 plus status line
  01 = 24 x 40 plus status line
  10 = 28 x 132 plus status line

  bit 7: 0 = monochrome
  1 = colour

- SCROLL (byte) is defined as follows:

  bits 0,1: Scroll speed in scan lines per frame. Recommended speeds are 1 for slow scroll, 2 for a faster smooth scroll and any value in the range 16-127 for jump scroll.

  bit 7: 0 = up
  1 = down

- CURSOR BLINK (byte) is as follows:

  00b = no cursor
  01b = static cursor
  10b = fast blink
  11b = slow blink

- CURSOR SIZE (byte) contains the start line in bits 4-7 the end line in bits 0-3.

- POSITION (word) contains the row number in the MSB of the word and the column number in the LSB. Both are numbered from zero.

- BLANK (byte) is 1 to blank the display, 0 to unblank it.

- ATTRIBUTE (byte) is defined as follows:

Bit	Monochrome	Colour
0	Blank	Blue foreground
1	Grey background	Green foreground
2	High intensity	Red foreground
3	Underlined	Underlined
4	-	Blue background
5	Inverse video	Green background
6	-	Red background
7	Blinking	Blinking

- NUMBER OF CHARACTERS (byte) is the number of times the character is to be written.

- BUFFER (dword) is an area holding or to receive screen data formatted as one word per screen character with the character code in the low byte and the attribute data in the high byte.

- NUMBER OF PAIRS (word) is the number of attribute-character pairs in the BUFFER

- FONT BUFFER (dword) is an area holding or to hold font data. There are 16 bytes per character (i.e. 1 per scan line) with the topmost scan line first. The least significant bit of each byte is the leftmost dot. If the character cell width is greater than 8, the most significant bit is repeated to fill the remaining dots at the right of the character cell.

- FONT OFFSET (word) is the offset in the font memory at which a font read or write is to begin (i.e. 16 * the code of the first character whose font is to be read or written).

- NUMBER OF FONT BYTES (word) is the number of bytes of font data to read or write (i.e. 16 * the number of characters whose fonts are being read or written).

11	SCRN_MODE
Set the screen mode	

```
PUSH    mode
MOV     AL,scrn_mode
PROM_ENTRY
```

12	SCRN_INT
Screen interrupt routine	
MOV AL,scrn_int PROM_ENTRY	

This entry executes the kernel of the screen interrupt routine. It should normally only be called when a screen interrupt is pending and does nothing if the screen is not in the vertical retrace period. On return, A = 0 and the Z flag is set if after execution of the routine no scroll operation is pending; A is non-zero and the Z flag is clear if further calls to the interrupt routine are required to complete a scroll operation. In either case it is necessary to send an End of Interrupt command to the 8259A but not to the CRT controller.

13	SCRN_BLANK
Blank/unblank the screen	

```
PUSH    blank
MOV     AL,scrn_blank
PROM_ENTRY
```

14	SCRN_SCROLL
Scroll the screen	

```
PUSH    scroll
MOV     AL,scrn_scroll
PROM_ENTRY
JMP     <not_finished_code>
        <finished code>
```

The call returns to the initial return point if a previous scroll is still incomplete. A RETF must be used to re-enter the routine; only the SS and SP registers need be preserved, but if any other registers are destroyed then their values at the final return may not be the same as those at the initial entry.

15	SCRN_ATTRIB
Set attributes for screen write	

```
PUSH    attribute
MOV     AL,scrn_attrib
PROM_ENTRY
```

16	SCRN_WRITE
Write to the screen	

```
PUSH    character
PUSH    number of characters
MOV     AL,scrn_write
PROM_ENTRY
```

The character is written the specified number of times. Writing beyond the end of a row is not supported.

17	SCRN_WR_BLOCK
Write a block to the screen	

```
PUSH    position
PUSH    number of pairs
PUSH    buffer segment
PUSH    buffer offset
MOV     AL,scrn_wr_block
PROM_ENTRY
```

Wrap around from the end of a line to the start of the next is not supported.

18	SCRN_RD_BLOCK
Read a block from the screen	

```
PUSH    position
PUSH    number of pairs
PUSH    buffer segment
PUSH    buffer offset
MOV     AL,scrn_rd_block
PROM_ENTRY
```

19	SCRN_CURS_POS
Set cursor position	

```
PUSH    position
MOV     AL,scrn_curs_pos
PROM_ENTRY
```

20	SCRN_CURS_TYPE
Set cursor personality	

```
PUSH    cursor blink
PUSH    cursor size
MOV     AL,scrn_curs_type
PROM_ENTRY
```

21	SCRN_WR_STAT
Write to the status line	
PUSH buffer segment PUSH buffer offset MOV AL,scrn_wr_stat PROM_ENTRY	

The number of attribute-character pairs in the buffer must equal the number of screen columns.

22	SCRN_RD_STAT
Read the status line	
PUSH buffer segment PUSH buffer offset MOV AL,scrn_rd_stat PROM_ENTRY	

The buffer must be large enough to receive the complete status line in attribute-character form.

23	SCRN_WR_FONT
Write to the fount RAM	

```
PUSH    font buffer segment
PUSH    font buffer offset
PUSH    number of font bytes
PUSH    font offset
MOV     AL,scrn_wr_font
PROM_ENTRY
```

3.1.2.6 - KEYBOARD ROUTINES

keybd_map	25 - Keyboard to ASCII translation
keybd_check	26 - Check for key repeat
keybd_type	27 - Keyboard type

25	KEYBD_MAP

Up/down matrix code to ASCII conversion

```
PUSH    keycode
MOV     AL,keybd_map
PROM_ENTRY
JMP     test_fn_key  ; final return point
JMP     done         ; final return point
```

This processes the up/down code from the keyboard. The routine returns to the final return point as follows:

NZ, NC - a typematic key has been pressed, AL contains the ASCII or function key code

NZ, C - a function key used as a supershift has been pressed, AL contains the function key code

Z, C - a function key used as a supershift has been released; AL contains the function key code

Z, NC - nothing interesting

In all cases, at the final return AH indicates which shift keys are down, as follows:

bit 0 - set if CAPS LOCK key is down
bit 1 - set if SHIFT LOCK key is down
bit 2 - set if REPEAT key is down
bit 3 - set if CONTROL key is down
bit 4 - set if first SHIFT key is down
bit 5 - set if second SHIFT key is down
bits 6,7 - undefined, not necessarily zero

NB - this information is normally only of interest to Concurrent CP/M since the ASCII code in AL already takes account of shifting etc.

The returned values for function codes on the LSI layout keyboard are as follows:

F1 - F24	80h - 97h respectively at first return point, or at final return point if unshifted 98h - AFh respectively at final return point if either SHIFT key or SHIFT LOCK is down
F25 - F28	B0h - B3h respectively
Up arrow	B4h
F29	B5h
Left arrow	B6h
Home	B7h
Right arrow	B8h
F30	B9h
Down arrow	BAh
F31 - F32	BBh - BCh

26	KEYBD_CHECK
Finds the keyboard status	
MOV AL,keybd_check PROM_ENTRY	

Checks the key status based on information sent via keybd_map. On return flags are set as follows:

NZ means the last character key depressed has not yet been released.

C means the repeat key is down and a character key is or was down with it.

In either case, AL is the ASCII or function code of the key concerned and AH indicates which special keys are down, as for KEYBD_MAP.

27	KEYBD_TYP
Find the keyboard type	
MOV AL,keybd_typ PROM_ENTRY	

Returns in AL the keyboard type:
00 = Cherry (LSI layout)
01 = IBM

3.1.2.7 - CALENDAR/CLOCK ROUTINES

clock_read	28 - Read clock registers
clock_write	29 - Write clock registers

Definition of parameters:

- o BUFFER SEGMENT (word) is the segment address of buffer
- o BUFFER OFFSET (word) is the offset address of buffer within segment
- o NUMBER OF REGISTERS TO READ/WRITE (byte) is the number of registers to read from or write to the MC146818 clock chip.
- o START REGISTER NUMBER (byte) is the first register in the MC146818 clock chip to be read or written, in the range 0 to 63.

28	CLOCK_READ
Read the clock chip	

```
PUSH    buffer segment
PUSH    buffer offset within segment
PUSH    number of registers to read/write
PUSH    start register number
MOV     AL,clock_read
PROM_ENTRY
```

If reading from any register in the range 0-9, the system waits until the 'valid date and time' bit is true; if this bit does not become true within a suitable time (due to faulty or absent clock hardware) then the routine returns with the zero flag clear. In all other cases the routine returns with the Z flag set.

29	CLOCK_WRITE
Write to the clock chip	

```
PUSH    buffer segment
PUSH    buffer offset within segment
PUSH    number of registers to read/write
PUSH    start register number
MOV     AL,clock_write
PROM_ENTRY
```

Note that no special action is taken when writing time and date registers, i.e. the caller should stop the clock via an additional call to clock_write before using clock_write to set the time and date and start it again afterwards.

3.1.2.8 - CONFIGURATION DATA

get_switch	30 - System type switch
mc_version	31 - Get hardware firmware & OEM versions

30	GET_SWITCH
Reads the switch data	
MOV AL,get_switch PROM_ENTRY	

This returns the Dip switch configuration in AX

Bit 0,1 = number of double sided floppy drives
Bit 2 = quad drives
Bit 3,4,5 = Winchester type
Bit 6 = unused
Bit 7 = colour monitor connected
Bits 9-11 = unused

bit1	bit0	
0	0	no floppies
0	1	1 floppy
1	0	2 floppies
1	1	not used

bit5	bit4	bit3	
0	0	0	no winchester
0	0	1	RO201
0	1	0	RO202
0	1	1	Reserved
1	0	0	RO204
1	0	1	Reserved
1	1	0	RO208
1	1	1	Reserved

NOTE: The assignment of Winchester drive types is subject to change, except that bits 3,4,5 all zero will always mean that no Winchester is present. For information on Winchester disc capacity use the HDSK_SIZE entry.

31	MC_VERSION
Reads the machine version string	
MOV AL,mc_version PROM_ENTRY	

Returns with the hardware version number in AL, the I/O PROM version number in AH, the OEM number in DL (zero for LSI Octopus), and zero in DH (reserved for future use).

3.1.2.9 - SOUND GENERATION ROUTINES

sound_cont	32 - Continuous sound generation
sound_beep	33 - Beep generation

Definition of parameters:

- o PITCH (word) is the divisor of a 2.4575 MHz clock to produce the pitch.

32	SOUND_START
Start a continuous tone	
PUSH pitch MOV AL,sound_pitch PROM_ENTRY	

If pitch is zero, the sound generator is turned off; if non-zero the sound generator is turned on until a zero call is made.

33	SOUND_BEEP
Generate a beep	

```
PUSH    pitch
MOV     AL,sound_beep
PROM_ENTRY
```

This call produces a fixed length beep without using a wait loop.

3.2.1 - INTERRUPT ASSIGNMENTS

Interrupts are handled by a pair of cascaded 8259A devices programmed in fully nested mode to generate the following interrupt numbers:

Int number	8259A level	Interrupting device
60	0	RS-422 (on comms option board)
61	1	Daisy-chained DARTs and SIOs
62	2	Option bus 2
63	3	Hard disc
64	4	Option bus 4
65	5	Floppy disc
66	6	Option bus 6
67	7	(cascaded hardware device level - reserved by LSI)
68	0	CRT controller
69	1	Option bus 9
6A	2	Real time clock
6B	3	Option bus 11
6C	4	Keyboard
6D	5	Maths coprocessor
6E	6	Printer
6F	7	8259 cleanup
70-7F		Revectored interrupts from on-board SIO (DARTSEL0)
80-8F		Revectored interrupts from DART on comms option board (DARTSEL1)
90-9F		Revectored interrupts from RS-232 SIO on comms option board (SIOSEL0)

3.2.2 - END OF INTERRUPT ACTIONS

During or at the end of each interrupt service routine, the interrupt must be dismissed within the PIC.

For the master channels (interrupts 60-66) this is achieved by outputting an EOI of 60H + level number to INTSEL0_ICW.

For the slave channels (interrupts 68-6F) this is achieved by outputting an EOI of 60H + level number to INTSEL1_ICW followed by an EOI of 67H to INTSEL0_ICW.

For the revectored DART and SIO interrupts, the end of interrupt routine consists of sending an End of Interrupt command (38H) to channel A of the interrupting DART or SIO followed by an EOI of 61H to INTSEL0_ICW. Software which uses the revectoring MUST initialise the interrupt vector registers in the DARTs and SIOs to the correct value (i.e. 70h for the on-board SIO or DART, 80h for the comms board DART and 90h for the comms board SIO) and ensure that the 'status affects vector' bit in the SIO or DART interrupt control register is set. This action is performed for the on-board SIO or DART by all Octopus operating systems.

3.3 - CALLING 8088 ROUTINES FROM THE Z80

The operating systems CP/M-86/80 Plus, Concurrent CP/M-86/80 and MP/M-86/80 for the Octopus provide a means for 8-bit programs to call 16-bit subroutines. This feature may be needed in the following circumstances:

a) To access I/O ports from the Z80. The Z80 can only access memory; IN and OUT instructions have no effect.

b) To access memory beyond the 64K directly addressable by the Z80 at any time, e.g. to access the System Data Area of CP/M.

c) To make use of 8088 firmware routines, e.g. to change the character fount.

The 8088 routine to be called must lie in the Z80 memory space and must finish with a RETF (far return) instruction with the SS and SP registers as they were on entry. Other registers need not be preserved. There is adequate 8088 stack available for most applications. On entry to the routine the CS, DS and ES registers will address the 64K segment in which the Z80 is running, but the stack (and hence the SS register) lies in a different segment.

To call an 8088 routine, execute a BDOS call 255 with the DE register pointing to the routine.

A sample use of this procedure to read data from an I/O port is illustrated over.

```
; Z80 routine to read from port whose address is in HL
; The value read is returned in A

PORTIN:                     ; Z80 part of the routine
    LD   (PORTNO),HL
    LD   DE,PORT88
    LD   C,255
    CALL 0005H
    LD   A,(PORTDATA)
    RET

PORT88:                     ; 8088 part of the routine
    DB   08BH, 016H       ; MOV  DX,PORTNO
    DW   PORTNO
    DB   0ECH             ; IN   AL,DX
    DB   0A2H             ; MOV  PORTDATA,AL
    DW   PORTDATA
    DB   0CBH             ; RETF

; Workspace

PORTNO    DW   0          ; holds port number
PORTDATA  DB   0          ; holds value read from port

; End of routine
```

3.4 - ACCESS TO SERIAL PORTS

This section should be read in conjunction with the Concurrent CP/M Programmers Guide, which is available from LSI Computers Ltd. If you wish to access the ports directly you will also need the Z80 - SIO Technical Manual or the Mostek equivalent, available from Zilog or Mostek or their distributors.

1. Available ports

The basic Octopus configuration includes two asynchronous RS232 ports which are brought out on the back panel in 25-way male D-type connectors labelled Port 1 and Port 2. The connector is wired according to the convention for Data Terminal Equipment (DTE). Port 1 also has current loop capability.

The communications option board (where fitted) has an additional two asynchronous RS232 ports (Ports 3 & 4), two synchronous or asynchronous RS232 ports (Ports 5 & 6) and one RS422 port (Port 7). The four RS232 ports also have current loop capability.

2. Operating system usage of ports

The CP/M Plus and Concurrent CP/M 2.0 operating systems use Port 1 for Printer 1 and Port 2 as the auxilliary I/O device. The Concurrent CP/M 3.1 operating system allocates serial ports to additional physical consoles and printers according to the configuration (e.g. a 3 console 3 printer system would use Ports 1 & 2 for the additional consoles and Ports 3 & 4 for the additional printers; the first console is always the built-in screen and keyboard and the first printer always uses the parallel port). Any remaining ports are addressable as auxilliary I/O devices.

Programs can write to the ports assigned as printers using the BDOS calls L_SETNUM, L_WRITE and L_WRITEBLK. Note that flow control can be achieved using XON/XOFF characters sent back to the Octopus or using the DSR or CTS lines.

Programs can read from and write to the auxilliary ports using BDOS function S_BIOS subfunctions 6 (output, character in CL register image) and 7 (input, character returned in AL). Flow control may be achieved using the DSR or CTS lines (both must be active for transmission to take place). Status routines can be achieved using the multi-tasking approach described below. Under Concurrent CP/M (and CP/M Plus) 3.1 the DX register image should be set to the auxillary device number required (where the first free port is numbered 0); under Concurrent CP/M (and CP/M Plus) 2.0 only the first free port is accessable.

3. Direct access to ports

The chip used for these ports is a Zilog Z80A-SIO or Z80A-DART (or the Mostek equivalents); a DART is an SIO without the synchronous capability. The chips are described in the Z80-SIO Technical Manual which is available from Zilog and their distributors. Users wishing to drive the serial ports directly should obtain this manual.

The port addresses of the main board SIO are:

A0h	Channel A data (Port 1)
A1h	Channel A control/status
A2h	Channel B data (Port 2)
A3h	Channel B control and status

The port addresses of the DART on the option board are:

1A0h	Channel A data (Port 3)
1A1h	Channel A control/status
1A2h	Channel B data (Port 4)
1A3h	Channel B control/status

The port addresses of the SIO on the option board are:

1A4h	Channel A data (Port 5)
1A5h	Channel A control/status
1A6h	Channel B data (Port 6)
1A7h	Channel B control

By default, receive interrupts are enabled on all channels for which there are operating system drivers (i.e. all ports used for additional physical consoles or printers or the auxilliary I/O device) and received characters are buffered for use by the operating system. Transmit interrupts and external/status interrupts may or may not be enabled for these ports depending on the operating system. Interrupts are not enabled on ports beyond the port used for the auxilliary I/O device.

The interrupt vector in Write Register 2 of channel B of the main board DART or SIO is preset to 70h and must not be changed. The interrupt vectors in the option board DART and SIO may or may not be preset; if these devices are used with interrupts then the vector(s) must be set to 80h and 90h respectively.

In all cases it is necessary to write an interrupt control word to Write Register 1 of the channel to be used in order to enable or disable interrupts as required. The Status Affects Vector bit (bit 2) should always be set as the value written to channel B affects channel A also and the operating system interrupt handlers assume this feature is used.

If interrupts are used, the vectors generated by the main board device are:

70h	Channel B Transmit Buffer Empty
72h	Channel B External/Status change
74h	Channel B Receive Character Available
76h	Channel B Special Receive Condition
78h	Channel A Transmit Buffer Empty
7Ah	Channel A External/Status change
7Ch	Channel A Receive Character Available
7Eh	Channel A Special Receive Condition

The interrupt numbers generated by option board devices follow the same sequence but with a base address of 80h for the DART and 90h for the SIO.

All interrupt routines must reset the SIO interrupt system before returning by writing and End of Interrupt command to Write Register 0 of Channel A of the SIO and then writing a Specific End of Interrupt command (61h) to the 8259A interrupt controller chip at port address B0h.

When accessing a control register other than register 0 you should disable interrupts (CLI) before writing the pointer to Write Register 0 and enable them (STI) only after you have read or written the required register. This avoids the possibility of another task or interrupt routine accessing the port just after you have written the pointer, causing both of you to access the wrong register. This is particularly important for channel A since interrupt routines for channel B have to write and end of interrupt command to channel A.

If the SIO is being driven by 8-bit software then the above sequence should be coded as a single 16-bit subroutine (see the application note 'Calling 8088 Routines from the Z80 on the Octopus'). DO NOT use separate routines for enabling and disabling 8088 interrupts as is is possible to hang the system if an 8088 routine called from the Z80 finishes with interrupts disabled.

4. Multitasking considerations

CP/M Plus and Concurrent CP/M are multitasking operating systems so you cannot assume that you will always have the undivided attention of the processors. This can result in loss of received characters. You can avoid this problem in three ways:

a) By stopping all other activity (this will include updating the status line and the system time & date). This is achieved by using the BDOS call P_PRIOR to set your priority higher than other tasks. A value of BCh is usually suitable; this will stop other user tasks and the CLOCK process but will allow the PIN process (which handles the keyboard) to continue.

b) By handling received characters via an interrupt routine and a large buffer. This requires you to drive the SIO or DART directly and is therefore not portable to other Concurrent CP/M systems.

c) By using a separate task to do character input via the auxillary I/O drivers in the operating system. This also makes input status available and is achieved as follows.

   Use BDOS function P_CREATE to create the character input task, having set up its Process Descriptor, User Data Area and stack. The PRIORITY field of the PD should be high (i.e. BCh or less). The starting address on top of the stack should address code to read a character (via the S_BIOS call), put it in a ring buffer and loop back to read another character.

   The original task can then obtain input status by seeing if there are any characters in the buffer and can take characters from the buffer as required.

   A similar approach can be used for character output if output status is required.

Before terminating (i.e. returning to CP/M) the main task should abort the child using the P_ABORT system call.

This approach is illustrated in the example code in below :-

```
; Demo routines for auxilliary device i/o
; The following routines are provided:
;
;       INIT            Initialisation (sets up the AUXIN task)
;       STATUS          Gets aux device input status
;       INPUT           Gets a character from the aux device
;       OUTPUT          Writes a character to the aux device
;       FINISH          Cleans up before return to CP/M
                        (aborts the AUXIN task)


prio    equ     080h    ; Priority of AUXIN task
buflen  equ     256     ; Buffer length (must be a power of 2)

; System call equates

c_write         equ     2
c_writestr      equ     9
p_create        equ     144
p_abort         equ     157
s_bios          equ     50

uda_sp  equ     word ptr 34h
uda_cs  equ     word ptr 50h
uda_ds  equ     word ptr 52h
uda_ss  equ     word ptr 56h
udalen  equ     100h


        CSEG
```

```
; Initialisation routine.  This creates the AUXIN task. If the
; create fails it
; prints a message and returns to CP/M.
; DS must address our data segment

INIT:
; Set up UDA for new task - must be on a paragraph boundary

 mov    bx,offset udaspace
 add    bx,0Fh                      ; round up to next paragraph
 and    bx,0FFF0h

; Initialise UDA (addressed by BX)

     mov        ax,bx
     mov        cx,4
     shr        ax,cl
     mov        pd_uda,ax       ; store par offset from DS in PD

     mov        di,bx
     push       ds      ! pop   es
     mov        cx,udalen/2
     xor        ax,ax
     cld        ! rep   stosw                   ; clear out UDA

     mov        uda_cs[bx],cs     ; initialise register images
     mov        uda_ds[bx],ds
     mov        uda_ss[bx],ds
     mov        uda_sp[bx],offset task_sp
     mov        task_cs,cs

     mov        dx,offset pd                ; create AUXIN task
     mov        cl,p_create
     int        224
     test       ax,ax
     jz         create_ok

     mov        dx,offset bad_create           ; create failed
     mov        cl,c_writestr                  ; print message
     int        224
```

```
     mov        cl,0
     int        224

create_ok:
     ret

; Aux input status routine
;  Returns  NZ  if  character available,  Z  if  no  character
   available

STATUS:
     mov        ax,buf_getp
     cmp        ax,buf_putp                ; compare pointers
     ret

; Aux in routine - returns character in AL
INPUT:
     mov        bx,buf_getp
waitin:
     cmp        bx,buf_putp
     je         waitin                     ; wait for character

     mov        al,buf[bx]
     inc        bx
     and        bx,buflen-1
     mov        buf_getp,bx
     ret

; Aux out routine - pass character in AL

OUTPUT:
     mov        auxout_chr,al          ; store in control block
     mov        dx,offset auxout_bcb
     mov        cl,s_bios
     int        224
     ret
```

```
; Routine to be called prior to termination
; It is necessary to abort the AUXIN task since it shares our
  memory,
; which will be released when we terminate.
; Difficulty  may  be experienced when trying to abort  AUXIN
  under
; version 2.0 of Concurrent CP/M - version 3.1 is preferable

FINISH:
     mov        dx,offset acb            ; try to abort AUXIN
     mov        cl,p_abort
     int        224
     test       ax,ax
     jnz        finish                   ; try again if failed

     ret

; Code for AUXIN task

task_start:
     mov        dx,offset auxin_bcb
     mov        cl,s_bios
     int        224                ; get a character from AUX
     mov        bx,buf_putp
     mov        buf[bx],al               ; store it in buffer
     inc        bx
     and        bx,buflen-1
     mov        buf_putp,bx
     jmps       task_start
```

```
        DSEG        ; Data
; Message printed if task creation fails
bad_create      db      'Create failed.$'

; Process descriptor for AUXIN task
pd      dw      0, 0
        db      0, prio
        dw      0
        db      'AUXIN   '
pd_uda  dw      0, 0, 0, 0
        dw      0, 0, 0, 0
        dw      0, 0, 0, 0
        dw      0, 0, 0, 0

; Space for UDA of new task
udaspace        rb      udalen + 15

; Stack for new task
        rw      100
task_sp dw      offset task_start                       ; IP
task_cs rw      1                                       ; CS
        dw      0200h                                   ; flags

; Abort control block
acb     dw      0, 00ffh
        db      0, 0
        db      'AUXIN   '

; Buffer for characters read from AUXIN
buf             rb      buflen
buf_getp        dw      0
buf_putp        dw      0

; Bios control block for reading from AUX
auxin_bcb       db      7                               ; reader
                dw      0, 0

; Bios control block for writing to AUX
auxout_bcb      db      6                               ; punch
auxout_chr      db      0, 0
                dw      0
```

3.5 - IMPLEMENTATION NOTES FOR MS-DOS

Contents:-

3.5.1 WHAT THIS SECTION COVERS - WHAT THE MANUAL COVERS

This file covers the LSI specific features of the Octopus MS-DOS implementation. The manuals you should also have are:

- The Microsoft MS-DOS Operating System Users Guide (for general MS-DOS information)
- The LSI Octopus System Guide, (for general Octopus information on hardware, setting up the Winchester if you have one, CP/M utilies to run under the emulator, and other information common to CP/M users).

If you intend to do much assembly language programming you should obtain the following two manuals, plus some extra software (contact Softpac - 0278 421 020 for price and availiability):

- The Microsoft MS-DOS Programmers reference Manual (for details of operating system calls).
- The Microsoft Macro Assembler Manual.

3.5.2 - MS-DOS EDITING KEYS

The editing keys discussed in chapter 6 section 1 of the Operating System Users Guide are assigned as follows

<COPY1>	esc S
<COPYUP>	esc T
<COPYALL>	esc U
<SKIP1>	esc V
<SKIPUP>	esc W
<VOID>	esc E
<INSERT>	esc P
<NEWLINE>	esc J

esc R can be used to introduce escape sequences.

These escape sequences have been assigned to the function keys in the example key file MSDOS.KEY which also contains some useful MS-DOS commands.

F1	introduce escape char (esc R)
F2	<COPY1>
F3	<COPYUP>
F4	<COPYALL>
F5	<SKIP1>
F6	<SKIPUP>
F7	<VOID>
F8	<INSERT>
F9	<NEWLINE>
F10	a: <return>
F11	b: <return>
F12	c: <return>
F13	dir/w <return>
F14	a:chkdsk <return>
F15	copy
F16	del
F17	rename
F18	type

3.5.3 - KEYGEN, FONTGEN, PARMGEN, ARCHIVE, CPEM and RDCPM.

MS-DOS versions of the CP/M utilities KEYGEN, FONTGEN, PARMGEN and ARCHIVE are not supplied. Instead we provide an emulator program CPEM.COM under which the CP/M versions can be run.

To run these programs proceed as follows

First use RDCPM to copy the files KEYGEN.CMD, PARMGEN.CMD, FONTGEN.CMD and ARCHIVE.CMD from the CP/M release disk to a copy of your MS-DOS master or to the Winchester drive; eg

```
RDCPM B:KEYGEN.CMD          (floppy system)
RDCPM C:KEYGEN.CMD          (Winchester system)
```

Now run the program under the emulator;

```
CPEM KEYGEN
```

Not all of PARMGENs facilities are applicable to MS-DOS. LOADPARM will report an error if it thinks you are trying to load inappropriate parameters. The current version of LOADPARM will load parameters for the serial ports, the cursor definition, screen mode and attributes (status line not used), scroll speed and keyboard repeat/delay. The P: drive is considered inappropriate for MS-DOS since Microsoft define disk parameters to be on the boot sector of each floppy (see section 3. of this document). PARMGEN's printer number selection is not supported; use DEV instead (see section 5. of this document).

CPEM is not intended to be a general purpose CP/M emulator but is provided solely for running the above utilities. One of CPEM's limitations is that it will not process command line arguments.

Note that RDCPM will read from your CP/M Winchester partition if you specify A or B on a Winchester system. It does not support the CP/M P: drive although it will automatically support LSI and IBM PC CP/M floppies. To obtain a directory of a CP/M floppy or Winchester partition enter

```
RDCPM DIR B:                (or A: or C:)
```

3.5.4 - READING AND WRITING DIFFERENT FLOPPY DISC FORMATS

The MS-DOS implementation on the Octopus can handle a variety of disc formats. These are

1. IBM PC-DOS version 1 single and double sided

2. IBM PC-DOS version 2 single and double sided (this is our standard format)

3. Any IBM System 34 compatiable 5" format that implements the boot sector BPB and has a sector size of 128, 256 or 512 bytes.

This should cover most formats found on currently available MS-DOS micros (the major exception is the Sirius, its floppy format is totally non standard and needs special drives to read it).

The above formats are automatically sensed by our MS-DOS implementation, the user does not need to specially configure the system in any way.

Note that Octopus are supplied with double sided 48 or 96 tpi (tracks per inch) drives. If you have a 48 tpi machine you can only read and write 48 tpi formats, you cannot read or write 96 tpi discs. If you have a 96 tpi Octopus you can read but not write 48 tpi format discs. Writing data to a 48 tpi format disc from a 96 tpi drive cannot be done reliably (since the written track will be half the normal width) especially if you are then going to move the disc to another machine. For this reason 48 tpi discs in 96 tpi drives appear write protected. Note - FORMAT supports multiple disc formats - see below.

3.5.5 - NOTES ON FORMAT

The normal syntax of format is

```
format <d>: {switches}
```

where d is a valid MS-DOS drive reference and {switches} is one or more option switches defined as follows.

/?	show switches for this version of format
/m	verify only
/1	on 48 tpi machines format a single sided disc
/c	clear - just wipes out data, no physical format
/0	IBM PC-DOS v1.x compatiable directories
/v	volume label required
/s	make system disc
/8	eight sectors/track (PC-DOS v1.x, CAL PC)

Later versions of FORMAT may add more switches to this list, use /? to find what switches your version supports.

The default disc format produced is

```
512 bytes/sector
9 sectors/track
double sided
```

On 48 tpi machines this is the same format as IBM PC-DOS version 2.0. The earlier PC-DOS standard of 8 sectors/track is also supported, for example entering

```
format b:/8/0/1
```

on a 48 tpi machine will produce a floppy that could be used on a single sided IBM PC running PC-DOS v1.25

The /s option copies the hidden system files and COMMAND.COM from the default drive to make a bootable disc.

The /v option allows you to put a volume label onto the new disc - this will then show in all directory listings.

Note that FORMAT must be used for initialising Winchesters, see the Octopus System Guide for details.

3.5.6 - MS-DOS DEVICES

These are the devices built in to the LSI BIOS.

```
CON   - the console
LST   - the parallel Centronics interface
PORT1 - the first  built in RS232 port
PORT2 - the second   "   "    "     "
PRN   - see below
AUX   -  "    "
```

LSI have implemented two assignable drivers that are accessable to MS-DOS. These are

PRN - intended as a printer driver, expands tab characters to spaces and supports XON/XOFF.

AUX - intended as a communications channel that could be passing binary data, input/output not modified in any way.

Output from these two drivers can be assigned to any physical port driver using the DEV program. The default port assignments set up by the Octopus are

```
PRN is set to LST, the parallel printer port
AUX is set to PORT1, one of the RS232 ports
```

Here's an example of using DEV to change the assignments:

```
dev prn=port1
```

This assigns the printer driver PRN to the first RS232 serial port. Now entering

```
copy afile.txt prn
```

will print the file AFILE.TXT on the serial printer, expanding tabs to spaces and using the XON/XOFF protocol.

DEV will give you help at any time, simply type

```
dev
```

for a short help screen.

3.5.7 - HINTS ON DEVICE,ATTRIBUTES

To change screen or printer characteristics quickly it is possible to use the MS-DOS COPY function to send the appropriate escape sequence direct from the keyboard. For example, to put your Octopus screen into its fastest scroll mode enter

```
copy con con <return>
<esc>RsZ <control-Z> <return>
```

Explanation:

"con" is the Octopus console device (keyboard and screen), the keyboard is being copied directly to the screen.

<esc>R is the MS-DOS key sequence to introduce an escape sequence into the text (if you have auto run the LSI supplied AUTOEXEC.BAT file succesfully at boot time you should find the key F1 is set to this).

<esc>sZ is the Octopus screen code for jump scroll. <control-Z> is the MS-DOS end of file code, this will terminate the COPY operation. As the transfer terminates you should see the scroll rate change.

Instead of using copy to change screen attributes you can use the batch command echo instead; for example

```
echo <esc>RsZ
```

sets the Octopus to fast scroll in the same way as the first example. This method can be used for quickly setting other Octopus screen attributes. See the Octopus System Guide for details of the other codes availible.

Note that a space must always follow echo, as shown above, otherwise echo fails. The space is not echoed.

The advantage of using COPY is that the same technique can be used to send escape sequences to other devices eg printers -

```
copy con prn
<esc>R...  <esc>R....  etc etc  <control-Z>
```

Consult your printer manual for details of the appropriate escape codes.
It is emphasised that the above methods are quick techniques for convenience. Programs that regularly need to modify screen or printer attributes should be made to issue the relevant codes directly.

3.5.8 - ANSI ESCAPE SEQUENCES

The Octopus implements most of the ANSI escape sequences (used by IBM) in addition to the standard Heath/Zenith (Z19/Z89) compatible Octopus escape sequences detailed in the Octopus System Guide.

Here are the ANSI escape sequences implemented:

CUP (cursor position)	esc [Pl ; Pc H
HVP " "	esc [Pl ; Pc f
CUU (cursor up)	esc [Pn A
CUD (cursor down)	esc [Pn B
CUF (cursor forward)	esc [Pn C
CUB (cursor back)	esc [Pn D
DSR (device status report)	esc [6 n
SCP (save cursor posn.)	esc [s
RCP (restore cursor posn.)	esc [u
ED (erase display)	esc [2 J
IL (insert line)	esc [Pn L
DL (delete line)	esc [Pn M
EL (erase line)	esc [K
SGR (set grphx rendition)	esc [Pn ; ... ; Pn m
SM (set mode)	esc [Pn h
RM (reset mode)	esc [Pn l

Where Pn is a decimal parameter

Possible values of Pn for the SGR sequence are:

0	all attributes off
1	bold on
4	underscore on
5	blink on
7	reverse video on

Possible values of Pn for the SM and RM sequences are:

0	40 x 24 black and white
1	40 x 24 colour
2	80 x 26 black and white
3	80 x 26 colour

SM and RM have exactly the same effect on the Octopus since wrap at end of line is always enabled.

If the Pn parameter is missed out a default value of 1 is taken by all escape sequences. IL and DL are not actually implemented on the IBM PC. Note that you cannot use the IBM's special ANSI sequence to set function key - you should use KEYGEN/LOADKEY for this.

3.5.9 - TRANSFERING DATA TO CP/M

This can be done with the CP/M program RDMS.CMD (availiable at extra cost from Softpac at the address in section 0). RDMS will read files from any LSI MS-DOS floppy into CP/M. It has the same syntax as PIP so entering

```
RDMS A:=B:*.DAT
```

will copy all files with extension of DAT from the MS-DOS floppy in drive B to the CP/M disk in A. In addition entering

```
RDMS DIR B:
```

will give a directory of the MS-DOS files on the specified floppy.

```
RDMS STAT B:
```

will give you the characteristics (total capacity, sectors per track etc) of the MS-DOS disk.

Note that RDMS can only be used to read files into CP/M from MS-DOS (use RDCPM to go the other way). RDMS uses direct calls to the Octopus firmware and will not run on any other machine.

3.5.10 - WINCHESTERS

The procedure for setting up Winchesters is fully detailed in the revised Octopus System Guide which should be consulted for step by step instructions. This is a very brief overview of the process.

1. Under CP/M physically format the Winchester and create an MS-DOS partition (+ any other partitions) using WDFORMAT.

2. Boot up MS-DOS from floppy and initialise the Winchester directories with FORMAT:

```
FORMAT A:/S          (transfer MS-DOS as well)
FORMAT B:
```

You may now use Winchester drives A: and B: as normal. Remember to back up your MS-DOS master floppy.

3.5.11 - IBM PC COMPATIABILITY

Although MS-DOS and PC-DOS are compatiable operating systems most of the popular software for the IBM PC accesses the hardware directly by reading and writing to ports, writing to the memory mapped screen and using the ROM BIOS interrupts. The standard Octopus MS-DOS implementation will not support this, typical symptoms being uninitialised interrupts and the machine hanging up. As an expansion option LSI are offering the Octopus 127 board, plus an installable MS-DOS device driver which will enable many IBM packages (Lotus 123, DBase III, Open Access etc) to run under Octopus MS-DOS. Expansion option boards are documented in SECTION 2.

1. 8088 iAPX 86,88 User's Manual.
Intel Corp (U.K) Ltd.
Pipers Way,
Swindon,
Wilts SN3 1RJ
Tel. (0793) 488 388

2. 8237A-5 Microprocessor and Peripheral Handbook
Intel Corp (U.K) Ltd.
Pipers Way,
Swindon,
Wilts SN3 1RJ

3. 8284A Microprocessor and Peripheral Handbook
Intel Corp (U.K) Ltd.
Pipers Way,
Swindon,
Wilts SN3 1RJ

4. 8259A Microprocessor and Peripheral Handbook
Intel Corp (U.K) Ltd.
Pipers Way,
Swindon,
Wilts SN3 1RJ

5. 8255 Microprocessor and Peripheral Handbook
Intel Corp (U.K) Ltd.
Pipers Way,
Swindon,
Wilts SN3 1RJ

6. 8253 Microprocessor and Peripheral Handbook
Intel Corp (U.K) Ltd.
Pipers Way,
Swindon,
Wilts SN3 1RJ

7. Z80 CPU

Microcomputer Components.
Zilog (U.K) Ltd.
Babbage House,
Maidenhead,
Berks SL6 1DU
Tel. (0628) 36131

Microelectronic Data Book.
Mostek (U.K) Ltd.
Masons House,
1-3 Valley Drive,
Kingsbury Road,
London NW9
Tel. 01-204 9322

8. Z80 SIO

Microcomputer Components.
Zilog (U.K) Ltd.
Babbage House,
Maidenhead,
Berks SL6 1DU
Tel. (0628) 36131

Microelectronic Data Book.
Mostek (U.K) Ltd.
Masons House,
1-3 Valley Drive,
Kingsbury Road,
London NW9
Tel. 01-204 9322

9. 1793

SMC Data Catalog.
Manhatten Skyline Ltd.
Bridge Road,
Maidenhead,
Berks FL6 8DB
Tel. (0628) 39735

10. 9229 SMC Data Catalog.
Manhatten Skyline Ltd.
Bridge Road,
Maidenhead,
Berks FL6 8DB
Tel. (0628) 39735

11. MC146818 Motorola Ltd.
York House,
Empire Way,
Wembly,
Middx
Tel. 01-902 8836

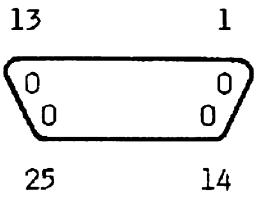
12. 2674 SCN2674 Data Sheet
Signetics Corp.
Mullard Ltd.
Mullard House
Torrington Place
London WC1E 7HD
Tel. 01-580 6633

13. 2675 SCN2675 Data Sheet
Signetics Corp.
Mullard Ltd.
Mullard House
Torrington Place
London WC1E 7HD
Tel. 01-580 6633

14. 75188/189 The Linear and Interface Circuits Data Book.
Texas Instruments Ltd.
Northern European Semiconductor Division,
Manton Lane,
Bedford MK41 7PA
Tel. (0234) 67466

15. 8251A Microprocessor and Peripheral Handbook
Intel Corp (U.K) Ltd.
Pipers Way,
Swindon,
Wilts SN3 1RJ

PORT 1 AND PORT 2



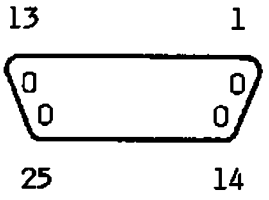
	PIN No.		RS232
	2	-	TRANSMITTED DATA (M)
	3	-	RECEIVED DATA (M)
	4	-	REQUEST TO SEND (M)
	5	-	CLEAR TO SEND (O)
	6	-	DATA SET READY (O)
	7	-	SIGNAL GROUND
*	15	-	TRANSMITTER SIGNAL ELEMENT TIMING (DCE) (F)
*	17	-	RECEIVE SIGNAL ELEMENT TIMING (DCE) (F)
	20	-	DATA TERMINAL READY (O)
*	23	-	TEST (+12 Volts)
*	24	-	TRANSMITTER SIGNAL ELEMENT TIMING (DTE) (F)

CURRENT LOOP

*	11	-	TRANSMIT, POSITIVE SUPPLY
*	13	-	CURRENT LOOP, NEGATIVE SUPPLY
*	14	-	TRANSMIT, NEGATIVE SUPPLY
*	16	-	RECEIVE, NEGATIVE SUPPLY
*	18	-	CURRENT LOOP, POSITIVE SUPPLY
*	25	-	RECEIVE, POSITIVE SUPPLY

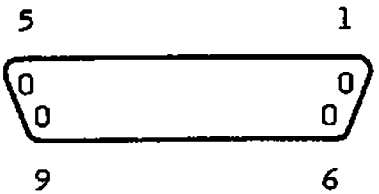
* ONLY USED ON PORT 1

PARALLEL



PIN No.		SIGNAL
1	-	STROBE (L)
2	-	DATA BIT 1 (H)
3	-	DATA BIT 2 (H)
4	-	DATA BIT 3 (H)
5	-	DATA BIT 4 (H)
6	-	DATA BIT 5 (H)
7	-	DATA BIT 6 (H)
8	-	DATA BIT 7 (H)
9	-	DATA BIT 8 (H)
10	-	ACKNOWLEDGE (L)
11	-	BUSY (H)
12	-	PAPER EMPTY (H)
13	-	SELECT OUT (H)
14	-	AUTO FD XT (L)
15	-	ERROR (L)
16	-	INITIALIZE (L)
17	-	SELECT IN (L)
18-25	-	GROUND (0 Volts)

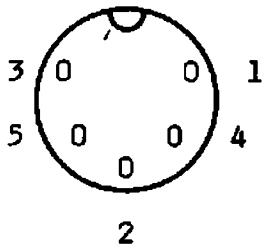
VIDEO "TTL"



PIN No.		SIGNAL
1-2	-	GROUND (0 Volts)
3	-	RED
4	-	GREEN
5	-	BLUE
6	-	LUMINENCE
7	-	VIDEO
8	-	HORIZONTAL SYNC (H)*
9	-	VERTICAL SYNC (H)*

* Hardware selectable (default values shown)

KEYBOARD



5 PIN CONNECTOR - 180° DIN SOCKET

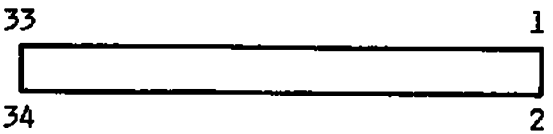
PIN No		SIGNAL
1	-	KEYBOARD PRESENT (L)
2	-	KEYBOARD DATA IN (L)
3	-	KEYBOARD DATA OUT (L)
4	-	GROUND (0 Volts)
5	-	+5 Volts

OPTION BUS CONNECTOR

Pin No.	A	B	C
1	0V	0V	0V
2	+5V	+5V	+5V
3	+12V	+12V	+12V
4	-12V	-12V	-12V
5	BDB0(H)	BREQ0(L)	BDB1(H)
6	BDB2(H)	BREQ3(L)	BDB3(H)
7	BDB4(H)	BREQ5(L)	BDB5(H)
8	BDB6(H)	BREQ6(L)	BDB7(H)
9	*BDB8(H)	BACK0(L)	*BDB9(H)
10	*BDB10(H)	BACK3(L)	*BDB11(H)
11	*BDB12(H)	BACK5(L)	*BDB13(H)
12	*BDB14(H)	BACK6(L)	*BDB15(H)
13	0V	0V	0V
14	BAB0(H)	VSYNC(L)	BAB1(H)
15	BAB2(H)	HSYNC(L)	BAB3(H)
16	BAB4(H)	DCLK(L)	BAB5(H)
17	BAB6(H)	CCLK(L)	BAB7(H)
18	BAB8(H)	*BHE(L)	BAB9(H)
19	BAB10(H)	BINT0(L)	BAB11(H)
20	BAB12(H)	BINT1(L)	BAB13(H)
21	BAB14(H)	BINT2(L)	BAB15(H)
22	BAB16(H)	BINT4(L)	BAB17(H)
23	BAB18(H)	BINT6(L)	BAB19(H)
24	BMRD(L)	BINT9(L)	BMWR(L)
25	BIORD(L)	BINT11(L)	BIOWR(L)
26	BM(H)/IO(L)	RESERVED	BSMC(L)
27	BCLK(L)	BIEO(H)	BSYSIO(L)
28	BREFRESH(L)	BAUDCLK	BRESET(L)
29	BPFAIL(L)	BALE(H)	BWAIT(L)
30	RESERVED	RESERVED	BAUDIO
31	+5V	+5V	+5V
32	0V	0V	0V

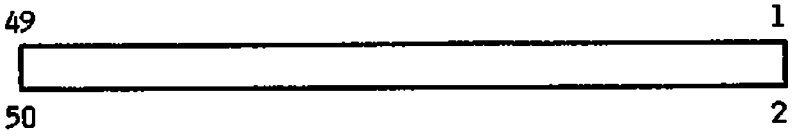
Notes: 1. BIEO(H) is used for completing the SIO and DART daisy chain to the main logic board.
2. Signals marked * are for 16 bit operation and are not controlled by the main logic board. BHE(L) is pulled up, BDB8-15(H) are floating.

FLOPPY



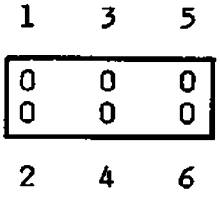
PIN No.		SIGNAL
8	-	INDEX (L)
10	-	DRIVE SELECT 0 (L)
12	-	DRIVE SELECT 1 (L)
16	-	MOTOR ON (L)
18	-	DIRECTION INPUT (L) / OUTPUT (H)
20	-	STEP (L)
22	-	WRITE DATA (L)
24	-	WRITE GATE (L)
26	-	TRACK 0 (L)
28	-	WRITE PROTECT (L)
30	-	RAW READ (L)
32	-	SIDE 0 (H) / SIDE 1 (L)
34	-	READY (L)
1-49	-	GROUND (0 Volts)

WINCHESTER

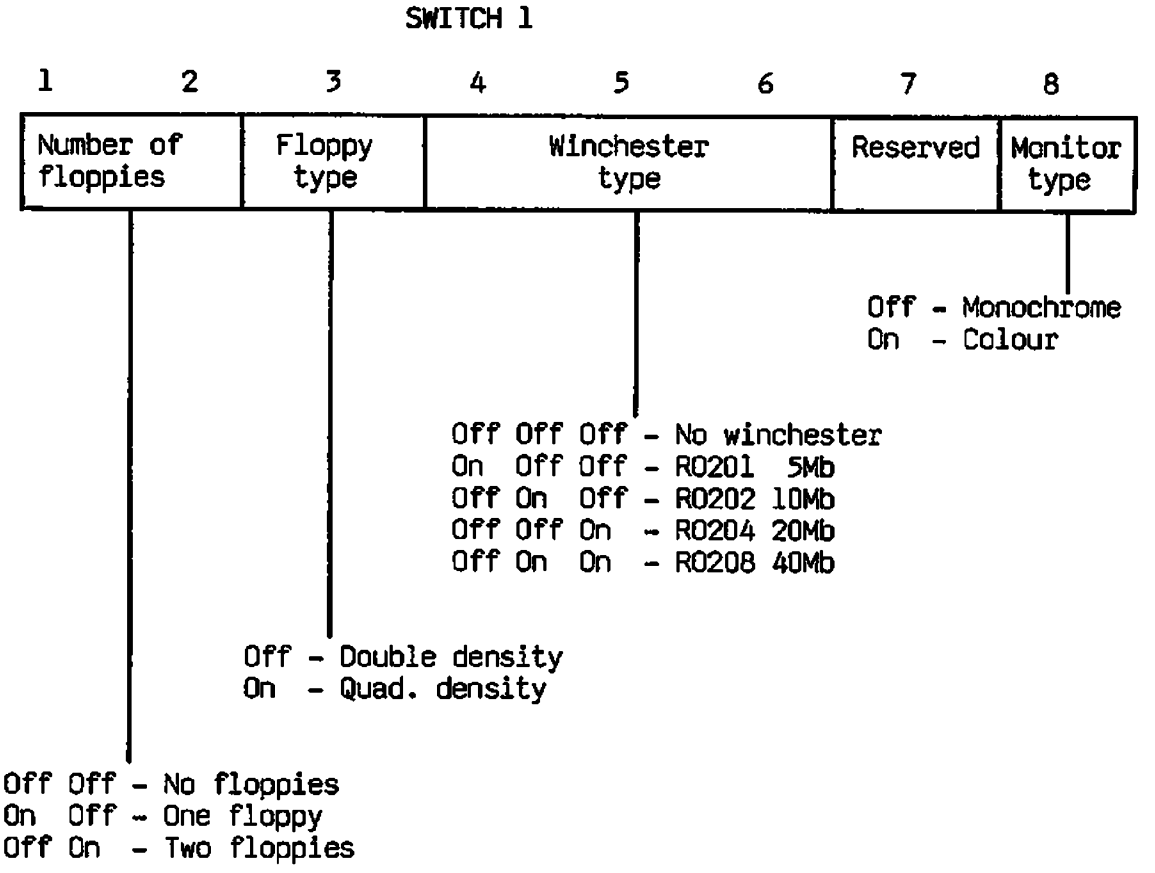


PIN No.		SIGNAL
2	-	DATA 0 (L)
4	-	DATA 1 (L)
6	-	DATA 2 (L)
8	-	DATA 3 (L)
10	-	DATA 4 (L)
12	-	DATA 5 (L)
14	-	DATA 6 (L)
16	-	DATA 7 (L)
36	-	BUSY (L)
38	-	ACKNOWLEDGE (L)
40	-	RESET (L)
42	-	MESSAGE (L)
44	-	SELECT (L)
46	-	COMMAND (L) / DATA (H)
48	-	REQUEST (L)
50	-	INPUT (L) / OUTPUT (H)
1-49	-	GROUND (0 Volts)

LED AND SPEAKER CONNECTIONS

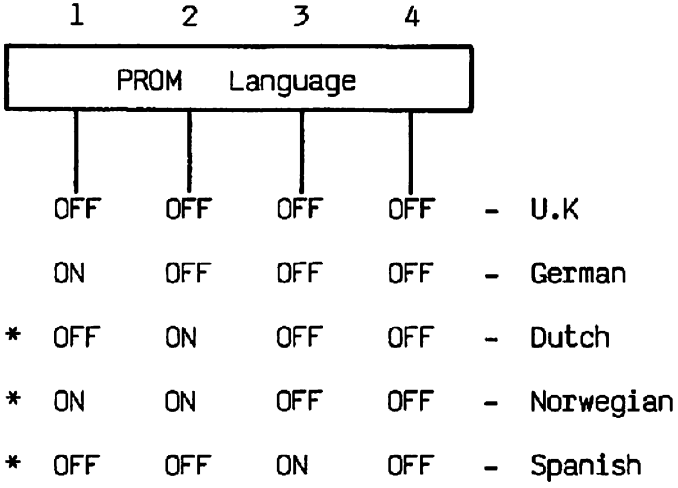


PIN No		SIGNAL
1	-	ON
2	-	SPEAKER 1
3	-	ON RETURN
4	-	SPEAKER 2

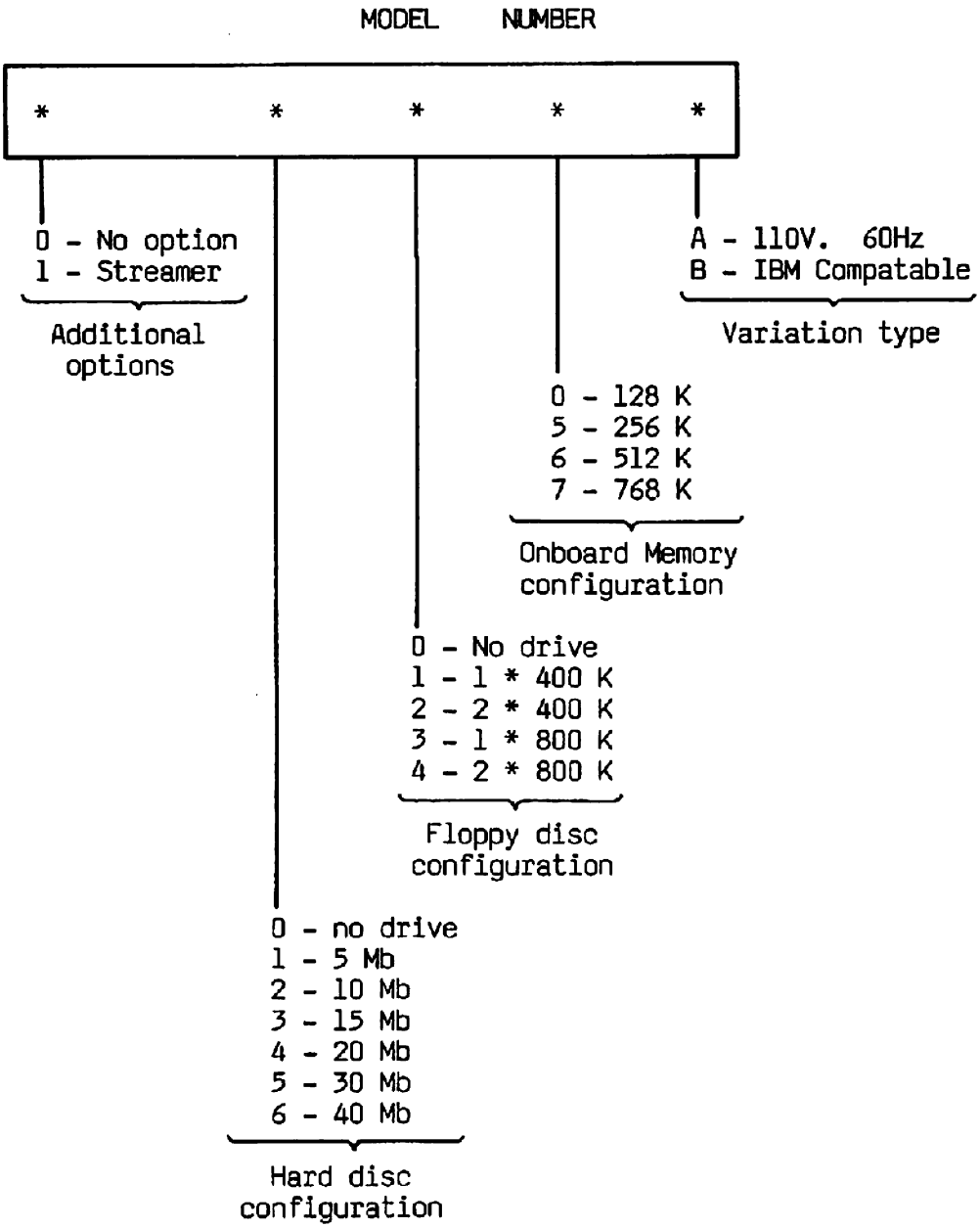


All other combinations are reserved for system expansion

SWITCH 2

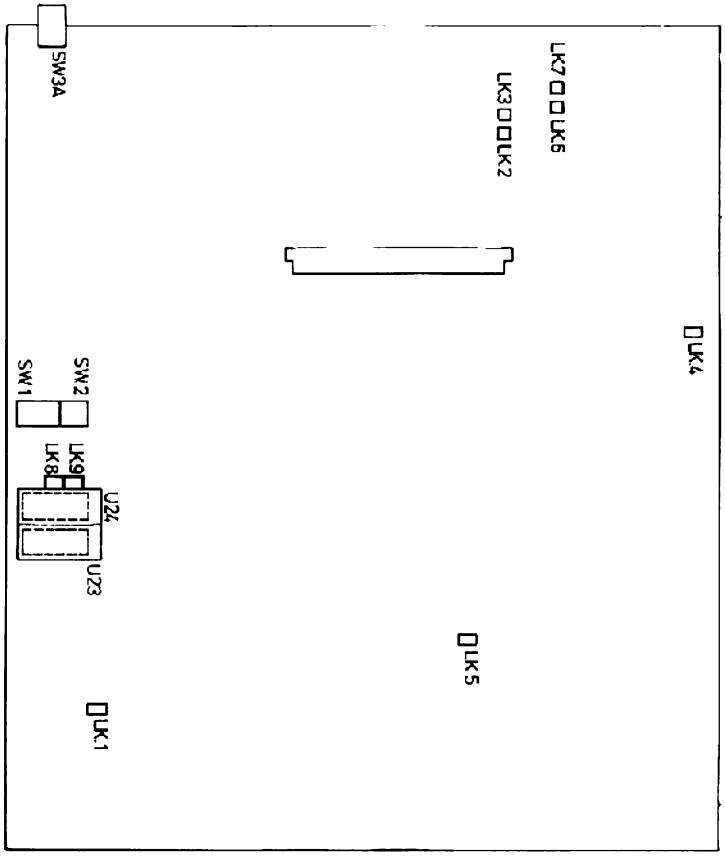


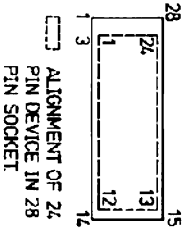
NOTE : Options marked * have been Provisionally allocated. All other combinations are reserved for system expansion.


MODEL NUMBER
* * * * *
0 - No option
1 - Streamer
Additional options
A - 110V. 60Hz
B - IBM Compatable
Variation type
0 - 128 K
5 - 256 K
6 - 512 K
7 - 768 K
Onboard Memory configuration
0 - No drive
1 - 1 * 400 K
2 - 2 * 400 K
3 - 1 * 800 K
4 - 2 * 800 K
Floppy disc configuration
0 - no drive
1 - 5 Mb
2 - 10 Mb
3 - 15 Mb
4 - 20 Mb
5 - 30 Mb
6 - 40 Mb
Hard disc configuration

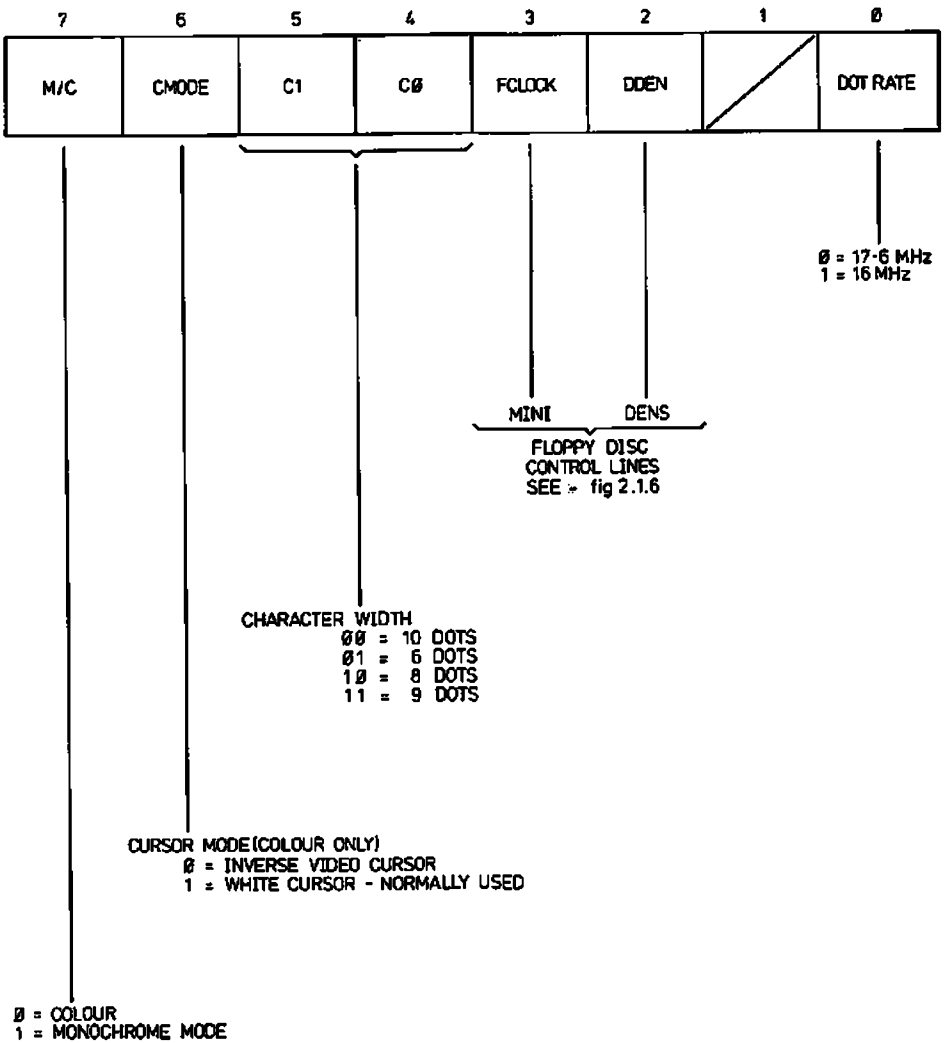
LINK REFERENCE		DESCRIPTION	SHIPPED FROM FACTORY
LK.1	1 TO 2	MP TO 128K EPROM'S IN U23,U24	X
	2 TO 3	256K EPROM'S ONLY IN U23,U24	
LK.2	1 TO 2	EXTERNAL TRANSMIT CLOCK PORT 1 PIN 15	
	1 TO 3	INTERNAL TRANSMIT CLOCK PORT 1	X
LK.3	1 TO 2	INTERNAL RECEIVE CLOCK PORT 1	
	3 TO 2	EXTERNAL RECEIVE CLOCK PORT 1 PIN 17	X
LK.4	OPEN	CLOCK BATTERY ISOLATED	
	SHORTEN	CLOCK BATTERY IN CIRCUIT	X
LK.5	OPEN	256K BIT DRAM DEVICES	
	SHORTEN	64K BIT DRAM DEVICES	X
LK.6	1 TO 2	HSYNC NON INVERTED	
	1 TO 3	HSYNC INVERTED	X
LK.7	1 TO 2	VSYNC NON INVERTED	
	1 TO 3	VSYNC INVERTED	X
LK.8	A TO B	U24 IN PROM MODE	X
	B TO C	U24 IN STATIC RAM MODE	
LK.9	A TO B	U24 IN PROM MODE	X
	B TO C	U24 IN STATIC RAM MODE	

BOARD SETTINGS





VIDEO CONTROL LOGIC
-CONTROL REGISTER



CRT ATTRIBUTES

MONOCHROME (SWITCH 8 OFF)

7	6	5	4	3	2	1	Ø
BLINK	GP1*	REVERSE VIDEO	GP2*	UNDER LINE	HIGH INTENSITY	GREY BACKGROUND	BLANK

*GP1,2 ARE GENERAL PURPOSE USER DEFINED BITS - MAY BE USED FOR PROTECT ETC.

COLOUR (SWITCH 8 ON)

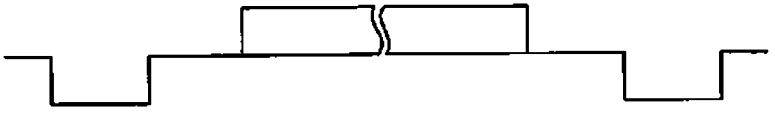
7	6	5	4	3	2	1	Ø
BLINK	RED	GREEN	BLUE	UNDER LINE	RED	GREEN	BLUE
	BACKGROUND				FOREGROUND		

VIDEO TIMING-25/27×80 MODE

ESC,m,0 } 25
ESC,m,1 }

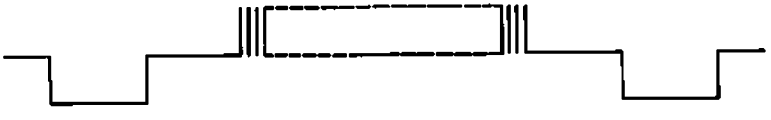
ESC,m,6 } 27
ESC,m,7 }

HORIZONTAL TIMING



hsync	hbp	hdisp	hfp	TOTAL
4	11	81	6	102 CHARS
2·25	6·1875	45·5625	3·375	57·375 µs

VERTICAL TIMING



vsync	vbp	vdisp	vfp	TOTAL
5	16	325	4	350 LINES
0·287	0·918	18·647	0·23	20·08 ms

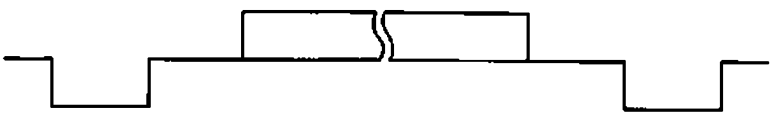
TO BE USED WITH OCTOPUS MONITORS

(CONVENTIONAL TVS WILL NOT SYNC IN THIS MODE)

VIDEO TIMING-25/27×40 MODE

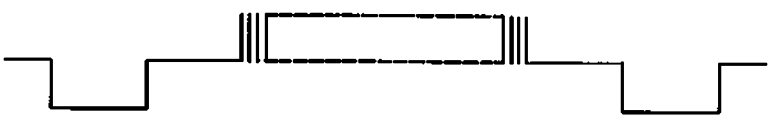
ESC,m,2 } 25
ESC,m,3 }
ESC,m,8 } 27
ESC,m,9 }

HORIZONTAL TIMING



hsync	hbp	hdisp	hfp	TOTAL
10	15	40 DW (81)	8	114 CHARS
5·625	8·4375	45·5625	4·5	64·125 μs

VERTICAL TIMING



vsync	vbp	vdisp	vfp	TOTAL
7	20	275	10	312 LINES
0·449	1·28	17·634	0·641	20·00 ms

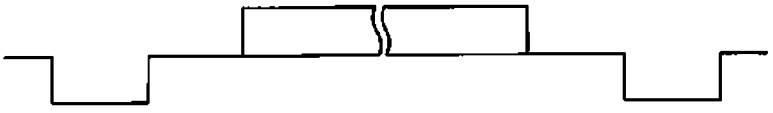
25×40 MODE TO BE USED WITH UHF MODULATOR

(OCTOPUS MONITORS WILL NOT SYNC IN THIS MODE)

VIDEO TIMING-29x132 MODE

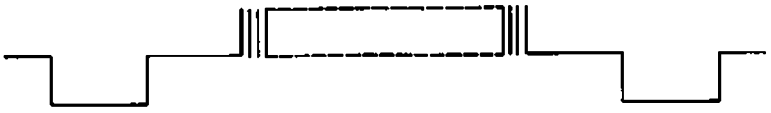
ESC.m.4
ESC.m.5

HORIZONTAL TIMING



hsync	hbp	hdisp	hfp	TOTAL
10	19	133	6	168 CHARS
3·4	6·48	45·34	2·046	57·272 µs

VERTICAL TIMING



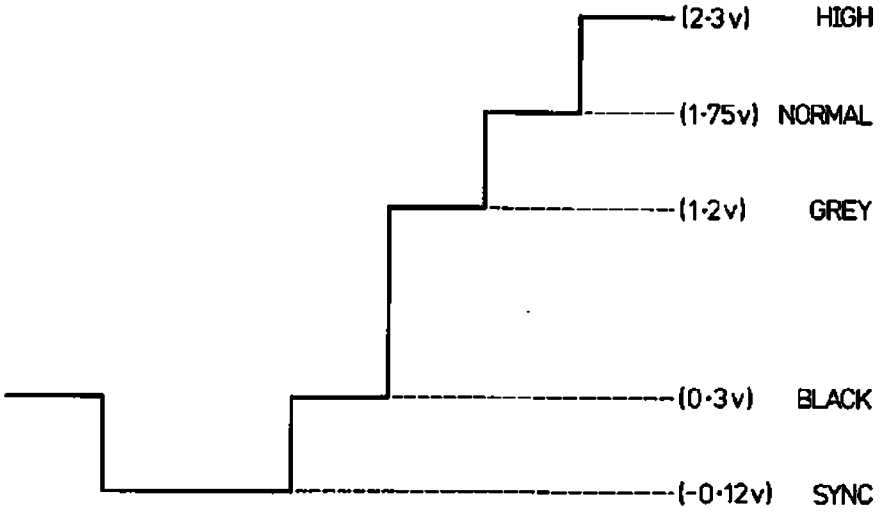
vsync	vbp	vdisp	vfp	TOTAL
5	20	319	4	348 LINES
0·286	1·145	18·270	0·229	19·93 ms

TO BE USED WITH OCTOPUS MONITORS

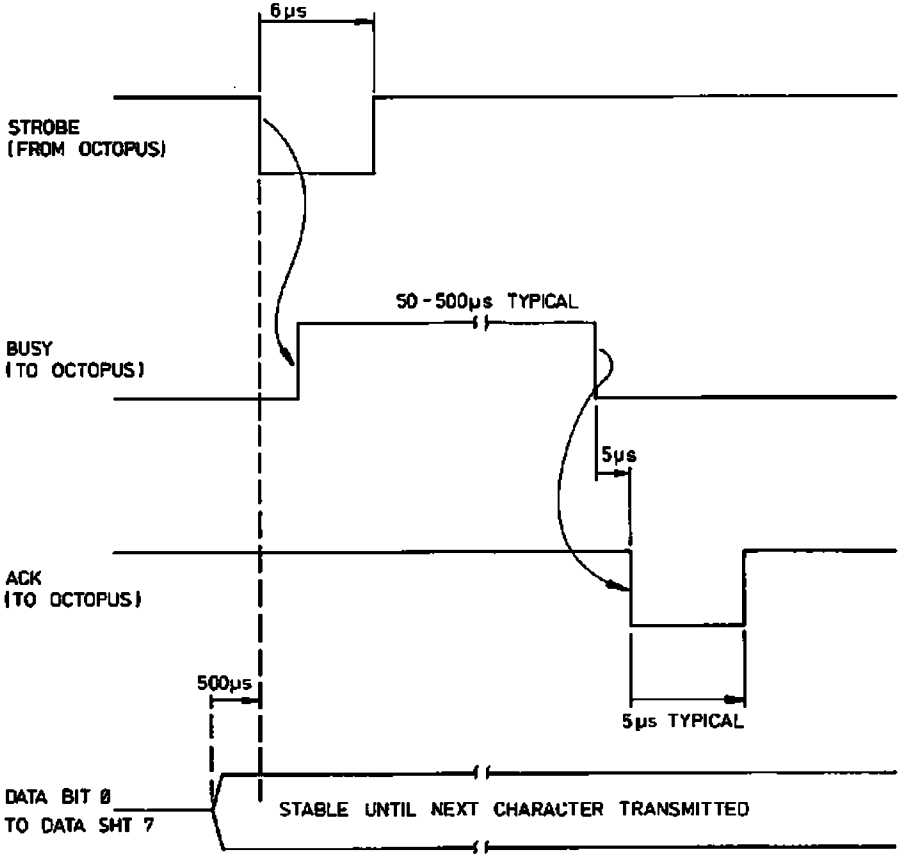
(CONVENTIONAL TVS WILL NOT SYNC)

COMPOSITE VIDEO LEVELS

VOLTAGES INTO A 75^ LOAD



CENTRONICS INTERFACE TIMING



8088 INPUT/OUTPUT

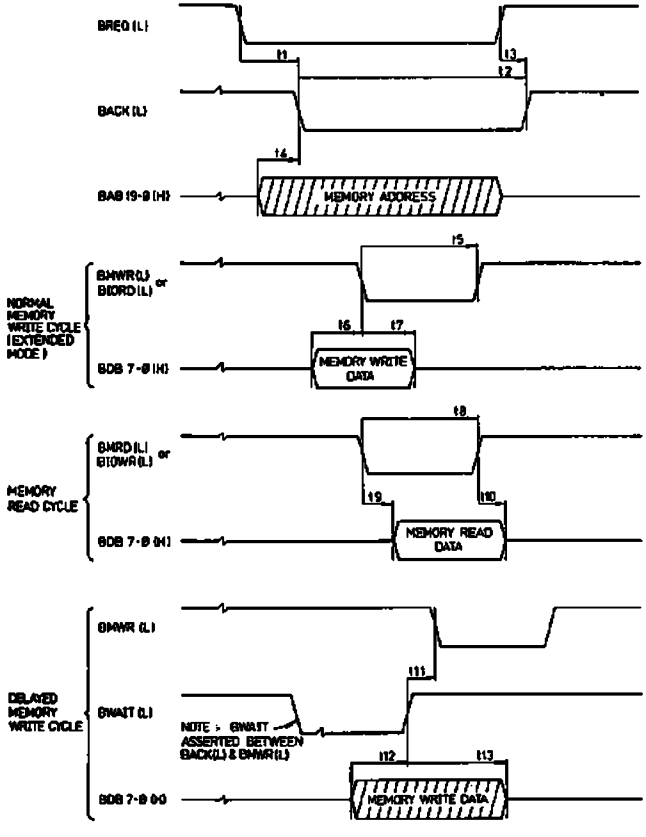
BUS WAIT TIMING

BUS RESET TIMING

REMARKS	REF	8088 CYCLE	
		MIN	MAX
	t1	86	268
	t2	0	63
	t3	0	20
	t4	81	238
	t5	6	53
	t6	0	95
	t7	6	53
	t8	6	53
	t10	15	-
	t11	55	-
	t12	55	-
	t13	0	150
	treset	450ms	650ms

* ALL TIMING IN NANO-SECONDS UNLESS OTHERWISE STATED

DMA READ AND WRITE CYCLES



REMARKS	REF	8088 CYCLE	
		MIN	MAX
	t1	130	190
	t2	610	880
	t3	450	-
	t4	0	190
	t5	150	300
	t6	70	-
	t7	69	-
	t8	450	550
	t9	-	195
	t10	-	110
	t11	-	142
	t12	20	-
	t13	69	-

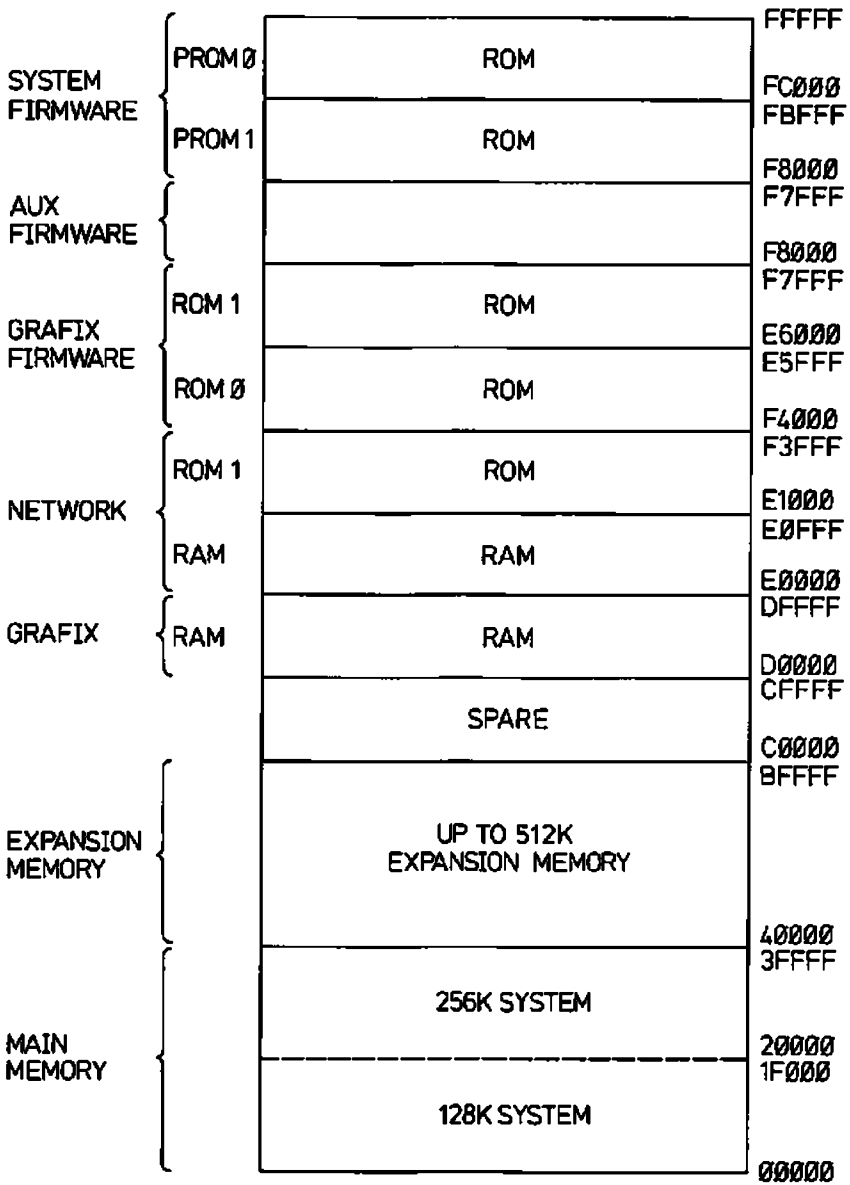
* ALL TIMING IN NANO-SECONDS. UNLESS OTHERWISE STATED

8088 AND Z80 MEMORY ACCESS CYCLES

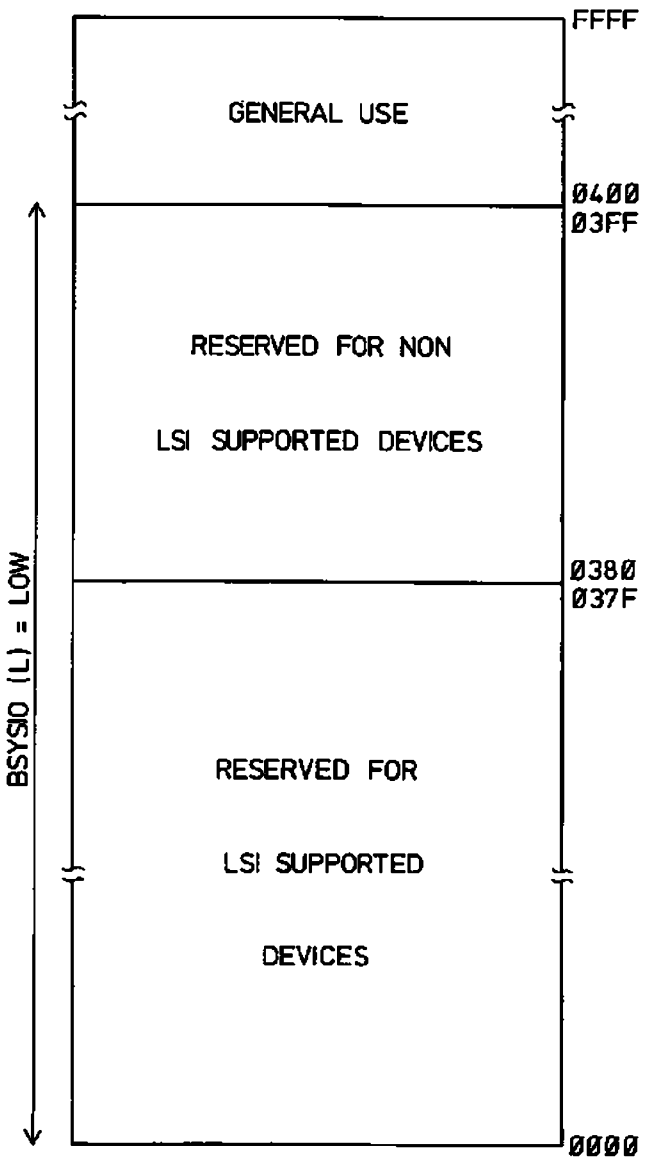
REMARKS	REF	8088 CYCLE		Z80 CYCLE	
		MIN	MAX	MIN	MAX
	t1	68	81	79	86
	t2	44	57	80	87
	t3	125	125	166	166
	t4	14	48	-	-
	t5	58	95	-	-
	t6	9	75	-	-
	t7	23	151	-	-
	t8	59	175	37	180
	t9	-	241	184	-
	t10	22	146	-	73
	t11	200	320	256	389
	t12	94	-	104	-
	t13	0	-	0	-
	t14	20	100	-	-
	t15	190	239	135	292
	t16	20	121	30	-
	t17	0	-	20	-
	t18	48	121	84	241

* ALL TIMING IN NANO-SECONDS, UNLESS OTHERWISE STATED

MEMORY ADDRESS MAP



I/O ADDRESS MAP



Main logic board

DMA controller (i8237-2). Device 0 (00-FF)

DMASEL0_BADD0	000h	Ch 0 Cascaded - base & current addr(wr)
DMASEL0_CADD0	000h	Ch 0 Cascaded - current address (rd)
DMASEL0_CCNT0	001h	Ch 0 Cascaded - current word count (rd)
DMASEL0_BCNT0	001h	Ch 0 Cascaded - base & current word count (wr)
DMASEL0_BADD1	002h	Ch 1 Hard dsk - base & current addr(wr)
DMASEL0_CADD1	002h	Ch 1 Hard dsk - current address (rd)
DMASEL0_CCNT1	003h	Ch 1 Hard dsk - current word count (rd)
DMASEL0_BCNT1	003h	Ch 1 Hard dsk - base & current word count (wr)
DMASEL0_BADD2	004h	Ch 2 Ram rfsh - base & current addr(wr)
DMASEL0_CADD2	004h	Ch 2 Ram rfsh - current address (rd)
DMASEL0_CCNT2	005h	Ch 2 Ram rfsh - current word count (rd)
DMASEL0_BCNT2	005h	Ch 2 Ram rfsh - base & current word count (wr)
DMASEL0_BADD3	006h	Ch 3 Cascaded - base & current addr(wr)
DMASEL0_CADD3	006h	Ch 3 Cascaded - current address (rd)
DMASEL0_CCNT3	007h	Ch 3 Cascaded - current word count (rd)
DMASEL0_BCNT3	007h	Ch 3 Cascaded - base & current word count (wr)
DMASEL0_STAT	008h	Status register (rd)
DMASEL0_CMD	008h	Command register (wr)
DMASEL0_REQ	009h	Request register (wr)
DMASEL0_MASK_SR	00Ah	Single mask register bit (wr)
DMASEL0_MODE	00Bh	Mode register (wr)
DMASEL0_FLOP	00Ch	Clear byte pointer flip/flop (wr)
DMASEL0_TEMP	00Dh	Temporary register (rd)
DMASEL0_CLR	00Dh	Master clear (wr)
DMASEL0_MASK	00Fh	All mask register bits (wr)

DMA controller (i8237-2). Device 1 (10-1F)

DMASEL1_BADD0	010h	Ch 0, Cascaded - base & current address (write)
DMASEL1_CADD0	010h	Ch 0, Cascaded - current address (read)
DMASEL1_BCNT0	011h	Ch 0, Cascaded - base & current word count (write)
DMASEL1_CCNT0	011h	Ch 0, Cascaded - current word count (read)
DMASEL1_BADD1	012h	Ch 1, Flop dsk - base & current address (write)
DMASEL1_CADD1	012h	Ch 1, Flop dsk - current address (read)
DMASEL1_BCNT1	013h	Ch 1, Flop dsk - base & current word count (write)
DMASEL1_CCNT1	013h	Ch 1, Flop dsk - current word count (read)
DMASEL1_BADD2	014h	Ch 2, Cascaded - base & current address (write)
DMASEL1_CADD2	014h	Ch 2, Cascaded - current address (read)
DMASEL1_BCNT2	015h	Ch 2, Cascaded - base & current word count (write)
DMASEL1_CCNT2	015h	Ch 2, Cascaded - current word count (read)
DMASEL1_BADD3	016h	Ch 3, Cascaded - base & current address (write)
DMASEL1_CADD3	016h	Ch 3, Cascaded - current address (read)
DMASEL1_BCNT3	017h	Ch 3, Cascaded - base & current word count (write)
DMASEL1_CCNT3	017h	Ch 3, Cascaded - current word count (read)
DMASEL1_STAT	018h	Status register (read)
DMASEL1_CMD	018h	Command register (write)
DMASEL1_REQ	019h	Request register (write)
DMASEL1_MASK_SR	01Ah	Single mask register bit (write)
DMASEL1_MODE	01Bh	Mode register (write)
DMASEL1_FLOP	01Ch	Clear byte pointer flip/flop (write)
DMASEL1_TEMP	01Dh	Temporary register (read)
DMASEL1_CLR	01Dh	Master clear (write)
DMASEL1_MASK	01Fh	All mask register bits (write)

System Control

SYSSEL	020h - System type switch (read)
Z80NMI	020h - Z80 NMI (write)
PFRESET	021h - Parity fail reset (write)
SLCTOUT	021h - SLCTOUT from parallel interface (bit 5, read)
PFAIL	021h - Parity fail (read) - bit 6 - option board - bit 7 - main board
Z80SEL	028h - Z80 enable (write)

Bank select registers (30-3F)

DMAXTNO_HARD	031h - Hard disk bank
DMAXTNO_FLOP	032h - Floppy bank
DMAXTNO_RAM	033h - Ram refresh / Z80 bank
Z80_BANK	DMAXTNO_RAM

Video interface (*0-*F)

For FPLA pre-Revision 3 (PROM Ver. 13) * = 4
For FPLA Rev 3 (PROM Ver. 13) and onwards * = c

CRTSEL_INT	0*0h - Interrupt control (read)
CRTSEL_INIT	0*0h - Initialization (write)
CRTSEL_STAT	0*1h - Status (read)
CRTSEL_CMD	0*1h - Command (write)
CRTSEL_1SSL	0*2h - Screen start 1 - lower
CRTSEL_1SSU	0*3h - Screen start 1 - upper
CRTSEL_CURL	0*4h - Cursor - lower
CRTSEL_CURU	0*5h - Cursor - upper
CRTSEL_2SSL	0*6h - Screen start 2 - lower
CRTSEL_2SSU	0*7h - Screen start 2 - upper
CRTSEL_VID	0*8h - Video control
CRTSEL_CDAT	0*9h - Character data
CRTSEL_ADAT	0*Ah - Attribute data
CRTSEL_UHF	0*Fh - Mode control (read)

Keyboard (i8251A) (50-53)

KYBDSEL_DATA 050h - Keyboard data in
KYBDSEL_CNTL 051h - Keyboard control
KYBDSEL_STAT 051h - Keyboard status

Floppy disk interface 1 (1793) (*0-*3)

For FPLA pre-Revision 3 (PROM Ver. 13) * = 6
For FPLA Rev 3 (PROM Ver. 13) and onwards * = d

FDCSEL0_CMD 0*0h - Command port
FDCSEL0_STAT 0*0h - Status register
FDCSEL0_TRK 0*1h - Track register
FDCSEL0_SEC 0*2h - Sector register
FDCSEL0_DATA 0*3h - Data register

Hard disk adapter 1 (70-73)

HDKSEL0_STAT 071h - SASI status
HDKSEL0_CMD 071h - Controller select
HDKSEL0_RESET 072h - Clear select/controller
HDKSEL0_DATA 073h - Data i/o

Baud rate generators (on board) (i8253 counter/timer 1)(80-83)

BAUDSEL0_CT0 080h - Dart 1 Ch A baud rate
BAUDSEL0_CT1 081h - Dart 1 Ch B baud rate
BAUDSEL0_CT2 082h - Sound Transducer Freq. gen.
BAUDSEL0_CTL 083h - Control

SIO 1 (on board) (Z80 SIO) (RS232) (A0-A3)

DARTSEL0_ADATA 0A0h - Dart 1, channel A, data
DARTSEL0_ASTAT 0A1h - Dart 1, channel A, status
DARTSEL0_ACNTL 0A1h - Dart 1, channel A, control
DARTSEL0_BDATA 0A2h - Dart 1, channel B, data
DARTSEL0_BSTAT 0A3h - Dart 1, channel B, status
DARTSEL0_BCNTL 0A3h - Dart 1, channel B, control

Programmable interrupt controller (Master) (i8259A) (B0-B3)

```
INTSEL0_STAT    0B0h - Interrupt request/serviced status
INTSEL0_ICW     0B0h - Initialization control words
INTSEL0_IMR     0B1h - Interrupt level masks
INTSEL0_OCW     0B1h - Operational control words
```

Programmable interrupt controller (Slave) (i8259A) (B4-B7)

```
INTSEL1_STAT    0B4h - Interrupt request/serviced status
INTSEL1_ICW     0B4h - Initialization control words
INTSEL1_IMR     0B5h - Interrupt level masks
INTSEL1_OCW     0B5h - Operational control words
```

Pseudo Z80 Interrupt Acknowledge (E0-EF)

```
VECSEL0         0E0h - Vector input for RS232 devices (E0-E3)
VECSEL1         0E4h - Vector input for RS422 device  (E4-E7)
```

Parallel I/O interface (F0-F3)

```
PTRSEL_DATA     0F0h - Data i/o
PTRSEL_CNTL     0F1h - Control
                     - bit 0
                     - bit 1
                     - bit 2
                     - bit 3
                     - bit 4
                     - bit 5

PTRSEL_STAT     0F1h - Status
                     - bit 0
```

Real time clock control (i8255A) (F8-FF)

```
RTCSEL_DAAD     0F8h - Data i/o and Address
RTCSEL_CNTL     0F9h - RTC control + FDC control
                     - bit 4 - write precomp. bit 0
                     - bit 5 -   "      "     bit 1
                     - bit 6 - drive select 0
                     - bit 7 - drive select 1
RTCSEL_GPO      0FAh - General purpose outputs (3 bits)
                     - bit 2 - side select (0 = side 0)
                     - bit 1 - parallel data i/o (0 = out)
                     - bit 0 - parallel control i/o (0 = out)
RTCSEL_MODE     0FBh - Mode control
```

RESERVED OPTION BOARD ADDRESSES

DMA controller (i8237-2).Device 2 (Disk option board)(100-10F)

DKDMA_BADD0	100h - Ch 0 Streamer - base & current addr(wr)
DKDMA_CADD0	100h - Ch 0 Streamer - current address (rd)
DKDMA_CCNT0	101h - Ch 0 Streamer - current word count (rd)
DKDMA_BCNT0	101h - Ch 0 Streamer - base & current word count (wr)
DKDMA_BADD1	102h - Ch 1 Flop dsk - base & current addr(wr)
DKDMA_CADD1	102h - Ch 1 Flop dsk - current address (rd)
DKDMA_CCNT1	103h - Ch 1 Flop dsk - current word count (rd)
DKDMA_BCNT1	103h - Ch 1 Flop dsk - base & current word count (wr)
DKDMA_BADD2	104h - Ch 2 Not used - base & current addr(wr)
DKDMA_CADD2	104h - Ch 2 Not used - current address (rd)
DKDMA_CCNT2	105h - Ch 2 Not used - current word count (rd)
DKDMA_BCNT2	105h - Ch 2 Not used - base & current word count (wr)
DKDMA_BADD3	106h - Ch 3 Not used - base & current addr(wr)
DKDMA_CADD3	106h - Ch 3 Not used - current address (rd)
DKDMA_CCNT3	107h - Ch 3 Not used - current word count (rd)
DKDMA_BCNT3	107h - Ch 3 Not used - base & current word count(wr)
DKDMA_STAT	108h - Status register (read)
DKDMA_CMD	108h - Command register (write)
DKDMA_REQ	109h - Request register (write)
DKDMA_MASK_SR	10Ah - Single mask register bit (write)
DKDMA_MODE	10Bh - Mode register (write)
DKDMA_FLOP	10Ch - Clear byte pointer flip/flop (write)
DKDMA_TEMP	10Dh - Temporary register (read)
DKDMA_CLR	10Dh - Master clear (write)
DKDMA_MASK	10Fh - All mask register bits (write)

DMA controller (i8237-2).Device 3(Comms option board)(110-11F)

COMMDMA_BADD0	110h - Ch 0 RS422 Rx Ch A -	base & current address (write)
COMMDMA_CADD0	110h - Ch 0 RS422 Rx Ch A -	current address (read)
COMMDMA_BCNT0	111h - Ch 0 RS422 Rx Ch A -	base & current word count (write)
COMMDMA_CCNT0	111h - Ch 0 RS422 Rx Ch A -	current word count (read)
COMMDMA_BADD1	112h - Ch 1 RS422 Tx Ch B -	base & current address (write)
COMMDMA_CADD1	112h - Ch 1 RS422 Tx Ch B -	current address (read)
COMMDMA_BCNT1	113h - Ch 1 RS422 Tx Ch B -	base & current word count (write)
COMMDMA_CCNT1	113h - Ch 1 RS422 Tx Ch B -	current word count (read)
COMMDMA_BADD2	114h - Ch 2 RS232 Rx Ch A -	base & current address (write)
COMMDMA_CADD2	114h - Ch 2 RS232 Rx Ch A -	current address (read)
COMMDMA_BCNT2	115h - Ch 2 RS232 Rx Ch A -	base & current word count (write)
COMMDMA_CCNT2	115h - Ch 2 RS232 Rx Ch A -	current word count (read)
COMMDMA_BADD3	116h - Ch 3 RS232 Tx Ch B -	base & current address (write)
COMMDMA_CADD3	116h - Ch 3 RS232 Tx Ch B -	current address (read)
COMMDMA_BCNT3	117h - Ch 3 RS232 Tx Ch B -	base & current word count (write)
COMMDMA_CCNT3	117h - Ch 3 RS232 Tx Ch B -	current word count

```
COMMDMA_STAT     118h - Status register (read)
COMMDMA_CMD      118h - Command register (write)
COMMDMA_REQ      119h - Request register (write)
COMMDMA_MASK_SR  11Ah - Single mask register bit (write)
COMMDMA_MODE     11Bh - Mode register (write)
COMMDMA_FLOP     11Ch - Clear byte pointer flip/flop (write)
COMMDMA_TEMP     11Dh - Temporary register (read)
COMMDMA_CLR      11Dh - Master clear (write)
COMMDMA_MASK     11Fh - All mask register bits (write)
```

DMA extension addresses

```
DMAXTN1_QIC      130h - Streamer DMA extension
DMAXTN1_FLOP     131h - Floppy DMA extension
DMAXTN2_RX42     134h - RS422 Rx DMA extension address
DMAXTN2_TX42     135h - RS422 Tx DMA extension address
DMAXTN2_RX32     136h - RS232 Rx DMA extension address
DMAXTN2_TX32     137h - RS232 Tx DMA extension address
                 138-13B reserved for DMAXTN3
```

Graphics interface (140-14F)

GFXSEL_INIT	140h - Initialization (write)
GFXSEL_STAT	141h - Status (read)
GFXSEL_CMD	141h - Command (write)
GFXSEL_1SSL	142h - Screen start 1 - lower
GFXSEL_1SSU	143h - Screen start 1 - upper
GFXSEL_CURL	144h - Cursor - lower
GFXSEL_CURU	145h - Cursor - upper
GFXSEL_2SSL	146h - Screen start 2 - lower
GFXSEL_2SSU	147h - Screen start 2 - upper
GFXSEL_CNTL	148h - Graphics control
GFXSEL_PALA	149h - Palette access
	14Dh-14Fh reserved

Streamer adapter (170-173)

QIC_DATA	170h - QIC data port
QIC_CTL	171h - QIC control port
QIC_CMD	172h - QIC command port

Baud rate generators
(Comms option board)(i8253 counter/timer 2)(180-183)

BAUDSEL1_CT0	180h - Dart 2 Ch A baud rate
BAUDSEL1_CT1	181h - Dart 2 Ch B baud rate
BAUDSEL1_CT2	182h - Sio 2 (RS422) baud rate
BAUDSEL1_CTL	183h - Control

Baud rate generators
(Comms option board)(i8253 counter/timer 3)(184-187)

BAUDSEL2_CT0	184h	- SIO 1 (RS232) Ch A Tx baud rate
BAUDSEL2_CT1	185h	- SIO 1 (RS232) Ch A Rx baud rate
BAUDSEL2_CT2	186h	- SIO 1 (RS232) Ch B Tx/Rx baud rate
BAUDSEL2_CTL	187h	- Control

CLKSEL	188h	- Baud rate clock Int/Ext select (0 = internal, 1 = external) - bit 0 - RS232 port 5 Tx clock (184) - bit 1 - RS232 port 5 Rx clock (185) - bit 2 - RS232 port 6 Tx clock (186) - bit 3 - RS232 port 6 Rx clock (186)

DART 2 (Comms option board) (Z80 DART) (RS232) (1A0-1A3)

DARTSEL1_ADATA	1A0h	- Dart 2, channel A, data
DARTSEL1_ASTAT	1A1h	- Dart 2, channel A, status
DARTSEL1_ACNTL	1A1h	- Dart 2, channel A, control
DARTSEL1_BDATA	1A2h	- Dart 2, channel B, data
DARTSEL1_BSTAT	1A3h	- Dart 2, channel B, status
DARTSEL1_BCNTL	1A3h	- Dart 2, channel B, control

SIO 1 (Comms option board) (Z80 SIO) (RS232) (1A4-1A7)

SIOSEL0_ADATA	1A4h	- SIO 1, channel A, data
SIOSEL0_ASTAT	1A5h	- SIO 1, channel A, status
SIOSEL0_ACNTL	1A5h	- SIO 1, channel A, control
SIOSEL0_BDATA	1A6h	- SIO 1, channel B, data
SIOSEL0_BSTAT	1A7h	- SIO 1, channel B, status
SIOSEL0_BCNTL	1A7h	- SIO 1, channel B, control

SIO 2 (Comms option board) (Z80 SIO) (RS422) (1A8-1AB)

SIOSEL1_ADATA	1A8h	- SIO 2, channel A, Rx data
SIOSEL1_ASTAT	1A9h	- SIO 2, channel A, status
SIOSEL1_ACNTL	1A9h	- SIO 2, channel A, control
SIOSEL1_BDATA	1AAh	- SIO 2, channel B, Tx data
SIOSEL1_BSTAT	1ABh	- SIO 2, channel B, status
SIOSEL1_BCNTL	1ABh	- SIO 2, channel B, control

ARCNET controller (Network option board 1D0-1D7)

NETSEL_INTM	1D0h	- Write interrupt mask (write)
NETSEL_STAT	1D0h	- Read status register (read)
NETSEL_CMD	1D1h	- Command register (write)
NETSEL_CTL		- Control register (write)

AC Input: Changed by link
90 - 140 VAC
180 - 265 VAC
47 - 63 Hz
Input Current: Limited to 20A max.

DC Outputs:	Op 1	Op 2	Op 3	Op 4
Voltage:				
Min	4.95	10.8	10.8	-10.8
Typ	5.00(1)	12.0	12.0	-12.0
Max	5.10	13.2	13.2	-13.2
Current:				
Min	2.50	0.0	0.0	0.0
Max(2)	12.00	3.5	2.5	0.5
Load Regulation:				
20% - 100% FL	+2%	+5%	+5%	+2%
Rippe & Noise:				
Max	50mV	100mV	100mV	100mV

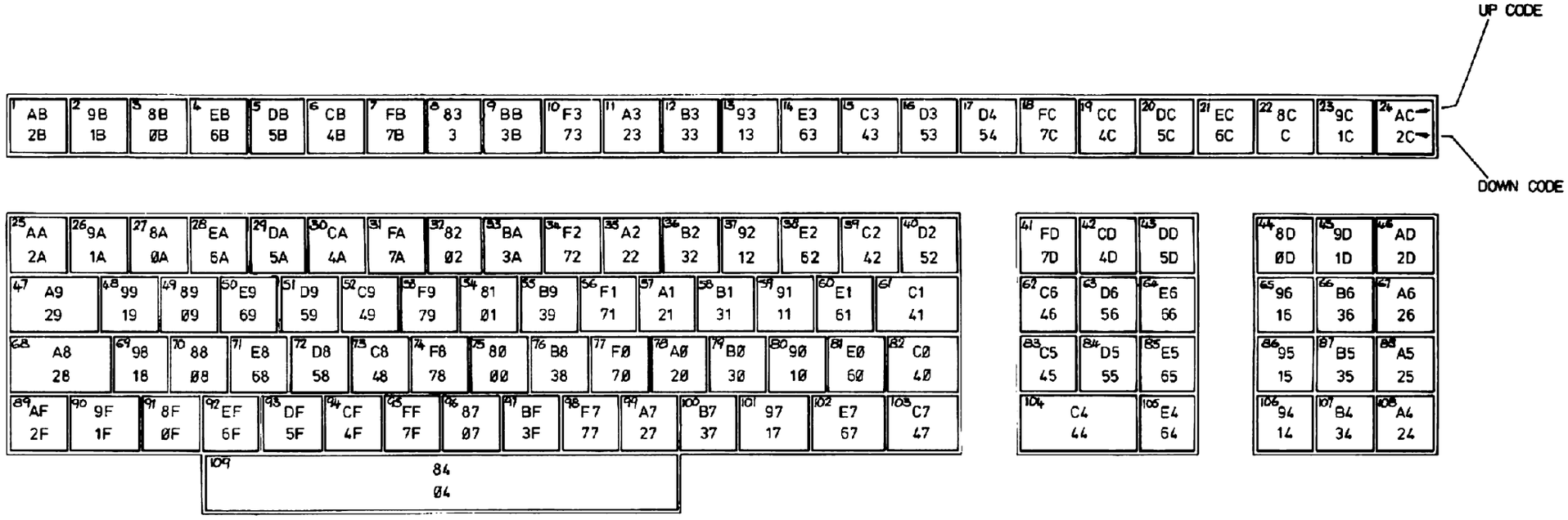
Note. (1) Output adjustment of 5V rail 4.8V to 6.2V minimum
(2) Max' total power not to exceed 125 Watts

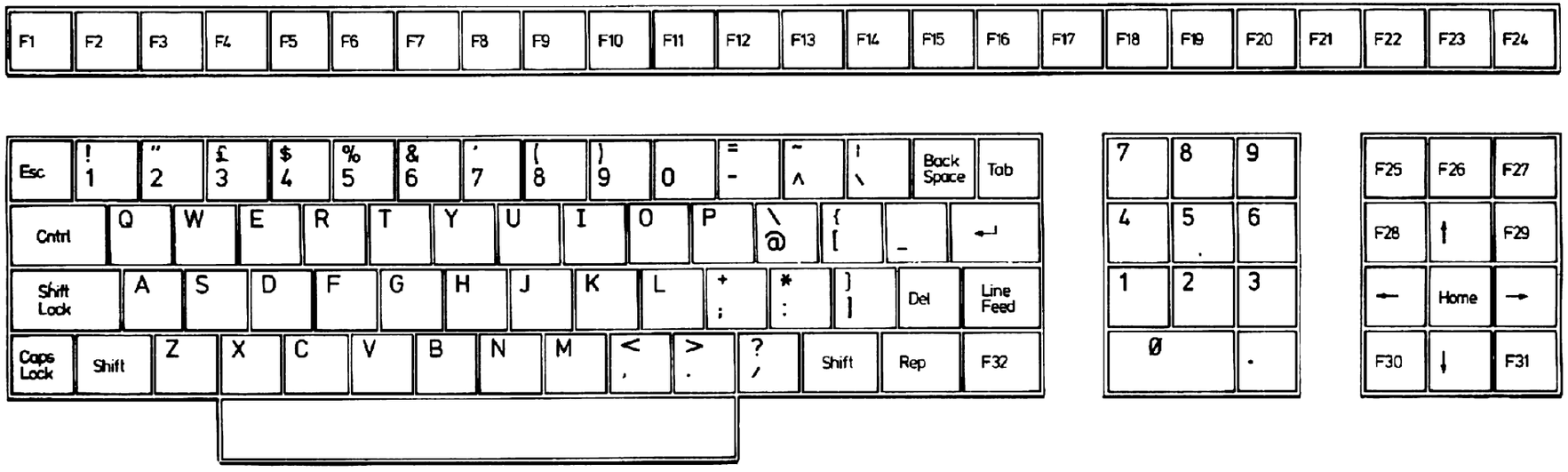
Line Regulation: Better than +0.1% on all outputs over full input votage range

Hold up time: 20 mS
Isolation: 4kV peak input to output
Overvoltage: Protection to 6.25V +0.75V on 5V rail

Connections on barrier terminal:

AC Input:	P1	Live	
	P2	Neutral	
	P3	Earth	
DC Outputs:	P4	Op 3	+12V (1.5A)
	P5	Op 2	
	P6	Op 4	+12V (5.0A)
	P7,8,9	Common	
	P10,11	Op 1	+5V


UP CODE
DOWN CODE
1 AB 2B
2 9B 1B
3 8B 0B
4 EB 6B
5 DB 5B
6 CB 4B
7 FB 7B
8 83 3
9 BB 3B
10 F3 73
11 A3 23
12 B3 33
13 93 13
14 E3 63
15 C3 43
16 D3 53
17 D4 54
18 FC 7C
19 CC 4C
20 DC 5C
21 EC 6C
22 8C C
23 9C 1C
24 AC 2C
25 AA 2A
26 9A 1A
27 8A 0A
28 EA 6A
29 DA 5A
30 CA 4A
31 FA 7A
32 82 02
33 BA 3A
34 F2 72
35 A2 22
36 B2 32
37 92 12
38 E2 62
39 C2 42
40 D2 52
41 FD 7D
42 CD 4D
43 DD 5D
44 8D 0D
45 9D 1D
46 AD 2D
47 A9 29
48 99 19
49 89 09
50 E9 69
51 D9 59
52 C9 49
53 F9 79
54 81 01
55 B9 39
56 F1 71
57 A1 21
58 B1 31
59 91 11
60 E1 61
61 C1 41
62 C6 46
63 D6 56
64 E6 66
65 96 16
66 B6 36
67 A6 26
68 A8 28
69 98 18
70 88 08
71 E8 68
72 D8 58
73 C8 48
74 F8 78
75 80 00
76 B8 38
77 F0 70
78 A0 20
79 B0 30
80 90 10
81 E0 60
82 C0 40
83 C5 45
84 D5 55
85 E5 65
86 95 15
87 B5 35
88 A5 25
89 AF 2F
90 9F 1F
91 8F 0F
92 EF 6F
93 DF 5F
94 CF 4F
95 FF 7F
96 87 07
97 BF 3F
98 F7 77
99 A7 27
100 B7 37
101 97 17
102 E7 67
103 C7 47
104 C4 44
105 E4 64
106 94 14
107 B4 34
108 A4 24
109 84 04


F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 F13 F14 F15 F16 F17 F18 F19 F20 F21 F22 F23 F24
Esc ! 1 " 2 £ 3 $ 4 % 5 & 6 ' 7 (8) 9 0 = - ~ ^ | \ Back Space Tab
Cntrl Q W E R T Y U I O P \ @ { [_ ←
Shift Lock A S D F G H J K L + ; * : }] Del Line Feed
Caps Lock Shift Z X C V B N M < , > . ? / Shift Rep F32
7 8 9
4 5 6
1 2 3
Ø .
F25 F26 F27
F28 ↑ F29
← Home →
F30 ↓ F31

MECHANICAL

- size 400mm (depth) x 450mm (width) x 145mm (height)
- weight approx 7kg depending on configuration

ENVIRONMENTAL

- storage:

  -30 C to +70 C
  0% to 90% relative humidity

- operational:

  0 C to +40 C
  15% to 70% relative humidity
  (non condensing)

ELECTRICAL

- 210-250V AC; 45-65Hz
- Optional 100-110V AC; 45-65Hz
- 170 watts max (full options) 100 watts typically
- Fan cooled. Less than 31 dB noise at 50Hz
- 3 pin mains inlet
- 3 pin 2A mains outlet (unfused) for monitor
- switching power supply

OCTOPUS DISK FORMAT

LSI 48 tpi 5" format (Octopus, M-Four, CalPC)

3120	:	128 byte record capacity
390	:	Kilobyte drive capacity
128	:	32 byte directory entries
128	:	checked directory entries
256	:	records/directory entry
16	:	records/block
5	:	sectors/track
2	:	reserved tracks

From the above you can see the block size is 2K and there are 2 directory blocks.

Here are the physical parameters:

40	:	cylinders
2	:	sided
48	:	tpi
1024	:	byte sectors
5	:	sectors/track

There is an index mark on each track.
The data is non-inverted.
The first sector is numbered 1.
The access mode is cylinder (i.e. the disk is filled cylinder by cylinder, rather than up one side then down the other).
There is no physical or logical interleave.

96 tpi format is identical but has 256 directory entries since there are now more than 256 blocks on the disk (16 bit block numbers).

COPSE ROAD
ST. JOHN'S
WOKING
SURREY GU21 1SX

TELEPHONE: WOKING (04862) 23411
TELEX: 859592

VAT REG. No. 358 7042 36
BANK: NATIONAL WESTMINSTER
1 HIGH STREET
WOKING, SURREY
CODE 60 24 20
ACCOUNT No. 1827188

L.S.I. Computer Ancilliaries Ltd.

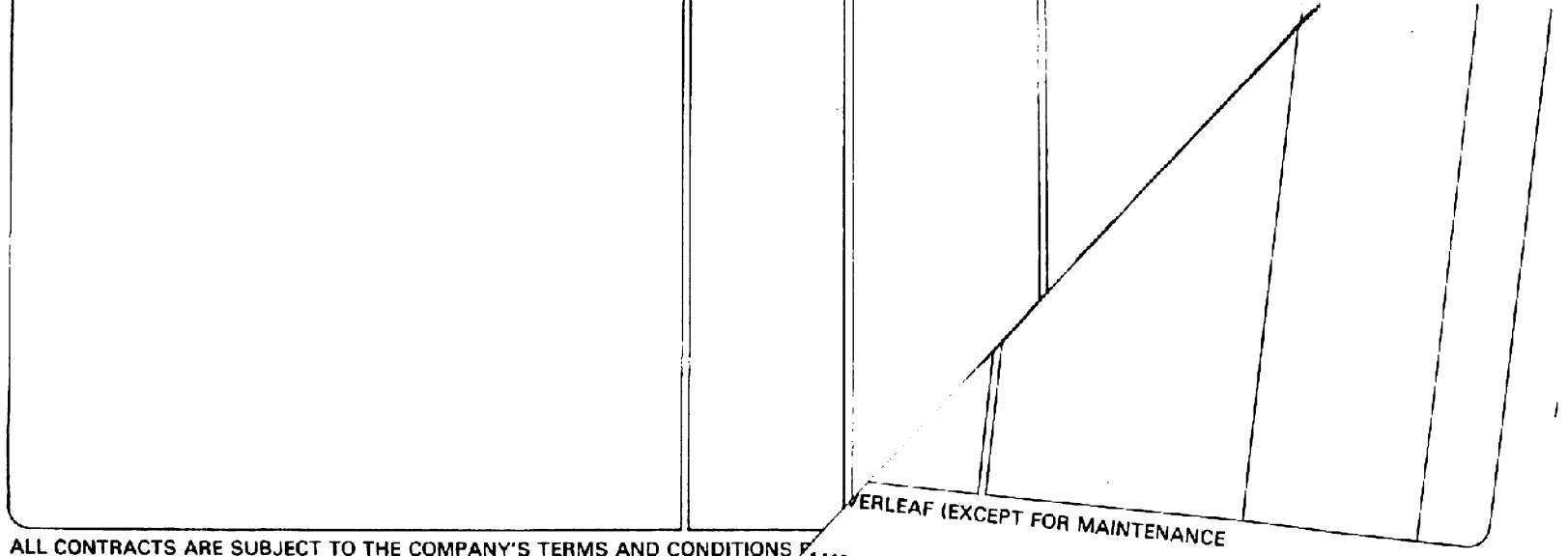
DELIVERY NOTE

NUMBER 18860

INVOICE TO CUSTOMER 4087 REFERENCE 80884A 1508 TAX POINT 12-03-85

HORLEY DIGITAL DATA SYSTEMS LT
4 THE CORONET
HORLEY
SURREY
02934-6375

PRODUCT DESCRIPTION REF.	QTY				
901100 OCT BROCHURES SMALL	5 EA				
901099 OCT BROCHURES LARGE	5 EA				
901133 OCT TECH MANUAL	1 EA	—Due	3/3/85		



VERLEAF (EXCEPT FOR MAINTENANCE

ALL CONTRACTS ARE SUBJECT TO THE COMPANY'S TERMS AND CONDITIONS F
CONTRACTS AND SOFTWARE PACKAGE RENTAL CONTRACTS).

1412251 ENGLAND

REGISTERED OFFICE: SHERWOOD HOUSE, COPSE ROAD, WOKING, SURREY GU2