Greenbank Electronics

Document Ref: XMMZC1

Issue: 0

Date: October 1988

MZC-1 INTERAK HIGH PERFORMANCE CPU CARD

USER MANUAL - ISSUE 0

"THE Z80280 DOSSIER"

Copyright Note:

No unauthorised copies of this
Manual may be made.

c 1988 Greenbank Electronics

Manual Price £5.00

PREFACE

This "Issue 0" Document is the current issue of a collection of my own notes and designs for an Interak Card using the Zilog Z80280.

The document was formerly known as "The Z80280 Dossier", but as the dream approaches reality of Interak continuing in development to become (in its ultimate form) a system with a fast CPU addressing a 16 Megabyte address space, high resolution graphics etc, I thought it was about time I started gathering the information in a form more nearly approaching the form it will take in a final manual.

We are all hoping that the price of the Z80280 will drop, but at the moment it is a fairly expensive chip (ie half the price of the 2.5 MHz Z80-CPU when we first met it!)

Therefore it is only the few far sighted Interakers who have such a chip - we managed to obtain a small quantity of the very first production chips available, and they are mostly now in Interak experimenters' hands.

This document will be made available to this intrepid band, so that they will know what we are up to. I have organised the information in the same way as I do for my other Interak manuals so if you can find your way round them you can find your way round this.

Needless to say it is all new and unchecked - where I did not know a fact, I simply made it up - so any feedback on the contents of this document and the philosophy of the design would be welcome; particularly practical experience, so that when we get to "Issue 1" we will have something that is well tried and tested.

I have now been involved with Interak since my youth (10 years or more ago) and have the benefit of much experience of what is required in an Interak card design. If you have any bright ideas on enhancements or improvements to this design then I shall be glad to hear them, but please do not be offended if I appear to ignore them. As we at Greenbank carry the ultimate responsibility for my successes and failures we have do do things "our way"; it may look to any Interak individual that we do not listen to him, but we listen to everybody, and the finished product becomes the result of an aggregate opinion from everybody, with myself having the casting vote!

David M Parkins
October 1988

CONTENTS

*** NOTE, ISSUE 0 MANUAL: MANY OF THE SECTIONS BELOW MAY BE ABSENT - THEY HAVE NOT BEEN WRITTEN YET ***

(CONTENTS LIST CONTINUED ON NEXT PAGE)

(Total number of pages Issue 0 Manual: 81)

ERRATA

Issue 0.0 October 1988

page 2-4 "expence" should be "expense"

page 2-16 wishing to (Paragraph Forming)

page 2-17 DIAGRAMS) (bracket missing)

page 2-21 Delete critical mention of ZBUS p 2-21; if our the Z80280 catches on, then ZBUS is well worth considering, as it's made for the job. The point is that we are not going to kill the Z80 off while it still has a use.

page 3-1 Re-word: It probably does matter if socket is wrong way round

(various) page numbers need alteration; text writing and inserting etc.

1-2 "do do" shd be "to do"
1-4 7-1 Out of line
5-4c-1 xx in front of all
p. 1 X MZC-1.

SPECIAL HANDLING REQUIREMENTS
CMOS ETC. DEVICES.

Precautionary Notice

Some of the Integrated Circuits used on this card are supplied packed in special anti-static packing (tubes, foil, or foam), and have a warning notice affixed to the packing.

Do not be alarmed, experience shows that damage due to mis-handling very rarely happens.

The damage is due to static electric charges, being transferred from an object or person through the leads of the integrated circuit to the tiny chip inside. Only some types of IC are vulnerable, e.g. MOS or CMOS types, and some of the latest shallow diffusion high-speed bipolar types.

The initials CMOS stand for "Complementary-Metal-Oxide-Semiconductor". The oxide is, for example silicon dioxide - an excellent insulator, which insulates the metal from the silicon. (The design of the latest high speed CMOS has moved on from the metal gate type used originally, and the more correctly the internal construction of the integrated circuits is based on IGFETs: Insulated-Gate-Field-Effect-Transistors. So far as I know nobody claims to manufacture CIGFET logic circuits, so I'm not going to stick my neck out and call my ics anything but "High Speed CMOS", even though the term is something akin to calling a motor car a "Petrol Engined Horse Drawn Carriage")

However the scale of IC chip manufacture is so small that whatever the construction of the internal insulating layers they are easily damaged by excessive static charges which can "punch through" an insulating layer.

Sometimes the damage is not immediately noticed (which might explain how some people disregard all precautions and appear to get away wiith it), but during the months and years which follow, contaminating chemical "ions" can migrate through the hole and failure can eventualy result. The descriptive American term for these chips is the "walking wounded".

We would liken the risk of causing this damage to the risk of getting caught "speeding" in a motor car. Of course there are people who disregard all speed limits and never get caught, just as there are people who disregard handling precautions with no adverse effects. But do remember, every so often you meet someone who has been disqualified from driving due to being caught speeding, so do not let him handle your ICs!

The full set of handling precautions is given on the page after next; as mentioned, not everybody follows them all to the letter, but at least you cannot say you haven't been warned!

PLCC Package

This board may mark the introduction of a new IC package to some users: the PLCC (Plastic Leaded Chip Carrier). The CPU on the MZC-1 card is a 68-pin device, containing 140,000 transistor elements (compared with the 40,000 of the Z80A-CPU chip which we also use in the Interak system). However better things appear to come in smaller packages: the Z80A-CPU measures about 50 x 15 mm; the Z80280 is about 25 mm x 25 mm, which is a smaller area. Examination of the Z80280 (obeying the handling precautions of course) will cause the stoutest heart to sink for a moment: you will see that the connections are only 50 thousandths of an inch apart, and they are clearly intended for surface mounting techniques (where the IC is glued and soldered to the surface of the board, and if a mistake is made, or a defect shows up you not only throw the IC away you throw the whole board with it!)

Fortunately there is a socket available, with pins on normal 0.1" centres, which will accept the PLCC IC and allow it to be used on conventional boards. It is easy to get the chip into the socket, not so easy to get it out (an expensive extractor tool is necessary; we shall probably sell these to those who want them). The socket is intended only to get users out of the jam the PLCC designers have got them into, you can see by the very small dimensions that the socket design is itself a miracle of engineering, so treat it with the respect it deserves - you do not gaily whack chips in and out every five minutes as you may have become accustomed to in your activities so far.

You will be amused to hear (when you get the bill for your expensive socket, and the expensive extraction tool) that the benefit of the PLCC package is that it is cheaper to manufacture. I'm not sure wh is kidding whom, but like lams to the slaughter, if we want to carry on playing in this field we will have to pay the price; (come back Z80 - all is forgiven!)

This is all I shall have to say about handling the PLCC chips. On the next page is the full list of standard handling precautions for CMOS etc devices (which of course apply to the Z80280 too).

HANDLING PRECAUTIONS TO AVOID DAMAGE TO VULNERABLE INTEGRATED CIRCUITS BY STATIC DISCHARGE

Before unwinding any wire shorting together the pins of the ICs, or removing the ICs from their protecting metal or anti-static carrier tube, container, or anti-static foam, please read the following precautions:

1. Never use an isolated bit ("low leakage") soldering-iron to work on a circuit with the ICs in place. The bit should be earthed. If in any doubt, earth it by clipping on a small crocodile clip connected to earth. Similarly, all test equipment should be earthed before it is connected to a finished circuit.

2. Work on an earthed metal plate about a few feet square, (e.g. a stainless steel kitchen sink, or cooking foil), as a work-bench, when the time comes to install the ICs.

3. Keep all your tools on this earthed metal plate, and connect yourself to it, either by touching, or by using a piece of connecting wire formed as a wrist-strap. (Note: if you are using a wrist-strap, it is considered less hazardous to personal safety if the connection to earth is made via a 1 Megohm resistor.)

4. Before fitting the ICs, earth your circuit board, the IC sockets, and yourself; make sure that the power supply has been turned off and all electrolytic capacitors have been completely discharged.

5. Never leave unprotected ICs on a plastic or other non-conductive surface and never store them in ordinary white polystyrene without protection. (If a conductive tube or similar container is used, it is not possible for a damaging static potential to be built up inside such a container, nor could such a charge normally be introduced to the ICs from outside.)

6. Damage is less likely in humid conditions than dry ones. Try to avoid nylon and similar clothing, seating and carpeting, when working with these chips.

7. Use some form of IC sockets if you possibly can, as once the devices have been soldered, any guarantee which existed becomes void. If it is essential to solder the ICs, the supply pins should be soldered first, in order that the internal protection circuits have the maximum chance to carry out their task.

MZC-1 ADVANCED INTERAK CPU CARD FEATURES

* International Size Card (4.5" x 8", 114 x 203 mm)
* Uses Zilog Z80280 "Super Z80" Microprocessor.
* Provides enhanced performance in Interak systems, without rendering existing cards obsolete.
* On chip memory management to handle 16 Megabyte Addressing range.
* On chip I/O Port space memory management to handle up to 16 Megabyte of I/O Ports (although in the Interak system we choose only to take advantage of a space of 256 ports with an expanded space of 64K or 16 Megabyte reserved for the unlikely event that it might be needed)
* Programmable 10 bit Dynamic Ram Refreshing
* Fully Buffered in and out
* 16-20 MHz crystal input allows operation at high speed (but scaling of bus timing allows all existing and future cards made from "normal" speed peripheral chips, RAMs etc to be accessed without difficulty)
* On board memory decoder provides for memory space for up to 64K of on-board ROM (eg Boot EPROM), with suitable logic to allow the ROM to seize control at address zero at power on yet be mapped high up in memory later.
* On board decoder provides 64K space for existing bus Memory Request line to access existing cards without bus contention, or need to modify the cards.
* (FUTURE) Plated-through holes, Epoxy-glass PCB.
* (FUTURE) Green Solder Resist on both "A" and B sides.
* (FUTURE) Gold-plated edge connector on both A and B sides.
* (FUTURE) Silk screened component identification printing to side B.
* ISBUS-A, ISBUS-B, and INTERAK 1 bus compatible.
* Buffered where necessary to reduce bus loading to 1 load per line.
* No manufacturer's name appears on the card, thus ideal for OEM (Original Equipment Manufacturer) use.

2.1 INTRODUCTION

Introduction to XMZC-1 CPU Card Design

This document describes a design for an additional CPU card to the Interak range.

The card is to be known as **"MZC-1"**, but while it is still in development it will have the prefix "X", ie for now it will be XMZC-1.

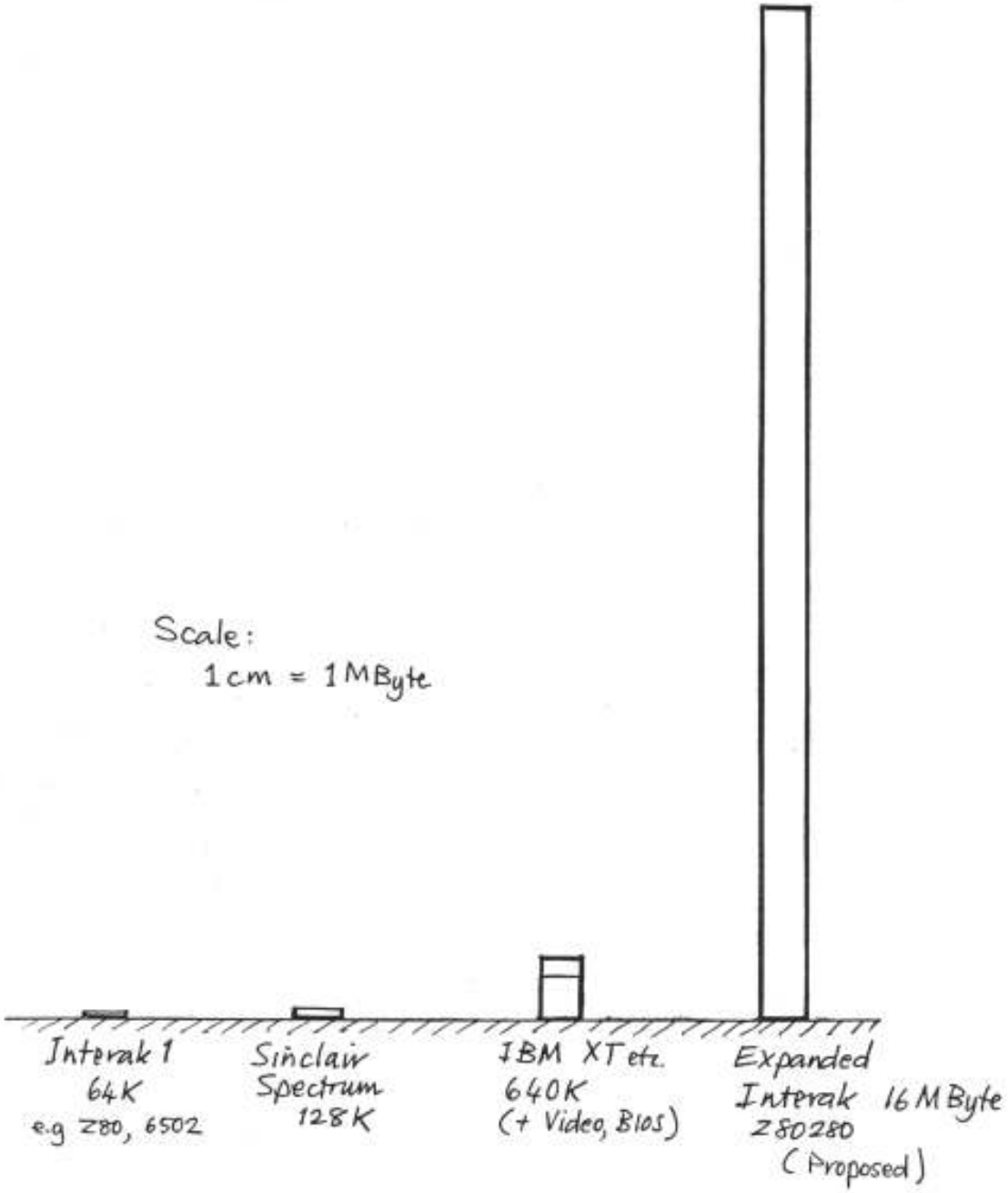
The present design has been arrived at after much careful thought, and experience with how Interak is sold and used. The strength of the Interak system has always been that the system is modular, reliable, easy to construct, easy to understand. The most popular cards have been the simplest - for example a simple RS-232C interface. Less popular has been for example the QS-1 quad serial interface. This latter card is bristling with features and sophistication, options, jumper links, programmable baud rates, interrupt daisy chain lines, card selects, and so on. As a result it has ended up one of the more complex cards and one of the more expensive (although half the price of competing products from RS Components, Farnell etc). However it is too sophisticated, and thus too expensive for the typical Interak user.

The appropriate phrase is that the product has been "over engineered"; our motto should be "keep it simple"!

I shall not burden the reader with the initial design I drew up for the XMZC card. It had 40 or more chips, batteries, interfaces, latches, switches, memories, CMOS RAMs, etc etc. All very desirable, but completely over the top. Too complicated to build, too complicated to use, and too expensive to sell. The CPU card in a system has to last a long time (10 years or so for example in the case of the venerable Kemitron MZB-3 Z80 card) and so has to use chips which will be around for years to come. Imagine if Kemitron had used "state of the art" RAM chips on the MZB-3 - we would today be burdened with a row of 8 off 2102 Static RAMs to give a massive 1K of RAM (and incidentally costing more than 50 times the quantity of modern DRAM).

The whole idea of a modular pluggable system is that things can be plugged in and out of it, hardware and software too we hope. If everything is built in to a given card then we are forcing people to buy more than they want sooner than they want, or we are riddling the card with options so nobody knows what makes a standard system. And once in track on a card, no part of the circuit can be redesigned or modified without throwing the whole card away.

Greenbank Electronics



COMPARISON OF MEMORY SPACE
IN VARIOUS SYSTEMS

DMP April 1988

Facing Page 2-3

It is easy to think of "bells and whistles" to be added to any new card (eg Real Time Clock, Electrically Erasable and Programmable ROM instead of EPROM, interface to IBM type of keyboard, printer and serial interfaces, CMOS battery backed RAM, bus monitor and mapping lights, etc etc) , but I now think the bells and whistles should be the subject of a separate bells and whistles card. A change of bell or whistle will then not necessitate scrapping the CPU card.

I have now returned from dream land, and have produced a design which I believe is manufacturable. I have taken advantage of the space I have released by simplifying the concept of the card, to put on extensive buffering, so that the card will be able to drive a full system without future problems.

There is no plan to kill off the Z80A-CPU, which is still selling in ever increasing quantities. Therefore a good deal of thought is required to make sure that Z80 applications can still be supported, programs developed, debugged etc, ditto hardware, even on the new system.

Naturally we cannot make a Z80280 be a Z80, nor vice versa, unless we cripple the Z80280 or bolt on so many extra chips to the Z80 that it becomes a Z80280. Therefore we can allow ourselves to accept that say a 1 Megabyte RAM card cannot be addressed by the Z80. A few users have suggested th use of memory mapping chips such as the Texas Instruments 74LS612 as used in the IBM AT for the same purpose. When we learn that the current price of the 74LS612 (see RS Components Catalogue) is more than that of the Z80280 then this takes a bit of the shine off the idea. If we're going to buy a memory mapping chip at all, we may as well buy the one built into the Z80280, and get all the other features for free rather than pay more for the 74LS612. On the other hand it is reasonable to expect the Z80280 to be able to make use of the existing range of cards, the 8255A (SBC-1) card, floppy disk card, DRM-64 RAM card, serial interface card and so on.

A new monitor program / disk boot will be required, but this can be based on the existing DMON, without the need to commission a new program. We already have the "Plus" version of CP/M, which is well organised for handling a memory space of grater than 64K, so we hope the alterations will be relatively painless.

I believe I have now fully justified the philosophy of the new design, and I now proceed to the specification without any further debate:

Specification (Design Goals)

About 20 chips

Full 24-bit addressing

8 bit Z80-style bus (to suit existing cards)

I/O controlled by existing NIOREQ bus line (to allow use of existing I/O cards, keyboard, disk, tape, serial, parallel printer, etc)

28-pin socket for "Boot/Monitor" ROM (This allows up to 64K EPROMs to be accommodated), smaller EPROMs simply repeat themselves in the one 64K space.

"Boot/Monitor" ROM to remain in the memory map during and after use, ie not to be switched out.

Predefined 64K space in new memory map to activate the existing NMREQ Interak 1 control line, ie to gain access to all of the existing Interak memory mapped cards, eg, DRM-64, EPROM programmer, VDU-2K etc.

All other memory spaces (ie other than the "64K" ROM and the 64K Interak 1) are accessed via the NEMREQ (extended memory request) control line. This gives a new space of 16 MByte minus 128 KByte space, ie 16646144 Bytes, which should be enough for the time being.

Refresh, when carried out by the Z80280, should activate both the ordinary and extended memory request lines, ie both NMREQ and NEMREQ. At all other times only one or other of the memory request lines will be active, to prevent conflicts.

In a similar manner to the allocation of 64K bytes to the purpose of Interak 1 memory access, a portion of the Z80280 I/O space will be allocated to the existing I/O select control line, NIOREQ. If we think of the existing Z80 I/O space as being 256 bytes (the "usual" space, not the full 64K of space which even the Z80 could access by using the address bus AB8-AB15 to select extra I/O port addresses), then 256 bytes of the Z80280 I/O space are to be devoted to the needs of the existing cards. Accesses to the I/O space outside this range will be via the "extended" I/O request control line, NEIOREQ.

At power on (reset), the design will arrange things so that the first instructions will be fetched and executed from the on board ROM. Once the Z80280 has control the memory mapping and management can be arranged (by software instructions, operating the hardware) so that the ROM is mapped high up in memory, allowing RAM to be located in the system from address zero upwards. The

mapping of the 64K for the existing Interak system will similarly be arranged high up in memory, clear of the RAM.

The policy for the production of new Interak memory cards will be as follows: Memory cards greater than 64K will (necessarily) use the extended memory request line, NEMREQ, and will not be suitable for use (except in the most limited of fashions) in a 64K Z80 system. This applies too to cards which use the memory space, eg graphics RAM, RAM disks, Hard disk buffer caches, EPROM programming areas, etc

The policy for the production of new Interak I/O cards will be as follows: I/O cards will continue to use the existing 256 byte I/O space, and a new table will be drawn up giving guidance on general allocations of space for specific purposes. In this way new I/O card developments will be suitable for use in either a Z80 or a Z80280 system. This is not to say that use of the expanded Z80280 space will be prohibited, but it will only be used if there is a good reason, eg the card for some reason will never be used in a Z80 system - using too many ports for example, or ports which control the use of a card which is already a Z80280-only card, eg large graphics management. (The benefit of this policy is that address decoding for I/O cards may continue to be 8-bit; a larger space would force the expense and inconvenience of 16-bit decoding and the need to buffer the extra address lines used in the decoding.)

The new card should be suitable for 16MHz (preferably 20MHz) crystal operation in existing systems, and should also be suitable for use with faster Z80280s (eg 50 MHz) should they ever be produced by Zilog. This implies management of the bus clock scaling factor at power on, and the provision of wait states (if needed) for accesses to existing Interak cards and the on board ROM. (The wait states which can be programmed into the Z80280 by software are too coarse for this purpose as they cover an 8 Megabyte range - all or nothing!)

Wait states for the I/O space need not be generated by on board hardware. If wait states are needed for an I/O access the appropriate register(s) can be altered by Z80280 software before the access is made. Any individual I/O cards which need wait states can of course (and should) generate these themselves by controlling the wait control line, NWAIT.

Reset will be automatic at power on, and also controllable by a switch. The necessary additions to the reset circuit will be included to ensure that the contents of DRAM are not corrupted by a reset operation by the switch, and a suitable circuit will be devised to

allow the "Bus Timing and Initialisation Register" to be set automatically at power on or reset.

Extensive use will be made of the high speed CMOS 74HC and 74AC families of logic, which are expected to take over from 74LS during the lifetime of this board. Wherever possible chips which are available in all families (including 74LS) will be used, so that users can use up existing stocks of 74LS if they wish. However the design will not be prejudiced by this; if HC logic is the most suitable then this will be used.

The on board UART in the Z80280 is not very sophisticated, having no handshake lines, and combined transmit and receive clocks etc. Its most likely use would be for serial communication via a terminal or another computer in a minimum configuration. No separate baud rate clock will be provided in the design, and the UART clock (if used) will be derived from the Z80280 clock, via an internal counter timer circuit. It is possible that a deviation from a good round number (eg 16 MHz or 20 MHz) for the crystal clock would make for more accurate baud rates, but if the error is likely to be less than a few percent, then the general untidyness of using an "ugly" frequency clock will not be accepted.

The external parts of the "on-chip" peripherals will not be built into the system, with the obvious exception of the master clock input circuit, and the likely exception of the UART for serial communication. (Indeed the TTL to RS-232 voltage level translator chip type 145406 will be built into the design for this purpose.) The remaining on-chip peripherals, and my comments, are as follows: CTC circuits - inputs and outputs merely terminated in holes on the pcb for a connector for external use if required by the user (this of course does not prejudice their use for internal software counting and timing). Similarly the DMA circuits, the Local Bus Request and Control Circuits, the Global Bus Request and Control circuits and the handling of the Extension Processor mechanism. Some of the signals may be brought to the edge connector where this seems appropriate, but largely they will not be used in our systems.

The idea of multiple processors sharing Local and Global resources, at first seems quite attractive, but the method of managing this activity, arbitrating between different requesters of resources, and so on demands special logic which is it is unwise to build into the main system board. For example once the Z80280 is switched off for DMA accesses to proceed, or for a second processor to gain access, who is going to look after the refresh of the dynamic RAMs in the system?

I am not against the idea of many microprocessors in a

system, but I have my doubts about them all fighting for and sharing the same resources. What use is the introduction of say the M25 Motorway if too may people fight to use it. At the best unwanted delays and inefficiencies result, and at the worst the whole system locks up and crashes are caused - a crash in a computer is not perhaps as serious as a crash on the M25, but we still don't want to take the risk without a good reason.

There are (arguably) applications which can take advantage of multi-processors all sharing the same resources, but the management of such system design pushes Interak too far away from its origins ("Keep it Simple" - remember?) A very good bus and system design already exists for those who want it: the VME (Versa Modular Europe) Bus, however it is an acquired taste cards for VME applications often cost literally 10 (ten) times that of a similar Interak card to fulfill the same purpose.

As a final word on this subject: If we want our computers to run faster we should add more memory (and use it wisely, to avoid disk accesses) or add hard disk drives. The effect of this will be far more dramatic than jamming in extra processors on the same bus. (A Porsche (yuk) in a traffic jam goes just as slowly if it has a normal one engine or if it has two.)

(This ends the Introduction to the design and the Specification List and discussion of Design Goals)

2.2 MZC-1 CARD GENERAL DESCRIPTION

Memory Mapping

The memory space accessed by the Z80280 is 16 Megabytes. In the diagrams which follow hexadecimal notation is used, "low" addresses are at the bottom of the diagram, and "high" addresses are at the top. This is in keeping with the general jargon which refers to the "top" of memory, and "stack overflow". The higher we go the bigger the numbers get.

(See diagram "Proposed Memory Map")

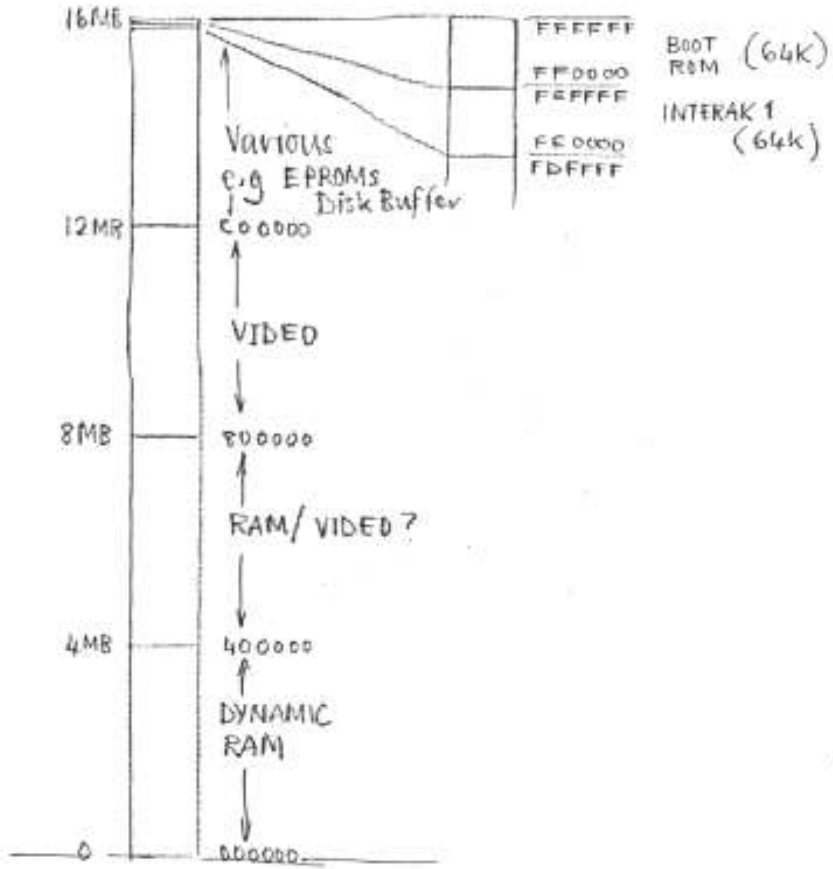
Starting at the bottom, we have 4 Megabytes (000000-3FFFFF) which will definitely be RAM, almost certainly Dynamic.

The use of the next 4 Megabytes (400000-7FFFFF) is less certain. I am not sure what we will run out of first - RAM for the glorious Memory Mapped Graphics, or general purpose RAM. Leave this space blank for a while.

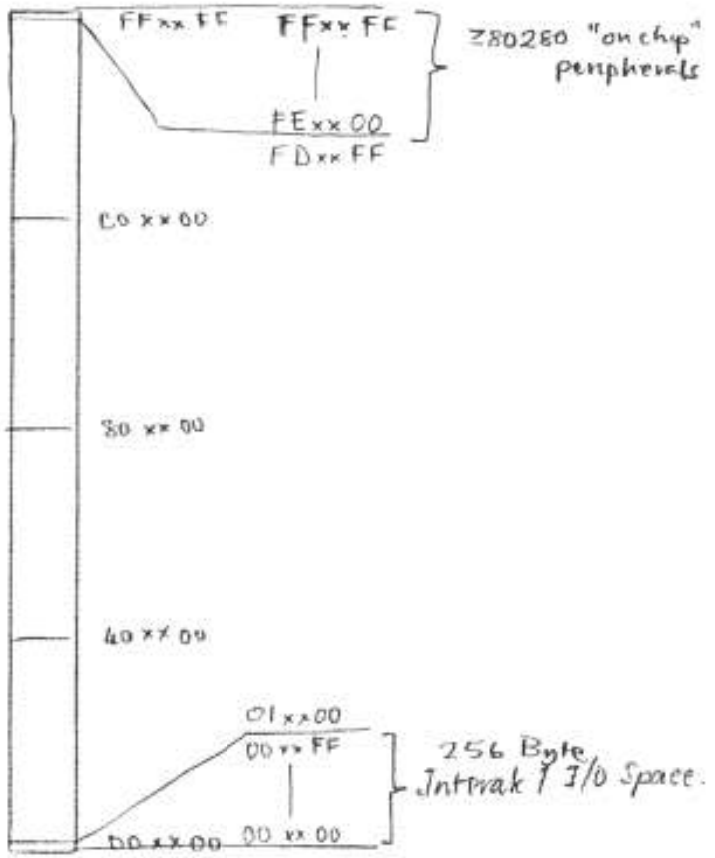
The next 4 Megabytes (800000-BFFFFF) are reserved for memory-mapped graphics purposes ("Video RAM").

The last 4 Megabytes (C00000-FFFFFF) are largely used for odds and ends; things which have not been invented yet. Examples are a memory mapped EPROM card, and EPROM programmer. (I know we have designed an I/O accessed EPROM programmer, but it proved too complex to fit on one board; I am sure an easier technique would be to have it memory mapped in the same way as the original Kemitron PP-2 Design used by ZYMON; "Keep it Simple"?) Other ideas would be in debugging other systems, C00000-FFFFF could include areas which are an image of the system being debugged, and which can be examined at our leisure in our system. An intelligent hard disk interface, or a whole track floppy disk buffer, could be devised, each with their own private (Z80?) CPUs, with common RAM in this area. (See, I'm not totally opposed to multi-processors; I just like them simple.)

At the very top of memory are two dedicated areas, each of 64K. First we have FE0000-FEFFFF for Interak 64K, which translates to our existing 0000-FFFF. When the Z80280 accesses FE0000 then the normal (for the Z80280) extended memory request control, NEMREQ, is not generated; instead the original 64K Interak memory request, NMREQ, is produced, thus enabling all of the cards in the 64K Interak we already know and love. Outside this 64K range no NMREQ is produced, and so the old 64K Interak cards are not accessed. (It would be disaster if they were because they would put their data



PROPOSED MEMORY MAP



PROPOSED I/O MAP

DMP 20/9/88

on the bus in contention with that from the various 4 MByte areas already discussed.

Finally we come to the last 64K, FF0000-FFFFFF. This is all reserved for the on board ROM firmware. A 28 pin socket will be provided, which defines the maximum EPROM size readily available as being 64K. (There are bigger than 64K EPROMs which will fit in a 28-pin socket, but they themselves use a paging mode, and are trickier to access, particularly at power-on time when there's already more than enough paging goin on. EPROMs larger than 64K which have conventional addressing require larger sockets, eg 32-pin or 40 pin. Let us say that 64K will be enough to get the system booted up.) Read operations from to FF0000-FFFFFF generate an extended memory request, NEMREQ, like the other addresses in the Z80280's range but the data on the bus is not allowed on the card; instead the data comes from the on board firmware ROM, not the external system. Writes to that address do in fact get out on to the system bus (because there is little point in writing to an ROM, it takes more than that to make its data change). This may be useful some time in the future as it lends itself to easy modification if somebody ever invents an ROM you can write to (in fact I think they already have), or perhaps the use of those "paged" EPROMs I mentioned.

There is something else to be mentioned regarding the on-board firmware: At power on, or reset, a special arrangement of hardware ensures that not only does the ROM occupy the address already mentioned (FF0000-FFFFF), it occupies <u>every</u> address in the system. This is necessary to ensure that at power on, when the microprocessor fetches its first instruction from location 000000, there is a program at that address. After the ROM gains control, it can establish the required memory map, wait states, etc, by means of the special registers in the Z80280, and finally turn off the mechanism which first established it at location 000000 onwards (and everywhere else). The "turning off" of this power on mechanism is carried out under software control as being the result of performing any I/O Read (ie "IN") instruction. I/O Writes ("OUT" instructions) are permitted; it is only an I/O Read which establishes the ROM in its final position. The ROM program can retain control, or control can be passed to a program loaded from disk or tape or via a serial port, or from ROM elswhere in memory.

I/O Space Mapping

(Please see diagram "Proposed I/O Map")

The total I/O space addressable by the Z80280 is, similarly to the memory space, 16 Megabytes. However this is not often used, certainly not in our systems anyway. 16 Megabytes of I/O is just too much. Typical I/O peripherals, such as PIO (peripheral input output) chips, and UARTS (universal Asynchronous Receiver Transmitters) only require a handful of port addresses. If we generously allowed for an average of say 10 I/O port addresses per card then even an Interak rack full of I/O cards would only require 120 addresses. 16 Megabytes of I/O would provide for more than 100,000 such racks full of cards without ever repeating an address.

Being aware of this, Zilog in the design of the original Z80, (and now the Z80280), decided to make better use of the address bus lines. During an I/O access the eight address bus lines AB8 to AB15 are treated in a special manner. On these lines the contents of a register, either A or B, are output during an I/O access. Conceivably these lines could be used as an address (thus boosting the normal Z80 I/O port addressing range from 256 bytes to 64K, and the Z80280 range from 64K to 16M), but most systems designers (me for example) either ignore the feature or use it instead to allow 16-bit data writes in a single output instruction. In this last application 8 bits are output on the data bus, as usual, and the other 8 bits are output on the "unused" address lines AB8 to AB15.

This feature is often disregarded because of a (perhaps misguided) wish to keep programs compatible with ones which run on close relatives of the Z80, for example the Intel products 8080 (the Daddy of them all) and the Intel 8085 (a half brother to the Z80). Intel are proud of boasting about the 8080, but they hardly mention the 8085 now - probably sour grapes because the introduction of the 8085 was eclipsed by the runaway success of the Z80.

But back to the plot. Depending on how you look at it, the Z80 then has either a 256 byte I/O space addressed on lines AB0 to AB7, (with an optional additional 8 bits of data output on lines AB8 to AB15), or if you like, it has a 64K I/O space addressed on the address lines AB0 to AB15. We generally quote 16 bit addresses in the form of a 4 digit hexadecimal number, eg if the sixteen address lines in a memory access were 1011000010110001, we can express this address in hexadecimal notation as "B0B1". Because of the two ways of looking at the addressing of the I/O space (256 bytes or 64K) a hexadeximal address in the I/O space is often written for example "xxB1". Here the "xx" means "don't care" in that it is the same I/O port address on AB0

to AB7 regardless of the contents of the address bus lines AB8 to AB15. Of course there are times when we "do care", but I like this notation myself because its slightly ambiguous nature reminds me of the two ways of looking at the AB8 to AB15 bus lines during an I/O transaction.

All this is so I can refer to the I/O port space in the Z80280. As the Z80280 has an additional 8 address lines (AB16 to AB23) the range of I/O addresses is increased by the same factor as was the range of memory addresses. 16 Megabytes of I/O if you like, but more practically 64K, with those middle 8 address lines as "don't cares". I demonstrated earlier that 256 bytes of I/O is quite sufficient, so 64K of I/O must be more than sufficient. Therefore my view of the I/O space available in the Z80280 is that is 64K running from address 00xx00 right up to FFxxFF. Zilog have swiped the top few addresses (FExxFF to FFxxFF) for their own use in accessing what they term the "on chip peripherals" (thus explaining how they have added to the Z80 facilities but retained the Z80 operation codes).

In the discussion on the proposed memory map, I showed how the space for the Boot ROM and the 64K Interak was pushed high up into the attic, like an unwanted relative, out of sight of the brave new world of Megabytes of RAM and the like. Such unkind treatment is not to be the lot of the 256 byte Interak I/O space. The I/O cards are still fully franchised members of the family; such items as disk interface, serial RS-232, parallel printers, etc are all still required in a large system, and so I have decided that the 256 ports of the existing Interak system can be in common with the first 256 ports of a Z80280 system. As in the case of the two control lines for the Memory space ("memory request" and "extended memory request") we have two lines for control of the I/O space: the conventional "I/O Request", NIOREQ, and "Extended I/O Request", NEIOREQ. An access to one of the first 256 bytes of I/O (ie hexadecimal 00xx00 to 00xxFF) will generate the existing I/O request; outside this range (ie from 01xx00 to around FDxxFF) the extended I/O request will be generated instead. I am a little vague as to the precise upper limit, the Zilog Technical Manual for the Z80280 should be consulted to see exactly what behaviour results on the control lines during accesses to the on chip peripherals, and to see just how many port addresses they occupy.

Policy on Production of New Interak Cards

This document describes the design of one new Interak card: the XMZC-1 CPU card. Also planned is a new Z80 card, with substantially the same performance as the existing MZB-3. The future in Interak will see the parallel development of two types of system: Z80 based, for those applications where 64K of RAM is enough, and Z80280 based, where 64K isn't enough. We will become a little like a two car family - a mighty Mercedes for the husband to take the whole family on a tour to the South of France, and a modest litle Mini for the wife to pop down to Sainsbury's in. (Or, to show I am not a male chauvinist, the wife can go to the South of France whilst I enjoy myself at Sainsbury's.) Sometimes the Z80280 will be the more appropriate, and sometimes the Z80.

We shall also keep a weather eye on the Zilog/Hitachi 64180, or Z180 as Zilog call it. This lies somewhere between the Z80 and the Z80280, having an address range of 1 Megabyte, ie 16 times greater than the Z80 but in turn only one sixteenth of the address range of the Z80280. Its on board peripherals are as good if not better than the Z80280's, but its compatibility with the Z80 is not so complete as the Z80280's. (For example the Z80280 can be arranged to power on looking exactly like a Z80, the 64180 has power on port mapping for the on chip periperals which means that an existing Z80 program will not run without some minor changes to the I/O addressing.)

The policy on new Interak cards will be as follows: Memory cards (and cards which use the memory space) of 64K or less, will be designed so that they can be used in either a 64K Z80 system or a 16M Z80280 system, controlled by the conventional Memory Request line.

Memory cards or cards which use the memory space, occupying more than 64K will be designed to suit only the Z80280, controlled by the Extended Memory Request line.

Most new I/O cards will be designed to use the first 256 I/O ports, controlled by the conventional I/O Request line, and thus can be used in either Z80 or Z80280 systems. If for some good reason (it would have to be good) cards needing to use an I/O port space of 64K were introduced then they will be accessed via the Extended I/O Request line, and thus be predominantly suitable for only the Z80280 systems. (It would be quite simple to include jumpering on small I/O cards so that they were controlled either by the conventional or the extended I/O control lines. In this case the system would allow an I/O space of 256+256 to be developed - this might be useful in development work when there might be a conflict of development system port usage and "target" system port usage)

2.3 DIL SWITCH SETTINGS, LINKS, OPTIONS, etc.

It is a little early to be describing these as there is no card in production.

However we hope there will be very little to be set up and altered; probably no more than the following:

There are two 3-pin 0.1" pitch pin assemblies shown on sheet 4 of the diagram. One is marked FE(L) and the other is marked FF(L). These introduce wait states (independently of those introduced by the contents of the control registers in the Z80280, and independently of any that may be produced by other cards in the system via the normal method of operating the wait line in the system).

The easiest positions to describe are those where the links are in positions 2-3; in this case there are no wait states set and this part of the circuit has no effect.

If a link is fitted to join 1-2 on the FE(L) pin assembly then wait state(s) are introduced for any access to the 64K space which uses the normal (ie not the expanded) memory request line for control. In an Interak system this will access cards which only need a 64K space, and which generally will be designed to work with a Z80A-CPU. Adding wait states in the memory area reserved for these "old 64K" cards will allow them to be accessed correctly without the need to slow down the rest of the Z80280 system. With the link fitted the wait state(s) will be introduced for accesses to (24-bit) addresses FE0000-FEFFFF.

If a link is fitted to join 1-2 on the FF(L) pin assembly then wait state(s) are introduced for any access to the on board ROM, since this is located at (24-bit) address FF0000-FFFFFF. At power on every access to memory is forced to the on board ROM (as part of the "boot" procedure) and so wait states set here apply over the whole memory map until the booting procedure is finished.

In the initial stages we propose to test the Z80280 with a 16 MHz crystal, and to run the bus at the same speed as before, ie nominally the same as the speed of a Z80A-CPU running at 4 MHz crystal frequency. Thus we are going no faster than before, and the link on the FE(L) assembly can be set for now on positions 2-3. 8K and upward EPROMs are all nowadays supplied with access times shorter than 250 ns, so by the same token the link on the FF(L) assembly likewise should be 2-3.

(continued overleaf)

Also on sheet 4 are shown the 8 jumper links which establish the initial settings of the Z80280 "Bus Timing and Initialisation Register". (See Zilog Technical Manual Chapter 3, page 3-1, section 3.2.1)

Reading from ZD7 to ZD0 the links perform the following function (in what follows, "loading a bit with a 1" means link 1-2 on the appropriate pin assembly "0" means link 2-3):

ZD7. Reserved by Zilog; always load a 0.

ZD6. BS Mode Enable bit; we do not use this mode so you should load a 0.

Multiprocessor Configuration Enable bit; we do not use this mode, so you should load a 0.

ZD4. Reserved by Zilog; always load a 0.

ZD3, ZD2
These two act together and set the number of wait states initially applied to the lower 8 Megabytes of memory. Whatever the setting is here, it can be overruled by software during the initialisation procedure in the "boot" ROM, so it does not really matter. To suit those hoarders who still have stocks of slow old 450 ns EPROMs, we can set these bits up to give one wait state, ie load a 0 to ZD3 and load a 1 to ZD2.

ZD1, ZD0
These two act together to establish the speed at which the Z80280 will access the system bus. The ISBUS back board in Interak, and all of the cards designed for Interak at present require a bus operating at what would be equivalent to a Z80A-CPU operating with a 4 MHz. On the Z80280 with a 16 MHz crystal this is means that the required bus scaling factor is one half. To achieve this load a 0 to both ZD1 and ZD0.

Summarising, the initial settings of the Register in the order given above should be:

ZD7	ZD6	ZD5	ZD4	ZD3	ZD2	ZD1	ZD0
0	0	0	0	0	1	0	0

(Note that for my purposes all 0's are perfectly acceptable, and this is the condition which the Z80280 starts off in if the method of holding the wait line low during power up is used. Thus in the event of trouble during your first experiments it is probably OK not to bother with this feature and to let the Z80280 initialisation look after itself).

The remaining links requiring comment would be on diagram sheet 5 if I had drawn any. They would be related to the handshaking on the RS-232 serial in/out link, and anything that turns up when we start to experiment with the CTC and DMA channels.

Finally, in the finished manual I shall have to say something about the connections to components mounted on the front panel, for example the proposed pinouts and connector for the RS-232 in/out channel (a 25-way D type of course), the LEDs (if fitted on the panel), and the reset push button.

2.4 DETAILED CIRCUIT DESCRIPTION

The circuit diagrams towards the end of this Manual will be taken as the basis for the detailed circuit description. The purpose of this Manual sadly (for the reader, not the writer) does not extend to being a course on digital circuit design, and so it will have to be assumed that a certain amount of background knowledge is possessed by the reader.

Mostly, the components on the diagrams and in the parts list use the designation letters specified in ANSI standard Y32.2-1970. In a way these letters are fairly irrational (eg "J" is a connector, "Q" is a transistor, "U" an integrated circuit, and so on), however as it is a standard it has been adopted here. The component overlay diagram uses the same designations as for the circuit diagrams.

(FUTURE) A block diagram is provided, which also serves as a "map" showing the user where he may find the particular section of the circuit in which he has an interest. Each of the sections represents a logical unit of the whole circuit. Particular care has been taken when preparing the diagrams to ensure, as far as is reasonably possible, that the logic flow begins at the top-left hand corner of the drawing, and continues downwards from left to right. Another convention which has been followed as rigorously as possible is to have the inputs to the left of each circuit block and the outputs to the right. Although this is a very logical and sensible approach to drawing circuit diagrams, it is surprising how often these ideas are disregarded, although perhaps not so surprising to those who have ever performed the mental contortions required to make a circuit diagram lie down on a page and stop wriggling.

(FUTURE) In order to include the information which it is desired to present on the circuit diagrams (pin numbers, function names, power supply pins etc.), it has been necessary to use several sheets of paper. However great care has been taken in partitioning the diagram to turn this apparent disadvantage into a positive advantage, by breaking the circuit up into individual sub-functions which are more easy to understand a step at a time. Even the order of the various sheets has been given close attention, so that as far as possible the source of a signal is shown on an earlier diagram before it used on a later one. All signals which connect to a part of the circuit on a different page are given names, and their source or destination is shown in the form (e.g.) SIG1 3/10,12(4);6/9(5). This (fictitious) example means the signal called SIG1 is connected to U3 pins 10,12 on the circuit diagram sheet (4), and also U6 pin 9 on sheet (5) of the diagram.

So as not to interfere with the understanding of the logic flow, not all power supplies are shown on the integrated

circuit drawings. Instead, each sheet of the diagram contains a table of the integrated circuits on that sheet, their type number, and their power supply connection pins. This does not apply to power supply connections which represent a logic level input, for example 0V for a logic "0", or +5V for a logic "1"; these are shown connected to the integrated circuit pins. An effort has been made to show all pins of each integrated circuit, even the ones which are not connected to anything else. Such pins are marked "NC", which means "not connected".

(FUTURE) The Parts List is to be found at the very back of the Manual, for easy reference. It is organised in two ways; firstly by component reference number, which gives the component value if the reference number is known, and secondly by component value, which gives the reference numbers of all the components which have that value. The latter method is more convenient when checking a kit of parts and assembling the card; for example it is often useful to know where all of the components of a given value should be located if you finish construction and find one or two items left over. Later, you will want to know the value of a component of a given reference number, and the first method of forming the parts list will be more appropriate. To minimise the well known chore of searching all over a circuit board seeing if you can find say C11, or R17, which have become temporarily invisible, the components on the component overlay have been numbered in ascending numerical order, starting at the top left-hand corner and working across and down to the bottom right. (So if you are looking for an elusive R17, you will know you are getting warm if you see R16 or R18: R17 won't be far away.)

(FUTURE) To help users carrying out tests on the card, or wishing to modify it, both ends of such components as resistors and non-polarised capacitors are identified on the circuit diagram; for example if it is wished to check the output pin __ of __ (on sheet _ of the circuit diagram) it can be seen that it is connected to end _ of __, end _ of __ being connected to __.

On the component overlay diagram and the card itself, viewed on side B in "landscape" position with the edge connector to the right, horizontally positioned components should be taken as having end 1 to the left and end 2 to the right, and vertically positioned components should be taken as having end one to the top of the diagram, and end 2 to the bottom. In an effort to help users all components have been placed on the card in a similar direction, so that for example all the ICs are the same way round, similarly the electrolytic capacitors and diodes.

(FUTURE) Section 6 of this Manual also includes various timing diagrams, which may be an aid to understanding certain parts of the circuit.

As the circuit diagram has been drawn split up into logical blocks, these represent convenient sections into which the circuit description may be divided.

(FUTURE) Sheet 0. The Block Diagram.

Detailed Circuit Description

```
* * * * * * * * * * * * * * * * * * * * * * * * *
*  WARNING! THE CIRCUIT DIAGRAMS ARE            *
*  PRESENTED ONLY FOR THE PURPOSE OF            *
*  DEMONSTRATING THE PHILOSOPHY BEHIND THE      *
*  PRESENT DESIGN. THEY CONTAIN A NUMBER OF     *
*  ERRORS AND INACCURACIES AND SHOULD BE        *
*  CHECKED THOROUGHLY AND REVISED BEFORE        *
*  BUILDING COMMENCES. (CONTACT DM PARKINS      *
*  ON 051-645 3391 IF YOU SEE ANY ERRORS, OR    *
*  IF YOU WANT THE LATEST ISSUE OF THE          *
*  DIAGRAMS)                                    *
* * * * * * * * * * * * * * * * * * * * * * * * *
```

The Circuit Diagram comprises 5 Sheets. This is easier for me to draw and edit, and I have divided the circuit into different logical units, which I find easier to deal with during discussion and construction. If you prefer one big diagram then I am afraid you will have to stick all the pieces together yourself. If it then looks like one big ugly mess which is impossible to understand then all I can say is "I told you so!"

Sheet 1. Address and Data Buffers

There are 24 address lines and 8 data lines on the XMZC-1 card. All are buffered on and off the card by HC541s. These have been chosen because they have 2 output enable lines which are gated together internally, which saves on an external gate whenever two control lines are used, they are also available in other family ranges such as advanced high speed logic "AC", which may become relevant in the future, but most importantly they have "flow through" pinouts which make it possible to lay the circuit out on one of our cards without suffering the expense and complexity of a multilayer construction. Separate input and output buffers are used on the data bus, instead of a device such as HC245. This enables us to use different family chips for input and output (useful at the moment whilst the Interak buses are all TTL levels), and allows the insertion of series input protection resistor networks in the input circuits without them appearing in the output circuits.

Other items of note on this sheet of the diagram are the use of a HC573 transparent latch to demultiplex the 8 lowest

address lines (these are output by the Z80280 on its data bus to keep the number of package pins used to as "few" as 68), and the general use of BUSACK(H) (derived from the bus acknowledge response from the Z80280) to disable all of the buffers.

Finally notice that the input and output buffers on the data lines are controlled by separate signals. The Z80280 generates two controls: OE(L) output enable, and IE(L) input enable. This saves quite a bit of logic producing a set of control signals which cater for everything, eg both interrupt acknowledge inputs and normal read inputs, whilst avoiding bus contentions. The Z80280 OE(L) signal is used directly as ZOE(L) on the diagram, and the Z80280 IE(L) is used to derive the signal shown here as RDBUFF(L).

Sheet 2. Control Signals

This diagram deals with all of the CPU control signals, read and write strobes, refresh, halt etc, and the input signals such as Interrupts, Wait, etc. Inputs are on the left, outputs on the right. The inputs are all taken from open collector lines on the Interak bus and a plug in resistor pack provides the necessary pull up. As the pull ups are being used for a logic function rather than simply protecting a CMOS input they have a lower value, about 1 kilohm instead of hundreds of kilohms. The series resistors are protection for the inputs of the HC541 buffers.

The outputs on this sheet fall into 4 classes: (1) unused, (2) those used on the board only which are are unbuffered so as to cause no delay; (3) open collector dirvers and (4) those driving tristate bus lines.

The unused control outputs ("Pause" and "Reserved") are simply unterminated. We have received a suggestion that the unused control outputs should be brought out to a connector for some future purpose but alone they will be of little use: "Pause" is used with an extended processor (which Zilog have no plans to produce) - if they did it would surely need more than this one signal to work it, total changes to the entire design, buses, buffering etc, would be required. "Reserved" equally is difficult to forsee being of fundamental importance now or in the future.

The unbuffered outputs are ones which do not leave the card but are used by other circuits on the card. The Z80280 is a CMOS (Complementary Metal Oxide Semiconductor) device and its outputs are therefore able to swing from rail to rail (0V to +5V) and correctly drive the CMOS 74-series logic chips correctly. Inverting chips are used where the active low signal from the Z80280 is more convenient for subsequent operations if it is converted to an active high signal. Examples of the signals used directly without buffering are

the inputs to the HALT LED drivers, and the signals ZIOREQ(L) ZMREQ(L) etc, at the bottom of the diagram, Signals unbuffered and inverted before use are ZBUSACK (L) and ZADS (L).

The third group are the group of outputs which drive so called "open collector" bus lines. In the CMOS transistors used in these modern integrated circuits there are no such things as collectors, bases, emitters, but the term hangs on as a reminder of the past. It still hangs on in the term used for the positive supply in the integrated circuits: Vcc (voltage of collectors). Those of us who call memory in a computer "core store" are showing our great age - few modern computers now have memories made from a core of magnetic rings. Another odd idea from the old days is the distinction between ROM and RAM memories; ROM means Read Only Memory, fai enough, and RAM means Random Access Memory, again fair enough. But does that mean that a random access may not be made to a ROM? In fact modern ROMs can be accessed completely at random, it was only the ancient ROMs in computers (magnetic drum stores, and the like) which could not be accessed randomly.

If anything, the open collector lines should be called "open drains" unsavoury as that sounds because the transistors in the integrated circuits used now have sources, gates, and drains as their terminals. But I digress: in this design the effect of an open collector driver is simulated by the appropriate use of an HC125 tri-state buffer. To do this the input of the HC125 is connected to 0V, and the signal which is to drive the open collector line is connected to the control input of the HC125; when the control signal is low the output is low, and when the control signal is high the output is tristate or "open".

The "open collector" output bus lines shown here are NBAO (Bus Available Output), NRST (System Reset), and NWAIT (Wait State Control).

The outputs which drive tristate bus lines employ either a HC541 or HC125 bus driving integrated circuit. 74HC bus drivers can sink or source 6 mA at logic 0 and logic 1 outputs respectively. This may not sound a lot compared with the 74LS types, but "HC" logic has the benefit that the amount of current drive available during a transition from one logic level is quite substantial, in the order of 70 mA or so; ideally suited to driving the long lines on a bus. Up our sleeve for the future if extreme levels of drive, or higher speed is required is the availability of the same function in the "AC" (advanced CMOS) range - these have about 4 times the sink or source capability. (Incidently these very high speed chips should not be substituted casually in all positions; the higher is the speed and pulse currents often is the higher the noise and interference generated, particularly in hand wired prototypes.)

Perhaps unwisely, I have allowed for an input to the Bus

Request of the Z80280, which can render most of its outputs high impedance. In order to terminate the inputs of the output buffer chips etc under these circumstances, high value (100k or so) resistors have been provided for the lines which otherwise would "float". When the Bus Request input is activated not only does the Z80280 relinquish its own buses but the whole XMZC-1 card should as well; this is the reason for the signal called BUSACK(H) being connected into the output tristate buffer control (on this sheet of the diagram and sheet 1). The uninverted Bus Acknowledge signal itself of course does not go tristate, so this is connected to the bus via a permanently enabled buffer, the bus signal name being NBAO, Bus Available Output.

Despite all this there is little use for the Bus Request mechanism in any system I can forsee. If this card relinquishes the bus then we had better be pretty certain that there is something else ready to take over - this implies quite complex bus controller logic, arbitration mechanisms and so on. The matter is complicated further in that the Z80280 also has a "Global Bus Request" and "Global Bus Grant" set of control lines (these are alternative uses for the Counter Timer In/Out 0, and Counter Timer Input 0, if you are searching for them). Remember when turning over the bus to some other bus master, that bus master becomes responsible for refreshing the dynamic RAM, managing the normal and expanded Memory Request and I/O request mechanism, and so on. All matters which are in conflict with our motto "Keep it simple".

The Halt output, by way of a gimmick, drives a pair of LEDs, "Red" means Halt, "Green" means Run. If a dual red and green LED is used it will change colour in the same way, and would even show as yellow if there were regular transitions between red and green. In present Interak systems "Halt" is never used, but once interrupts are brought into play, the Halt (or waiting for interrupt or reset) indicator will have more application. The Z80280 has a sophisticated method for trapping and detecting various other violations of correct operation. If nothing else the Halt indicator is useful for indicating the existence of the ominous sounding "Fatal Condition" (See Zilog Provisional Technical Manual Section 6 page 6-11.) This produces a Halt state which can only be exited by a reset.

Pin 34 "Opt" is connected to 0V. This establishes the operation of the Z80280 as being suitable for the Z80 environment. If this pin were connected to +5V then the chip would behave as a "ZBUS" type of processor, ie 16-bit data bus and timing and control similar to that of Zilog's Z8000 and Z80000 offerings. The idea of using the Z80280 in this last mode in the Interak system has been ruled out entirely. ZBUS lacks signals like the Z80 clock, M1 and so on, (not that in itself is a bad thing) so it certainly cannot be used with existing cards in the Interak range which require those signals. Such items as the ZBUS equivalent of say the Z80

dual serial chips are often twice the price, so use of the ZBUS cuts us off from existing supplies of chips we know about. Without the Z80 clock the edges of strobe signals become important, and the noise, crosstalk, etc performance of the bus becomes more critical (if there are bad edges on strobes in a Z80 system it does not matter because transactions are timed by the Z80 clock, which operates within the control pulses, after data has settled.

I do not mean to say there is not room in the world for us to design a bus other than the Interak Z80 Bus, but if Zilog succeed in killing off the Z80, it would not do them any good as far as I am concerned. If they let me (and the rest of the Interak community) down on the Z80 I am hardly likely to come back for another pasting on the ZBUS!

Similarly if the design of this new card were to force all existing users to buy a new set of cards, then I would not be surprised if they chose not to buy them from me!

Sheet 3. Boot ROM, I/O and Memory Expansion Control Signals

Much has already be said of these matters and how the memory I/O and Boot ROM spaces are mapped. On this sheet we see the circuit to achieve all this. At the top left hand side of the diagram is an HC30 8-input NAND gate whose output is labelled FE.FF(L) This goes low when the top seven address lines ZA17-ZA23 are 1, ie at the very top of the memory map, addresses FE0000 to FFFFFF, for the two 64K areas having addresses FExxxx and FFxxxx. (FExxxx is used to map in the 64K Interak system, and FFxxxx provides 64K for the on board ROM.) Which of the two it is to be depends of course on adress ZA16, since that is the one which distinguishes FFxxxx from FExxxx. ZA16 is taken to an HC04 inverter so the output of the following HC02, (output labelled FF(H), goes low only for addresses FFxxxx.

The signal path to the chip enable of the Boot ROM (for that is our destination in this part of the discussion) continues as follows: in a further HC02 the FF(H) signal is combined with one called PWRON(H); this comes from a circuit on sheet 4 which will be discussed shortly, and is a signal which is high when the board is first powered on. If either (or both) of the PWRON(H) or FF(H) lines are high, the output of this HC02 goes low. In this case ZMREQ(L) which is effectively connected into the HC32 OR-gate we come to next, will activate the Boot ROM chip Enable line (active low). In short, at first power on and/or when addresses FFxxxxx are accessed, the on board ROM is enabled. In fact only reads of that address cause the ROM to output data because you can see that the ZRD(L) line is connected to the active low Output Enable control of the ROM.

There is a little bit further to go now before we can leave the ROM. Its active low Chip Enable pin is inverted and combined with the ZIE(L) (Z80280 Input Buffer Enable control

signal) in a further HC32 OR-gate, whose output is RDBUFF(L). If you refer to diagram Sheet 2 you will see that RDBUFF(L) is the control for the data bus read buffer. Thus this follows the ZIE(L) signal, allowing data reads from the system bus except when the Boot ROM is enabled, ie at first power on, or at any time when address FFxxxx is selected, the read operation receives data from the Boot ROM, not the system bus.

So much for the Boot ROM. We now turn our attention to the circuit which deals with the addresses FExxxx (which you will remember represent the 64K which is chosen to activate the 64K Interak memory cards via the conventional Memory Request line on the bus, all other addresses (16 MByte or so) being accessed via the Extended Memory Request line on the bus). Most of the work is done in an HC153 dual 4-line to 1 line multiplexer. I am sorry for the obscure use of this chip instead of conventional gating. Its benefit here is that it contains a lot of gates of just the right kind to suit my needs, thus saving space on the finished board, but its outstanding benefit here is that passage of signals through the innards of the HC153 takes much less time than the journey through discrete gates which would perform the same function. (And there isn't much time to spare; the decision on which bus Memory Request line to activate - conventional or extended - has to be taken before the ZMREQ(L) signal which qualifies it. The addresses are issued only marginally before the ZMREQ(L) lines. To ease things a bit I delay the ZMREQ(L) signal by one gate, but it is a close run race.)

The outputs of the HC153 depend on the inputs and the state of the selection inputs S0 and S1. The required truth table, which is implemented here as follows:

Address Bus	S1	S0	Output 2Y ie MREQ(L)	Output 1Y ie EMREQ(L)
FE0000-FEFFFF	0	0	0	1
FF0000-FFFFFF	0	1	1	0
(neither)	1	0	1	0
(neither)	1	1	1	0

When the outputs have settled, whichever one is active (the conventional or extended memory request) is gated through by ZMREQ(L) in the HC32 gates and buffered on to the system bus.

So much this far in fact could of course easily have been accomplished by descrete gates. However there is a further complication, and that is the refreshing of Dynamic RAM in both the conventional and the extended memory areas. During refresh this both of the two memory request lines need to be activated simultaneously (there is no commotion on the bus as a result of this because of course the Z80280 is wise enough to withold the read and write strobe signals during refresh.

The ZRFSH(L) signal from the Z80280 is inverted and enters the

two (active low) enable inputs of the HC153. At refresh time the HC153 outputs are both disabled (they were enabled before in our discussion so far). The output state when the HC153 is disabled is that both outputs are "0" QED.

The remaining circuit on this sheet is the similar one which deals with the extension of the conventional I/O Request Line, by developing an "Extended I/O Request" control line. The conventional I/O Request is activated for I/O accesses to addresses in the range 00xx00 to 00xxFF, and the Extended I/O Request line is activated for the rest, ie 01xx00 to FFxxFF (although at the very top of the I/O space, FExx00 to FFxx00, the Z80280 hides its on-board peripheral addresses, and I/O transactions here are not conducted outside the Z80280.)

The technique is basically similar, but this time as it is eight 0's we are decoding we can take advantage of a gate which does not exist in the old 74LS world. The gate to be used here is an HC4078, a combined 8 input OR and NOR function. When the top address lines ZA16 to ZA23 are all 0 the INV output of the HC4078 is 0, ultimately activating the conventional I/O Request line on the bus; for all other I/O addresses it is the NONINV output which is 0, ultimately activating the Extended I/O Request line on the bus. The ZIOREQ(L) signal from the Z80280 qualifies the output in the HC32 gates after the output of the HC4078 has settled. As for the Memory Request lines there is a time when both the normal I/O Request and the Extended I/O Request lines must be activated simultaneously. This is during an interrupt acknowledge - the Z80280 does not issue an address at this time, so the system has no way of knowing which of the two I/O request lines should be activated: was the interrupt from the conventional I/O space, or the Extended I/O space? The solution is, by means of the operation of the 2-input AND gates, HC08, to introduce a signal to force both I/O lines low together, but only during an interrupt acknowledge cycle. This is uniquely identified by the simultaneous activation (by the Z80280) of ZM1(L) and ZIOREQ(L), therefore it is sufficient to include ZM1(L) in the HC08 gates on the way to the output. When ZM1(L) is high, ie during normal I/O accesses, the operation of the circuit is as desribed, but when ZM1(L) is low both of the bus I/O Request outputs are active (low) if ZIOREQ(L) is also low. Again, the Z80280 is wise enough to avoid the contention on the bus which would result when both I/O Request lines are simultaneously active by witholding the ZRDS(L) read data strobe during an interrupt acknowledge cycle.

****** WARNING There is a discrepancy between the timing diagrams for I/O transactions in the two Zilog Publications Preliminary Product Specification February 1987, and The Preliminary Technical Manual March 1987. On page 78 of the former, figure 61 "Z80 Bus Read Type Transactions" shows IOREQ(L) active around the time A8-A23 are changing, giving me no safety margin whatever to allow for the delays in my

gating. Much more palatable is the diagram on page 12-11 of the Technical Manual: Fig 12-10 "I/O Read Timing". This shows IOREQ(L) becoming active a whole CLK cycle later, which gives me plenty of time for my gates to process the which I/O Request bus output decision. Until I can find out from Zilog, or until I can get a Z80280 up and running I shall cross my fingers and assume the true timing is that in the Preliminary Technical Manual. END OF WARNING *********

The outputs MREQ(L) EMREQ(L) IOREQ(L) EIOREQ(L) are buffered onto the system bus but buffers shown on sheet 2 of the diagram.

Finally, there is an output IORD(L) produced by the HC32 at the foot of the diagram, having at its input ZIOREQ(L) and ZRD(L). This output goes low for any I/O read (ie INPUT instruction). It is used on the power-on circuit on sheet 4.

Sheet 4. Power on Reset and Wait State Circuits

Compared with those for the Z80, the requirements of the reset input in a Z80280 are quite simple. The Z80 reset was not internally synchronised with its instruction execution cycle, and external hardware had to be added to synchronise a hardware reset with an M1 cycle if the penalty of possible corruption of a row of dynamic RAM data could not be accepted. (Once a Dynamic RAM access operation has started it must be allowed to finish completely, otherwise data in a whole row of RAM locations could be spoiled.

Some systems expect to reload everything in entirety from RAM at power on, or after a manual reset. This is perhaps acceptable in an office or home "personal computer", but in a development system like Interak it is not. Many home computers (eg the IBM range) get over their reset difficulties by not providing a reset switch at all, but this is rather running away from the problem. If we are to preserve the previous contents of dynamic RAM after a manual reset then the duration of a reset must be kept as short as possible - if reset is sustained for more than a few milliseconds the data in dynamic RAM is put at risk owing to the lack of refresh activity in the Z80280 during reset.

In this design the reset is limited to about 50 us by the use of a section of an HC123 monostable. A pulse is produced whenever the "B" input rises from logic 0 to logic 1. The "B" input is suitable for a slowly rising waveform such as a capacitor charging through a resistor. The diode ensures rapid discharge of the capacitor even if the power to the computer is interrupted only for a moment. The manual reset button is connected via a 330 R resistor to limit the rush of current which would otherwise pass through the switch contacts, possibly reducing the life or quality of the switch connection. (Short circuiting a charged capacitor is bad for

the capacitor too. We use aluminium types which can stand a bit of abuse, but if you use tantalum, which are inferior in this one respect you definitely shouldn't do it!)

The negative going reset pulse from the HC123 "not-Q" output appears on the line labelled RESET(L); refer to sheet 2 of the diagrams to see how this is buffered onto the reset line on the bus, (NRST), via an HC125 connected as an "open collector" driver.

Output Q of the reset monostable is a positive going pulse, which is delayed slightly (the required figure is at least 2 processor clock cycles) by the resistor and capacitor, and inverted by the HC1. It then passes through an HC08 2 input AND gate (we shall deal with the other input in a moment) and so on to an HC125 connected as an open collector driver to the wait line on the bus, NWAIT. Thus as the system comes out of the "reset" condition, the wait line NWAIT (and ultimately the wait line to the Z80280 MPU) is held low. This is a special initialisation condition, (see Preliminary Technical Manual Chapter 11, 3rd paragraph page 11-1), and allows the Z80280 to begin with data selected by the user loaded into its internal Bus Timing and Initialization Register. The logic states of the 8 data lines ZD0 to ZD7 at this particular moment are copied into the internal register. On this sheet of the diagram we see how this is achieved. A high value resistor is connected to each of the data bus lines and the other end of each resistor can be connected to either +5V or 0V at the user's option, to set up the required initial conditions. Some of these conditions can be altered later by program instructions, but some are fixed at power on, so this is our only chance to alter them if we want something different.

Whilst we are in the area of the wait state circuit I shall just finish describing this, before returning to the power on reset arrangements.

The other half of the monostable type HC123 is shown on the lower part of the diagram. Connected to the "A" input (which is the negative edge trigger input) is an HC08, 2 input AND gate. Two signals are available for input to the HC08, and may be selected by links on the associated 3 pin assemblies: the signals are produced by the circuit on sheet 3 of the circuit diagram; FE(L) corresponds to an access to the original Memory Request area (ie 64K Interak mapped at FExxxx) and FF(L) corresponds to an access to the on board ROM.

Of course enthusiasts of the Z80280 will be aware that it is capable of applying wait states under program control, but they are quite coarsely applied over large areas of 8 Megabytes. 64K is (relatively) small and the provision of this hardware circuit will ensure that the occasional need to access the 64K Interak space and the on board ROM will not hold up accesses elswhere. Once the wait state monostable fires the negative going pulse at the not-Q output passes

through the HC08 gate and so on to the line labelled WAIT(L). See sheet 2 of the diagrams to see how this is connected via an HC125 connected as an open collector driver for the NWAIT system bus line.

All we have to describe now is the power-on circuit. At the top right of the diagram is an output PWRON(H) which is taken from the upper one of a pair of 2 input NAND gates (HC00) connected as a cross-coupled bistable. In essence at power on, or reset, PWRON(H) is jammed high, which forces (see sheet 3 of the diagram where we see this signal used) every memory access, regardless of its actual address to be routed to the on board ROM (Boot ROM). Once the ROM has taken control and established the desired conditions, memory map, timing etc, the program which is running issues an I/O Read instruction, eg IN A,(0FFH); this turns off the power on mechanism (ie establishes PWRON(H) to "0") and thenceforth the on board ROM appears in the memory map only at its designated address (FFxxxx).

The cross coupled bistable is set to give a "1" output if its lower input is taken to "0" (provided the other input is "1"). The upper input is taken from the Z80280 master reset input labelled here as ZRST(L). This therefore sets the bistable every time the monostable is triggered, ie at power on, and by the reset switch. The lower input to the bistable is provided by the signal called "IORD(L)" which is derived from an OR gate on sheet 3 of the diagrams. This is an HC32 OR gate whose output goes low when any I/O read (INPUT) instruction is executed by the processor. Thus as soon as this happens, PWRON(H) becomes logic "0" which marks the conclusion of the power on sequence. All future I/O reads have no further effect, until the system is reset again, either by power on or by manual operation of the reset switch.

Sheet 5. Z80280 "On Chip" Peripherals.

Included inside the Z80280 chip are a number of peripherals, which in Z80 days would have been the subject of other separate integrated circuit designs. Some of these we are likely to use, some less likely.

At the bottom left is the clock circuit. A fundamental mode crystal is used of the required frequency although in the finished design an alternative crystal oscillator module will be permitted. This is in case the frequency does approach the roumoured 50 MHz; the on-chip design of a 50 MHz oscillator may prove difficult (unless Zilog are cleverer than I thnk they are).

To the top left are the three counter timer channels. Channel 0 has an alternative use as a Global Bus request and Global Bus acknowledge system. We shall disregard this latter use because the use of these would necessitate a total redesign of

the Interak system, making it a completely different product; not likely to be greeted with enthusiasm by the owners of the tens of thousands of cards already in circulation. For the time being they will simply be terminated electrically and physically. One suggestion we have received and which is under considerationi is to bring these lines to an edge connector on the front side of the card.

The counter timer channels can of course be used by the Z80280 without the need for external connection. There may be some benefit in allowing a track pattern which would allow the three channels to be chained together to make a very long counter.

Below the CTC channels are the DMA (direct memory access) channels. These again would cause quite a bit of alteration to the Interak system if they were to be implemented into the structure. Using these to any marked degree would force the system to be too incompatible with the existing Z80 set up, and much of the charm of the system would be lost. I have not investigated thoroughly, but I would imagine the DMA channels are capable of doing work inside a Z80280 system, without the need for further hardware - a super fast Megabyte Block move facility perhaps?

Finally to the right is the serial in/out channel. This is very simple in that it operates merely with data in and out lines, ie no handshaking or modem control lines are built in. One of the internal counter timer circuits can be used for generating the baud rate (we may have to fiddle the crystal frequency to get the baud rate 100% correct, or we could accept its being a few percent out). A 145406 TTL to and from RS-232 level interface chip will be included. Software handshaking (eg XON/XOFF) will be possible, but if hardware handshaking is required we will have to look into the possibility of borrowing some of the lines from the CTC or DMA channels if we can find any which can be set and tested independently.

the Interak system, making it a completely different product; not likely to be greeted with enthusiasm by the owners of the tens of thousands of cards already in circulation. For the time being they will simply be terminated electrically and physically. One suggestion we have received and which is under consideration is to bring these lines to an edge connector on the front side of the card.

The counter timer channels can of course be used by the Z80280 without the need for external connection. There may be some benefit in allowing a track pattern which would allow the three channels to be chained together to make a very long counter.

Below the CTC channels are the DMA (direct memory access) channels. These again would cause quite a bit of alteration to the Interak system if they were to be implemented into the structure. Using these to any marked degree would force the system to be too incompatible with the existing Z80 set up, and much of the charm of the system would be lost. I have not investigated thoroughly, but I would imagine the DMA channels are capable of doing work inside a Z80280 system, without the need for further hardware - a super fast Megabyte Block move facility perhaps?

Finally to the right is the serial in/out channel. This is very simple in that it operates merely with data in and out lines, ie no handshaking or modem control lines are built in. One of the internal counter timer circuits can be used for generating the baud rate (we may have to fiddle the crystal frequency to get the baud rate 100% correct, or we could accept its being a few percent out). A 145406 TTL to and from RS-232 level interface chip will be included. Software handshaking (eg XON/XOFF) will be possible, but if hardware handshaking is required we will have to look into the possibility of borrowing some of the lines from the CTC or DMA channels if we can find any which can be set and tested independently.

called "flow-through" pin outs. These have all the inputs in order on one side of the chip and all of the outputs on the other. (Contrast for example the pinouts of say the flow-through 74HC573 and 74HC541 against the similar function 74HC373 and 74HC244.) The use of these better pinouts will make the construction of the prototype much easier too, and the short neat wiring which will result can only improve matters regarding the "noise" and interference which dogs handwired prototype designs of high speed circuits.

Mount the buffer chips (input and output) near the edge connector. Ensure there is a decoupling capacitor between the supply pins of each buffer, connected directly as possible to the supply pins of the same ic.

The Z80280 must have similar decoupling (across its many supply pins); keep the control lines away from the address and data buses if possible, and keep everything away from the crystal.

3.2 SETTING UP AND TESTING

When a pcb is in production we shall go on and on in the following vein:

1. Check that all components fitted so far are in the correct place, and have the correct polarity where appropriate. Re-read the constructional notes - important points are preceded by the loud word "Note!", and can be checked again now.

2. Inspect the board for dry joints, solder bridges and solder splashes, paying particular attention to areas where tracks run between IC pins. Shine a strong light through the board, and use a magnifying glass if you have one.

3. Apply power to the board and check that +5V is present and correct at all the IC sockets, i.e. at the upper right-hand corners of each. If you consider you are likely to cause damage by carrying out this test, or are very confident of your workmanship then omit this step at your own discretion.

etc, etc.

If you have just constructed a prototype card then you hold in your hand something that has never existed before in the history of mankind. Treat it with care. The most critical chip is of course the big one. All you can do to look after it is to check and recheck your wiring, and the translation of the pins on the chip to pins on the socket. Before you plug in make sure +5V and 0V are correctly wired (perhaps probe the empty socket with a voltmeter whilst applying power) and then plug the chip in, the right way round.

Good Luck!

3.3 FAULT FINDING, RETURN FOR SERVICE

Not on your nelly: if it doesn't work, you keep it (I've enough problems of my own thanks!)

There are a few things you can try; for example no matter how scrambled you have got your wires a microprocessor once switched on pretty well has to do something. If there is no activity at all then your clue is to look at the Halt line output (although even a "halt" leaves many signals thrashing about furiously), interrupt inputs, reset input, wait line input, bus request and acknowledge, M1 (opcode fetch), Refresh, CLK and so on. Only a complete write-off of a chip or absent of incorrect power supplies will produce a complete zero of activity on every line.

The type of socket I amusing has little holes specially for the purpose of probing the signals at the pins of the chip. This makes life a bit easier than having to guess.

More likely your problem will be that the chip is working but not doing what you want. (This is in fact the definition of a bug, hardware or software.) All you can do is to burn progressively simpler and simpler test programs into your boot EPROM until you can isolate the trouble. At the very simplest try "00" in every location of your EPROM - even if you have totally mixed up the order of the address and data lines then the Z80280 should be able to find that data (since 00 backwards is still 00). If you find your boot ROM is not being selected, fiddle and jumper the circuit until you force all reads to come from the boot PROM regardless of where they want to come from. Gradually build up a program byte at a time until you can see what is going wrong. For testing I like little programs like: perform an I/O instruction, and, go back and do it again. There are many "reads" even in such a short program but there is only one I/O instruction. The I/O line is the one to trigger your scope on then, to get a stable trace whilst you are prowling round the other lines on the circuit. A nasty fault is wrongly controlled buffers, where for example a read and write operation is simultaneously selected. If you have used sockets for your buffers, then you can easily pull one or another out during testing, probing the control lines to se just why it is the buffer which shouldn't be enabled is being enabled.

If you are accustomed to testing boards with TTL (74LS etc) chips then note you need slightly different techniques with CMOS. An open circuit 74LS input shows a characteristically odd signal on a 'scope - neither 0 nor 1, but something in between. Open circuit CMOS on the other hand takes up the voltage at the end of your scope probe, typically 0V. On the other hand CMOS is easy to push around; for example you can test the operation of gates by sticking in a signal from your (anti static) finger into an unterminated gate input and watch the output toggle up and down at 50 Hz mains frequency; you

can't do that with 74LS.

(Finally, you did tie pin 34 of the chip to ground didn't you? It explains a lot of misbehaviour if you didn't.)

Nothing can possibly go wrong, go wr*ng, g* wr*ng

There are a few known bugs in the Z80280, and goodness knows how many unknown ones. One rumour I have heard is that the 20 MHz part will not function completely correctly with input frequencies above 16 MHz. (Although I read this in an issue of a newsletter for a competing product to Interak, known as the "SC84" Scientific computer - they have decided to use the Hitachi 64180, so perhaps have an axe to grind. Mind you, if a Hitachi 64180 user criticises the Z80280 it is a case of the pot calling the kettle black because the 64180 is full of little dodgy bits, so much so it proved necessary for the manufacturers to produce a "Z" (= even more Z80 compatible) version of that chip). I shall try of course to gather as much information as I can, and any bug lists I have will be included with this document.

Another possible problem area once you get going applies only to those wicked programmers who think they are very clever in using the "undocumented" instructions in the Z80 instruction set. If you have used only legal instructions you will be all right. "So what, it works" may now be answered by "So what, it doesn't, that's what!"

Tips on Software, and Boot ROM Design

The easiest way to start the system up for evaluation is of course to make it appear to the system exactly like the existing MZB-3 card. This implies a crystal of 16 MHz (ie an internal clock of 8 MHz) and a bus clock scaling factor of 2, ie equivalent to a 4 MHz Z80A. By a happy coincidence, Zilog have so organised the power on initialisation of the Z80280 that it defaults to the above conditions, ie it appears at power on exactly like a 4 MHz Z80A. (I daresay that the use of a 20 MHz crystal would not be prejudicial, but to be absolutely certain 16 MHz is safest.)

Existing programs will not run without modification in this design because of the simplifications I have made to the power-on-jump arrangements of the existing MZB-3 card. As an example of the necessary initialisation procedure I shall take ZYMON as an example, although the technique will apply to DMON or any of the programs which are used at present as on-board fimware on the MZB-3 card. First we will remind ourselves of the power-on procedure on the MZB-3:

> ZYMON 2 is located in a 2K EPROM from addresses E000-E7FF, although the decoding on the MZB-3 card only extends to a 4K select thus a repeat ("image" or "ghost") of ZYMON appears at E800-EFFF.
>
> Hardware at power on in the MZB-3 ensures that a Z80 instruction fetch from address 0000 actually comes from E000, so at power on ZYMON is immediately in control.
>
> ZYMON does the following:
>
> Tests to see if it is indeed in ROM at E000 (for historical reasons - very early Interak systems had no RAM at 0000 as even 2K was too expensive to waste, and the ZYMON EPROM could also be used at 0000).
>
> If so a "natural jump" is then executed to an address on page E, ostensibly a few bytes, but in reality right down from page 0 where the Z80 begins right up to ZYMON on page E. The Z80 program counter is now fetching instructions from addresses on page E and thus the hardware mechanism which forces 0xxx to become Exxxx at power on is redundant.
>
> This mechanism is disabled by executing any input instruction from Port FF, eg IN A,(0FFH). Accesses to addresses 0000 onward no longer are routed to ZYMON but take place in the system RAM which begins at 0000.
>
> Initialisation of the power on sequence is now complete ZYMON now has total control and proceeds in the way detailed in the ZYMON manual (copies itself to RAM at 0000, and jumps to RAM to run).

ZYMON is still present in the memory map at addresses E000-EFFF, but it may be switched off if not required by any output instruction to Port FF, eg OUT (0FFH),A When it is switched off whatever else in the system is at address E000-EFFF (eg RAM) "shines through" and can now be accessed in the normal way.

So much for ZYMON etc. The following characteristics of the new design mean that similar results must be achieved in different ways:

At power on the XMZC-1 card ROM appears throughout the memory map, ie 000000-FFFFFF, thus any access to any address (including the starting address 000000) will be routed to the on board ROM. So far this is the same as the MZB-3 Z80 card. A natural jump to page E would in fact still be successful, although in the new scheme of things this is not the whole story.

Unlike on the MZB-3 card, the ROM cannot be turned off. As explained earlier the ROM is always present at address FF0000 to FFFFFF. The Z80280, like the Z80 before it only "sees" a 64K space, and so the method of removing the ROM from the sight of the Z80280 is merely to map it somewhere outside its current 64K area of interest; it does no longer has to be switched off physically.

The method of turning off the power-on mechanism (which forces initial accesses to 000000 to be re-routed to the ROM) is different from that used on the MZB-3. On the MZB-3 you will recall it was any input from Port FF which accomplished this. In the new XMZC-1 design the required instruction is an input from any I/O port, eg IN A,(xx), ie it does not matter which precise port is used. You may be alarmed at this if you know that I/O transactions are required to initialise the memory management unit inside the Z80280; however (see section 12.5.4 on page 12-10 of Zilog's Preliminary Z80280 Technical Manual) internal transactions to devices on I/O pages FExx00 to FFxxFF do not generate external bus transactions. I am not sure other processors will be so forgiving, therefore in case a more precise port allocation is required later, I suggest that you keep to the traditional IN A,(FF).

Suggested initialisation procedure to bring in ZYMON, DMON etc on the new card.

The smallest practical EPROM nowadays is 8K, and taking advantage of this I can simplify the following discussion, and can demonstrate how a standard initialisation procedure can bring in EPROM programs which are byte for byte the same as they were, running

at exactly the same addresses as they were. If you are limited to 2K EPROMs then you can follow the same principle, but you will have to tinker with some of the bytes in the 2K program itself.

Burn a new EPROM with the new initialisation in the first 4K (ie at location 0000-0FFF) and the old program in the last 4K (eg 2K ZYMON from 1000-17FF; use 1000-1FFF if the program is 4K; use a bigger EPROM or waste less space on initialisation if the program is bigger)

Procedure:

1. Power on, with EPROM on XMZC-1 card.

2. Initialise any Z80280 control registers which need initialising - wait states, mapping etc (use the "LDCTRL" instruction, see Chapter 3 in the Zilog Technical Manual). Set the I/O Page register to FF ready to allow accesses to I/O addresses FFxxxx at step 3. Note ROM is in blanket control at the moment, RAM accesses, use of stack, memory tests etc, are denied. Also prohibited at this stage (by my hardware) is any I/O input instruction.

3. Alter memory management unit (by accessing internal registers around addresses FFxxF0, see Zilog Technical Manual Chapter 7) to achieve memory mapping as follows (use "address translation without Program/Data Separation" which establishes mapping in 4K pages):

CPU Address	Mapped to	
F000-FFFF	FEF000-FEFFFF	(64K Interak)
E000-EFFF	FEE000-FEEFFF	(64K Interak)
D000-DFFF	FED000-FEDFFF	(64K Interak)
C000-CFFF	FEC000-FECFFF	(64K Interak)
B000-BFFF	FEB000-FEBFFF	(64K Interak)
A000-AFFF	FEA000-FEAFFF	(64K Interak)
9000-9FFF	FE9000-FE9FFF	(64K Interak)
8000-8FFF	FE8000-FE8FFF	(64K Interak)
7000-7FFF	FE7000-FE7FFF	(64K Interak)
6000-6FFF	FE6000-FE6FFF	(64K Interak)
5000-5FFF	FE5000-FE5FFF	(64K Interak)
4000-4FFF	FE4000-FE4FFF	(64K Interak)
3000-3FFF	FE3000-FE3FFF	(64K Interak)
2000-2FFF	FE2000-FE2FFF	(64K Interak)
1000-1FFF	FF1000-FF1FFF	(ZYMON in ROM)
0000-0FFF	FF0000-FF0FFF	(Initial'n ROM)

The restrictions (see of Zilog Manual page 7-6 last paragraph) on keeping to an untranslated page when changing an MMU register can be ignored here, because at power on my circuit always routes every access to the on-board ROM, so the transition is quite smooth from the code running in the initialisation ROM at power on, to the same code a few bytes further on in the same ROM. Turn off the power-on initialisation mechanism by executing an IN A,(0FFH) instruction,

4. Where are we now? At this stage we have turned off the power-on initialisation mechanism, we have enabled the memory management unit, mapping the majority of the 64K Interak RAM into the upper 14 pages of the current memory map and 8K of onboard ROM into the lowest two 4K pages. The CPU program counter has moved on from location 0000 and is running at shall we say location 0100 or 0200 or so.

   Now execute a block move, copying page 1 (ie 1000-1FFF, containing ZYMON 2 in ROM at present) to page E (ie E000-EFFF, the usual starting point for ZYMON 2)

5. We are almost ready to jump to E000 to begin execution of the ZYMON 2 program, but there is one more procedure to face. At the bottom 2 4K pages we still have 8K of the on-board ROM. We need to alter the page descriptor registers one more time to map out the ROM on pages 0 and 1 and to map in the remainder of the 64K Interak system RAM. Unfortunately it is not so easy as that, because it is a rule which must not be broken in Z80280 systems, that we don't remap memory when we're actually executing code in it!

What we do next is this:

Transfer a fragment of code (the fragment which performs this last piece of remapping, then jumps to E000 to begin ZYMON), from its address low down in ROM to somewhere in RAM, using a block move instruction. Pick a place in RAM where it does not matter too much if it gets corrupted, for example well away from the transient program area and jump tables used by CPM. For example I use rrrr. Now jump to the fragment of code which has been copied, so long as it is outside the area of the bottom 2 4K pages the final piece of remapping will continue without incident and the memory map becomes Interak RAM from 0000 to FFFF, with an image of ZMON at E000-EFFF. The required changes to the page descriptor registers are those needed to produce the following result:

1000-1FFF	FE1000-FE1FFF	(Interak)
0000-0FFF	FE0000-FE0FFF	(Interak)

Before the final jump to E000, remember to use the LDCTRL instruction one more time to restore the I/O Page Register back to its initial value of all zero, to enable I/O accesses to address the Interak I/O cards. (See section 3.3.4 on page 3-5 of the Zilog preliminary Technical Manual.)

6. There is one remaining thing to be done if ZYMON 2 is indeed the program to be used for this demonstration. Inspection of the listing for ZYMON 2 (ZYMON 2 Manual Issue June 1982, page 15) reveals that ZYMON 2 begins by testing the contents of location E000 to see if they are ROM, jumping to a label "ZERO" if not. I suggest that location 000B in the listing (ie E00B in the memory) be changed from 10H to 00 before the jump is made. Of course if ZYMON was seriously going to be used on the XMZC-1 card (not very likely, as there seems little demand for a 16 Megabyte tape-based system) then suitable change(s) can be made to ZYMON 2 before it is burnt into the EPROM for this card. (Even at 2400 baud, my calculations reveal that it would take about 20 hours to load a 16 Megabyte program from tape into RAM.)

5.1 APPENDIX 1: LIST OF ISBUS CONNECTIONS (WITH PARTICULAR REFERENCE TO MZC-1 CARD)

1A NIOREQ I/O Request (Output from CPU)

Input/Output request line. Active low. Indicates that the address bus lines AB0 to AB7 hold a valid address for an I/O access. (During an interrupt acknowledge cycle, and only then, NIORQ is produced at the same time as NM1(OCF); at all other times NM1(OCF) is produced together with NMREQ or NEMREQ, never with NIOREQ.)

The 8 address bits define which one of 256 ports are to be accessed. The XMZ280-1 card can access up to 64K of I/O ports (even 16 Megabytes if pushed to its ultimate). To prevent these ports conflicting with the existing 256 ports NIOREQ is denied to the bus during accesses to the extended space; instead a signal NEIOREQ "extended I/O request" is produced.

Both NIOREQ and NEIOREQ are produced simultaneously during an interrupt acknowledge cycle. This is the only time they ever occur together.

2A NMREQ Memory Request (Output from CPU)

Memory request line, active low; indicates that the 16-bit address bus, AB0-AB15, holds a valid address for a memory access, ie an access to one of the first 64K memory addresses is in progress.

The XMZ280-1 card also provides a further 8 address lines (AB16-AB23), defining a 16 Megabyte memory space. To prevent the addresses in the larger space conflicting with the existing 64K space NMREQ is denied to the bus during accesses to the extended space; instead a signal NEMREQ "extended memory request" is produced.

Both NMREQ and NEMREQ are produced simultaneously during an memory refresh cycle. This is the only time they ever occur together.

3A NWDS Write Data Strobe (Output from CPU)

Write line, active low. Indicates that the data bus has valid data to be written to a memory location or output to an I/O device.

4A NRDS Read Data Strobe (Output from CPU)

Read Line, active low. Indicates that the CPU wants to read data from a memory location or I/O device.

5A-20A AB15-AB0 16-bit Address Bus (Outputs from CPU)

These are the address lines for a normal memory access (ie when NMREQ or NEMREQ are present). In ISBUS "A" systems these 16 are all that are used , but in ISBUS "B" systems there are a further 8 lines (AB16-AB23) defined for addresses. The positive logic convention is used, i.e. a "0" is represented by a low voltage and a "1" by a more postive voltage. The address lines are usually driven from the CPU, but if the CPU card is rendered high impedance (disabled) these lines can be driven from other controllers (e.g. Direct Memory Access (DMA) devices).

When the access is to the I/O space (ie when NIOREQ or NEIOREQ are present) the lower 8 lines, AB7-AB0, carry the address of the I/O port being accessed and the other address lines carry information which depends on the particular microprocessor instruction being executed (usually the contents of a CPU register on lines AB8-AB15, and further addressing information on lines AB16-A23).

A final use of the address bus is to carry a "refresh address" for dynamic RAMs. This is issued at a time when the MPU is not using the bus (ie immediately after an opcode fetch, while the instruction the opcode represents is being decoded) and allows an easy refresh of dynamic RAMs. The indication that the address bus bears a refresh address is provided by the NRFSH signal. NMREQ and/or NEMREQ are also present but NRDS and NWDS are denied so that an actual memory access does not take place. As a result it is possible therefore to refresh an unlimited quantity of memory cards at once, with the same refresh signals. (Unlike the refresh address produced by the Z80-CPU which is only 7 bits, (ie AB0-AB6) the Z80280 produces a 10-bit Refresh, thus allowing easy use with larger than 64K dynamic RAMs)

21A NRST System Reset (Open Collector to or from CPU)

This signal is active (ie low) when the reset switch on the CPU card is reset. It is used to reset circuitry on all of the cards in the system which require such a signal. If the reset line is brought low by the action of a device other than the CPU then the CPU is also reset. The line is normally held high by a 1k pull-up resistor on the CPU card.

(The Kemitron MZB-3 CPU card does not comply with this specification in all details.)

So as not to run the risk of damage to data in the dynamic RAMs when the reset switch is operated, the CPU card produces a short duration pulse rather than a permanent signal for the duration of the reset switch operation. Furthermore the Z80280 internal timing ensures that its reset does not begin until the end of an internal cycle, and thus the CPU never resets during a memory access (because an incompleted access could corrupt a row of data in a dynamic RAM).

The manual and power-on reset is normally produced on the XMZ280 card, where the appropriate timing components are specified and fitted; if the alternative is adopted of applying a reset via the open collector NRST line, then the duration of this reset should be limited to about 50-100us to reset other peripherals satisfactorily whilst still preserving the contents of dynamic RAM.

22A-29A DB7-DB0 8-bit Data Bus (Bidirectional, 3-state)

These are the data lines used for 8-bit bidirectional data exchanges between the CPU or Controller and Memory or Input/Output Cards. Positive logic is used (as for the address lines previously described). This is an 8-bit system and these 8 lines are the only 8, but for systems using a 16-bit bus there are a further 8 lines defined on the "B" side of the bus.

30A (Unallocated) Daisy Chain 1 In (Daisy Chain Input)

31A (Unallocated) Daisy Chain 1 Out (Daisy Chain Output)

These two lines allow an additional and as yet unallocated priority daisy chain to be set up. Viewed from the plug in card side of the edge connector, the output of the card on the left is connected to the input of the card on the right. The convention used will be active low.

The two lines Daisy Chain 1 In and Daisy Chain 1 Out are linked in track on the XMZ280-1 card.

32A NRFSH DRAM Refresh (Output from CPU)

Refresh line, active low, indicates that the lower 10 address bus lines (AB0-AB9) hold a refresh address for dynamic RAMs. It is the duty of the CPU or Controller card to ensure that refresh addresses are issued frequently enough for proper refresh of dynamic RAMs in the system. If

the dynamic RAM card in use has its own refresh counter the CPU or Controller is relieved of this duty; the NRFSH signal can then merely be used to signal to the dynamic RAM card that now is an appropriate time for a refresh to take place. The Z80280 can be programmed for various different refresh rates etc, including no refresh. Naturally this programming must suit the RAM in use.

33A CLK System Clock (Output from CPU)

This is the single phase system clock produced by the Z80280 to sysnchronise transactions with other cards in the system. It can be the same as, or one half, or one quarter the frequency of the internal clock used by the Z80280 itself, which in turn is one-half of the frequency of the crystal oscillator or input to the Z80280. Depending on the crystal usedthe scaling factor chosen (by jumper links on the XMZ280-1 card) should be such as to correspond to the clock used in a Z80A system, ie 4 MHz.

34A NWAIT Wait State Request (Open Collector Input to CPU)

This signal, if active (ie low) during a specified time during a Z80280 machine cycle indicates that an addressed device requires "wait" states to be inserted into the current Memory or I/O operation. This is usually to accommodate slow devices which require an increased time before data can be provided or taken, but another use for "wait" states is to allow synchronisation between the CPU and the Memory or I/O device if timing is critical (e.g. for some floppy disk controllers, or VDU designs). If the CPU is providing dynamic RAM refresh, prolonged use of "wait" states should be avoided as it may predjudice proper refreshing of the dynamic RAM. (There is no need to panic on this point - the maximum wait state allowed can be calculated fairly easily for specific cases.) A 1k pull-up resistor is fitted on the CPU card.

The Z80280 allows for wait states to be added into the part of the cycle which establishes the settling time for the "interrupt priority daisy chain", see Technical Manual on Z80280 for details.

35A +12V +12V Power Supply Rail
36A +12V +12V Power Supply Rail

Together with their "opposite numbers" 35B and 36B on the other side of the connector, these are the conductors of +12V regulated d.c. power from the power supply to the system. Generally these rails carry no more than an Ampere or two, and if this is the case there is no need to reinforce the copper track on the bus board.

37A Pol Polarisation Slot

This line is not available for signals as this position is removed for polarisation. An important secondary purpose for the polarisation key fitted in each connector at this position is to pull the cards into line - to prevent individual connectors shorting between adjacent contacts.

38A -12V -12V Power Supply Rail
39A -12V -12V Power Supply Rail

Together with 38B and 39B on the other side of the connector, these are the conductors of -12V regulated d.c. power from the power supply to the system. Generally these rails carry much less than an Ampere, and if so there is no need to reinforce the copper track on the bus board.

40A 0V 0V Power Supply Rail
41A 0V 0V Power Supply Rail

Together with 40B and 41B on the other side of the connector, these provide the return path to 0V and Earth for the currents flowing in the +12V, +5V and -12V rails. Usually the algebraic sum of these currents is more than the absolute value of any one of them, so these rails carry most current of all. If the current is substantial, say 10 Amperes or more, it may be advisable to reinforce these rails with lengths of heavy gauge tinned copper wire e.g. 18 swg. This is quite easy to do because the wire can lie between the two adjacent rails provided for this voltage on each side of the connector.

42A +5V +5V Power Supply Rail
43A +5V +5V Power Supply Rail

Together with 42B and 43B on the other side of the connector, these are the conductors of +5V regulated d.c. power from the power supply to the system. If the current drawn from this rail is substantial, say approaching 10 Amperes or more, it may be advisable to reinforce these rails with lengths of heavy gauge tinned copper wire e.g. 18 swg. This is quite easy to do because the wire can lie between the two adjacent rails provided for this voltage on each side of the connector.

"B" Side Connectors

1B NEIOREQ Extended I/O Request (Output from CPU)

Extended Input/Output request line. This is an active low signal which is only used in systems with an extended I/O space (i.e. using addresses AB16 to AB23). It has a similar function to the NIOREQ line on 1A of the bus, but NEIOREQ is only issued when the extended port space is brought into play.

Indicates that address bus lines AB0 to AB7 and AB16 to AB23 (and AB8 to AB15 if they are used for I/O) hold a valid address for an I/O access. (During an interrupt acknowledge cycle, and only then, NEIORQ is produced at the same time as NM1(OCF); at all other times NM1(OCF) is produced together with NMREQ or NEMREQ, never with NIOREQ.)

Both NIOREQ and NEIOREQ are produced simultaneously during an interrupt acknowledge cycle. This is the only time they ever occur together.

2B NEMREQ Extended Memory Request (Output from CPU)

Extended Memory Request. This is a signal (active low) which is used to control access to the 16 Megabyte Memory Space (i.e. 24-bit Address Bus). It has a similar function to the NMREQ line on 1B of the bus, but NEMREQ is only issued when the extended address space is to be used.

Both NMREQ and NEMREQ are produced simultaneously during an memory refresh cycle. This is the only time they ever occur together.

3B NADS Address Strobe (Output from CPU)

This is an active low signal which has been allocated for possible future use. Once 32-bit (and beyond?) CPUs are introduced, the idea of separate address and data buses (i.e. 32+32=64 bits) becomes unwieldy; it is possible that the normal implementation will be to use a multiplexed address-data bus. As ISBUS has 16 data bus lines defined, and 24 address bus lines, i.e. 16+24 = 40 lines in total this will be more than enough for 32-bits. When this happens, an extra strobe will be required: NADS, which indicates that the address-data bus holds a 32-bit address. When NADS is complete over the bus will then be driven with the data corresponding to that address.

The possibility of non-multiplexed 32-bit address and data buses has not been ruled out, but if they are introduced this will need very extensive alterations to the bus signals. Indeed so much will have to change that the user may then have to say goodbye to the old cards forever. A better idea in our view, if a lot of investment has been made in non 64 address and data bit systems, is that then more than one CPU be run simultaneously, on separate buses, sharing the computational load between each other.

Notwithstanding the above, the Z80280 does have an address strobe signal; used for demultiplexing those elements of its address and data buses which share common package pins. In case the signal is of use to any user it has been brought out to this bus pin, but we ourselves do not anticipate using it.

4B NDIRIN Direction In (to MPU) (Output from CPU)

The Z80280, like its predecessors the Z80 and the 64180 inputs data at times other than during conventional I/O or memory reads. (Time eg when an "interrupt vector" is being fetched, or in the case of the CPU chips which operate DMA (Direct Memory Access), when data is being read from memory for that purpose). In the past the required buffer control gating has been tedious to implement; the Z80280 carries this out internally, and has a pin called input enable for controlling the input buffer. To preserve compatibility with more mundane processors, eg the Z80, which is far from being dead yet, we probably will not use the direction signal, but it is brought out here for the benefit of those users who do require it.

It is an active low signal, ie it goes to a logic "0" when the direction of data transfer is towards the CPU.

5B NM1(OCF) M1 (Opcode Fetch) (Output from CPU)

This is the M1 signal from the Z80280 CPU. It is used to indicate that the CPU is fetching an op-code. (M1 is also asserted by the CPU with IORQ during the special conditions of an interrupt acknowledge cycle.)

6B (Unallocated) Extended Signal 1
7B (Unallocated) Extended Signal 2

Although it is an irritating feature of a bus standard to have unallocated lines, these are so at present. They should not be employed for the user's own purposes, as this may conflict with whatever signal is allocated to them in

the future. The direction and purpose is naturally unspecified at the moment, but if in the future they carry control signals they probably will be active low, and outputs from the CPU card.

8B NZCMAINS Zero Crossing (Mains) (Output from CPU in direction, although not produced on CPU card)

This is not implemented on this card.

This signal is a series of short pulses (active low), synchronised with the zero crossings of the ac mains input to the computer. It has three main purposes - firstly for triggering ac power controlling devices and the like (so that ac powered circuits can be switched on only when the mains voltage is passing through zero thus avoiding current surges and unwanted generation of electromagnetic interference) - secondly to provide advance warning of an imminent power failure (since when the ac mains fails there will be a period of continuous "zero crossing" before dc power fails) - and finally to offer a simple timing pulse of known repetition frequency which is independent of any software (this can be used for example to set a "watchdog" timer to alert of or restart a "looping" or "crashed" program).

9B NNMI Non Maskable Interrupt (Open Collector to or from CPU)

Active low, non-maskable interrupt line. 1k pull-up resistor fitted on the CPU card.

10B NINTA Interrupt A (Open Collector to or from CPU)

Active low, maskable interrupt line, 1k pull-up resistor fitted on the CPU card.

11B NINTB Interrupt B (Open Collector to or from CPU)

This is a further active low line similar in function to Interrupt A, 1k pull-up fitted on the CPU card.

12B NINTC Interrupt C (Open Collector to or from CPU)

This is a further active low line similar in function to Interrupt A, 1k pull-up fitted on the CPU card.

13B-20B AB16-AB23 Extended Address Bus (TTL Level Output from CPU)

These provide an extra 8 positive logic address lines to extend the memory space memory space to 16 Megabytes. The extra lines are produced by the CPU card. When these lines are used the NEMREQ (Extended Memory Request) signal replaces the ordinary NMREQ signal of a non-extended (64K) system, to prevent any conflict where the old 64K system memory space is shared with the new 16 Megabyte system. NMREQ is issued for a 64K space within the full 16 Megabyte space. The precise memory address range chosen on this card for the 64K space is FE0000-FEFFFF.

Similarly these address lines provide an extended I/O Port Address space, activating NEIOREQ (Extended I/O Request) for the larger space, and denying the smaller NIOREQ (I/O Request signal of earlier systems.) The precise memory address range chosen on this card for the smaller (unextended) I/O space is 00xx00-00xxFF, wher "xx" means don't care, or is the special case of the outputs of one of the CPU register during an I/O transaction where this feature is being used to provide the ultimate 16 Megabytes of I/O addressing.

21B NBUSREQ Bus Request (Open Collector Input to CPU)

This is an active low signal which is output by some other card (e.g. Direct Memory Access Controller) to request the XMZ280-1 to relinquish its control of the bus. Most of the bus output lines from this card then go tri-state, at which point it generates the signal NBUSAK (q.v.) Care must be taken when using this technique to ensure that arrangements for dynamic RAM refresh (for example) are properly made as the Z80280 on this card is effectively switched off.

(We have chosen that the clock signal from this card is also rendered tri-state, so if one is needed by other cards it should be provided by the new controller of the bus. Beware also that the question of what effect interrupts should have is very vague.)

22B-29B DB15-DB8 Extended Data Bus (Bidirectional, 3-state)

Not implemented on this card, which is designed exclusively for 8-bit operation, in order to maintain compatibility with the entire range of cards produced over the last decade or so.

If they had been used their purpose would have been as follows: When systems require a 16-bit data bus, these lines carry the upper 8-bits of that data. They are directly

opposite the bus lines for the lower 8-bits and this enables easy conversion of existing 8-bit cards to use the uppper 8-bits instead, so that cards can be used in pairs to provide an easy upgrade to 16-bits. The positive logic convention is used, as described for the address lines on the "A" side.

30B NIEI Interrupt A Daisy In (Daisy Chain Input)
31B NIEO Interrupt A Daisy Out (Daisy Chain Output)

These active low signals are used by Z80 peripherals to form an interrupt daisy-chain for determination of interrupt priority. This is done in accordance with the rules set out for the Z80 family peripheral chips by the the manufacturers of the chips.

On this card the two lines NIEI and NIEO are linked in track.

32B NBAI Bus Available Daisy In (Daisy Chain open collector input)

On this card this signal is used in conjunction with 21B NBUSREQ and 33B NBAO (qv). If both the Bus Request line NBUSREQ and the Bus Available Daisy In line NBAI are low simultaneously then the Z80280 finishes its current operation and relinquishes its buses, ie the ones on the pins of the CPU chip itself and the ones on the XMZ280-1 card. It also drives low its Bus Available Daisy Out line, NBAO which is part of a daisy chain, thus the next card to the right (viewed from the front of the computer if the cards are mounted in the conventional fashion) receives permission to use the bus. The permission passes down the daisy chain until the card is reched which requested the use of the bus. (All of this is entirely hypothetical as we have no immediate intentions of using this feature. The necessary arbitration between the various bus "masters" which we can imagine, and other matters like dynamic RAM refresh, timeout of prolonged transactions and so on, is by no means as simple as this discussion has made it appear.)

A possible practical use we could forsee would be during experimental work with a new processor card where memory contents, disk accesses etc could be carried out using the a familiar system and control passed to and from the new card during the debugging phase.

33B NBAO Bus Available Daisy Out (Daisy Chain open collector output)

This active low signal forms a daisy chain with BAI. The output from this card is held at a logic "0" when the Z80280 has received and fulfilled a request to release the system bus. NBAO is high when this card is using the system bus.

Because NBAI and NBAO form part of a daisy chain, the physical order of the cards using it will determine their priority to access the bus.

34B 2XCLK 2 x MPU Clock (Output from CPU)

This is not implemented on this card.

We originally had planned that this line would carry a clock of twice the frequency of that on bus line 33A (ie 8 MHz for a Z80A-CPU, and that the signal CLK on pin 33A would be derived from this clock; CLK clock changing state on the negative transition of 2XCLK. Events have overtaken us in the design of this card. Zilog have included in the Z80280 on chip peripherals, a complete clock oscillator and control circuit. The output clock which like us they call "CLK" is a scaled factor of the input to the chip, and they have made no provision for a double frequency bus clock. This is not a matter of great importance to us, as we have not designed any cards which need a double frequency bus clock, and we have no plans to do so.

35B +12V +12V Power Supply Rail 36B +12V +12V Power Supply Rail
These are connected to the same +12V Power Supply Rail as described for connections 35A and 36A on the "A" side of the connector.

37B Pol Polarisation Slot -
This line is not available for signals as this position is removed for polarisation. Also it is used with the key fitted in each connector to pull the cards into line and prevent individual connectors shorting between adjacent contacts.

38B -12V -12V Power Supply Rail 39B -12V -12V Power Supply Rail
These are connected to the -12V Power Supply Rail - see the description for connections 38A and 39A on the "A" side.

40B 0V 0V Power Supply Rail 41B 0V 0V Power Supply Rail
The 0V Rail is on these two pins, and the 40A and 41A on the "A" side. See the description for 40A and 41B for full details.

42B +5V +5V Power Supply Rail 43B +5V +5V Power Supply Rail

These, and 42A and 43A on the other side of the connector form the +5V Power Supply Rail. See the description for 42A and 42B for more information.

Derived Signal

NINTAK: This signal indicates that the CPU is acknowledging an interrupt and expects an interrupt vector to be placed on the data bus.

The NINTAK signal can be produced as the output of an "OR" gate (e.g. 74HC32) which has NIOREQ and NM1(OCF) connected to its inputs. As it can be produced in this way anywhere it is needed there is no need to allocate a bus line to it.

5.2

APPENDIX 2
DISCUSSION OF CMOS INTERFACING

5.3 APPENDIX 3: WAIT STATE GENERATION.

The subject of wait state generation has been covered implicitly at various appropriate stages in this Manual. The purpose of this Appendix is to make some general points on the subject, and to explain specifically why the various arrangements for wait states were made as they were.

Why have Wait States?

(Space reserved for general discussion, similar to LKP-1 Manual Appendix 2)

Why use a Monostable to Set the Wait States?

(Space reserved for general discussion, similar to LKP-1 Manual Appendix 2)

Timing Diagram

(FUTURE) A detailed timing diagram is given in Section _._ of this Manual. It shows the various waveforms on the standard bus when an access is made to an address which triggers the wait state generation circuit on this or another card.

5.4 APPENDIX 4: MZC-1 APPLICATIONS.

(Space reserved for general remarks similar to LKP-1 Manual Appendix 3.)

5.5 APPENDIX 5: EDUCATIONAL ASPECTS.

Although the Interak System represents a "real" computer rather than an artificial microprocessor training aid there are some distinct benefits in demonstrating various aspects of digital logic design on such a "real" computer where the student can appreciate that the theoretical techniques, such as truth tables, Boolean logic and the like have some practical application.

. . . . . etc etc (Space reserved for general discussion as in LKP-1 Manual Appendix 4)

5.6 APPENDIX 6: BOOT ROM SPECIFICATIONS

As I was talking earlier of the requirements of a new ROM to run on this card I think it is time to draw up some formal specifications.

Unfortunately lack of experience with the Z80280 in our system means that I cannot give specific guidelines yet on how the memory management should be handled by application programs, whether or not the ROM should contain any elements of an operating system and so on.

However, here is a framework on which to build:

The program should begin like this:

```
XX0000      Jump to program cold start
XX0003      Jump to program warm start
XX0004-     ASCII text:
               Name of program          "XXXXXXXX.XXX"
               Date of creation         "YYMMDD"
               Version Number           ".NNN"
               Latest update            "YYMMDD"
               Copyright notice         "(c)XXXXXXXX...."
               Supplier Name/Code       "XXXXXXX"
               Supplier Phone           "1234567890"
               Supplier Town, Postcode  "XXXXX ..123"
               Author Name/Code         "XXXXXX..."
               Author Phone             "1234567890"
               Author Town, Postcode    "XXXXX ..123"
               Spare                    "       ...."
XXHHHH (a good round hexadecimal number)
               ROM Location             "0034ABCD"
               Last byte of Code        "00NNNNNN"
               Last address in ROM      "12345600"
               Check Sum of ROM         "1234"
               Processor type           "ZZZZ"
               Processor speed          "0000"
               Spare                    "       ...."
XXHHH2 (a good round number)
               Start of program
               Jump Tables
               Data Tables, eg lookup tables
               Text Strings used in program
               Main Program loop
               Subroutines used by Main Program
               End of Program

  Fill to end with FF bytes
```

Any other suggestions, comments let me know.

5.7 APPENDIX 7: LATEST Z80280 INFORMATION

In this section is included any extra information I have found on the Z80280, notes of bugs, changes in timing specs etc.

17 JUL 1987

Z28[illegible]

MACRO

an affiliate of
EXXON Corporation

[illegible] Dell Avenue
Campbell California 95008

Telephone 408 37[illegible]
TWX 910 338 7621

This document is a list of known bugs on the Z280 silicon, Revision G.

1. Possible Cache Corruption

This bug happens only when the Z280 is used the Z80 bus mode. It can be attributed to the fact that the Z280 is designed as a 16-bit device with its internal logic geared towards the transfer of words instead of bytes.

In the Z80 bus mode with the cache enabled, a memory read of word quantity data (not instruction) causes the cache to be updated incorrectly. This happens only if the word data address is even aligned. This is best explained in the following example:

The stack printer (xSP) is pointing to location 1004H. The contents of external memory (not cache) locations 1004H and 1005H are "BB" and "CC" respectively. When a "POP HL" instruction is executed, the problem will exhibit itself.

For correct operation, the following events should occur:

a) Memory Read Transaction with address "001004" valid at time of address strobe (AS). When read (RD) is asserted, the data bus should contain "BB." Cache location "001004" should be updated to "BB" and validated. Register L should now contain "BB" as well.

b) Memory Read Transaction with address "001005" valid at time of address strobe (AS). When read (RD) is asserted, the data bus should contain "CC". Cache location "001005" should be updated to "CC" and validated. Register H should now contain "CC" as well.

Due to the errata, the following takes place instead:

a) Memory Read Transaction with address "001004" valid asserted at time of address strobe (AS). When read (RD) is asserted, the data bus contains "BB." Cache location "001004" inadvertently gets updated to "10" (contents of A15-A8) instead of "BB." Cache location "001005" gets updated to "BB" instead of "CC." Register L now contains "BB" as well.

b) No second Memory Read Transaction takes place externally due to the fact that the cache now has "valid" data at location "001005" and supplies that data to the instruction. Register H now contains "BB" as well.

Macro 69XGM
PHLIPPINES

This anomally can be circumvented through any of the following methods:

a) Use Z-Bus mode instead of Z80.

b) Ensure that word data is odd-aligned. This will force two read cycles to occur in order to fetch the data (Z80-bus mode).

c) Ensure that no code resides in the same 16-byte memory space (0-F0 that data resides in (A23-A4 cannot be the same) (Z80 Bus mode).

d) Disable cache (Z80-bus mode).

2. Burst Mode Memory Transactions (Z-bus mode only)

It has been discovered that the burst memory cycle does not function as described when in the X2 and X4 bus clock modes. It does function as specified in the X1 bus clock mode. What happens is that the IE signal behaves as if it were a normal memory access cycle.

This should not impact any applications because the burst mode is used for high performance, and the highest performance should be observed with the X1 bus clock mode.

SECTION 6:

DIAGRAMS AND TABLES

6.__ MZC-1 BUS ALLOCATIONS

(ISBUS SIGNALS USED OR CREATED ON XMZC-1)

Overleaf is a list of the ISBUS signals with specific reference to this card.

ISBUS-1 SIGNAL ALLOCATIONS

No.	Name	Type	Signal Description	No.	Name	Type	Signal Description
1A	NIOREQ	(1)	I/O Request	1B	NEIOREQ	(1)	Extended I/O Request
2A	NMREQ	(1)	Memory Request	2B	NEMREQ	(1)	Extended Memory Request
3A	NWDS	(1)	Write Data Strobe	3B	NADS	(1)	Address Strobe
4A	NRDS	(1)	Read Data Strobe	4B	NDIRIN	(1)	Direction In (to MPU)
5A	AB15	(1)	Address Bus 15	5B	NM1(OCF)	(1)	M1 (Opcode Fetch)
6A	AB14	(1)	Address Bus 14	6B	-*-	(1)	Extended Signal 1
7A	AB13	(1)	Address Bus 13	7B	-*-	(1)	Extended Signal 2
8A	AB12	(1)	Address Bus 12	8B	-*-	(1)	Zero Crossing (Mains)
9A	AB11	(1)	Address Bus 11	9B	NNMI	(O)	Non Maskable Interrupt
10A	AB10	(1)	Address Bus 10	10B	NINTA	(O)	Interrupt A
11A	AB9	(1)	Address Bus 9	11B	NINTB	(O)	Interrupt B
12A	AB8	(1)	Address Bus 8	12B	NINTC	(O)	Interrupt C
13A	AB7	(1)	Address Bus 7	13B	AB23	(1)	Extended Address Bus 23
14A	AB6	(1)	Address Bus 6	14B	AB22	(1)	Extended Address Bus 22
15A	AB5	(1)	Address Bus 5	15B	AB21	(1)	Extended Address Bus 21
16A	AB4	(1)	Address Bus 4	16B	AB20	(1)	Extended Address Bus 20
17A	AB3	(1)	Address Bus 3	17B	AB19	(1)	Extended Address Bus 19
18A	AB2	(1)	Address Bus 2	18B	AB18	(1)	Extended Address Bus 18
19A	AB1	(1)	Address Bus 1	19B	AB17	(1)	Extended Address Bus 17
20A	AB0	(1)	Address Bus 0	20B	AB16	(1)	Extended Address Bus 16
21A	NRST	(O)	System Reset	21B	NBUSREQ	(O)	Bus Request
22A	DB7	(2)	Data Bus 7	22B	-*-	(2)	Extended Data Bus 15
23A	DB6	(2)	Data Bus 6	23B	-*-	(2)	Extended Data Bus 14
24A	DB5	(2)	Data Bus 5	24B	-*-	(2)	Extended Data Bus 13
25A	DB4	(2)	Data Bus 4	25B	-*-	(2)	Extended Data Bus 12
26A	DB3	(2)	Data Bus 3	26B	-*-	(2)	Extended Data Bus 11
27A	DB2	(2)	Data Bus 2	27B	-*-	(2)	Extended Data Bus 10
28A	DB1	(2)	Data Bus 1	28B	-*-	(2)	Extended Data Bus 9
29A	DB0	(2)	Data Bus 0	29B	-*-	(2)	Extended Data Bus 8
30A	-*-	(D)	Daisy Chain 1 In	30B	-*-	(D)	Interrupt A Daisy In
31A	-*-	(D)	Daisy Chain 1 Out	31B	-*-	(D)	Interrupt A Daisy Out
32A	NRFSH	(1)	DRAM Refresh	32B	NBAI	(D)	Bus Available Daisy In
33A	CLK	(1)	MPU Clock	33B	NBAO	(D)	Bus Available Daisy Out
34A	NWAIT	(O)	Wait State Request	34B	-*-	(1)	2 x MPU Clock
35A	+12V	(P)	+12V Power Supply	35B	+12V	(P)	+12V Power Supply
36A	+12V	(P)	+12V Power Supply	36B	+12V	(P)	+12V Power Supply
37A	Pol	-	Polarisation Slot	37B	Pol	-	Polarisation Slot
38A	-12V	(P)	-12V Power Supply	38B	-12V	(P)	-12V Power Supply
39A	-12V	(P)	-12V Power Supply	39B	-12V	(P)	-12V Power Supply
40A	0V	(P)	0V Power Supply	40B	0V	(P)	0V Power Supply
41A	0V	(P)	0V Power Supply	41B	0V	(P)	0V Power Supply
42A	+5V	(P)	+5V Power Supply	42B	+5V	(P)	+5V Power Supply
43A	+5V	(P)	+5V Power Supply	43B	+5V	(P)	+5V Power Supply

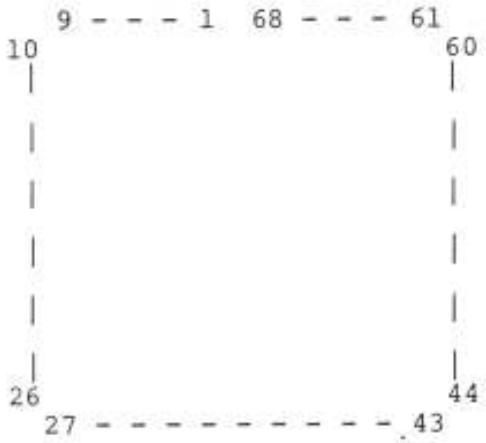
"O" = Open Collector; "1" = Unidirectional Bus; "Z" = Bidirectional Bus;
"D" = Daisy Chain In/Out; "P" = Power Supply;
"-*-" = Signal not implemented on this board

6.3 68-pin Chip Pins to Socket Translation

Here is a chart which shows how the pins on the PLCC carrier integrated circuit translate to the pins on the socket.

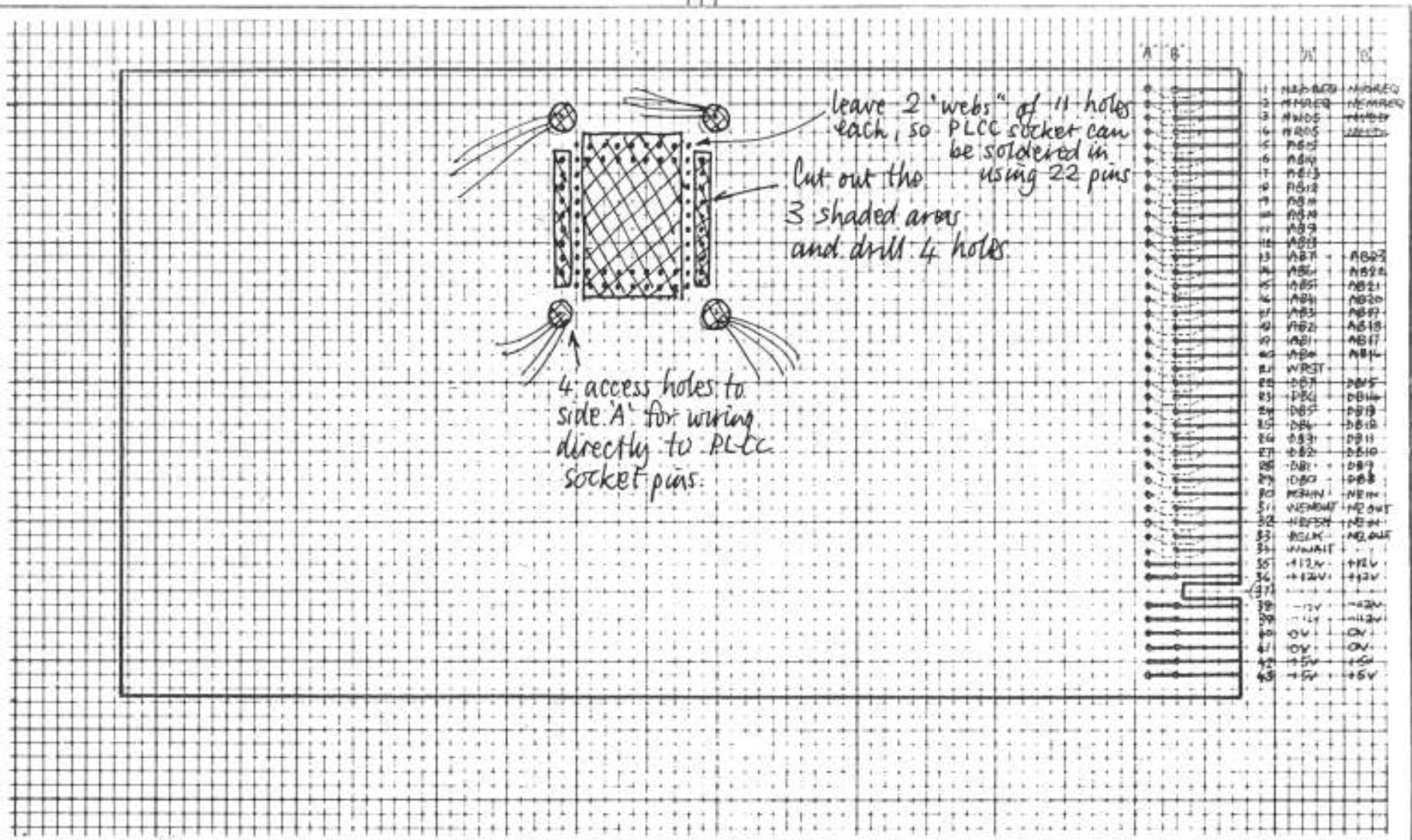
*** WARNING THESE ALL REQUIRE CHECKING ***

PLCC PINS



SOCKET PINS

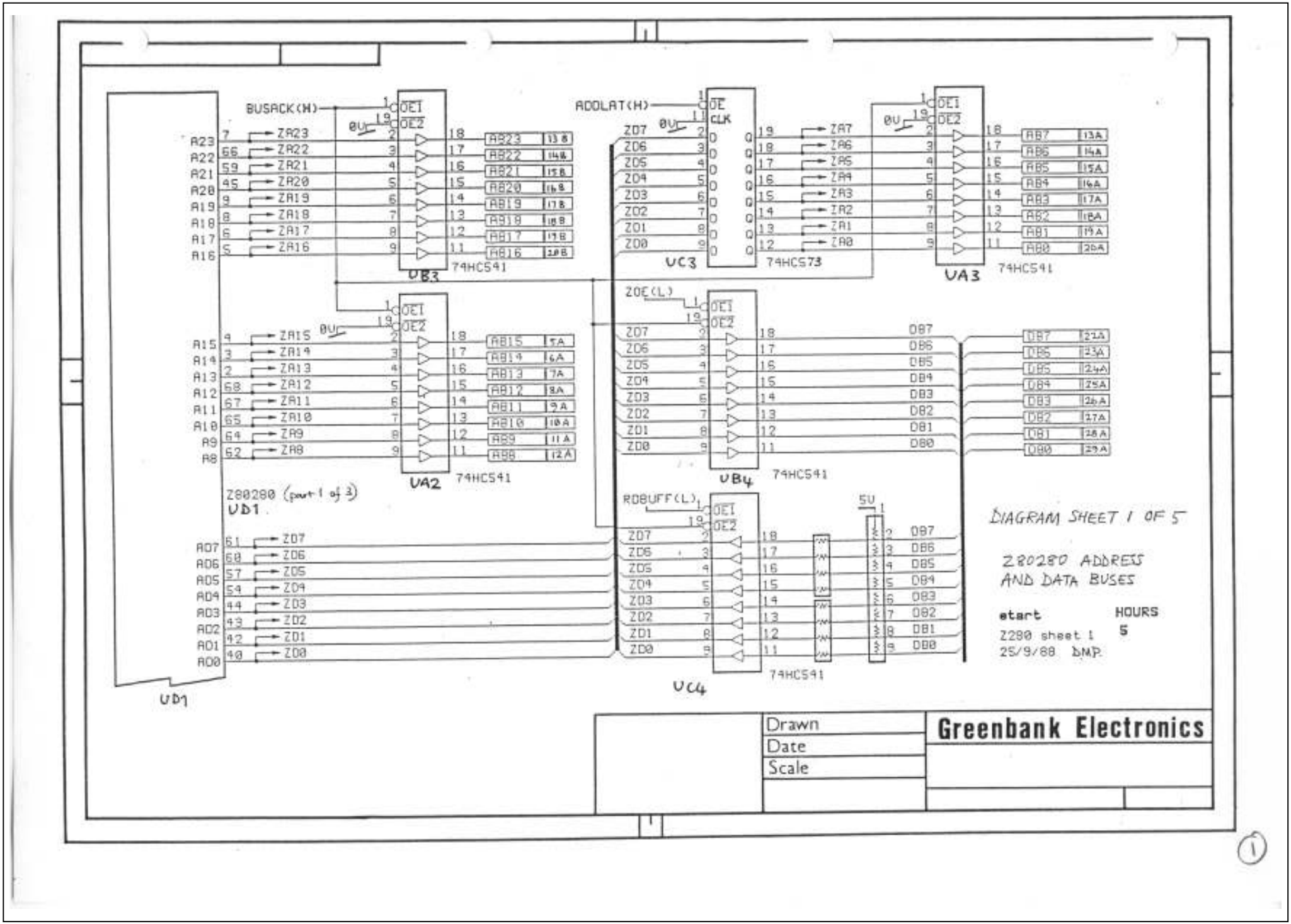
	9	7	5	3	1	67	65	63	61	
10	11	8	6	4	2	68	66	64	62	60
12	13								59	58
14	15								57	56
16	17								55	54
18	19								53	52
20	21								51	50
22	23								49	48
24	25								47	46
26	28	30	32	34	36	38	40	42	45	44
	27	29	31	33	35	37	39	41	43	

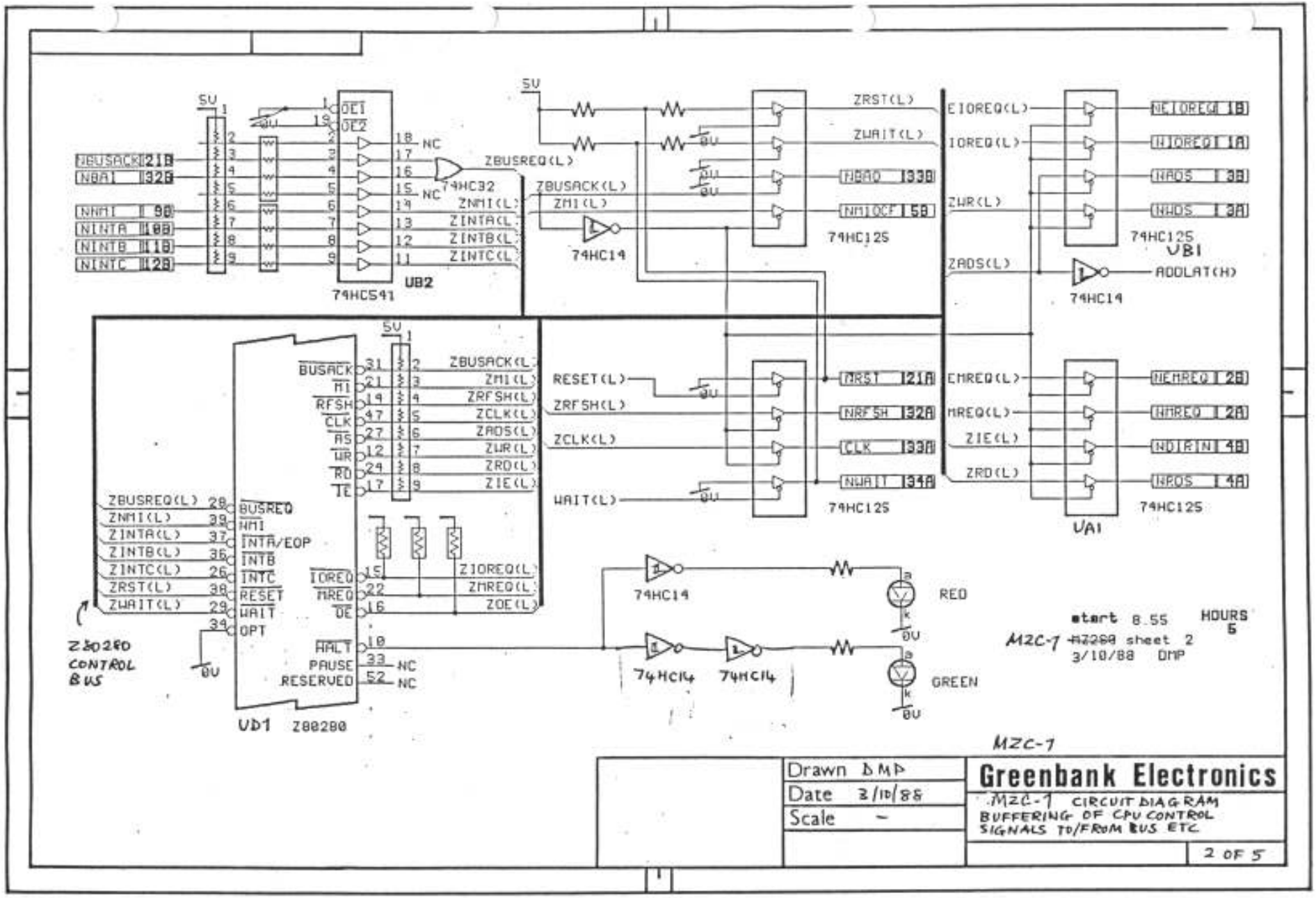


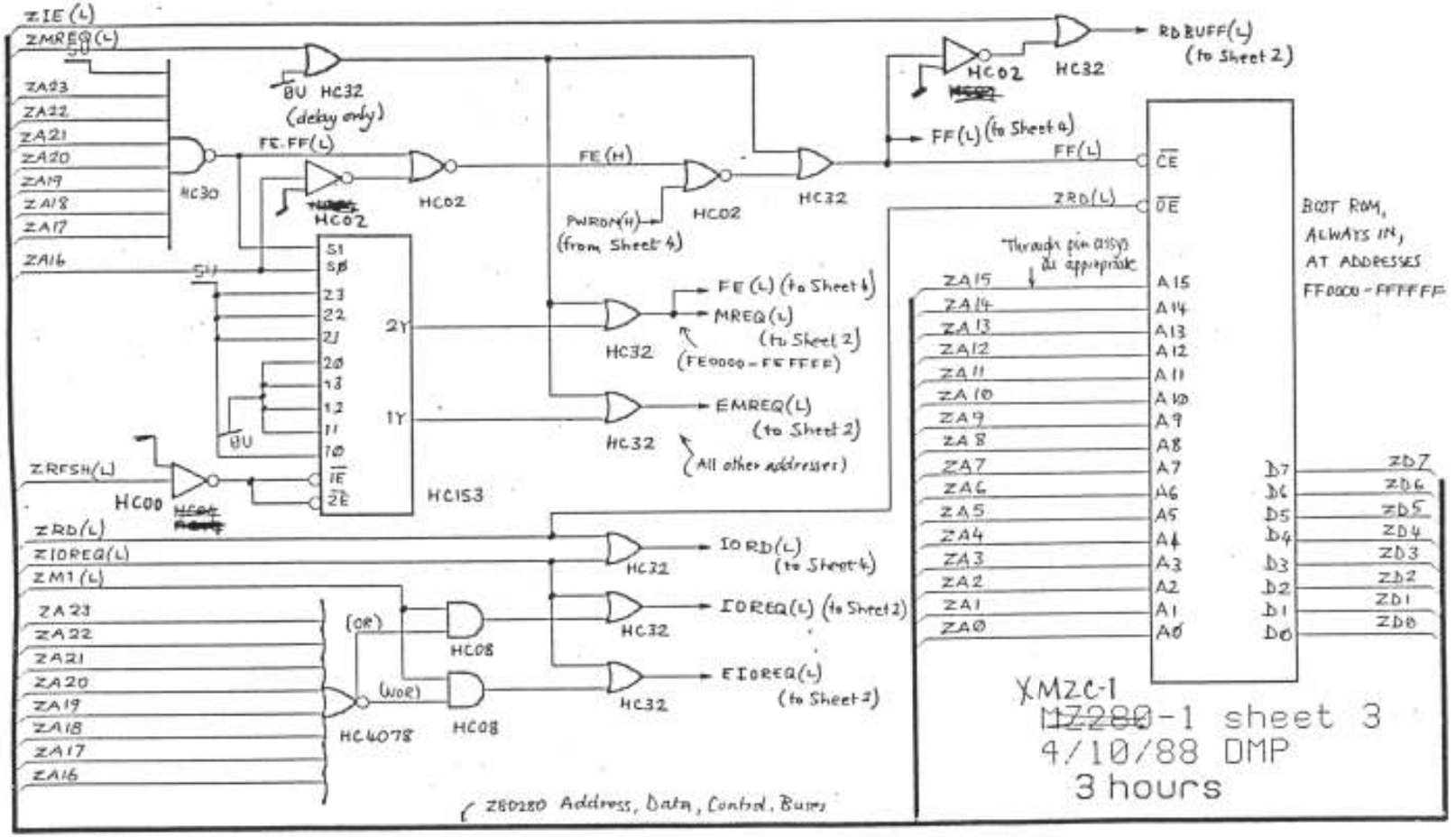
Drawn	BMP
Date	21/9/88
Scale	1:1

Greenbank Electronics

SUGGESTED HOLE CUT-OUT FOR PROTOTYPING 68 PIN PLCCS ON DIP-1

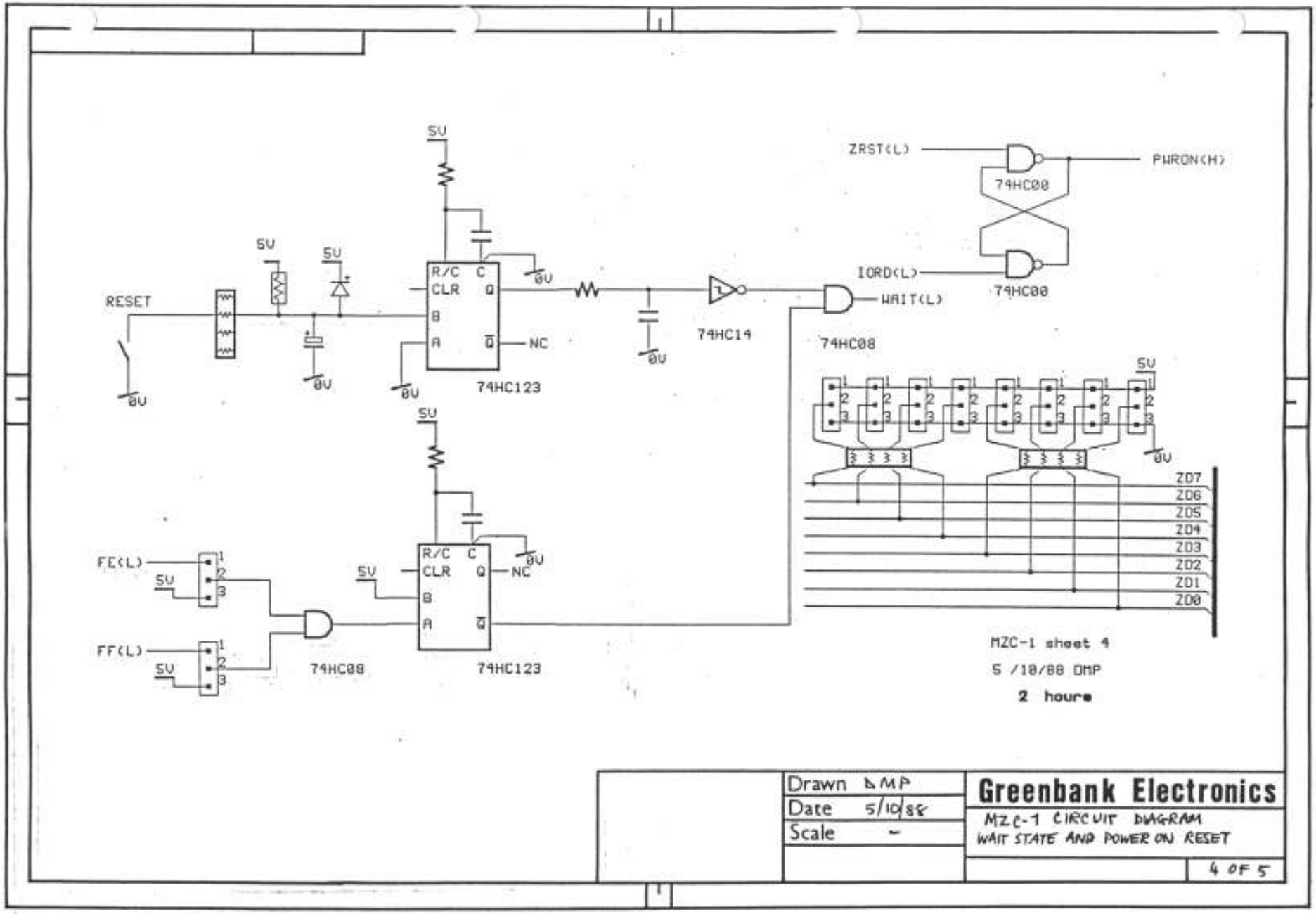

BUSACK(H)
ADDLAT(H)
ZOE(L)
RDBUFF(L)
5V
0V
74HC541
74HC573
UA2
UA3
UB3
UB4
UC3
UC4
UD1
Z80280 (part 1 of 3)
DIAGRAM SHEET 1 OF 5
Z80280 ADDRESS AND DATA BUSES
start
HOURS
5
Z280 sheet 1
25/9/88 DMP
Drawn
Date
Scale
Greenbank Electronics

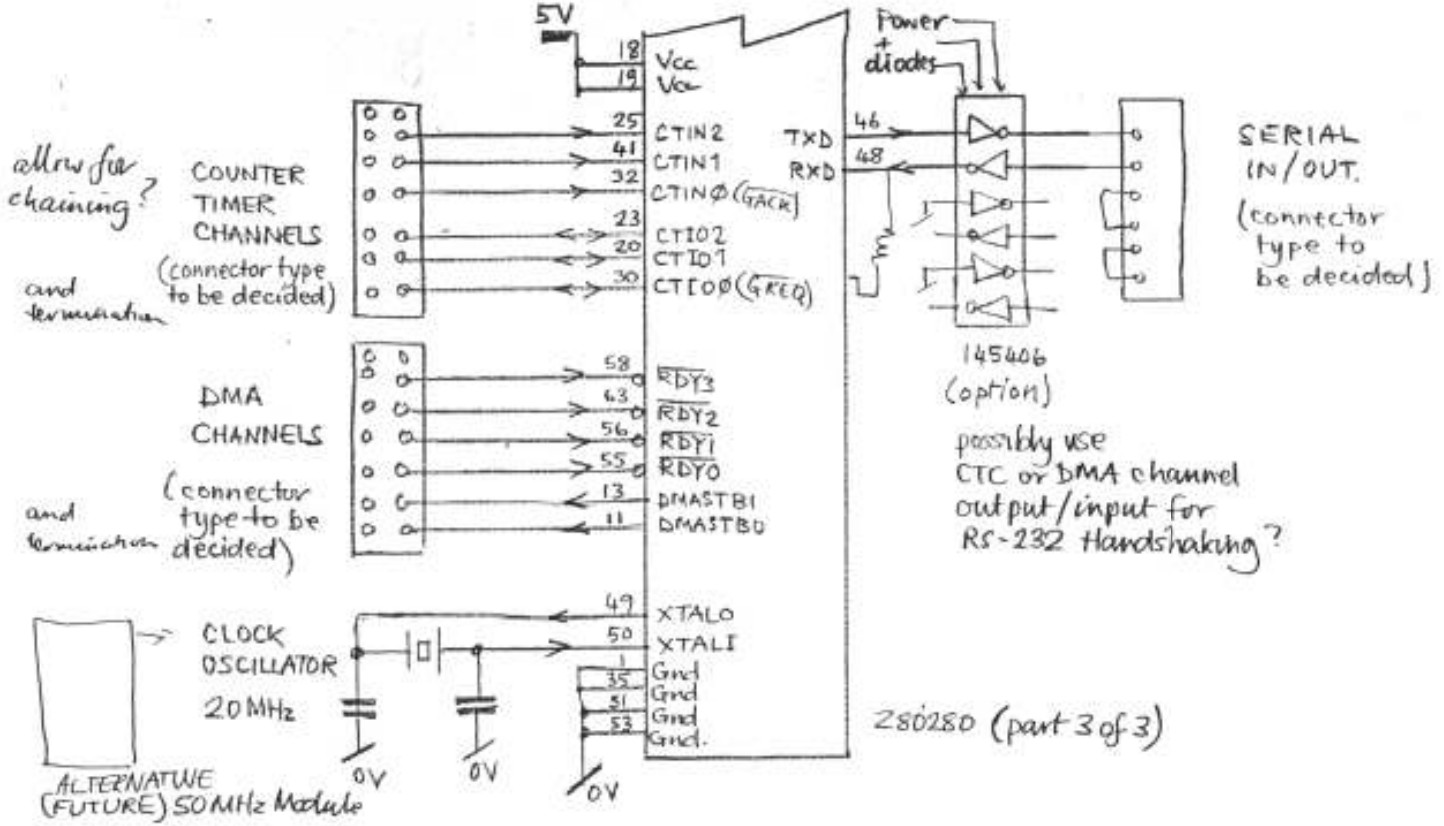

5V
ZBUSACK 21B
NBAI 32B
NNMI 9B
NINTA 10B
NINTB 11B
NINTC 12B
74HC541
UB2
74HC32
ZBUSREQ(L)
ZNMI(L)
ZINTA(L)
ZINTB(L)
ZINTC(L)
ZBUSACK(L)
ZMI(L)
74HC14
ZRST(L)
ZWAIT(L)
NBAQ 33B
NM1OCF 5B
74HC125
EIOREQ(L)
IOREQ(L)
ZWR(L)
ZADS(L)
NEIOREQ 1B
NIOREQ 1A
NRDS 3B
NWDS 3A
VB1
ADDLAT(H)
BUSACK
MI
RFSH
CLK
AS
WR
RD
IE
ZBUSACK(L)
ZMI(L)
ZRFSH(L)
ZCLK(L)
ZADS(L)
ZWR(L)
ZRD(L)
ZIE(L)
RESET(L)
ZRFSH(L)
ZCLK(L)
WAIT(L)
NRST 21A
NRFSH 32A
CLK 33A
NWAIT 34A
EMREQ(L)
MREQ(L)
ZIE(L)
ZRD(L)
NEMREQ 2B
NMREQ 2A
NDBIN 4B
NRDS 4A
VA1
ZBUSREQ(L)
ZNMI(L)
ZINTA(L)
ZINTB(L)
ZINTC(L)
ZRST(L)
ZWAIT(L)
BUSREQ
NMI
INTA/EOP
INTB
INTC
RESET
WAIT
OPT
IOREQ
MREQ
OE
ZIOREQ(L)
ZMREQ(L)
ZOE(L)
HALT
PAUSE
RESERVED
NC
Z80 CONTROL BUS
UD1 Z80280
74HC14
RED
GREEN
start 8.55 HOURS 5
MZC-1 ~~MZ80~~ sheet 2
3/10/88 DMP
MZC-1
Drawn DMP
Date 3/10/88
Scale -
Greenbank Electronics
MZC-1 CIRCUIT DIAGRAM
BUFFERING OF CPU CONTROL SIGNALS TO/FROM BUS ETC
2 OF 5



	Drawn D.M.P	Greenbank Electronics
	Date 4/10/88	~~XMZ280~~-1 CIRCUIT DIAGRAM
	Scale -	BOOT ROM; I/O AND MEMORY EXPANSION CONTROL SIGNALS
		3 OF 5

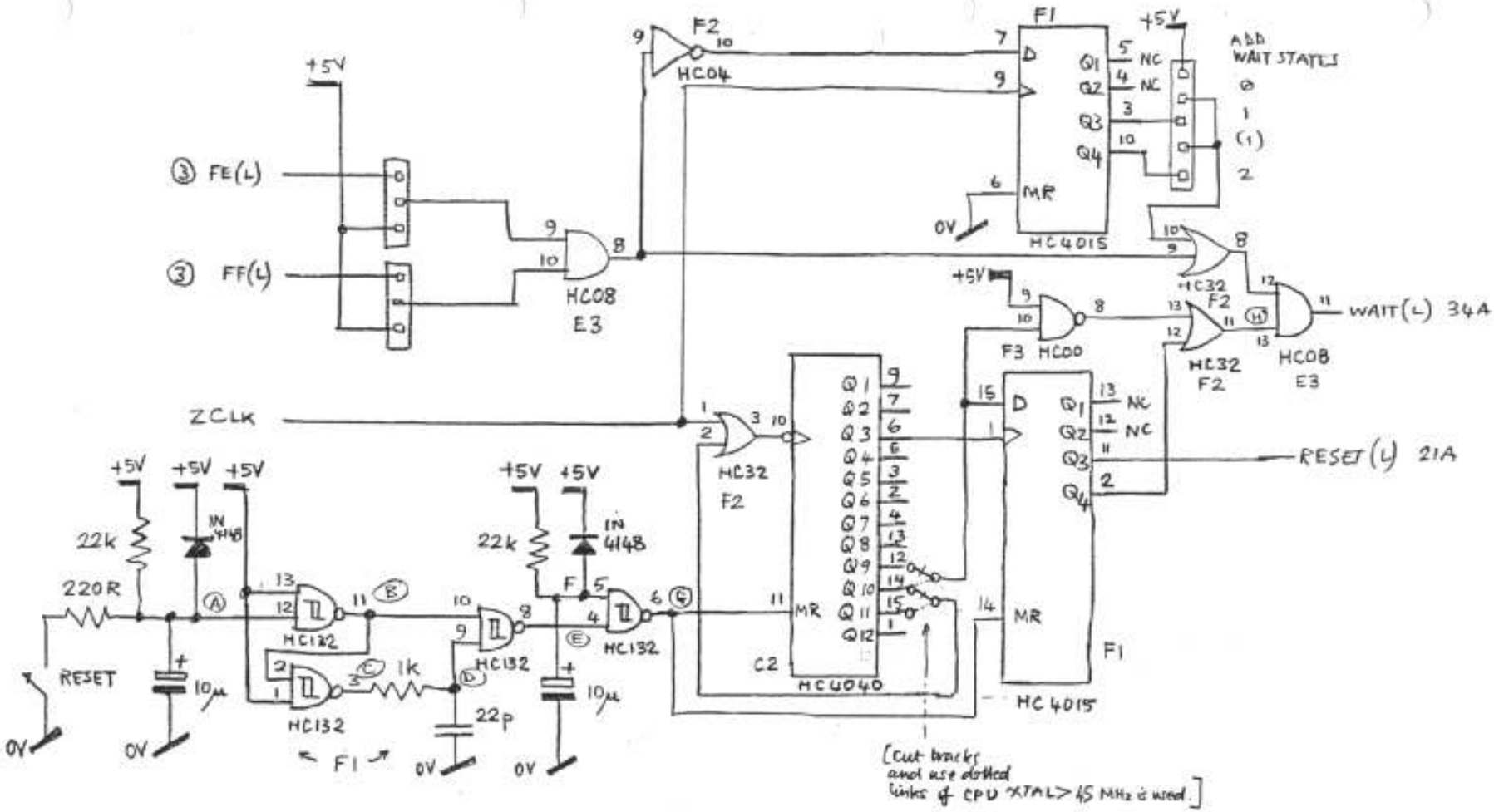
XMZC-1


5V
0V
RESET
R/C
C
CLR
Q
B
A
NC
74HC123
74HC14
74HC08
ZRST(L)
PWRON(H)
74HC00
IORD(L)
WAIT(L)
ZD7
ZD6
ZD5
ZD4
ZD3
ZD2
ZD1
ZD0
FE(L)
FF(L)
MZC-1 sheet 4
5 /10/88 DMP
2 houre
Drawn DMP
Date 5/10/88
Scale -
Greenbank Electronics
MZC-1 CIRCUIT DIAGRAM
WAIT STATE AND POWER ON RESET
4 OF 5



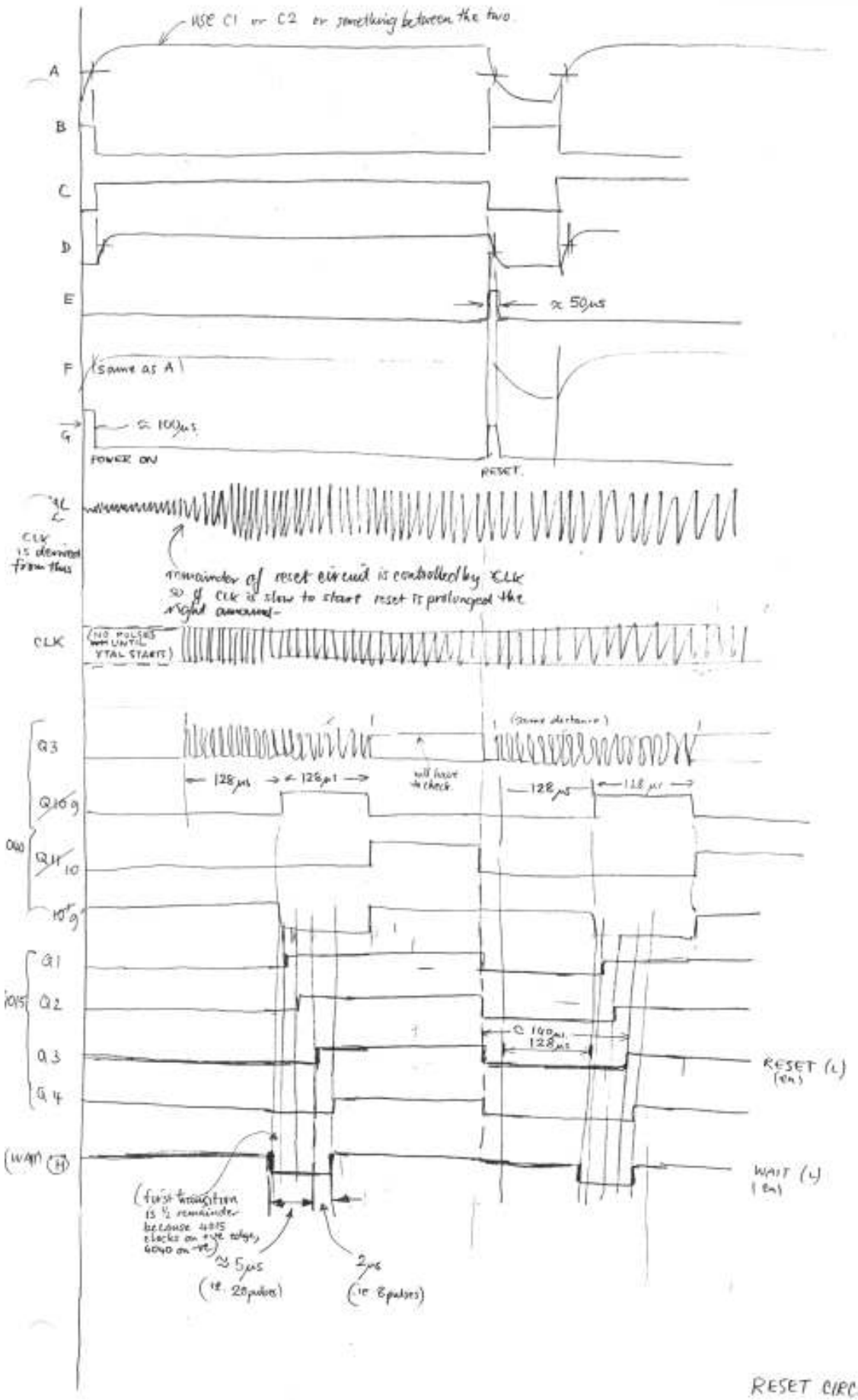
Z80280 "ON-CHIP" PERIPHERALS
DIAGRAM SHEET 5 OF 5

DMP 21/9/88



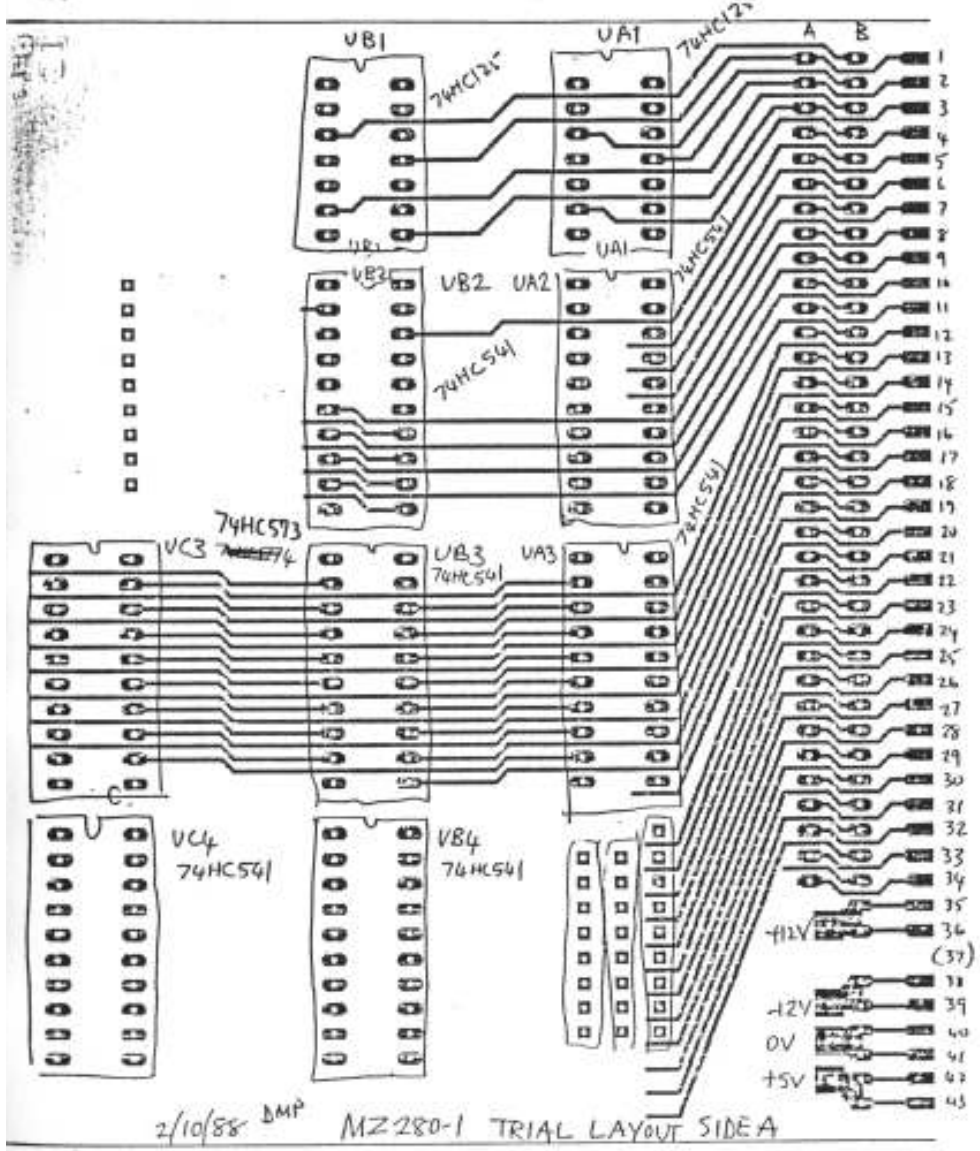
REVISED RESET AND WAIT STATE CIRCUIT

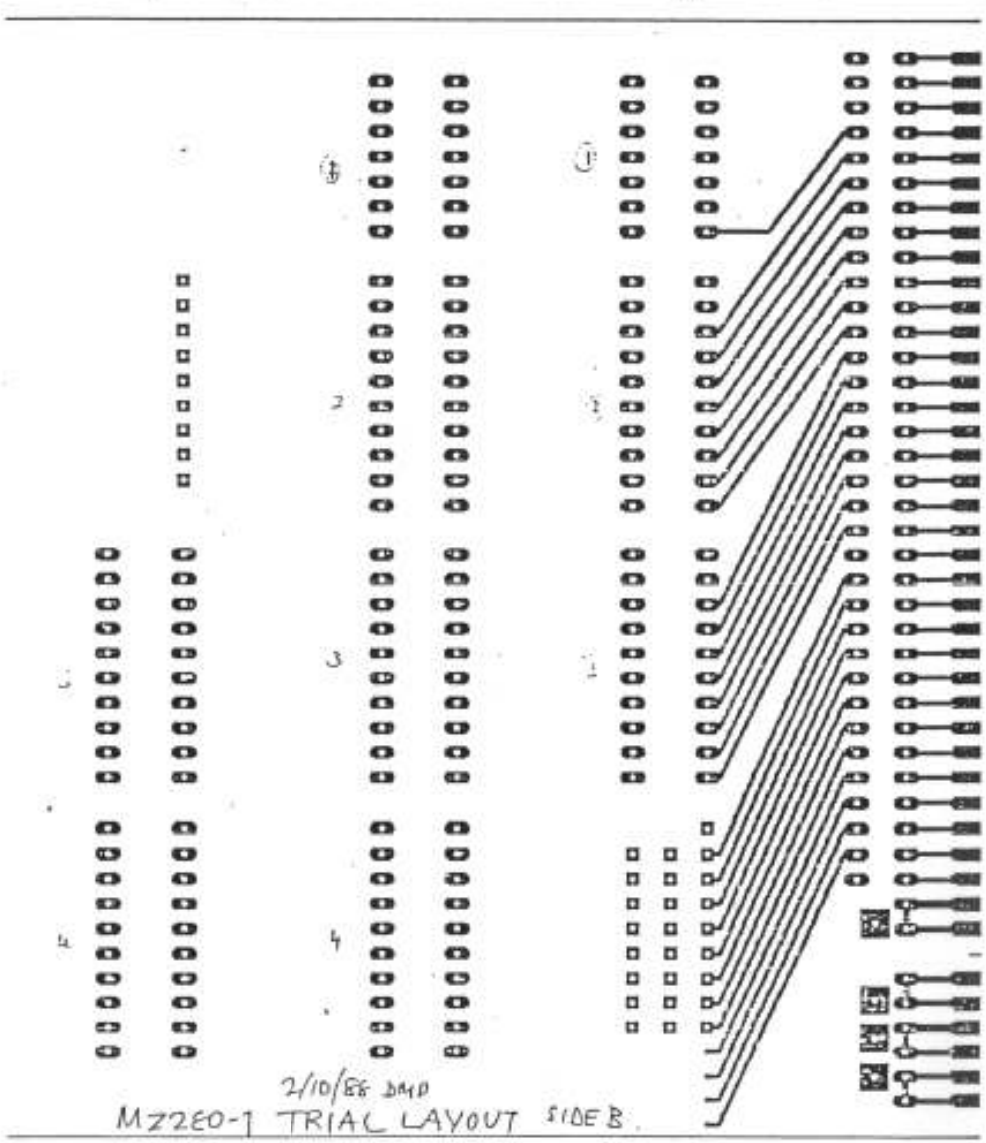
mods 30/11/88 DMP 29/11/88

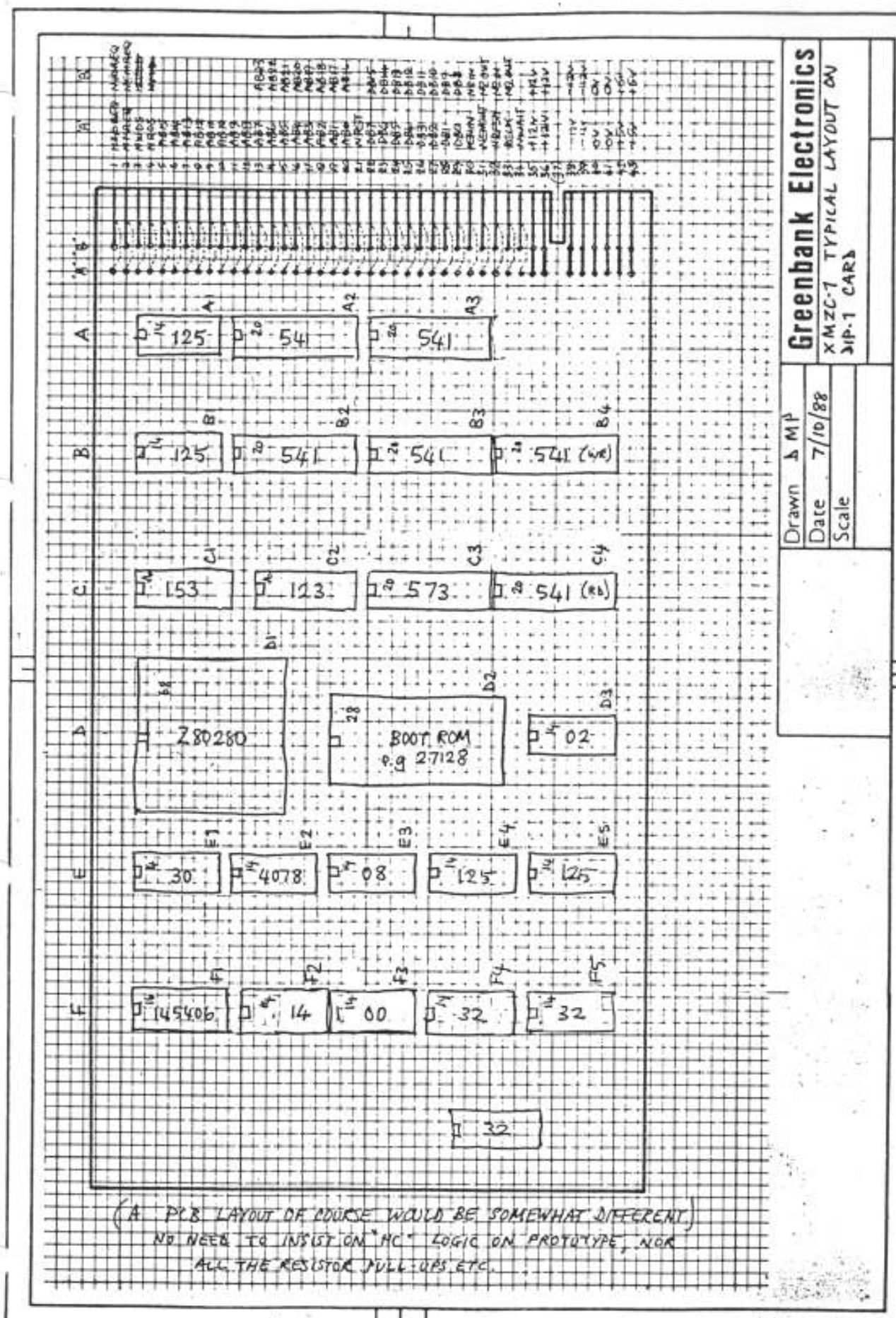


RESET CIRCUIT TIMING DIAGRAM

DMP 19/11/88


UB1
74HC125
UA1
74HC125
A
B
UB2
UA2
74HC541
74HC573
UC3
UB3
UA3
UC4
74HC541
UB4
74HC541
+12V
(37)
-12V
0V
+5V
2/10/88 DMP
MZ280-1 TRIAL LAYOUT SIDE A


C
B
A
2/10/88 DMP
MZ280-1 TRIAL LAYOUT SIDE B


Greenbank Electronics
XMZC-1 TYPICAL LAYOUT ON DIP-1 CARD
Drawn JMP
Date 7/10/88
Scale
A1
A2
A3
B1
B2
B3
B4
C1
C2
C3
C4
D1
D2
D3
E1
E2
E3
E4
E5
F1
F2
F3
F4
F5
125
54
541
125
541
541
541 (sk)
153
123
573
541 (sk)
Z80280
BOOT ROM e.g 27128
02
30
4078
08
125
125
145406
14
00
32
32
32
(A PCB LAYOUT OF COURSE WOULD BE SOMEWHAT DIFFERENT)
NO NEED TO INSIST ON "HC" LOGIC ON PROTOTYPE, NOR
ALL THE RESISTOR PULL-UPS ETC.

COMPONENT PARTS LIST FOR MZC-1 CARD

This part of the manual will have our normal parts list section, ie the dual listing which has proved so popular: listing by component identification, and also by component value.

Meanwhile, here is a provisional parts list:

Sheet 1

2 off 8 pin resistor pack 2k2 (4 separate resistors)
1 off 9 pin resistor pack 100k (8 resistors, 1 common)

1	Z80280-10VSC	
5	74HC541	Octal Buffer
1	74HC573	Octal Latch

Sheet 2

2 off 330R resistors (for LEDs)
3 off 8 pin resistor pack 2k2 (4 separate resistors)
1 off 9 pin resistor pack 1k (8 resistors, 1 common)
2 off 9 pin resistor pack 100k (8 resistors, 1 common)

5/6	74HC14	Hex Schmitt Inverter
1/4	74HC32	2 input OR
4	74HC125	Quad Buffer
1	74HC541	Octal Buffer
4	74HC573	Octal Latch
1	LED Red	
1	LED Green	

Sheet 3

1	27128	16K Boot EPROM
1/4	74HC00	2 input NAND
4/4	74HC02	2 input NOR
2/4	74HC08	2 input AND
1	74HC30	8 input NAND
8/4	74HC32	2 input OR
1	74HC153	Dual 4-1 multiplexer
1	74HC4078	8 input OR-NOR

Sheet 4

2 off	8 pin resistor pack 100k	(4 separate resistors)
1	330R for reset switch	
1	100k for reset	
3	Timing Resistors (3 values)	
1	Aluminium Elec Timing capacitor	
3	Timing Capacitors (3 values)	
1	1N4002	
2/4	74HC00	2 input NAND
2/4	74HC08	2 input AND
1/6	74HC14	Hex Schmitt Inverter
1	74HC123	Dual monostable
10	3-pin assemblies + links	

Sheet 5

2	47p capacitors	
1	16 MHz Crystal	
1	145406 (optional for RS-232)	
?	Various connectors, jumpers etc for RS-232	

Collated IC list:

1	145406	(optional for RS-232)
1	27128	16K Boot EPROM
1	74HC00	2 input NAND
1	74HC02	2 input NOR
1	74HC08	2 input AND
1	74HC14	Hex Schmitt Inverter
1	74HC30	8 input NAND
3	74HC32	2 input OR
1	74HC123	Dual monostable
4	74HC125	Quad Buffer
1	74HC153	Dual 4-1 multiplexer
6	74HC541	Octal Buffer
1	74HC573	Octal Latch
1	74HC4078	8 input OR-NOR
1	Z80280-10VSC	
25		