April-June 1987 Volume 1. Issue 2

Scorpio News

FOR OWNERS AND USERS OF:–

Gemini MultiBoard • Gemini M-F-B • Gemini Galaxy
80-BUS • Pluto • Quantum • 68K-BUS • Nascom
Gemini Challenger • Kenilworth • Gemini MultiNet

Scorpio Systems
P.O. Box 286 · Aylesbury · Bucks · HP22 6PU

ARCTIC COMPUTERS LTD.

6 Church Street
Wetherby
West Yorkshire
LS22 4LP

Tel: (0937) 61644

ARCTIC are specialists in Gemini 80-BUS products with over six years experience.

We are suppliers to industry and education.

Io Research PLUTO graphics specialists for the North of England.

A full range of equipment is on demonstration including Designer, Autocad and the Gemini CAD-8 PCB design system.

PCB Design and Plotting services available.

Suppliers of Graphtec plotters & Staedtler drafting pens.

Demonstrations and callers by appointment only please.
Mail order service available (carriage charged at cost).
ACCESS and VISA accepted.

SPECIAL OFFER TO SCORPIO SUBSCRIBERS

★ FREE SVC-Graphpac with every ★
GM832 SVC board purchased
(Offer ends June 30th 1987)

APRIL - JUNE 1987 SCORPIO NEWS VOLUME 1. ISSUE 2.

Contents

Editorial

First of all, thank you for the positive response to the first issue. You will find a representative sample of letters received later on in this issue (i.e. "representative" in that it is virtually all of them!).

The bad news, I'm afraid, is that we have "lost" a fair number of ex. 80-BUS News subscribers. This is probably for a number of reasons - Scorpio News may be considered inappropriate for those still using unexpanded Nascom 1s or 2s; apathy; cost of subscription; changes of computing interest. The good news is that we have "gained" quite a few new subscribers, including at least one NON 80-BUS/Nasbus user, and to all of them, welcome.

The direction that Scorpio News takes over the next issues will be guided by two main things. The first is what YOU the reader wants. Write a letter to us for publication expounding your views; write a brief (or long) article on your favourite computing topic. YOU can steer the direction probably more than you realise, IF you do something about it.

The second guiding force is consideration for expanding the number of subscribers. At the current level I'm sad to say that we are unlikely to continue into a second Volume, but if we can double or treble (or more!) the numbers then that's a different story. So there are at least two possibilities here. One is don't lend friends or colleagues your News, but each of you work on two or three of them to also subscribe. The other is to broaden the areas of coverage of Scorpio News. I'll say little more on this now. Read Dave Hunt's article and the letters pages carefully. A number of people are STILL using 80-BUS but now ALSO PCs/Clones. What do YOU conclude? What's YOUR own interest?

Finally, to any that fear that expanding the areas of coverage may dissipate the content too much, a few figures. Printing has high plate/setup/finishing charges, but lowish unit costs; I would estimate that with FOUR times the subscribers we could have TWICE as many pages for the SAME subscription. Hmmm!

The Dave Hunt Bits - Dateline March

Have new job - will travel.

I don't know why I bothered about a dateline, I'm in trouble with the editor for not having produced copy on time, rather than the old regime, where I would write something and find that it was printed a year later. I don't know if you'd noticed, but I usually used to contrive to include something in the text which would date a piece, something which it would be difficult to edit out without rewriting a paragraph or so. Now the tables are turned. Oh well ...

So lets have a round up of the DH news, people seem to be interested in what I get up to (or is it that I'm big headed enough to think that people OUGHT to be interested in what I get up to?). Anyway, as mentioned in the last issue, I've changed jobs since the demise of 80-BUS. Not an entirely voluntary change I must confess, but on the whole a change for the better. The new lot make me work harder (but I now enjoy what I do rather more), they send me gadding about the world (or at least Europe, and it's been hinted I ought to get a US visa in my passport).

Gadding about Europe is a pastime I normally rather enjoy, until, that is, you find yourself stuck in a taxi outside Frankfurt being driven by a manic octogenarian lady taxi driver who doesn't speak a word of English, and I'm late for my plane. My limited German led me to believe that she was either born in 1907 or that was the date she moved from the north to Altenstadt in the south. Either way, if Stirling Moss is as lively at that age, God help the average road user. Included in Europe, I suppose, should be a couple of visits to the submarine yards at Barrow in Furness, and several visits to the environs of Liverpool. The best my previous employer ever landed me with was a week or so in Baghdad, an episode best forgotten. (You can forget '1001 Nights..', etc.)

Micro-driven micro-film

So what is it I do? Well I now work for a microfilm company as 'The Guy Wot Knows about Computers'. Now microfilm is something I rather thought had died when large computer databases were invented. The first few weeks in the new job soon convinced me otherwise. Microfilm, I have discovered, is a far more viable archival storage medium for your average paper work than computer databases are, or are likely to be (optical disks, not withstanding). A rather sweeping statement, but one I now feel I can substantiate. There are some rather interesting statistics related to archival media. Did you know that only about 5% of the archived paper in this country is held on microfilm, whilst only 2% of archived paper is held in computer databases? I certainly didn't, and that's what my job is now about. Having stuffed archival paper work onto film, how the heck do you find it again?

Some time ago I was approached by these people to look into what has become termed by the awful jargon acronym, CAR (Computer Aided Retrieval). This means building a database for use on a micro, which contains references to the film. It works by whittling away at the given search criteria until you end up with a discrete frame. The frame information is then sent to an automated microfilm reader which goes away and finds the film you want, and ultimately displays the frame you're after.

The whole process takes about six seconds, about 5 and a half of which are whirring and clicking from the film reader. So having written a pilot program on the Gemini, I was commissioned to write it properly, but this time on an IBM as the potential customers had been identified as already having IBM PC's of various ilks coming out of their ears. I finished the job early last year. Since then, customers have come up with lots of bright ideas and what I thought was a one off has turned in to a never ending process. The current program being sold with all the kit is about six months old, but two newer and better and cleverer versions exist, and will receive their public launch at the INFO 87 exhibition at Olympia at the end of March.

Language problems

Thinking of public launches, the retrieval software has to be multilingual, and it's all done in a fairly straight forward fashion so the software speaks English, German, French and Dutch by answering the right questions when the program is initialized. This has caused me a lot of bother, because I don't speak French, German or Dutch. [Ed. - what about English?] The worst instance took place at the massive OrgaTechnik exhibition in Cologne at the beginning of November. There's me sitting in front of my machine on a stand at the exhibition, doing demos for all it's worth (me speaking English, whilst the program is speaking German, of course), when I typed something stupid and the program threw a wobbly and came up with an error message (in German). There's nothing quite so embarrassing as having to turn to your audience to ask them to translate what has happened, because you don't understand what your own software is telling you!

Unfortunately, the new job doesn't stop there. Writing this sort of stuff is a good challenge and great fun, but can get boring after a while. Oh no, customers are an original lot and come up with lots of bright ideas some of which take me back to the hardware days. Now the retrieval software is theoretically capable of handling in excess of 1,000,000,000 records, yes, that's 1 billion in English or 10 billion in American, yet the capacity of your biggest stand alone IBM micro is 32M bytes. One customer has 10.5 million bits of paper which would require 10.5 million records, which with indices would require some 170M byte of disk storage (bigger than the on-line storage of your average mini). To cut a long story short, I've managed to jack up the storage on this customer's IBM to 320M bytes and I now have potential storage of 750M bytes if someone is daft enough to want it and can pay the price. Other bright ideas include porting the stuff to a VAX and running it under emulation (a not very satisfactory exercise), and someone has even wanted it on a main frame (I told them not to be silly and run it under their network instead).

Life's a challenge

Life doesn't seem the same old drag anymore, problems are there as challenges; some you win, some you lose, either way, you learn something. The only sad thing is that over the last 8 months I seem to have had little time to do what I want. I haven't tried any new hardware (except my new toy, more later), I haven't written anything new or original, at least not Nascom/Gemini related, and apart from mucking about on bulletin boards on occasion, the machine at home has been collecting dust.

Well I suppose that's not true, I've been carting an IBM AT home at weekends and busily converting various bits of software from CP/M to MS-DOS. Nice bit of software the Gemini IBMCOPY. I've also been finding all the useful little tools that I have been using on the old machine, for use on the new. This has been leading up to something, and that is the acquisition of a new toy, a rather smart Japanese IBM AT clone, which arrived a couple of weeks ago.

I don't think it would be fair to say that I've outgrown the Gemini/DH machine, just that I need something to allow me to work at home without being incompatible with the machines at work. Ok, so I could continue to drag one of the works machines around after me, but that is dangerous. A real IBM AT with frills still costs about four and a half grand, it also weighs about 50 pounds (I think they include lead weights to help justify the cost). During that snowy bit a few weeks back, I slipped whilst carrying one these machines from the car, and although no harm was done, I had visions of the damage I might have done to the machine, and even worse, the damage I could have done myself in trying to catch it! No, the only answer was one of my own.

Hunt hunts the clone

I went and shook the piggy bank, and found just enough in it (provided the missus doesn't have funny ideas of trolling off to sunnier climes this summer, and provided the guttering can be persuaded to stay in place one more year). So off I went armed with the loot in search of my ideal machine. Nothing's easy is it? The magazine specs of machines are confusing, the prices seem to vary by about 25% for essentially the same things, and if you want to actually see one and try it out for an hour or two, you try and find someone who will actually show you one **BEFORE** you hand over the money.

I even tried a couple of the proprietary computer shops, who did know something about business software, but were amazed that I actually wanted a machine of this sort to actually USE. They knew naff all about what actually went on inside the machine; one lot were adamant that the processor inside an AT was actually an 8088, otherwise how could it be IBM compatible?

Just after Christmas there was a very twee but informative Channel 4 programme called 'What They Don't Tell You When You Buy a Computer'. I found the program amusing, but heaven help the ordinary punter. I subsequently found most of the programme true, and I'm supposed to know what I'm taking about! In the end the new machine came from one of my old employers' suppliers. At least I knew who to throw it at if it didn't work. In the event, it's all of three weeks old and hasn't hiccupped once (yet).

Despite the high speed of this machine, 10MHz clock speed with zero wait states (compared with the 'cooking' IBM AT's speed of 6MHz with one wait state, the new machine is actually just over twice as fast as an ordinary AT), and despite the 35mS latency on the 20M byte disk drive, I've had only one problem with software. The mags are all muttering about these high speed, so called 'Turbo' machines having problems with certain software. I've only found one and that was loading my Clipper dBASE compiler. I had to fit a 'Go Slower' switch to bring the thing down to normal speed because there seemed to be some hardware timing in the software protection. A software protection, incidentally, I have found exceedingly difficult to crack, I've no idea what they do to the disk, but I can't copy it!

So what have I done with it, well, I've written a cataloguing program (I couldn't find a half decent one anywhere), you'll find CAT in the MS-DOS download area of the CBBSLW bulletin board. I have revamped my address finder (I'll be up-loading that to CBBSLW shortly), my radio log book has been rewritten from end to end, and for fun and the practice, I've tried to re-assemble CRCK5 into 8086 code without a lot of success, using the 80TO86 program and then hacking from there. Funny, I can only find an MS-DOS version of CRCK4 on the public domain bulletin boards.

Include Clone info?

Somewhere else in this mag, you'll find a letter from me suggesting an IBM spot in future issues. I've no idea how that will go down, but if it's well received, it'll certainly give me lots of scope for writing about the intricacies of dBASE III Plus (twice as complicated and therefore twice as messy as dBASE II), what to look for in an IBM clone and what to avoid, driving the more doubtful of the MS-DOS utilities, what I've discovered inside MS-DOS and lots of other nutty goodies. All things I've learned the hard way over the last few months. Or perhaps a series on learning Modula-2, a language I've got to learn for the next job coming up at work.

Lastly, on the score of changing allegiance, I note that NASTUG, that ex-Windsor based Nascom Users' Club and bastion of the Nascom/Gemini cause, has recently changed emphasis, and whilst still supporting Nascom/Gemini/Map, recently had a meeting to decide the future of the club. They recognize that a majority of the membership now use different machines although they also elected to retain the name NASTUG. The venue has also changed and the club has been put on to a more substantial footing with an actual published programme! They now meet on alternate Thursdays at the Crown & Treaty, Oxford Road, Uxbridge. For further details see the article on NASTUG on page 15.

Dealer Profile - Arctic Computers

[Ed. - This issue Brian Wingfield, M.D. of Arctic Computers in Wetherby, writes about when, where, why and how he got involved with 80-BUS equipment.]

Arctic Computers. Who?

Yes, who are these people Arctic Computers who claim over 6 years of 80-BUS experience?

Well it all started back in late 1979 when I was looking for a way of sequencing an analogue music synthesizer and I happened across an advert for a Nascom 1. The beast was ordered and after the customary long wait (while it was designed I think), a box of bits arrived with a promise that the power supply would be along soon (still to be designed?). Well the kit was built and a home brew PSU was fabricated. Guess what? It didn't work. After many hours of fiddling with this, my first computer, I decided that the problem must be in the EPROM (NASBUG T1 in a 2708). So a test rig was made to single step the EPROM and monitor the data lines with LEDS. After the first three bytes, which were FF FF FF, I twigged that the EPROM was empty. It was promptly sent back to Nascom, where some weeks later during one of my begging phone calls I first spoke to the Editor of this very publication, the first sign of intelligence at Nascom. Within a couple of days my Nascom was alive. By this time I had forgotten the real purpose for buying the beast, and who cares I was hooked anyway.

Having realised that the Nascom 1 could be improved a friend and I set out to design add-ons. Amateurish they were but members of a local computer club, of which I was a founder, wanted to buy them. So from these small beginnings a decision was made to go live, and the name BITS & P.C.s was born. Adverts were placed in various publications and we were overwhelmed with demand. At this stage it seemed like a good idea to become a Nascom dealer and I was very flattered when I was accepted as such, when working from my lounge at home. In reality anyone buying product in quantity with cash up front would have been accepted.

During this period I was working full time for the P.O. and my wife was running the operation from home. Nascom's big boss John Marshall had been given the Special faults number at the telephone exchange where I worked in case of needing to contact me urgently. Either by design or by accident this number was published in the Nascom national advertising which led to rather a lot of embarrassment.

In March 1980 the company became BITS & P.C.s Computer Products Ltd. and I resigned from the P.O. We also moved into a quaint little shop in Wetherby, which meant that we were able to have most of the house back as living accommodation.

Our next few years saw us playing computer retailers snakes and ladders along with all the others. We fared better than most due to our policy of designing our own add-ons for various types of computers including Nascom, Sharp, Epson, Superbrain, and Gemini. This gave us a strong reputation among both customers and suppliers.

By 1983 we had moved into our second set of premises in Leeds City centre and the computer retail side was getting very tough. One thing we were not willing to do was sell at silly discounted prices because we wanted to give good customer support and a level of profit was required to maintain this. The company was at this stage consolidated in Leeds and was known as the LEEDS COMPUTER CENTRE. One of the biggest annoyances at this time was that Joe Public was buying from Bloggs Discount and was then pestering my staff for their expertise in getting his system running.

In January 1984 at a directors meeting the current decline in profitable business in Leeds city was discussed. The company was keeping alive thanks to a hard core of Gemini enthusiasts and thanks to a number of major OEM customers who were using us as suppliers and technical support. The decision was therefore made to go back to our roots and concentrate only on this part of the computer market.

New premises were found in Wetherby and, as the name Leeds Computer Centre would then have been geographically inaccurate, a new one was needed. I have constantly suffered by living in the North and having a vast number of business contacts in the South, the MD of Gemini being one of the worst offenders. You know the sort of thing where anything north of Watford is in "the frozen North", so what better name than ARCTIC COMPUTERS?

Things have gone very well since the move over two years ago. We are just abrut to branch out and expand the computer graphics side of the company in order to promote PLUTO graphic products on a much larger scale. One of the main areas will be a self drive bureau where time can be bought on our systems in order for potential purchasers to evaluate the equipment on a real project. We will be equipped with facilities for full colour frame grabbing from any source, colour printing, 35mm slide production, A0 pen plotting, CAD, PCB-CAD, Graphics design 2D & 3D, video productions to high band Umatic etc. Purchasers of systems will have access to our more expensive peripheral equipment at very reasonable prices.

I hope these ramblings serve to explain the company roots and all that remains is to thank all those who have supported us over the years.

B. S. Wingfield, MD, Arctic Computers Ltd.

Technical Query

SYS Questions

Could somebody please answer a couple of questions involving the disk routines in SYS18?

1) Why are Shugart SA200 drives only allowed 64 directory entries?
2) Why is "side nibbling" allowed only under Gemini DD format with PERTEC drives? I'm using a Mitsubishi drive and a Pertec drive and it works just as well on the Mitsubishi as the Pertec. Also why didn't Gemini keep the above format for 96tpi drives?

Yours, P. Smeesters, Forest Hill, London.

Richard Beal, Author of SYS replies:

1) I seem to remember that the Shugart SA200 was an 8" drive. In that case the reason for the 64 directory entries is because that was the original 8" CP/M disk format standard. In fact 64 is quite a sensible number of entries for a disk capacity of about 250K.
2) I assume "side nibbling" means putting data on both sides before moving on to the next track. This improves performance. The reason it was used for the Gemini DD/DS 48tpi format and not for the later 96tpi (80 track) formats is that the first Gemini double density 48tpi format was double sided, whereas the first Gemini 96tpi format used single sided drives. Naturally Gemini decided to use a format which would allow both single and double sided drives to be used, with single sided disks being immediately readable on the later double sided drives. In fact this required the additional forward thinking precaution of using 4K allocation blocks.

   The question of using Mitsubishi drives instead of Pertec appears to relate to the use of "Pertec" in early documentation - probably before alternate suppliers existed. It will work just as well on compatible units.

The Z80 CTC by Rory O'Farrell

Some time ago [1] I wrote on aspects of the Z80 microcomputer's interrupt structure. In that article, I dealt mainly with the Z80 PIO. Since then there have been many requests for another article to deal with the other Z80 family chips. In this and a subsequent article it is my intention to cover the intricacies of programming the following chips:

The CTC	(Counter Timer Circuit)
The SIO	(Serial Input/Output Controller)
The DART	(Dual Asynchronous Receiver/Transmitter)

Fortunately, the DART is a reduced specification SIO, providing only the asynchronous facilities, so for brevity I will simply refer to the SIO, and in general the DART will also be implied.

For ease of understanding, I propose to discuss the CTC/SIO pair as used on the MAP CPU and MAP MPI boards. On these boards the CTC/SIO group provide a most powerful, fully programmable, serial input/output unit. This is not the only possible use for these chips, but it will serve well as an introduction, besides being the most common configuration for them.

Serial I/O interconnections have been very adequately dealt with in [2] and [3], and interested readers are referred to those articles for detail coverage of the subject. For the newcomer, it is appropriate to give a minimum explanation to set the scene. A Serial I/O chip is one which is given a byte of data, and which squirts it out bit by bit onto a pair of wires - rather like a lump of dough being fed through a spaghetti maker - our new improved modified spaghetti maker which makes only one strand at a time! That is the act of transmission. Such a chip is also capable of receiving the individual bits of data and reassembling them into a byte - scrunching up all the fresh spaghetti to end up with a lump of dough again. There are other concerns, which need not affect us here, such as level shifting from TTL levels (0 - 5 volt) to RS232 levels (+/- 12 volt), as these problems are dealt with in the articles cited.

If we imagine that we have a spaghetti maker at one end of a table manufacturing a single strand of spaghetti, and a reverse spaghetti maker at the other, we can move pieces of dough from one end of the table to the other by feeding the output of one machine into the input of the other. It is easy to see that if the generator makes the spaghetti faster than the other machine consumes it, we will soon have a big pile of spaghetti on the table. If the consuming machine consumes faster than the maker, then the strand of spaghetti keeps getting pulled tight and breaking. So it becomes obvious that both machines must work at the same pace. Similarly when transmitting data from one serial device to another, they must both work at the same rate, or in jargon, be 'clocked at the same rate'. One of the major problems with serial communications is finding the *rate of transmission or reception, loosely called the 'baudrate'. We will* consider the use of a CTC to provide differing baudrates or 'clocks' for an SIO.

The CTC chip is a 28 pin member of the Z80 family. It comprises four channels, which are substantially the same. Addressed as four I/O ports, one for each channel, it can return an interrupt vector from each channel on the occurrence of certain conditions, which criteria can be programmed into that channel. For ease of reference, we number the channels from 0 to 3. Channel 0 differs from the others in that it remembers the interrupt vector for channel 0, and the onboard circuitry of the CTC is able to modify this interrupt vector to indicate the interrupting channel. This we will deal with shortly. Channel 3 differs from the other channels in that it (due to limitations of pin numbers) cannot output pulses, it can only count them.

The manual for the CTC [4] says that the CTC can be configured as a counter or as a timer. This is true, but confusing. I find the following picture easier to grasp: each channel of the CTC is a counter. The channel can be set up to count pulses arriving on the input pin for that channel (CLocK/TRiGger input, CLK/TRGn, where n refers to the channel number), or to count pulses arriving from the System Clock. If the System Clock is being used, it is routed through a prescaler, where it is divided by either 16 or 256. The counter can be set to count down from any number between 256 and 1. On reaching zero, the channel sends out a pulse on its Zero Count/Time Out (ZC/TOn) pin, except for channel 3 which lacks that pin. The counter can be caused to automatically reload the initial count everytime it reaches zero. The important event is the counter reaching zero, which can cause an output pulse on the ZC/TOn pin, an automatic reload of the counter ready to go again, and an interrupt if required.

The CTC, being capable of vectored interrupts, is best used in that mode. A typical example is the MAP CPU board, where channel 3 of the on-board CTC is configured to interrupt when the keyboard has a character ready. This is done by telling the CTC channel 3 that it is to count 1 transition on the CLK/TRG3 pin (the keyboard strobe), cause an interrupt, and automatically reload the count ready to go again.

In programming the CTC for a system using vectored interrupts, we must therefore supply a valid interrupt vector. This is the lowest of four consecutive slots in the interrupt service table, and is passed to channel 0. Channel 0 recognises it as an interrupt vector because the least three bits are 0 - typically the form of this vector is:

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	0	0	0

where the least byte of the address of the four word section of the interrupt table is V7V6V5V4V3000. Bits 2-0 are modified by the CTC adding in 000 for Channel 0, 010 for channel 1, 100 for channel 2 and 110 for channel 3, depending which originates the interrupt.

Then to program a channel of the CTC, a control byte and associated but optional parameters are written to the Channel Control Register. Each channel of the CTC can be programmed independently of the others. The channel recognises the control byte because bit 0 is always 1. The other seven bits of the control byte define differing alternatives from the channel's operating modes. These bits have the following meanings:

D7	D6	D5	D4	D3	D2	D1	D0
Int En	Mode	Range	Slope	Trig	load time const	reset	1

These bits have the following meanings:

D7 = 1	- Generate interrupt every time downcounter becomes zero. Before doing this, an interrupt vector must have been written to the CTC.
D7 = 0	- No interrupt from this channel
D6 = 1	- Select counter mode. Down counter is decremented by each triggering edge of the external clock from CLK/TRG pin. No prescaler used.
D6 = 0	- Select timer mode. System clock feeds through the Prescaler, thence to the Down Counter. The ZC/TOn pin is pulsed at a frequency of tc*P*TC, where tc is system clock period, P is prescaler factor of 16 or 256, as set by bit 5, and TC is the time constant data word.
D5 = 1	- is defined only in timer mode. Prescaler = 256
D5 = 0	- is defined only in timer mode. Prescaler = 16

D4 = 1	- In timer mode, positive edge starts timer. In counter mode, positive edge decrements counter.
D4 = 0	- In timer mode, negative edge starts timer In counter mode, negative edge decrements counter.
D3 = 1	- is defined in timer mode only. Timer starts immediately (well, almost)
D3 = 0	- defined in timer mode only. Timer starts after time constant has been loaded.
D2 = 1	- Time constant word is next word to be written to this channel.
D2 = 0	- No time constant will follow (use the existing one)
D1 = 1	- Reset channel. Channel stops counting or timing.
D1 = 0	- Continue current operation.
D0	- Always 1 for control byte.

To disable a CTC channel from causing an interrupt, the code is simple. Just write a 01H to the appropriate channel. Like the other Z80 chips, it is necessary to disable the system interrupts in case an interrupt occurs while the disable instruction is being assimilated by the I/O chip. The code is only slightly more complex than above:

```
LD   A, 01h
DI               ; Disable CPU interrupts
OUT (CTCn),A     ; Disable CTC channel
EI               ; re-enable CPU interrupts
```

A channel may not begin operations in any mode unless a valid time constant byte has been written into the Time Constant Register. This byte is expected on the next write to the channel in question following the write of the channel control byte, provided the channel control byte had bit 2 set (= 1). If this time constant byte is 0 it is interpreted as 256. If a new time constant byte is written to a channel already in operation, the new time constant byte is only reloaded at the next count down to zero.

The major difference between Counter mode and Timer mode is that Counter mode uses the external clock or pulse provided on the CLK/TRGn pin, whereas the Timer mode uses the system clock, divided by the prescaler (either 16 or 256 according to the setting of bit 5 of the channel control byte). In timer mode, the timing operation can be programmed to start from an external trigger to the CLK/TRGn pulse, or almost immediately from the system signals. The external starting trigger is usually not required, and the reader is referred to the manufacturer's data sheets for more precise details. It is also possible to define whether the trigger or count decrement takes place on the rising or falling edge of the timing clock (Bit 4). One constraint that is not very openly defined is to be found in the technical details of the data sheet; this states that the maximum frequency to be applied to the CLK/TRGn pin is to be half of the system clock. This means that on a 4MHz system, the maximum external clock to the CTC is to be 2 MHz.

Now for a worked example. In the case of a standard MAP CPU, we wish to set up the CTC as follows:

CTC0 is to drive SIO channel A to receive/transmit at 1200 baud
CTC1 is to drive SIO channel B to receive/transmit at 300 baud
CTC2 is not used
CTC3 is to receive the strobe from the keyboard, and to interrupt on each key.

Rather than give a full listing of code, I will use the method set out in [5] for I/O port initialization, with comments beside the relevant bytes to explain how they are arrived at. These initialization tables are set out like this:

```
     DEFB Number of bytes to be sent
     DEFB Port to address
     DEFB data1
     DEFB data2
     .......
     DEFB dataN
```

This differs slightly from the method used in the MAP80 BIOS in that they reverse the port address and byte count (you have been warned!)

```
     CPCTC      EQU  0B8H       ; define base address of CTC on MAPCPU

     ; channel 0 is clocked at 2MHz externally, and we must
     ; include the interrupt vector in channel 0's data
     CPCTC0:    DEFB C2 - C1    ; byte count
                DEFB CPCTC      ; define CTC port for channel 0
     C1:        DEFB LOW CPCTCV; low byte of interrupt vector
                DEFB 01000111B ; Counter mode           01000000B
                               ; Count follows          00000100B
                               ; Reset channel          00000010B
                               ; channel control byte 00000001B
                DEFB  104      ; divisor for 1200 baud (*16 mode)
     C2:

     ; channel 1 is clocked at 1 MHz externally
     CPCTC1:    DEFB C4 - C3    ; byte count
                DEFB CPCTC+1    ; define CTC port for channel 1
     C3:        DEFB 01000111B ; comment as above
                DEFB 208        ; divisor 300 baud (*16 mode)
     C4:

     ; no entry for channel 2
     ; channel 3 is to interrupt everytime the keyboard strobe
     ; is active
     CPCTC3:    DEFB C8 - C7    ; byte count
                DEFB CPCTC+3    ; define CTC port for channel 3
     C7:        DEFB  11010111B; Enable channel interrupt 10000000B
                               ; Counter mode               01000000B
                               ; Count follows              00000100B
                               ; Reset channel              00000010B
                               ; channel control byte       00000001B
     C8:
```

Of this, what needs explanation? Channels 0 and 1 are clocked to the CLK/TRGn pins with 2 MHz and 1 MHz clocks. Having defined the required baudrates (1200 and 300), we now need to calculate the divisors. Serial I/O chips usually require their clocks at 16* the required baudrate. in the case of channel 0, 1200 baud demands a frequency of 19.2 KHz. So the required divisor is:

```
                                    2000000
     Channel 0 divisor =            -------       = 104.167
                                    1200*16
```

for the purposes of the CTC, a divisor of 104 will be quite adequate.

Channel 1 presents a similar calculation:

```
                                    1000000
     Channel 1 divisor =            -------        = 208.333
                                    300 * 16
```

so we are all right with 208 as the divisor.

Channel 3 is explained by the comments. But, I hear someone say, just what is happening? The external clock is being divided down by the CTC channel using the divisors set out above. Everytime the counter hits zero, it outputs a pulse on the ZC/TOn pin. This stream of pulses is then fed as a clock to the appropriate channel of the SIO. This situation covers the MAP CPU with the default links as provided:

LB2/2 - LB2/10	LB2/3 - LB2/9	LB2/13 - LB2/12	LB2/1 - LB2/14
LB1/18 - LB1/4	LB1/19 - LB1/3	LB1/17 - LB1/5+6	LB1/16 - LB1/7
LB1/20 - LB1/1			

One other point which should be mentioned. Always use the assembler to calculate byte counts and ratios as far as possible. Remember that it does only 16 bit arithmetic, so we can't use it to compute the division ratios, but we certainly can and should use it to work out the byte counts.

Suppose we require to use a modem? The CCITT V23 standard will require that we can receive and transmit at different baud rates (1200/75). Under the set up above, we have the receive and transmit clocks for SIO channel A tied to the same channel of the CTC (LB1/17 - LB1/5+6), so we can't manage just as we are presently linked. We have spare CTC channel 2, and we could use channels 1 and 2 to provide 1200 baud and 75 baud clocks, and channel 2 to feed the second SIO channel. First of all, we must remake the links as follows:

LB2/1 - LB2/14	LB2/12 - LB2/13	LB2/1 - LB2/11	LB2/2 - LB2/9+10
LB1/1 - LB1/20	LB1/2 - LB1/19	LB1/3+4 - LB1/18	LB1/5 - LB1/17
LB1/6 - LB1/16	LB1/7 - LB1/15		

(These are as taken from my system. Work them out carefully - you may not need them all!). If we feed CTC channels 0 and 1 from the same clock (2MHz external) can we get the baud rates required? We have already done the calculation for the 1200 baud and know that 104 is the divisor. How about 75 baud?

$$\text{divisor} = \frac{2000000}{75*16} = 1666.7$$

We can't fit that into one byte. Even using 64* mode on our SIO, we can't do it. If we went to a 1 MHz clock, we can just manage. Unfortunately, I like to drive printers very fast, and for computer/computer links, with no modems, I like 9600 baud. If we use 1 MHz clock, I can't get 9600 baud. Is there another way?

Suppose we consider using the timer mode. and consider first the 1200/75 baud problem. If we divide the system clock (4Mhz) by 256 in the prescaler, could we get the 75 baud we need? Let's try:

$$\text{divisor} = \frac{4000000}{75*16*256} = 13$$

Lovely! Now we try for the 1200 baud. Immediately we realise that we'll have to use the /16 prescaler rather than the /256.

$$\text{divisor} = \frac{4000000}{1200*16*16} = 13$$

No problems here! How about 9600 baud?

$$\text{divisor} = \frac{4000000}{9600*16*16} = 1.6$$

So that won't work for 9600 baud. We are in the position that we can get 1200/75 baud clocks by using the divider mode of the CTC, but can't get 9600 baud. If we apply a 2 MHz external clock, we can get 9600 baud clock, and slower, but not 75 baud. What do we do? We set up the links as above, supplying a 2 MHz clock to the CLK/TRG pins of the appropriate channels, and using software, we reconfigure the CTC to use the divider mode and system clock for 1200/75 baud, and the counter mode and external clock for the faster baudrate of 9600.

The configuration tables for the CTC are:

```
; Port initialization table
; First byte indicates how many bytes to send
; Second byte defines port location of device
; On board CTC channel 0, include interrupt vector here
; This divides input clock of 2Mhz to provide clock for
; SIO channel A receive at 1200 baud
CPCTC0: DEFB C2-C1                ; byte count
        DEFB CPCTC                ; CTC port
C1:     DEFB LOW CPCTCV           ; Interrupt vector
        DEFB 00100111B            ; Counter mode, neg. edge triggers
                                  ; start on T2 cycle, Count follows,
                                  ; Reset channel
        DEFB 104                  ;  for 1200
C2:
; This divides system clock of 4Mhz to provide clock for
; SIO channel A transmit
CPCTC1: DEFB C4-C3                ; byte count
        DEFB CPCTC+1              ; CTC port
C3:     DEFB 00100111B            ; timer mode, prescale 256,
                                  ; Count follows, Reset channel
        DEFB 13                   ;  for 75
C4:
; This divides input clock of 2Mhz, to provide clock for SIO channel B
CPCTC2: DEFB C6-C5                ; byte count
        DEFB CPCTC+2              ; CTC port
C5:     DEFB 01000111B            ; Counter mode, neg. edge triggers
                                  ; start on T2 cycle, Count follows,
                                  ; Reset channel
        DEFB 104                  ;  for 1200 baud
C6:
; On board CTC channel 3
; Provides interrupt control for keyboard
CPCTC3: DEFB C8-C7                ; byte count
        DEFB CPCTC+3              ; CTC port
C7:     DEFB 11010111B            ; Interrupts, Counter,
                                  ; Count follows, Reset
        DEFB 1                    ; Divisor
C8:
```

To switch between 1200/75 and 75/1200 takes a little more work - I had to design a patch for my MODEM7, and for CP/M Plus, but it is possible. I hope that this long rambling article will persuade others to try using an unused channel of their CTC - it is a most useful chip. I know of one system where an unused CTC was employed to provide dooropen interrupts for three drives, and the fourth channel was used to provide a real time clock interrupt to update the on screen date and time every five seconds.

In the next article, I'll write about the SIO and its programming.

References

1. 80-BUS News Vol 2, No 1, The Interrupt Structure ofthe Z80.
2. 80-BUS News Vol 1, No 3, Teach Yourself Z80 - Part 7
3. 80-BUS News Vol 2, No 3, Serial Interface Problems made Easy
4. ZILOG Counter Timer Circuit Technical Manual
5. Z80 Assembly Language Subroutines by Leventhal and Savile, Osborne/McGraw-Hill, p385

NASTUG Expands Beyond Nascom by P.A. Greenhalgh

There have been, and may well still be, a number of Micro User Groups around the UK with Special Interest Groups dedicated to the Nasbus/80-BUS cause, and I am sure that the readers of Scorpio News would like to know of them. Therefore please send in a brief note if you are aware of any such Groups.

One Group that is still very much alive (or perhaps re-born is more appropriate) is NasTug. NasTug is an acronym for Nascom Thames Valley User Group and it was started several years ago. Originally purely Nascom, the type of equipment covered by the group has been expanding over the years, although the original name has remained. At a recent meeting it was decided that the activities of the club should now "concentrate on operating systems such as CP/M-80, CP/M-86, MS-DOS, etc rather than a particular hardware system......broadening the club membership whilst retaining the original enthusiastic spirit of the NasTug concept."

NasTug used to have frequent "formal" nights, when a guest speaker would talk on a specific topic for a short time, prior to the evening taking on its normal form of chatting, drinking, exchanging news, views, tips and hints, drinking, chatting some more, and drinking. Speakers included at various times were: Mike Rothery from MAP, myself on a couple of occasions, Dave Hunt, and Richard Beal.

The Group then went into a sort of limbo. It still met regularly, but it was just for the chatting, drinking, etc. Recently renewed enthusiasm has resulted in election of a new committee and re-formalisation. So, a few details are:

Venue:	The Crown & Treaty (Pub), Oxford Road, Uxbridge.
Dates:	2nd and 4th Thursdays of each calendar month. Normally 2nd Thursday will be a "formal" event.
Times:	8 p.m.-ish to 11 p.m.-ish
Costs:	£10 per annum including User Group letters. Visitors/guests £1 per visit.

By the time you are reading this recent speakers will have been Gemini on the Challenger and Chris Southern on the Atari 1040ST. Coming attractions are: Chris Mower demonstrating the Xerox Ventura Desk-Top Publishing Software on April 9th. Also soon: Andy Hay talking about the Pick operating system; Demonstration and a talk by PD-SIG(UK) (Public Domain Software Interest Group).

I went along to the Gemini talk on the 68000 based Challenger system. The presentation was given by Barrie Oliver, from the Sales Department, and Travis Benton, from the Technical Department. Barrie did most of the presentation, but unfortunately he did not seem to be aware of the technical bias and interest of his audience, and not even the most basic technical specification of the Challenger was outlined at any stage. The talk seemed to concentrate on the difficulties of selling a product such as the Challenger into a market dominated by IBM & its clones, and yet did not explain where people may benefit from having a Challenger instead of an IBM or Clone. Barrie went on to say that he was optimistic that over the next few months sales could climb to 10 per month.

Throughout the talk an impressive graphics demonstration program was left running, implying that the machine has a lot of potential in graphics areas, but no application packages were shown, either graphics or otherwise. The meeting was told that both 2D and 3D graphics packages were being ported onto the machine, and the audience's appetite was sufficiently whetted for a number of them to ask for an opportunity to see the Challenger with graphics again at a future date, once the graphics applications software can be demonstrated.

All interesting stuff. So, if you live within a reasonable distance of Uxbridge why not give NasTug a go? One guy likes it so much he travels from Chelmsford!

[Note to NasTug members. OK, you've had a good plug. At the moment only 2 of you subscribe to (as opposed to read) Scorpio News - why not now make it 22?]

Dr Dark's Diary - Episode 25 and still hacking away...

Own Up, and Make Money!

When I treated my system to its Gemini GM832 SVC (video) board, I found that is was possible for it to operate a keyboard, taking some of the load off the main processor. This would let me make things go faster, which is becoming an obsessive pursuit. But the SVC wanted to speak to either a serial or parallel keyboard from the Gemini range, and I happen to have a Nascom II keyboard.

It would have cost quite a lot to buy either of the Gemini keyboards, I think they are around the £140+VAT mark, so that idea was out. But the dealer who sold me the SVC (no names as I feel they should pay for their advertising) happened to mention that a member of a Thames Valley Nascom user group had made up some boards that would fit in between the Nascom II keyboard and the SVC. He said that these boards had a cpu of their own, which just scanned the keys and passed on what it found.

As well as the advantage of things running faster, this would also let me take advantage of the keyboard type-ahead that the SVC allows. So if you happen to be in possession of a spare one of the converter boards, I would be very pleased to hear from you. I might even buy one if the price is sensible. Incidentally, I have heard that perhaps this user group has been photocopying magazines. This practice is of course illegal, but copying my stuff also carries a curse the like of which has not been around since the Pharaohs...

There is always a catch!

The catch in this case is that if I do take the keyboard scanning load away from the main processor, it will mean that my CP/M will need to be sorted out to recognise the new keyboard, instead of carrying out the original scanning routine. But I have been meaning to do that for ages, in order to use the new, improved BIOS software that is now available (I hear it goes faster) and was described in the last Scorpio News.

No great problem, apart from finding the time, you might think. However, my CP/M has already been modified to provide a RAM disk on two MAP-80 RAM boards, and I have no idea what they have done, or where they did it. I suppose I will have to find where the old CP/M and the present one are different, and then disassemble the new bits. The result will then need to be incorporated in the improved system. Maybe MAP-80 will write an article about how their modification works. I think that would be very interesting to read, and it could also give some clues about another thing I would do, if I only knew how. It would be very handy in certain circumstances to be able to get CP/M to think the RAM disk is drive A, and vice versa. There is still some software around that insists on being on drive A, which I would like better if it went at drive P speeds! Has anyone managed to achieve this? Then tell us all how, don't keep it to yourself!

That Pascal Benchmark Again...

In case you didn't buy the first issue of Scorpio News, here is the Pascal version of a benchmark program from Dr Dobbs' Journal (no relation) no 83, which is supposed to show you how fast and how accurate the language you are using is.

```
{$t+ This tells the Hisoft compiler to include the run time routines that deal
     with the trigonometry and exponentiation, etc.}

PROGRAM BENCH;
VAR
  A : REAL;
  I : INTEGER;
BEGIN
  WRITELN('Started...');
  A := 1;
  FOR I := 1 TO 2499 DO
    A := TAN(ARCTAN(EXP(LN(SQRT(A*A)))))+1;
  WRITELN('Finished...');
  WRITELN('A = ',A:10:6)
END.
```

What it does is start with the number 1, and carry out a string of functions on it, which are intended to get back to the same value they started with, i.e. 1. This is then incremented, and the process repeated. So, if the computer is working to absolutely exact values in all its calculations, the final result will be 2500.

I ran this program using various Hisoft Pascals, and used a 20 year old Seiko automatic watch to time it. The results were as follows:-

```
     HP4                              44sec          320.347881
     HP5 using the Belectra HSA-88B   29sec            2.327E+03  (i.e. 2327)
     HP Pedigree                      44sec          320.347881
```

Now all of these are faster than the times quoted by P D Coker in issue 1, and the HP5 time, using that lovely floating point board, is miles faster than any of the competition. This pleases me immensely. But I know what you are thinking, "Those answers are miles out!" Well, Paddy Coker did ask if anyone could explain the apparently awful result the Hisoft Pascals give on this benchmark, and I am volunteering as I have done the Open University course M351, Numerical Computation (and I even passed!).

The program carries out 6x2500 = 15,000 floating point calculations, one after another. Any inaccuracy caused by the need to truncate the values to fit into the space allocated to a real variable will obviously mount up considerably over such a sequence of calculations. This tendency to reduce slightly the value of each result that will not fit in a real probably accounts for a large part of the deficit.

Clearly, anyone who is still awake will be wondering why all the other languages get so much closer to the "right" answer. They might feel that I am trying desperately to defend a compiler that I like, with no justification. In an attempt to find out what was at the root of the matter, I asked Hisoft how their tan and arctan functions worked. I thought they might use a Taylor series, or some such idea. I had, by fiddling about with the original program, discovered that almost all the inaccuracy was being created by the tan and arctan functions of HP4. Unlike every other software firm I have ever written to, Hisoft answered my enquiry. I quote...

"As you probably realise the Trig routines for the Pascal were written rather a long time ago. I think it was in March 1981! They were originally written in Pascal and then hand coded into assembler attempting to make the controlling code as tight as possible and with speed rather than space in mind. I can't actually remember whether they were written on a disk machine or on cassette. But either way the original Pascal source has long since disappeared. The algorithms used are mainly continuous fractions having divided the 0-90 degree range into a number of sections. Look at the assembler source; it doesn't make much sense since the only comments refer to the variables and constants from the Pascal version. The TAN function seems to use two entirely separate algorithms depending on the input value. One of the constants had to be fudged to avoid discontinuity which sadly leads to inaccuracy near 90. The algorithms were developed from a book published by HMSO in 1959 when speed of the algorithms used was really important. I can't remember its title but I'm pretty sure it involves 'digital computers'."

I would be prepared to bet that the "inaccuracy near 90" is the main reason for the bad result P D Coker got using HP4. Calculations carried out sensibly will give good results, while 15000 floating point operations in a row on a variable will indicate a silly program, and will give silly results. All the usual remarks about the usefulness or otherwise of benchmark programs are given even more emphasis by this result. I just wonder how the other languages gave so much "better" results? In the case of ComPas, this would be mostly due to its use of 6 bytes for a real number, instead of 4 almost everywhere else. This explains why David Parkinson didn't bother fixing ComPas to use the HSA-88B, as there would be quite a lot of work involved in that kind of patch! The time taken by ComPas was 330 seconds, seven times as long as HP4 and eleven times as long as HP5. Very much a case of horses for courses, I suppose...

The best result I have seen for the benchmark, in case you are still interested after all that, was 2.50000000000117e+03 given by a DEC VAX under VMS. Alas, we don't know how long it took, as the job put its results in a spooled file, and printed them later. My contact is hoping to get the program to send the "Started" and "Finished" to his terminal next time he runs it! Soon I will try it on the Open University DEC 20 mainframe...

Gratuitous Attacks on Other Contributors!

Not really, just a few comments on the content of issue 1. Quite a lot of it wasn't really directly concerned with the hardware listed on the front cover of the magazine. In particular, there were a lot of pages about Fortran, a language that should have died out years ago. And Dave Hunt's dictionary was rather incomplete. Items of this nature should be up to date, and include definitions relating to important languages like Pascal. You know the sort of thing I mean; "Recursion, see recursion"!

I was interested by the article about Gemini's Multinet, as I think there is far more sense in having at least one processor for each user than having lots of terminals tied onto one micro. Also, it sounds as if Multinet works properly, unlike the awful Econet. That being so, I wonder why it isn't mentioned in Network magazine? Back to Dave Hunt, whose reply to Bert Martin's letter was four times as long as the letter itself. Methinks he doth protest too much!

Installing ED80 for use with the SVC.

ED80 is the program editor I use, which is supplied with just about every language Hisoft produce. It comes with an installation program, to enable the user to set it up for a particular system's display hardware, and very good it is, too. What it doesn't do is word processing, with the word-wrapping and justification that implies. Programs tend not to need that sort of thing.

When I finished installing mine, following the instructions carefully (and the manual does explain the process clearly), I found I had a problem. The SVC was not showing any cursor. After a few minutes trying to work by pressing a key and watching where a character appeared, I decided I had to do something. Going back to the manual, I found out how to get ED80 to send an initialisation string to the SVC, and set it up to enable the cursor, in the form of a flashing block. I happen to prefer that sort to the more usual underline character. On running ED80, there was my flashing cursor blob. However, as soon as I touched a key, it vanished. Further investigation suggested that CP/M was resetting it to the original invisible type, for inscrutable reasons of its own. So I dug out the CP/M BIOS listings, and found the routine that was doing it. I made a CP/M with that routine removed, and ran ED80 again. The cursor still vanished as soon as it was moved from the top left of the screen. I went to the pub...

In the end, I had a look at the code of ED80, using Hisoft's MON.COM utility, and found that being jolly good chaps, and tidy programmers, they had routed all calls to the CP/M screen writing routine to a single routine of their own. Even better, there were useful areas of zeros in the program, which looked as if a patch inserted in them would do no harm. So I cobbled together a patch, and put it in. It may be ugly, it may be longer than it needs to be, it may even break some obscure law of Dijkstra or Codd, for all I know, but it also happens to work. Here it is, in case anybody needs it:

```
0005                 SYSTEM EQU  £0005        ; OPERATING SYSTEM CALL ADDRESS
001B                 ESC    EQU  £1B          ; ASCII ESCAPE CODE
0006                 DIRECT EQU  £06          ; DIRECT CONSOLE I/O ROUTINE
0150                        ORG  £0150
0150 CD0500          PATCH  CALL SYSTEM       ; DO A KEYSCAN
0153 B7                     OR   A
0154 C8                     RET  Z            ; RETURN IF NO KEY DOWN
0155 F5                     PUSH AF           ; SAVE KEY ON STACK
                                              ; SET CURSOR TO FLASHING BLOB
0156 0E06                   LD   C,DIRECT
0158 1E1B                   LD   E,ESC
015A CD0500                 CALL SYSTEM       ; SEND ESC TO SVC
015D 0E06                   LD   C,DIRECT
015F 1E59                   LD   E,£59
0161 CD0500                 CALL SYSTEM       ; SEND "Y" TO SVC
0164 0E06                   LD   C,DIRECT
0166 1E60                   LD   E,£60
0168 CD0500                 CALL SYSTEM       ; SEND £60 TO SVC
016B 0E06                   LD   C,DIRECT
016D 1E09                   LD   E,£09
016F CD0500                 CALL SYSTEM       ; SEND £09 TO SVC
                                              ; NOW ENABLE THE CURSOR
0172 0E06                   LD   C,DIRECT
0174 1E1B                   LD   E,ESC
0176 CD0500                 CALL SYSTEM       ; SEND ESC TO SVC
0179 0E06                   LD   C,DIRECT
017B 1E45                   LD   E,£45
017D CD0500                 CALL SYSTEM       ; SEND "E" TO SVC
0180 F1                     POP  AF           ; GET SAVED KEY FROM STACK
0181 C9                     RET
```

Having assembled that, and put it into the area of zeros at address £0150, it is necessary to alter the program so that the patch gets called. Replace the CD 05 00 found at £08A1 with CD 50 01, and save the new version of ED80. In order to do the same thing to the editor included in the new version of Hisoft Pascal, use the same patch, but put the CD 50 01 at £0AE6.

Using the GM870 MODEM to access Open University computers.

Open University students wanting to do this will be glad to know that it is easy to do, using the following line of data in the GEMTERM.DAT file: "OU,3,*********,Y,8,N". I have removed the telephone number, and put a row of stars, to make things more difficult for hackers. It is well known that they like a challenge, anyway. When they give up, they can always enrol for a course that makes use of the computers, and get given the number by the OU.

The only problem I had in running the TEST program, which tells you whether your terminal is suitable or not was finding what keys to press to generate the rubout character that is needed instead of backspace. The combination required turns out to be "control shift /" on a Nascom II keyboard. I searched high and low for that information, as I felt sure I had it somewhere, but finally decided it must have been in the Nascom I manuals I disposed of when I sold my Nascom I. Having resorted to a brute force search, I also found where the "|" is, that I had been looking for. C needs it, and it is "control shift ,". There, something useful, at last!

In the next exciting episode.

Reviews of Hisoft C and the updated version of Hisoft Pascal with its integrated programming environment, for the serious programmers. Recipes for anchovy wine for the deeply disturbed...

end.

The ZCPR3 System by C. Bowden G30CB

Part 1 - ZCPR Concepts and Features

In Issue 1 of Scorpio News I described some of the ways in which CP/M (Vers. 2) could be improved. In the article I described the ZCPR/CCPZ improved CCP, and briefly mentioned the ZCPR2 and ZCPR3 systems. I have now had a chance to try both of these upgrades, and in this article I will describe the features of this software. (Which I will call Z2 or Z3 from here onwards.) I will also describe some aspects of the various installation procedures, since these are not always explained in terms that are easy for less experienced programmers to easily follow. In some cases, there are a several different ways to implement the Z system, and these options are not always obvious. The installation information also assumes that the installer has the source code of his BIOS available. It is possible to install Z3 without this code, and several ways around this problem are suggested.

At the time that the article in Issue 1 was written, I did not have the most recent CP/M User Group Library Catalogue, and the one that I had included the Z2 disks. I assumed that this was the most recent upgrade of the Z system, and sent off my disks for the suite, (which occupies some 13 - 8in. disks and is between 2 and 3 Mb of software !). I also sent off for a new library catalogue. When it came I found that Z3 was available too. (Occupying another 2-3 Mb of disk space.) I probably drove poor Derek Fordred, the Librarian mad with so many requests for copies. Anyway, I resolved the confusion in my mind about the references to Z3 in the SB180 articles in BYTE, and ended up with a lot of disks full of programs.

There were a few weeks in between getting Z2 and Z3, which was useful, because the installation information with Z2 is rather more descriptive, and the installation less complex. Coming to Z3 after Z2 made the process much easier. I think that less experienced programmers starting straight off with Z3 would find the task rather difficult.

System Concepts

The original ZCPR/CCPZ brought the 'Path' concept to CP/M, displayed the USER number in the Prompt, and displayed the name of files being erased, and added a few extra useful features to CP/M which did greatly help make the system more friendly. Further expansion appeared to be restricted by the memory space allocations. The CCP and BDOS sizes were required to be kept to 2k and 3.5k for compatibility. The BIOS was the only place where extra features could be added, but these would be system specific, such as the screen edit, paging and dump features available in the BIOS's that we use on 80-BUS equipment.

The Z systems were written by Richard Conn, in the USA, and made available to all via the CP/M User Group Network. I do not know if other similar systems exist for CP/M, but the main feature of the Z system is that it uses extra memory ABOVE CP/M, thus retaining the unique structure of CP/M, and retaining compatibility with virtually all software.

The penalty to be paid for this is one of reduced T.P.A. size. Z2 requires 300H bytes extra for implementation which is not too bad, but Z3 requires about 5-6k for FULL implementation. It should be made clear however that it is entirely at the discretion of the user or installer which features of Z2 or Z3 will be implemented, and which omitted. Some features are virtually essential even on a 'small' system - i.e. one with say 2 floppy disks only, of about 300k capacity, such a an Alphatronic P.C., but others are much more suited to a 'large' system - i.e. 10Mb Winchester, 512k Vdisk, 2 - 800k floppies. Then again, some segments may be more or less desirable depending on the sort of application. The Flow Control Package and Redirected I/O Package might fall into this category.

One way around this problem could be to have several operating systems available, from a 'No frills' maximum T.P.A. CP/M where every bit of T.P.A. is needed, to a full Z3 system and fancy BIOS, for maximum flexibility, but with the loss of say 6k of T.P.A.

Both Z2 and Z3 are supplied with full source code for all of the software which includes some sixty or seventy utilities that are able to use the special features of their related Z system. It also includes SYSLIB which is a large library of subroutines that are used in the construction of the Z utilities, and may also be used in any other software that the user chooses. The user is thus supplied with all of the information needed to do whatever he likes.

Z3 is upward compatible with Z2, and so comments from here on will generally relate to Z3. The three memory extensions above CP/M with Z2 are the same as with Z3. There is one major difference which concerns the named directory facility. The Named Directory segment of Z2 is 200H bytes long, whereas that with Z3 is only 100H bytes long, (but could be made larger), and deals with 14 names. In practice this seems adequate. Also Z2 itself is not able to use or recognize the names, whereas the Z2 utilities can. This seemed to me to be a major deficiency of Z2, on a Winchester based system. On a Floppy based system, where the media is continually changing, names are perhaps of limited use, especially on systems where the disks are under say 500k. (Although the system allows the directory segment to be easily changed, and so a new one could be loaded for each application.)

```
              ZCPR2                                 ZCPR3

      --------------------- 0FFFFH      --------------------- 0FFFFH
       Z2 External Stack                 Z3 External Stack
      --------------------- 0FFD0H      --------------------- 0FFD0H
       Mult. Cmd. Buffer                 Mult. Cmd. Buffer
      --------------------- 0FF00H      --------------------- 0FF00H
          Named Directory                Named Directory Seg.
      --------------------- 0FD00H      --------------------- 0FE00H
           Modified BIOS                    External F.C.B.
      --------------------- 0EC00H      --------------------- 0FFD0H
             BDOS/BDOSZ                     Message Buffers
      --------------------- 0DE00H      --------------------- 0FD80H
               Z2CCP                          Shell Stack
      --------------------- 0D600H      --------------------- 0FD00H
                 ^                          Environment Data
                 |                      --------------------- 0FC00H
                 |                       Flow Command Segment
                 |                      --------------------- 0FA00H
                 |                       Input/Output Segment
                 |                      --------------------- 0F400H
               T.P.A.                    Resident Command Seg.
                 |                      --------------------- 0EC00H
             App. 53K (+2k)                   Modified BIOS
                 |                      --------------------- 0D800H
                 |                             BDOS/BDOSZ
                 |                      --------------------- 0CA00H
                 |                               Z3CCP
                 |                      --------------------- 0C200H
                 |                                 ^
                 |                                 |
                 |                               T.P.A.
                 |                           App. 48K (+2k)
                 |                                 |
                 v                                 v
      ---------------------             --------------------- 00100H

         External Path                     External Path      00040H
          Wheel Byte                        Wheel Byte        0003BH

      ---------------------             --------------------- 00000H
```

Fig 1 - ZCPR2 and ZCPR3 Memory Maps - Typical

The diagram of Fig 1. shows the segments and buffers that would be present in a full implementation of Z2 or Z3, although the addresses are flexible, and the size of the redirected I/O segment is variable or nil.

USER areas and moving about.

The only way in which the software can be effectively controlled on a Winchester drive is to use the USER facility of CP/M. To a non-technical user of the system, it appears that the disk is divided into a number of 'watertight' compartments, and that these know nothing about other user areas. This is rather how CP/M treats USER areas. In fact all that happens is that the first byte in the directory entry for any file is marked with a number that may be 0E5H, which means that the file is erased, or between 0 and 0FH (01FH - Z system supports users to 31).

This number indicates the USER area in which the file would be displayed for example by a DIR command, or be 'seen' by a PIP command.

CP/M utilities are unable (in general) to access files not in their USER area. The Z system and its utilities have much greater ability however, and can reach across USER areas, EVEN on different drives. Thus commands like:

```
A0>DIR C6:     or     C4>ERA A2:MYFILE.TXT
```

are perfectly legal. The CP/M method of accessing an USER area is also extremely inflexible. To move from A4: to B3: needs one to type B: and then USER 3. The Z method permits one to go as follows:

```
A4>B0: --->   B0>6: --->   B6>C:   --->   C6>D2:   --->   D2>
```

which is much easier. The Named Directory facility of Z3 adds even more flexibility, as will be described.

CCP and RCP Commands

The size limitation of 2k restricts the Z2 CCP to virtually the same commands as are provided with ZCPR/CCPZ, and prevents the CCP from being able to deal with Named directories itself, but does allow the movement described above. The USER command has been removed from Z2 and Z3 since it is redundant.

The size problem is overcome in Z3 by providing an additional 2k Command package in the RCP segment. This brings several advantages. The number and power of commands is extended. The RCP segment may be changed at any time to bring in other overlays with different features. The Z3 CCP can optionally be assembled to deal with Named directories, Password protection of directories, minimize the search 'path', give preference to DU: forms or to DIR: forms (i.e. to Drive/User or Named Forms) and so on. This increases the amount of code to the extent that there is little room for the standard and extra commands. It becomes a balancing act, trying various TRUE or FALSE combinations until the minimum amount of memory is wasted in the CCP area, and then providing the remaining commands in the RCP.

Options available in the CCP are DIR, LIST, TYPE, GO, ERA, SAVE, REN, GET, JUMP, NOTE. In my system the Z3 CCP provides ERA, SAVE, GO, and NOTE. GO will run a program in the TPA at 100H. NOTE allows a comment line to be displayed on the SCREEN.-

```
A0>NOTE This is a comment.
A0>NOTE A  semicolon  will  have  the  same effect
A0>;Like this
```

The RCP segment .LIB source code, like that for the CCP, will allow selection of more commands than will fit into 2k, so a compromise is necessary. In practice this is not much of a problem. The commands available for selection are CP, DIR, ERA, LIST, TYPE, PEEK, POKE, PROT, REG, REN, NOTE, ECHO, WHL and WHLQ.

The commands enabled in my system RCP are DIR, REN, TYPE and LIST (Send file to Printer). I can also fit in P)eek, POKE, PROT, REG, REN, WHL and WHLQ.

The **CP** command allows file copying. It allows renaming during copying but it does not perform a Verify. I like to always verify copies, so I use a utility like MCOPY, VFILER or NSWP.

The **P)eek** command will display 256 HEX and ASCII bytes from the given address, and the POKE command allows bytes to be changed using either HEX or ASCII.

PROT allows file attribute bits to be set/reset, e.g. SYS/DIR RO/RW.

REG allows manipulation of certain reserved memory locations that can interact with certain Z system utilities. I have not yet familiarized myself with this facility.

NOTE is the same as the optional command in the CCP.

ECHO will cause the text following the command to be sent to the screen or printer. It also allows command strings to be sent to the output device, since all characters are sent as typed.

WHLQ enables the state of the 'Wheel Byte' to be examined, and **WHL** permits it to be SET or RESET. The Wheel Byte is a system security device. Any or ALL CCP and RCP commands may be set to check the Wheel Byte before proceeding. Thus certain commands might be permitted, but others barred. On a Remote Access system, the Wheel Byte can prevent users from 'Vandalizing' the system. Access to other directories, drives, etc, can be barred.

If the user types an H, whilst at system level, the commands enabled in the current RCP will be displayed. Note that several of the commands provided in the RCP are also available as command files, which eases the problem of deciding which commands to include or omit from the RCP.

A description of the ZCPR3 Segments and Buffers

CP/M 2.2 can be 'moved' in 1k units via MOVCPM in the ordinary way. By altering a byte within MOVCPM however, moves as small as 100h bytes are possible. This is described in more detail in the installation part of this article. Thus once the decision to resite CP/M is made, at least another 256 bytes are added or removed.

If the additional software that caused the need to move CP/M down does not need all of these bytes, it is just as well to use them for something else, as to waste them. So if for example it is decided to implement the External FCB and this needs more space above CP/M, then the message buffers and shell stack become possible as well at no extra cost in memory.

External CCP Stack.

In order to 'cram' as many commands as possible into the CCP, the user has the option of placing the CCP stack inside or outside of the CCP memory area. This is especially important with Z2, where there is no extension of the CCP (RCP Segment). The External Stack area occupies 30H bytes, if enabled, which is generally the case.

Multiple Command Line Buffer (MCB).

Standard CP/M allows only one command per line of input. It is possible to get around this to some extent by using extended command processing, i.e. Submit type operation. However the occasion often arises when it would be nice to be able to type a 'one-off' and go away to make a cuppa. The type ahead available on the Gemini keyboards helps somewhat, but seems to gobble up characters at odd times even if a judicious number of returns are typed between commands. Z2 and Z3 permit the user to enable a multiple command line facility. This allows multiple commands to be typed, separate by a semicolon.

The buffer is (normally) 200 bytes long. Commands like:

```
   A0>M80 =BIOS;ECHO Linking;L80 BIOS/N,BIOS/E;ECHO Erasing;ERA *.BAK
```

then become possible which is very useful to the user. The MCB is in fact widely utilized by the Z system utilities for maximum power and flexibility. If a command string is found in this buffer on cold boot, the command(s) will be executed. Unless the multiple command facility is enabled, the Z system cannot function to any real advantage.

Memory Based Named Directory Segment.

As intimated above, I feel that this facility is not of any especial advantage on a single user system based on a couple of smallish floppies. On a system with several users, especially where a Winchester is used, the situation is completely different. If this facility is enabled, up to 14 (Could be extended at cost of more memory) Names may be allocated to Drives and USER areas. Thus A16: contains .HLP files. A15: contains most system utilities, etc. Typical name allocations are:

```
        A16 -- HELP, A15 -- ROOT,   A0 -- BASE,   C0 -- BAK,   P0 -- MDSK
```

The names allocated are a matter of user preference. I keep Z3 System development files in B0, so this I call Z3, and BASIC in A4: so this is called BAS. The main benefit of this is that one does not have to remember which DU one has allocated to a certain function. The directory name is displayed in the prompt:

```
               A0:BASE>      C0:BAK>      B6:PAS>      P0:MDSK>    etc;
```

To move about one may use either the DU: or DIR: form and so-

```
             A0:BASE>ROOT: {enter} ---->   A15:ROOT>   is a valid as
             A0:BASE>A15:  {enter} ---->   A15:ROOT>
```

In addition Z system utilities recognize either DU: or DIR: form so-

```
                     MCOPY ROOT:=BAK:THISFILE.BAS  or
                     MCOPY ROOT:=C0:THISFILE.BAS   or
                     MCOPY A15:=C0:THATFILE.DOC    or
                     MCOPY A15:=BAK:*.*
```

are ALL equally valid. Directory names may be up to 8 characters long, and may optionally be password protected by a password of up to 8 characters long. For more security, the DU: form of access may be disabled, so that password protected directories become more secure, especially to non technical users. Wheel protection or Disabling of the 'Peek' command, and removal of debug utilities would be strictly necessary to give added protection where more experienced users were using the system. A moderately secure system is possible however, in a multi user environment.

One particular point about allowing DIR names and USER areas to be displayed in the CCP prompts is that the screen edit feature of most of our BIOS's needs to be modified. This feature starts by looking for the '>' prompt in the EDIT line returned from the Screen, and in both SYS and MAP BIOS's starts at about the 4th character out from the start of the line. I have modified this to start out at about the 12th character as the extra data before the '>' can move this prompt character well out into the line. If the BIOS source is available this can be easily changed, otherwise it becomes a question of asking the supplier to tell you which byte changes the start point of this routine, or doing a bit of work with a debugger or disassembler. Generally no extra code is needed, (in fact I rewrote the routine in my BIOS and saved a number of bytes), - just a change of address to point further into the edit buffer and increase the loop counter. e.g. the original code in SYS18 was LD HL,EDBUF+3, LD B,3 and I changed this to LD HL,EDBUF+12, LD B,12. EDBUF is a block of about 83 - 00 characters in a system image, and the code to be modified usually lies just before this buffer.

The need to be able make such modifications to the BIOS, and Cold Boot initialization routines again underlines the need for system suppliers to make the BIOS source code available to users.

External File Control Block.

Use of the external FCB releases more room in the CCP area for commands, and provides Z system utilities with a means of accessing the data and using it. Only 48 bytes are used by this buffer.

Message Buffers.

This area uses only 80 bytes, and the author of Z3 emphasises the need to instal this feature if Z3 is to work properly. The area is used to convey system information between Z3 and utilities.

Shell Stack.

The shell stack uses 128 bytes of memory. Z3 supports the use of Shells, and the Shell Stack is necessary for this feature to be used.

I must admit to being unfamiliar with shells, and I quote from the Z3 documentation - 'A Shell is a front end processor which is invoked in place of the Z3 CCP input routine, allowing command input to proceed in a variety of different, perhaps more user friendly ways.'. Programs that are configured as shells lie in memory rather like the layers in an onion - one above the other, rather than replacing one another, so that control can be passed from one to the other. Thus MENU, which is a shell can run another shell like VFILER. When VFILER is exited, control would revert to MENU.

The Environment Descriptor.

This segment takes up 256 bytes of memory, but is essential if the Z system is to operate. It consists of two parts, each of which is allocated 128 bytes. One part describes the Z3 environment itself. It holds the addresses of all of the active segments and buffers so that Z3 and utilities know what is available and where to find it. It also contains some general data about Terminals and Printers, such as Screen Width, Screen Lines, Printer Width, Form Feed supported, and so on. Two terminals and four printers may be described.

The other part, called the TCAP - Terminal Capabilities, describes the special features of the system screen handling such as Cursor Key bytes, Cursor movement Byte string, Inverted Video Commands, Clear Screen and Line. These are then available to the Z system utilities, which can then use the features for better user interface. Z3 comes with a large library of Terminals described, but the Gemini IVC/SVC is not amongst them. It is quite easy to set up an ENV segment for any terminal however. There are over 80 bytes free in this part of the .ENV segment, which may be utilized in other ways, as suggested in the installation part of this article.

Flow Command Package.

This is another area of Z3 that I have not yet investigated to any extent. This segment takes 512 bytes of memory. It provides a number of conditional commands like IF, ELSE, FI, XIF that allow command sequences to be conditional. A command like:

```
IF EXIST MYFILE.TXT
    TYPE MYFILE.TXT
ELSE
    IF ERROR
        ECHO MYFILE.TXT not found
    FI
FI
```

The above example may appear trivial, but the concept can be extremely useful in Error handling, or controlling the flow of a command chain. The feature is not essential to the successful operation of the Z system however, and may be omitted to save the .5k it occupies, if memory conservation is important.

The Input/Output Segment.

This segment is optional, and its size is dependent on what facilities the user has included. In the memory map Fig 1., it is shown as being 1.5k bytes long. Since this segment is hardware specific, it becomes the responsibility of the user to write this software. An example source file for an IOP is provided, to show how it may be done. Since I have only one Screen and Printer, and have no great need of the facilities available in the IOP, I have not implemented this segment at present, so no memory is lost to it on my system.

The purpose of the IOP is to provide more flexible control of system I/O and to allow redirection and so on. It becomes easy to switch to different terminals and printers. If it is implemented, it can to a large extent replace BIOS I/O, and this could result in a smaller BIOS, offsetting the extra memory needed by the IOP.

The Wheel Byte.

The purpose of this Byte has been described above. It is just a one byte location, allocated somewhere in memory. In Z2, address 003BH was used. In Z3, the author has switched to 004BH in the documentation, but has not explained the reason for the move. Since this would restrict the External Path length, I have left the Byte at 003BH. The byte may be examined, set or reset as described above, and a utility to manipulate it is also provided.

The External Path

The path may be made internal to the CCP, but would then disappear when programs are loaded. The author chose to use locations from 0040H in both Z2 and Z3, and this seems reasonable. The path may be as long as one likes, but is in reality restricted by the memory space available for it from 0040H, and the fact that the longer it is, the longer a search takes, especially when one has type a 'dud' file name. Z3 can be assembled to minimize the path so that locations that appear more than once are only searched once. The current drive and user are ALWAYS searched first anyway, so do not appear in the path. On my system I currently search up to seven locations after the current DU. They are:

    $0, - A0, - A15, - A$, - B0, - B$, - C0 where $=Current Drive or User

bearing in mind that A:, B: are the two parts of a Winchester, C:, D: are Floppies. The free memory of the TCAP could be used for the Path and Wheel Byte.

ZCPR3 Utilities

There are too many Z3 utilities to describe more than a few in any detail. All of them are designed to work with Z3, although some will run on standard CP/M. Many of them read the ENV segment and make use of the screen facilities defined there, resulting in an improved display, and allow the cursor keys to be used. Many also make use of the various buffers and stacks described above.

In other words Z3 and its command utilities are an integrated system. Together they form a remarkable system. Z3 without its utilities is still good. Many of the Z3 utilities will not work on a non-Z3 system. They fall into several natural groups, depending on the function.

Copying -

MCOPY is appropriate when multiple copies or complex strings of commands are required. It copies files from any DIR/DU to any other (including users between 16 and 31), verifying the copy and reporting errors. It will perform multiple copies, allowing disk changes en route, and it accepts a string of commands. MCOPY, in common with most Z3 utilities, contains internal 'HELP'.

VFILER provides an alternative file manipulation utility. The many varied functions provided are reminiscent of SWEEP, but the system uses a pointer which moves around the file lists displayed on the screen. Files are tagged and may be copied, deleted, sized and so on. As with the other Z3 utilities, DIR: and DU: forms are allowed. A '?' switches to a HELP page, and another '?' switches back. VFILER supports the use of its own command files. One problem that I have found with VFILER is that, unlike SWEEP or NSWP, the files remain fully tagged after an operation, so if one has forgotten to tag one or two for the first operation, it is not possible to tag just those, copy or whatever and then retag all, so flexibility suffers a little bit.

Directory display and Maintenance -

Beyond the simple displays provided by the DIR command of the CCP, RCP or **DIR.COM**, two utilities are available for more sophisticated displays. **XD** and **XDIR** have a large repertoire of options available, but even in the simple form (i.e. no options), give a directory with very full information on files, sizes, attribute bits, space taken by the selected files, space remaining on disk, number of selected files, and number of files on disk. They also use the ENV data to page the screen output. The XDIR utility in particular is very powerful, allowing all users, disk output, acting as a file scanner and allowing the display of files to be negated. XD and XDIR support printer output, display format option and attribute display.

The ERA command of the CCP/RCP provides essential erase facility, and may be assembled to allow verify or inspect operation. **ERASE.COM** backs up the CCP/RCP ERA command, and greatly extends it. The user may opt to include 'SYSTEM' files, Read only Files, or to operate in an 'Inspect' mode, with user input for each file. Options may be mixed.

UNERASE will recover erased files, although the usual precautions are necessary. It is always best to recover files IMMEDIATELY, if they are accidentally erased. If several erased files of the same name exist in a directory, a by no means unusual occurrence, it is better to use **DU3** or **SPZ** to edit the directory directly. Each occurrence of the file in turn can be recovered, to an 'empty' user area if possible, and then each one can be checked. It should be remembered that a large file may have several entries in the directory for its extents.

The file **CLEANDIR** can be used to clean up a directory from time to time. Users of the SAP?? series from the CPMUG will be familiar with this operation. It will sort the names into order (ascending or descending at the users option), and rewrite the directory, removing all traces of erased file names. There is no way to recover erased files after a CLEANDIR, but it is a good exercise, because removal of the names of genuinely erased files gives a better chance of restoring the correct file if an ERA error is made subsequently.

The REN command of the CCP/RCP is backed up by **RENAME.COM**, which allows batch renaming of files, and supports control and inspect modes, and may be made to work with system files and read only files.

FINDF is another useful program. If one cannot remember where a file is on any drive on the system, FINDF will search for it and report all places where it is to be found. N.B. ALL drives are searched, so if a disk is not mounted the system will crash/hang. If a VDISK is in use and VFLIP operative FINDF gets confused, and will not find files on the first half of the Winnie.

Named Directories -

The command **PWD** will list the current Named directories and their associated DU's. This command may be made to display any associated Passwords, but ONLY if the Wheel Byte is set to allow this.

MKDIR will allow the creation of new named directory segment files. This is also possible by editing and assembling the SYSNDR.ASM/LIB files provided with the system, but MKDIR is interactive and easier to use.

CD.COM works with either DIR: or DU: forms. If CD is used to log into a new directory, instead of doing this by just typing say B7: or BAS:, when CD logs into the target directory, it looks for a file (**ST.COM**), which contains a command sequence. This could, for example, set up new function keys, reprogram the printer, change the Named directory segment, change I/O parameters and carry out other similar setting up operations.

File Protection and Maintenance -

The **PROT** command of the RCP allows attribute bits to be changed. **PROTECT.COM** is a much more sophisticated command which will set or reset the file name tag bits, filetype attribute bits, and will allow 'inspect' and 'control' modes so that the user may exercise varying degrees of control over the process.

CRC.COM is provided in Z3 compatible format. The same algorithm is used as in the other CRC command files that have been available from the CPMUG over the years, so that compatibility is maintained. The Z3 CRC command outputs the file name, Size in K Bytes and records, Counts lines of text (text files), supports printer output, Disk File output, and allows comments to be added to each file, supports optional inspect mode. Options may as usual be mixed. (On the Z3 disks, is included CRC.COM Vers. 5 which was described in a late 80-BUS News. Z3 CRC.COM does not compare CRC's with a current CRCKLIST, as CRC Vers. 5 does, so both files are useful.)

DIFF.COM is a file comparison utility. It may optionally be told to stop at the first difference, or list all differences on a byte for byte basis. Files may be in different directories. Multiple runs are supported. i.e. Disks may be changed. There are a number of similar programs that have been around for some time, some of which will report differences, and step a few bytes on one file until a match is again established, which is useful for comparing files that are essentially the same. It is therefore useful to have the Z3 utility and one or two of the others available.

Help Systems -

The version of **HELP.COM** supplied is tailored to Z3, and will find files in A16: which is suggested as the HELP: directory. Extensive help files are provided, totaling over 500k, although some refer to SYSLIB and VLIB which are not directly related to the operational aspects of Z3. Some of the HELP files operate on an elaborate 'tree' system. Sufficient help is provided to enable all of the utilities to be successfully used. With a Winnie, this HELP can all be online. Two supporting utilities **HELPCK.COM** and **HELPPR.COM**, are provided to check the format of HELP files that one might write, and to allow hard copy of the .HLP files without wasting paper.

PAGE.COM is a utility that can be used instead of the TYPE command to display files on screen. It is clever enough to make allowance for lines longer than screen length.

The **SAK** utility is extremely useful. It waits for the user to strike any key - hence the name. It can provide a number of features such as time out, ring the console bell, abort to CP/M.

SHOW.COM is the Z3 display utility. It will display data about the Z3 system, its' buffers and segments in a very informative and useful way.

PRINT.COM is a sophisticated file print utility. It supports a number of options like multiple file printing, page skipping and so on. I have customized my version so that it reads the system clock and prints this on the first page of the output.

Batch processing is supported in several ways. Normal **SUBMIT** type operation is possible. In addition to this Z3 provides **ZEX** command processor, and **ALIAS** alias generator. ALIAS is very easy to use. It prompts for a multiple command input line exactly as one might type at the console, and converts this into a small .COM file that can then be invoked at any time to run the commands. Parameters are supported, so the facility is both powerful and quick to set up. ZEX is more sophisticated, and to date I have not had a chance to explore its potential.

MU3 is a memory debug utility. It may be loaded as a transient, or if the TPA is to be preserved, a version is provided that loads to the RCP area. MU3 enables memory to be displayed and edited. Several other commands are also available. **DU3** is a Disk Editor similar to the various DU's in the libraries. It is rather more interactive than these, with a better display.

General Utilities -

PATH.COM allows inspection and redefinition of the PATH.

Z3 supports comments. A ';' in the first character position marks the line as a comment. The RCP and command file NOTE also mark the line as a comment. The command **COMMENT.COM** goes a little further, and replaces the CCP prompt with the word COMMENT.

The **ECHO** RCP and transient command will 'ECHO' the text that follows to the screen or printer. The main purpose of ECHO is to send messages to the screen during multiple command processing. Otherwise there might be no output for long periods, and one could wonder what was happening. Since characters are sent without modification, this can be used to reprogram the printer. Output to the printer must be preceded by a '$' character. ECHO $^L would send a form feed.

Menu Systems -

If the system is to be used by several persons, or where the users are non technical, extensive **MENU** type operation is provided. It is possible to lock the user into a MENU that does not permit escape back to the Z3 system. A menu check utility is also provided to ensure that the MENU is correct in format. A screen oriented system **VMENU**, is also available, with associated format checker. The Menu and Vmenu commands look for a command file of type .MNU/.VMN, and read it to obtain the commands to be processed. Menu's may access other menu's. Extensive .HLP files are provided for both these commands. Both MENU and VMENU are shells.

Flow Control -

Beyond the flow control available in the FCP segment, additional Flow control is provided in the command **IF.COM**. Several options are supported that may be tested for True/False, such as *Null Files*, *Error Flags*, *File Existence*, *Null* afn, Wheel, Register value, matching afn's.

The above summary covers the majority of the Z3 utilities. There are others, but they will generally be more specialized, concerning things like redirected I/O and are fully described in the HELP files.

ZCPR3 in Use.

If possible the BIOS cold boot 'sign-on' messages should be altered to announce the use of Z3, and any other enhanced features. The installation of Z3 will be gone into later, but the cold boot must also initialize some areas, and should set the Wheel byte, set up the default Path, and any startup commands. These can be quite extensive since there is room for 200 characters in the command buffer.

On my system, I send a different character set to the SVC (CPMUG Software), and use a Z3 command ECHO, to prompt for ^C or any key, and the Z3 command SAK (Strike a key), to wait for input. If it is a ^C, the system goes to the CCP prompt. Any other key goes straight to an online HELP file that I have provided, summarizing the main features of Z3.

On going to the CCP prompt, the first thing to be noticed is that the prompt includes the DIR: name as well as the DU: - for example A0:BASE>. A few tests with the DU: and DIR: forms confirm the ease of moving around, and cross user/disk commands like A15:ROOT>DIR P0: show the flexibility. With the various command and associated overlay files in the relevant names directories, it is so useful to be able to type BAS: or PEN: or WS: to access the required area, without having to remember on which DU: I have decided to put things.

Type an H, and see the RCP commands displayed, Type a P xxxx, and get a 256 byte Hex and ASCII dump from address xxxx. Try a few POKE commands, and use P to see the result. (Don't poke the Operating system unless you know what you are doing.)

I use a Winnie with .HLP files in HELP:, and Z3 commands and other utilities in ROOT:. It is very nice to be in say D6:, and be able to type HELP DIRS without moving or worrying where the files are. The path and HELP.COM sorts it all out.

On my system I flip Drives A: and Ramdisk - A: is now my vdisk. I have set up my Winnie as Drives H: and B: (The H: is for Hard). The system logs on to A0:MDSK> on Cold Boot and I normally use MDSK> for work as this greatly speeds any warm boot, and processing is much faster than even a Winnie).

This completes the review of ZCPR3 and its utilities. In the second part I will discuss the installation of the system.

Benchmark Update by P.D. Coker

Since carrying out the series of benchmarks which were published in Scorpio News V1 I1 (Jan.- Mar. 1987), I have acquired one or two more compilers and interpreters which I've tried out on a Gemini Multiboard system fitted with the GM888 processor, under CP/M-80 (2.2) and CP/M-86, as appropriate. For purposes of comparison, I ran the same benchmark on an Amstrad PC1512 using MSDOS and DOS Plus - the latter enables one to run CP/M-86 programs.

The results are quite interesting. The benchmark used was obtained from Dr Dobbs Journal in which the idea was to produce a result as near to 2500 as possible, using a range of intrinsic functions available in the interpreter or compiler.

The following language implementations were available:

Nevada Fortran	(Z80, CP/M 80)
Utah Fortran	(MSDOS)
ProFortran	(CP/M-86 and DOS Plus)
Microsoft Fortran-77	(MSDOS)
BASIC86 v. 5.28	(MSDOS)
BASIC86 v. 5.21	(DOS Plus)
GWBASIC	(MSDOS)

The Fortran versions were all compiled and the BASICs were all interpreted. The Amstrad uses the 8086 16 bit CPU with a system clock frequency of 8MHz, while the GM888 uses the 8088 8/16 bit CPU at 8MHz with I/O via the 4MHz 80-BUS.

The results were as follows:

Language	O/S	Clock freq. (MHz)	Result	Time (secs.)
Nevada Fortran	CP/M-80	4	2420.50	470
Utah Fortran	MSDOS	8	2420.50	320
ProFortran	CP/M-86	*	2775.34	54
ProFortran	DOS Plus	8	2775.34	30
BASIC86 (5.28)	MSDOS	8	2716.96	69
BASIC86 (5.21)	DOS Plus	8	2179.85	73
ProFortran	CP/M-80	4	2773.50	107++
BASIC86 (5.21)	CP/M-86	*	2179.85	130++

* Gemini quoted value 'about 6MHz'.
++ Results obtained in earlier Benchmark tests (see Issue 1)

The results are of considerable interest since they show that the quoted speed of the Gemini 8 bit CPU and 8/16 bit co-processor combination is much greater than the actual value. [Ed. - NO! See my comments below.] The Amstrad, working at 8 MHz is actually 1.8 times faster than the Gemini combination which appears to have a true overall speed of 4.44 MHz, marginally slower than the 4.77 MHz of the IBM PC. I was a little concerned about this so tried the program on a Samurai 16 bit machine (which runs MSDOS and CP/M-86) using the MSDOS and CP/M 86 versions of BASIC. The system clock is 4.608 MHz and the CPU is an 8086. The numerical values obtained were the same as before but timings were 115 secs for the MSDOS version and 122 secs for the CP/M-86 version, thus confirming the slowness of the Gemini combination.

[Ed. - Sorry, but there is one factor here that you appear to be omitting to take into account, although you did mention it earlier. The GM888 and IBM-PC both use the 808<u>8</u> processor which only has an 8-bit external data bus, as opposed to your Amstrad which has an 808<u>6</u> with 16-bit data bus. It is therefore not correct to assume that the IBM PC speed would be 4.77/8 times that of the Amstrad, it is considerably slower than this. Nor is it correct to calculate the effective Gemini clock speed by comparing its benchmark result to that of the Amstrad. Neither of these calculations are comparing like with like.

When the GM888 board was initially introduced Gemini ran benchmarks on it and on the 4.77 MHz IBM-PC. The GM888/MultiBoard combination ran the benchmarks approximately 25% faster than the IBM - hence the 'effective' speed of the Gemini is about 6MHz, as quoted.

The areas where your figures are obviously interesting are (a) in being able to calculate the performance improvement to be had from using the 16-bit data bus versions of the Intel family as opposed to the 8-bit ones, and (b) in seeing the ACTUAL performance differences of the Amstrad vs. the Gemini.]

Microsoft appear to have improved the accuracy of their BASIC in it's MSDOS version and out of interest, I looked at the performance of their 16 bit FORTRAN and PASCAL which run under MSDOS. Using appropriate versions of the benchmark program, Fortran-77 ran in 162 secs. on the Amstrad and produced an acceptable value of 2477.24 while the Pascal completed in 130 seconds and produced an appalling 3904.77. I'm glad I don't have to use it! I had hoped to try out the MSDOS version of ProFortran and the CP/M-86 version of ProPascal but I didn't have access to copies. I suspect that they would run fast and produce reasonably accurate results.

I included both versions of Ellis Computing's FORTRAN because they are cheap and quite accurate. Their execution speed is a little slow but for many applications, this is not a great disadvantage.

Letters to the Editor

Attention MultiNet Users

Dear Sir,

I would like to make contact with fellow subscribers of Scorpio News who use a Gemini MultiNet System, so we can exchange news, views, ideas and problems.

To encourage you to contact me, if you send me a formatted QDDS disk and return postage, I will give you a corrected version of NSWEEP207 that will run on MultiNet 2.4. Furthermore, it will copy to all 32 CP/M user numbers, has more facilities than NETSWEEP and is half its size (12K).

I hope soon to have a version working DIRECTLY with the 96 MultiNet user numbers!

Yours sincerely, John S. Radley. 81 Drayton Ave., West Ealing, London, W13 OLE.

Wrong Computer?

Dear Sir,

You may be interested to know that over the years I have only missed out on about 5 of the very first Nascom mags. I am grateful for the hints and information from Dave, Paul and Chris (not forgetting the many other contributors) and would like to thank them for their efforts.

I hope that Scorpio News will continue to maintain the standards which I'm sure most readers prefer. I personally feel that the mag has a unique way of presenting itself as a source of information without 'Talking Down' to its less knowledgeable readers. It would be a pity to see it turn into the usual 2 pages of info and 52 pages of adverts!

I have no ambitions to join the computer 'Professionals' but I have learned a great deal since buying my old Nascom 2 in 1978. Unfortunately the Nascom has now gone, but I intend to subscribe to Scorpio News for at least one more year if only to read the items of general interest.

I now possess a CBM 128 which suits my circumstances reasonably well but might have done better if its firmware writers had read more 80-BUS mags!!! I was thinking of trying to re-write Nas-S_'s in 6502 code but that's another story.

Once again my thanks to the people who have contributed to the Nascom mags so far; I hope you don't mind a CBM user reading your efforts.

Yours faithfully, Charles Tame, Nuneaton, Warks.

More on Benchmarks

The Editor,

You may be interested in the following benchmarks on some workstations I use at work.

```
     Whitechapel MG1 (32032 Xenix) - 13.4 sec -> 2500.0000

     DEC VAX (11/780 + F.P. Unit)  -  4.5 sec -> 2500.0000
```

Yours sincerely, R. Mohammed, Glasgow.

Short and to the point

Dear Sir,

Thank you very much for Scorpio News Vol 1 Iss 1 received yesterday. May I congratulate you on an excellent product in the best traditions of INMC, INMC-80 and 80-BUS News, all of which I have avidly read and re-read!

Yours sincerely, C.G. Woodford, Torquay, Devon.

Gemini - what's going on?

Dear Sir,

Dave Hunt's comments about IBM compatibles in the first issue of Scorpio News are timely, to say the least. What a pity that Gemini didn't exploit the potential of the 80-BUS to produce a card that would run MSDOS programs and graphics. It's too late now and some 80-BUS users will have purchased (depending upon their financial state) one of the PC clones that are widely available at present, in spite of the reported lack of technical backup. Perhaps we have been spoilt by receiving good service from the majority of companies supplying 80-BUS in the past but I am left with the impression that even this is declining fast. It's all rather sad.

Gemini, in spite of promises, don't appear to have produced any information on new 80-BUS products or even a price list recently and Lucas - well..! I asked them if I could have copies of the circuit diagrams for the RAM A and Buffer boards (really up to date stuff, this!) and was politely told that due to company policy and copyright laws, they do not issue circuit drawings. How pathetic! Gemini don't appear to answer letters these days - perhaps they are all beavering away on the Challenger. I (and many others) would be very interested to know what new boards are available for 80-BUS, and the advertisements from Map 80, EV Computing and Newburn in the first Scorpio News were most helpful - and Kenilworth Computers are amongst the most courteous and helpful of dealers.

Sadly, one has to forget about any significant help or new offerings from Lucas for the Nascom, but I am less willing to accept that Gemini have gone down the same path. Perhaps I am being naive but the 80-BUS (and the Z80) are by no means obsolete - one of my machines, with the aid of some excellent software from Map 80, is currently being used to transfer all sorts of software between different disk formats for both micros and mini-computers. No other manufacturer has anything even remotely approaching the versatility of this or the Gemini MFB or Mike York's All-disk systems. Dave Hunt mentioned the excellent Gemini modem card - and there are the Pluto boards from Io Research which have become a de facto industry standard (I'd like one of those but the piggy bank won't stand it now I've bought an Amstrad PC1512!) The quality of the customised BIOSs for CP/M 2.2 and Plus are outstanding compared with the feeble offerings for other machines, thanks to the efforts of Messrs. Beal, Parkinson and Watkins, to mention just a few contributors.

There's no point in moaning about the lack of information from Gemini or Lucas's peculiar company policies unless one is prepared to do something about it. Lots of polite letters from users of 80-BUS products to Mr Marshall might persuade him to reveal the goodies which we were assured were to come from Gemini in support of 80-BUS - possibly in the form of an insert to Scorpio News. Lucas are probably beyond hope, but I intend writing to Mr Marmion, the sales coordinator, to let him know what I think (politely, of course) of company policy!

Yours sincerely, Dr P.D. Coker, Orpington, Kent.

Withdrawal symptoms

Dear Sir,

When I received your leaflet in December '86 for Scorpio News as an 80-BUS News subscriber I took no action as I own a Nascom 2 and could not afford to update it to CP/M. Instead I purchased an IBM Clone.

Documentation of the IBM Clone is non-existent and I miss the ability to work on a machine which has hardware that I can understand.

However, the arrival of the first issue of Scorpio News triggered withdrawal symptoms and I now enclose my subscription.

I look forward to the rest of the series of articles on Disk Formats and CP/M Disk Routines by M.W.T. Waters and enjoy the Dave Hunt pages and Doctor Dark's Diary (having all previous 23 episodes) - keep up the good work.

Yours faithfully, G.H. Hestor, Brentwood, Essex.

Scorpio "IBM-Clone" News ?

Dear Editor,

Well I guess it's nice to be back in harness again, although I'm not so sure about being nagged about deadlines. I tend to write when the urge takes me, and producing something to be ready by a specified date is going to impose a discipline which I suppose will be good for me. Still never mind, it's good to see the old rag appearing under a new name even if it does cost a bit more. Let's all try and keep the quality up so that those who buy it think it's good value for money and so keep on buying it.

Next on my list is the content, I found the last issue chock full of the usual useful goodies from the usual contributors, particularly the article on 0.5" tape. In that particular instance, it was a little late, as I had done all the basic research a month or so before, for something I was up to. None the less, it confirmed all I had discovered and filled in a few gaps. It was nice to see MAP back in the fold, and I understand there will be a couple of useful articles coming up about some of their products which I confess I know very little about. Dr. Dark, of course, comes across strongly as usual, and so did Paddy Coker. Mike Waters' article I've already read, having had that on disk since last May. Very wise to make it a serial, it's fascinating but a bit heavy all in one go (apart from taking one whole magazine to itself).

Now, what's missing? I guess most of the readership are what I would call 'mature' micro users. By that I mean that they know what they (or someone else) is talking about, and are not grey haired and over weight, like myself, or knowing a few of them personally, perhaps, both. Might it not be a good idea to commission a few articles at the rank beginner level, you know like the Dodo's Guide I wrote for INMC all those years ago (no I'm not volunteering)? That way you might approach new readers who at present are being poorly served by the current stock of mags which seem to spend most of their time reviewing clever games, but not explaining how to write them or what makes the machines tick. Now I know we are primarily a Nascom/Gemini/Map based magazine, but some insight into the workings of some of the other Z80 based machines, such as the Einstein (which has a remarkable resemblance to a Nascom with disks), wouldn't do me any harm and I might pick up a few good ideas.

As threatened in my last offering I have had to defect (this week) to the ranks of IBM clone owners, as there was a limit to the amount of work-work I could do at home on my trusty old Gemini/DH machine (it had been reduced to producing documentation, and very little else). Now I know any defection from the ranks is sad, but in my case it's a case of expediency. If my family expect me to continue to earn the daily bread, then unfortunately, I must have the correct tools for the job.

Now one thing I have found about owning an IBM clone is that the amount of information which comes with it is minimal (like 10 pages of manual describing, in bad English, what the various links do). I need information, and one place I won't find it is here! As most of those afore-mentioned 'mature' micro users which I know personally are in a similar position and already have various breeds of IBM clone (ranging from Ying Tong Tiddle-I-Po nasties, through Amstrad 1512's to quite good Japanese AT clones) and all suffer from the same lack of information, might not a small section on IBM clones be interesting? After all they share derivatives of the same processor used in the GM888 8088 card and a small (readable) series on 8088/8086 assembler would be very useful, particularly about the segment registers which insist on inventing funny numbers all of their own when I try something other than the most mundane assembler program.

The inclusion of IBM clones won't necessarily be popular with the Nascom/Gemini/MAP purists, and such things might only appeal to myself and a few others, but it might be interesting to find out if the readership has a substantial number of clone owners/users. Why not a questionnaire in the next issue to find out what the readership wants?

Regards, Dave Hunt, Harrow, Middx.

Partial Defection

Dear Sir,

I wish Scorpio News well and hope that it can find a way of maintaining interest in 8-bit Z80 CP/M technology in the face of severe and ever increasing competition from IBM look-a-like machines. The development of a conversion path for users when they upgrade their home computing systems so that text and data files can be readily passed from from one type of machine to another.

I am obtaining a 16-bit MS-DOS machine but still intend to run a Nascom 2 based CP/M machine into the foreseeable future simply because I have 5 years of experience and understand the system very well when performing specialised tasks. Other users of older microcomputers are probably in a similar situation. I need the large storage capacity of a modern micro, but not many of its more advanced features.

Yours faithfully, Mr A.A. Bryan, Cambridge.

80-BUS <-> PC

Dear Sir,

Thank you for the first edition of Scorpio News with which I was very impressed. I am looking forward to the complete article on disk formats and if you have in mind an introduction to Pascal (along the lines of the Fortran article), that too would be great.

Having been forced to submit to the dreaded PC to be compatible with other poor unfortunates, I (along with some of your contributors) am of the view that the rest of the world knows not what they have missed. My Nascom and Galaxy systems get much more use simply because they can get through considerably more work in a given time. How about articles where the combination of 80-BUS and PCs can be used to advantage: for example, downloading programs and instructions from a PC to an 80-BUS which looks after various control functions and reports status back to the PC?

Incidentally, I have done something similar between the Galaxy and Nascom in which the Galaxy boots the Nascom through the serial interface, having first loaded it with a program to fill the NASPEN text space of memory with data sent down the RS232. The Nascom then acts as an editable (using ROM based NASPEN) buffer for data "printed" by the Galaxy. In addition, the Galaxy loads the Nascom with an output print routine for selecting the magnets on my very ancient Golfball which, touch wood, has got through some 350 letters without probobobobobobobolems in the last 9 months. [Ed. - sorry, I just had to insert that, you'll see why in the next paragraph!!]

Please keep up the good work and my thanks to the enthusiastic contributors and witty editor that make the journal such a good read.

Yours faithfully, C.J.T. Clarke, West Hampstead, London.

BDOSZ Bug

Dear Sir,

The last (first) issue of this magazine featured an article by Mr C Bowden on improving the user interface to CP/M. In this article Mr Bowden stated that he had found a bug in BDOSZ. As co-author of BDOSZ I feel duty bound to mention the following facts and throw users a lifeline.

Mr Bowden was correct in what he said when referring to the switching between 8 and 16 bit directory entries under BDOSZ. However, this bug was fixed in version 1.3 and I suspect that Mr Bowden has an old copy of BDOSZ. Users of BDOSZ with a version number of less than 1.7 should also note that there were a couple of bugs in the Read and Write Random routines. Admittedly, these were somewhat obscure and haven't been reported to me through the dealer chain but I feel duty bound to mention them. As far as I'm aware, there are few programs that are affected by the last two bugs, but I do know that the Library Utility (LU.COM) can come to grief as it uses random reads and writes. The current version of BDOSZ is 1.8 and features special handling of the IX and IY registers, for users who have BIOSs that use these registers, and a totally re-written error handler to improve compatibility with the DRI BDOS.

Armed with a copy of this magazine I am hopeful that users wishing to upgrade will be able to take their distribution disk to their friendly 80-BUS dealer and obtain version 1.8 for nothing more than a nominal copying fee. Failing this I should be pleased to provide this service. All I would require is the original BDOSZ distribution disk, together with either UK postage stamps to cover postage or a cheque made out to me to cover the cost of postage (you should know how much to enclose since you are sending the disk to me). Don't forget to enclose your name and address and please allow 28 days for delivery (I may need it if there is a rush).

My thanks to the Editor for publishing this letter and, of course, thanks for providing us 80-BUS addicts with an excellent magazine. There are some of us, at least, who would pay almost any price for a quality 80-BUS magazine like this, and there were those of us who would have attempted to start our own (Chris "Dr Dark" Blackmore for instance) if Paul had not done so.

Yours faithfully, M.W.T. Waters, 7 Trenchard Close, Stanmore, Middx. HA7 3SS.

Tail-piece

Dear Sir,

What can I say? I'm glad to see your arrival on the scene.

Yours, Caerwyn Pearce, London EC1.

MAP RAM Green (really red) Light by R. Mohamed

Have you ever wondered if your system had died during, say, a long compilation on your RAM-disk? I did, many times, shortly after configuring my MAP RAM card as 64K of memory and a 192K RAM-disk.

Mechanical disk drives can easily be seen to be active (audibly and/or visually), RAM-disks are by contrast totally silent and in my case there is also no apparent evidence of its being accessed. Very disconcerting!! However inspection of the MAP RAM card's circuit diagram revealed an unexpectedly simple solution.

DESCRIPTION

When the RAM card is used in the 32/64K page mode, a 8 bit latch (IC46 which is addressed via port 0FEh) forms an extended address which is subsequently used to select one bank of 64K of RAM from another. Additional logic allows the software to effectively select either the upper or lower 32K of memory (when in 32K page mode) from any bank of 64K. Three bits from the latch are further decoded on each card by IC47 (LS138), a 3 to 8 line decoder. The decoder converts an input binary number, in the range 0 to 7 (hence 3 bits), into a low voltage at the correspondingly numbered output (or select line). The 8 outputs of the decoder are used to select a particular row or bank of 64K of RAM situated on the card.

Because there are only four banks of 64K on each card, only four of the outputs of the decoder are actually used. The decoders, one on each card, are enabled (or selected) in pairs. Each pair of decoders can select up to 8 banks of RAM, spread over the two boards. Each card in a pair will usually use either the upper or lower four decoder select lines. For example, card number 1's decoder is wired to be selected when RAM banks 0 to 7 are permissible, and similarly for card number 2. But in card number 1's case, outputs 0 to 3 of the decoder are wired to the four banks of RAM located on the card. For card number 2, its 4 to 7 decoder outputs that are wired to the four banks of RAM on the card. Thus, if the software wishes to access bank 1 then the second row on card number 1 will be selected. If bank 6 is to be accessed then the third row on card number 2 will be selected.

After a reset, or when power is applied to the system, all the bits of IC46 are reset to zero. Bit 7 of IC46 selects operation in 64K page mode and as the 3 address bits into the decoder are also zero, bank 0 is selected (on card number 1). This bank of RAM now forms the main addressable memory for the Z80 microprocessor. Whenever another bank is selected, say in 32K page mode, and accessed (either upper or lower 32K) then line 0 of the decoder on card 1 will go high, deselecting bank 0, and one of the other outputs of that or possibly any other decoder will go low to select the wanted bank of RAM. When the latch is reset to all zeros, output 0 of the decoder goes low, selecting bank 0 once more.

It is easy to see that select line 0 on the decoder on card 1 can be used to give an indication of whether bank 0, the main Z80 RAM, is being accessed by the CPU. If the select line is low then bank 0 is selected and the CPU is accessing its main memory. Conversely, when the select line is high the CPU is accessing another bank of RAM. So all that is needed is a simple indicator to reflect the status of select line 0.

One could attach the indicator, a LED, directly to select line 0 but this approach has two disadvantages. Firstly, the 'sense' of the indicator will be wrong, i.e. the LED will be lit when bank 0 is being accessed and off otherwise. Secondly, the LED will add an extra load onto the select line and possibly degrade its performance.

Conveniently, there is a spare inverter on card number 1 which has its output and input pins wired to pins 14 and 8 (resp.) on LB1. Its use will both correct the 'sense' of the indicator and provide a buffer to select line 0. The specification sheets say the inverter (an LS04) can sink a current of 8mA maximum. I choose R to be 330 ohms as a good compromise between light output and sink current (see figure 1).

This is not quite the full story. The select line of bank 0 is treated as a special case and the output line of the decoder is first logically ANDed with the combined results of latch bits 6 & 7 and address line A15. The net result is that the LED indicator, as shown in figure 1, will be active (lit) whenever 32K page mode is selected. I have not found this any problem but purists may wish to use the 'R0' output at pin 13 of LB2 instead of select line 0 as shown in figure 1.

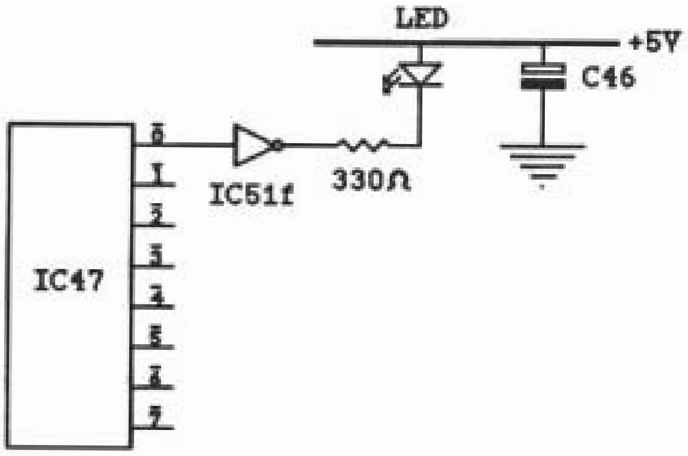


figure 1:circuit diagram

WIRING

The only components required are an LED and a 330 ohm 1/8 or 1/4 Watt resistor. The LED should be small (3mm) type but can be any colour, I just happened to have a red one to hand. Figure 2 shows the connections in both plan and frontal views.

1) Solder the 330 ohm resistor onto pins 9 and 14 on LB2 (component side) of card number 1.

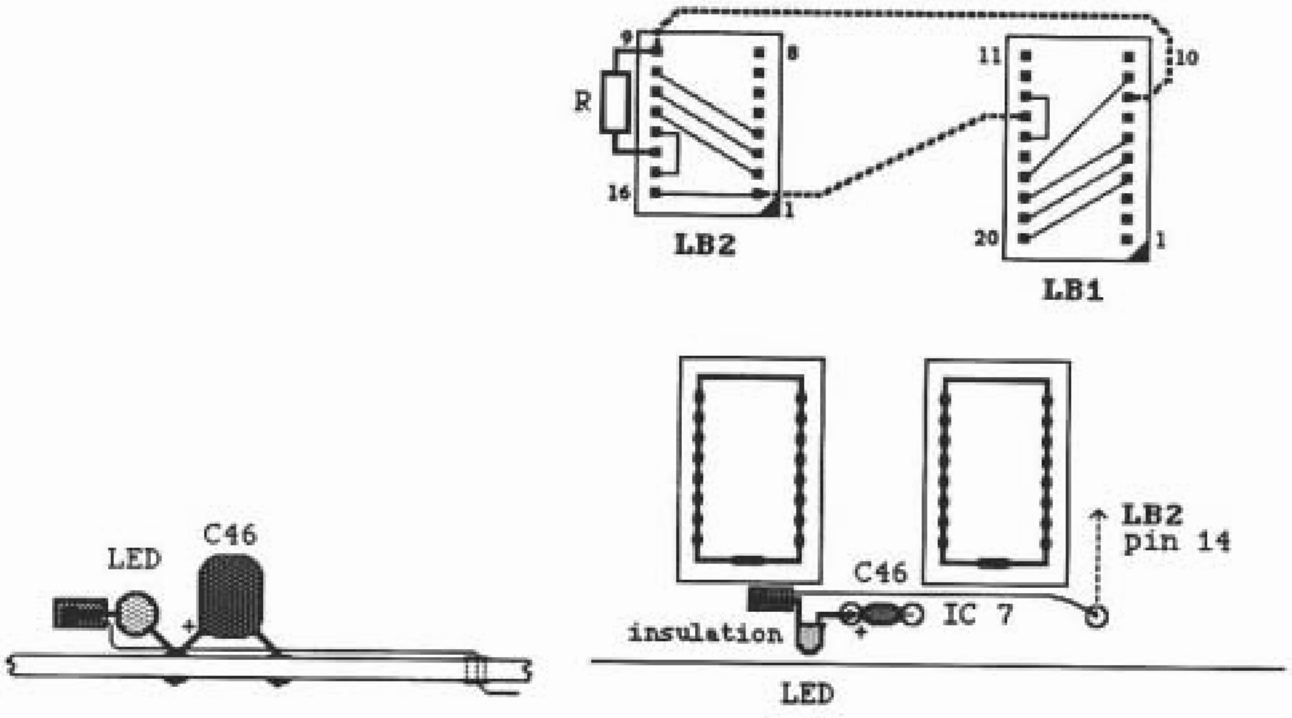
2) Using single cored connection wire (wire wrapping cable is ideal) make the following connections on the underside of the board.

   LB1 pin 8 to LB2 pin 9

   LB1 pin 14 to LB2 pin 1

3) Carefully bend and crop the leads of the LED as shown in figure 2. Note that the LED is correctly orientated, the cathode end (flat on package) is free. Solder the anode to the positive pin of C46 on the component side of the board.

4) Now solder a length of single cored wire to the LED cathode (free end), then poke the wire through the plate through hole adjacent to IC7 as shown in figure 2. Solder the free end of the wire to LB2 pin 14. Since the final link involves the wire passing through the hole, it is advisable to use wire that has a thick insulation layer, i.e. wire wrapping wire. If you plan to use enamelled wire, then I recommend you place a small piece of insulation over the wire and position it inside the hole. This will prevent the edges of the hole piercing the wire's insulation.



View from front

View from above

figure 2:wiring diagram

5) Insulate the free end of the LED with some heat shrink sleeving or a spare piece of PVC tubing.

6) Finally double check the wiring.

This completes the modification. Depending on the type of LED used, you may need to adjust the value of the 330 ohm resistor, but try to keep the current below 8mA.

TESTING

Re-install the card and switch on. The LED should be off. If you have a BIOS that supports a RAM-disk on the MAP RAM card (e.g. SYS) then boot it up and watch the results. If not then use the Monitor ROM to output values to port OFEh.

```
O FE 00        ;LED off
O FE 80        ;LED on
O FE 8x        ;LED on
O FE 00        ;LED off
```

CONCLUSION

The modification will work for both NASCOM 2 and GM811 users. I have successfully used it for the past year with no problems at all. Unfortunately the modification is not suitable for GM813 users who are using the MAP RAM board in its memory mapping mode. Additional logic is needed to detect when the extended address lies outside the first 64K (0 - FFFFh) i.e. when address lines A16-A19 are all zero.

Disk Formats and CP/M Disk routines - Part 2 by M.W.T. Waters

Part 2.

File sizes, disk sizes and directories

Having well and truly taken a disk to pieces in Scorpio News V1 I1, we can at last return to CP/M and the questions asked at the beginning of the first part of this article. (Can anyone remember what they were?)

Minimum file size on disk is determined by the block size (referred to by Digital Research as BLS). CP/M stores files on disk as one or more blocks of information, where each block may contain 8, 16, 32, 64 or 128 CP/M records (128 bytes long) depending upon the block size. Blocks may be 1K, 2K, 4K, 8K or 16K bytes long, chosen (usually) by the computer manufacturer when deciding upon disk size and format. As an example, there are four commonly used Gemini formats available. The SDDS format uses a 1K block size, the DDDS format uses a 2K block size while the QDDS and QDSS formats use a 4K block size. The blocks are sometimes referred to as allocation units since disk space is allocated to files as one or more blocks.

The minimum size of a CP/M file is one block. If you imagine that you wish to save a short utility program of, say, 128 bytes to disk, that program will appear to be 1K long on a system with a 1K block size or 16K long with a 16K block size. On a system with a 16K block size, there would be 16256 bytes of disk space wasted after the file was written to disk. Why then do manufacturers, such as Gemini, use larger block sizes? At first sight, it would appear that a 1K block size is the best choice all round. As we shall see later, the problem of which block size to use isn't quite that simple otherwise all manufacturers would have used 1K blocks (wouldn't they?).

What then, determines the maximum size of a file or disk? To quote Digital Research, CP/M 1.4 allows a maximum disk size and file size of 256Kbytes. The maximum disk size may be extended by the use of double density but the process is cumbersome and wasteful of directory space. For this reason, few computer manufacturers appear to have made double density CP/M 1.4 systems.

CP/M 2.2 allows a maximum disk size and file size of 8Mbytes. The reason for this is that a maximum of 65536 records of 128 bytes each may be written to a disk or file. The maximum file size is determined by the maximum value for the random record field of a CP/M 2.2 file control block which may range from 0 to 65535.

Under CP/M 3 (CP/M Plus), the maximum disk size allowed is 512Mbytes made up of a maximum of 32768 blocks. This of course assumes a 16K block size; smaller block sizes reduce the maximum disk size accordingly so that with a 1K block size, the theoretical maximum disk size is 32Mbytes. In practice, the maximum number of 1K blocks under CP/M 2.2 or CP/M 3 is 256. This is to provide compatibility with CP/M 1.4 disks as we shall see later. The maximum file size allowed by CP/M 3 is 32Mbytes and is determined again by the maximum value for the random record field of the CP/M 3 file control block which may range from 0 to 262143 giving 262144 records of 128 bytes each per file. To accommodate this higher figure, the lower 2 bits of the R2 byte of the FCB are now used to give an 18 bit random record number.

At this point, we need to see how directory information is stored on a disk. For those familiar with DU.COM, this may appear to be revision but don't skip this section too quickly as a couple of surprises may be in store for you.

A CP/M 2.2 directory entry on disk looks something like the example given below. The format used to show the entry is "borrowed" from DU.COM.

```
00544553 5446494C 45434F4D 00000001   *.TESTFILECOM....*
2D000000 00000000 00000000 00000000   *-...............*
```

The characters shown between the asterisks are the ASCII representation of the Hex characters on the left. A full stop is used to show unprintable characters.

The format for the directory entry corresponds almost exactly to that of a CP/M file control block and for good reason. Apart from the first byte, the directory entry is a direct copy of the first 32 bytes of the FCB.

The disk directory information is stored on the data tracks of the disk (as opposed to the system tracks) and occupies the first one or two (usually) blocks on the disk. Each directory entry requires 32 bytes and so it follows that for a 1K block size, each block will hold 32 directory entries. If you should wish to double the number of directory entries available to 64, then 2 blocks will have to be set aside for the directory. Therefore, the number of directory entries available and consequently the maximum number of files that may be stored on a disk is a trade off against the size of the disk and the number of blocks that can be spared for directory information. Gemini opted for 128 directory entries per disk with a 4K block size, so only one block is lost to the directory leaving 196 blocks for the files themselves.

Looking at the entry for TESTFILE.COM, the first byte tells us that the file is stored in user area 0. The user number is held in the lower four bits of the byte. The upper four bits are reserved by Digital Research and although they are unused in CP/M versions up to 2.2, CP/M 3 does use them so, if anyone was thinking of having 256 user areas, think again if you wish to remain compatible with future releases of CP/M.

A value of 0E5H in this position indicates to the BDOS that the file has been erased. This makes utilities such as UNERASE possible as, to restore a file, all that is necessary is to change this byte to a value between 0 and 15 depending upon which user area is required to hold the file. Unfortunately, this may not always work as if a file has been written to the disk since erasure, some of the data blocks used in the erased file may have been reallocated to the new file. For users of CP/M 1.4, it is not possible to unerase a file as the ERA command completely wipes the directory entries for the erased file by filling them with 0E5H bytes.

CP/M 3 uses different values of the first byte of the directory entry to identify disk labels, password information and date/time stamps, all of which occupy directory space and consequently reduce the number of files that may be stored on the disk. The disk label occupies one directory entry and may be identified by a 20H value for the first byte. Date/Time stamps and passwords are invoked by formatting the directory using the utility INITDIR.COM which reserves one directory entry for every three file entries in the directory; reserved entries being identified by the value 21H in their first byte. You will see that if Date/Time stamping is invoked under CP/M 3, the maximum number of files that may be stored on the disk is immediately reduced by one quarter.

Next in the directory entry comes the 11 bytes for the file name and type. The last four bytes in the top row give file size information and will be looked at later. Lastly, the 16 bytes in the bottom row contain the block numbers that have been allocated to the file. CP/M gets clever here and the block numbers are stored differently depending on whether the disk has more than 255 or less than 256 blocks available. If there are less than 256 blocks on the disk, CP/M stores the block numbers as 8 bit values and can consequently hold information on 16 blocks. If, however, the disk has 256 or more blocks, the numbers are stored as 16 bit values (stored in Low-High order) with the result that only 8 blocks may be allocated per directory entry. In the example given we can see that only one block has been allocated to the file although it isn't apparent from the entry whether 8 or 16 bit numbers are being used. The decision as to whether 8 or 16 bit numbers are used is made by the CP/M BDOS and is totally transparent to the user but the manufacturers of the Tatung Einstein force their implementation of CP/M to use 16 bit values, even though their disks have less than 256 blocks. This, presumably, was an attempt to make their disks unreadable on other systems (Didn't fool Dave Hunt though).

A quick tap on the calculator keyboard will have told you that, assuming say, a 2K block size and 8 bit directory entries will allow a maximum of 32K to be referenced by a directory entry. This implies (correctly) that more than one directory entry will be required to hold the block numbers for larger files. CP/M automatically takes care of this problem for us. When writing to disk, if a directory entry is filled, CP/M closes it (by writing it to disk) and opens a new one. It should be noted that each time CP/M has read or written all of the blocks allocated by a single directory entry, it has to read the next entry from, or write the next entry to disk. Logically, it would appear that to improve the performance of the disk system (i.e. speed up access), the block size should be as large as possible so that CP/M would need to access the directory track of the disk less often (since extra time is needed to move the disk drive head from where it is now to the directory track and back again). It would also appear that with a 16K block size and 8 bit directory entries, CP/M would only have to access the directory track once for every 256K of a file that it is reading or writing. However, maybe due to a hangover from CP/M 1.4 (which always held 16Ks worth of block information per directory entry), CP/M 2.2 and CP/M Plus STILL access the directory track every 16K.

If we return briefly to the number of blocks reserved for directory entries, we can see that we must have enough to allow access to the available disk space. With the Gemini QDDS format, each directory entry can refer to 64K since 8 bit entries are used with a 4K block size. The minimum number of directory entries required to access the 196 data blocks on disk will be 13. However, if we were to take the average sized file to be about 8K then to fill a disk with them we shall need at least 98 directory entries. In practice, Gemini chose 128 directory entries which, while entirely adequate for most purposes, try filling a disk with 4K files or programs. When all 128 directory entries have been used, there is still 272K of free space remaining. To be fair, this is an extreme example and will occur very rarely (unless you use dbaseII, in which case it will probably occur daily as the average dbaseII .CMD file seems to fit in a 4K block).

The choices, then, facing a manufacturer when deciding upon block size are a compromise of disk size, number of directory entries and speed of access. If Gemini had used a smaller block size for their QDDS format (2K for example), CP/M 2.2 (or CP/M 3) would have used 16 bit entries in the directory. Consequently, four times as many directory entries would have been required. Why four? Each directory entry would refer to 2K blocks - hence twice as many directory entries would be required to hold the block numbers for the same sized file BUT each block number would now occupy twice as much room in the directory entry so double the number of directory entries again. Similarly, the BDOS would have to access the directory track four times as often with a corresponding reduction in disk performance.

A further complication that prevents the use of a 1K block size with larger disks is due to the way CP/M itself works. Let's have a quick history lesson.

CP/M 1.4 uses single density disks (although some micro manufacturers have made it work with double density) with a fixed 1K block size and 8 bit directory entries. Each directory entry, therefore, refers to 16K (which we all recognise as an extent). CP/M 1.4 allows a file/disk to contain a maximum of 16 extents numbered 0-15, hence a maximum file/disk size of 256K (16 x 16K). A hangover from CP/M 1.4 is that each directory entry MUST be able to control at least one extent. If we are using 16 bit directory entries, each block must be at least 2K long since the directory can only hold 8 block numbers. This applies equally to CP/M 2.2 and CP/M 3.

We have already seen that the directory occupies at least one block in the data area of the disk. Under CP/M, the directory blocks are always the first ones in the data area and it follows that block 0 will always contain directory information. This being the case, at no time will CP/M ever need to allocate block 0 to a file. For these reasons, zeros in the directory entry may be used to signify that no further blocks have been allocated to a file.

This may have been stating the obvious as I had already said, when referring to the example directory entry, that only one block had been allocated to the file. However, the concept of using zeros to show unallocated blocks is fundamental to the way directory entries relating to random files are interpreted. I shall return to random files shortly.

Let's finish looking at the example directory entry by examining the last four bytes of the top row. The first of the four bytes is the extent byte. This shows the highest extent number accessed by the directory entry. In our example, the highest extent accessed is 0. Had the directory entry been full on a QDDS disk, the extent number would have been 3 since this directory entry would control extents 0-3. A subsequent entry would have controlled extents 4-7 and would have 7 in the extent byte if the entry was full. Under CP/M 1.4, as already stated, the highest extent number allowed is 15 (16 extents numbered 0-15) giving a maximum file size of 256K. Under CP/M 2.2 and CP/M 3 the maximum value allowed for the extent byte is 31 (32 extents) giving a maximum file size of 512K.....Hang on a bit! Digital Research told us that the maximum file sizes for CP/M 2.2 and CP/M 3 are 8Mbytes and 32Mbytes respectively.

Under CP/M 2.2 or CP/M 3, when a file exceeds 512K, the S2 byte is used to hold the multiples of 512K and the extent byte starts numbering again from zero. This use of the S2 byte is not at all well documented. In fact I have searched high and low through the Digital Research manuals for references to the S2 byte and its use in this context with little joy. The result of this lack of information has led to the production of programs, both commercial (Wordstar) and Public Domain (D.COM, SD.COM and DU.COM) that cannot handle files larger than 512K.

The maximum value for the S2 byte under CP/M 2.2 is 15 (16 permutations numbered from 0 to 15) and 16 x 512K is 8Mbytes made up of 65536 records of 128 bytes each. Under CP/M 3 its maximum value is 63 (64 permutations) and of course 64 x 512k is 32Mbytes made up of 262144 records of 128 bytes each. Under CP/M 1.4 the S2 byte is unused in this context. The S1 byte is unused by all versions of CP/M up to 2.2 and is probably not used by CP/M 3, at least I haven't yet found where it uses it, if in fact it does.

The last byte of the four is the record count byte and it shows how many records are stored in the last extent written (i.e. the one indicated by the extent byte). If this byte has the value 80H then the extent is full. In our example, one record has been written to extent 0. Had the extent number been 3, for example, then extents 0-2 would be assumed to be full. However, this isn't necessarily true when referring to random files as we shall see now.

With the block size chosen and having established that we are using 8 bit directory entries, as we have already seen, each directory entry controls up to 64K of a file. In fact the directory entry is more than a place to store block numbers, it is literally a map of the file.

Directory entries for random files

We can split the 64K controlled by the entry up into 512 CP/M records of 128 bytes with these records being numbered from 0 to 511. Each of the sixteen 4K block positions will contain 32 CP/M records and therefore the first block will contain records 0 to 31, the second 32 to 63 and so on. If only record number 32 is written to file TEST.RND, its directory entry will look like the one given below with only the second block position containing a block number.

```
00544553 54202020 20524E44 00000021   *.TEST    RND....*
002D0000 00000000 00000000 00000000   *.-..............*
```

If, on the other hand, only the 512th record had been written (record number 511), only the last block position would contain a block number.

The RC, EXT and S2 bytes will be set up as if the file had been written sequentially, i.e. writing the last record to an extent will cause the RC byte to read 80H. The only way that CP/M knows whether a block has been allocated or not is by whether there is a zero in the block position of the directory entry. What CP/M doesn't know is which records have been written to the block; this is left for the programmer to determine.

One way of doing this is to use the write random with zero fill function when writing to a previously unallocated block. To establish, from within a program, whether or not a block has been allocated to a file is to read the desired record using the read random function. If an error code is returned, you are trying to read a block that has not yet been allocated, i.e. CP/M found a zero in the relevant position in the directory entry. If this happens, you can write the record with zero fill. If no error code is returned, a block has already been allocated which could contain the wanted record. If you previously used the zero fill function when writing a record to this block, examination of the contents of the required record will reveal zeros if nothing has been written to it or data if something has been written.

A point worth noting is that the CP/M DIR command will only show a file if a directory entry exists for extent 0. For this reason, writing one record to record position 65535, for example, will result in two directory entries being created; an empty one covering extent 0 and one covering the 128th record of extent 512.

Another observation about random files is the way that PIP.COM handles them (or more accurately, it doesn't). PIP.COM reads and writes files sequentially and will therefore get an end of file code from the BDOS when it encounters an unallocated block in a file. Normally, this would be adequate as most files indeed end this way. If faced with the example of a file that contains only record 65535, PIP.COM will only copy the empty directory entry for extent 0.

BIOS data structures

Some questions that still need to be answered are: how does CP/M know how many blocks the disk contains, what size are they, how many of them are reserved for the directory and how does it keep track of which ones are free for use? The short answer is that the BIOS contains this information but to answer these questions in detail, some information about BIOS data structures is required.

The disk routines in the BIOS contain data areas and buffer space that will be used by the BDOS when it needs to have information that is system dependent. These areas are known as the Disk Parameter Headers (DPH's), Disk Parameter Blocks (DPB's), Checksum Vectors and Allocation Vectors. We'll look at these in the next issue.

MAP 80 Multi-Purpose Interface (MPI) Review by P.D. Coker

The user of 80-BUS compatible products has been quite well served for floppy disk controllers - initially by the Gemini GM809, then the GM829 with SASI support for a Winchester, closely followed by the even more supportive GM849 - Map 80's VFC was also capable of controlling 5.25" drives and Nascom/Lucas also produced an FDC card. So why have Map produced yet another card?

The MPI is more than just an FDC, catering for the usual range of disk sizes, with support for a Winchester drive or two. By changing a few links and the FDC chip one can convert to Nascom FDC compatibility rather than the more usual Gemini/Map "standard". It also has a CTC and two channels of SIO, one of which is a standard RS232 and the other conforms to the proposed standard for RS485 (high speed multi-drop interface). The CTC can be used to generate software selectable baud rates or, if the user wishes, on-board crystals can be used to generate two frequencies on the MPI board which are independent of the system clock. It uses 16 ports and, in my experience, can substitute for either of the two earlier Gemini FDCs - I have no experience of the most recent Gemini FDC.

The MPI is supplied either ready built, or as a kit. A number of options are available:

FDC and Winchester controller
(2797 or 2793 FDC Controller, depending upon system)
CTC
SIO

The board or kit is supplied with the appropriate combination of options as ordered. The PCB is the standard 8" square, with the usual Map 80 blue resist and high quality. Kit assembly is quite straightforward and the instructions are reasonably clear - there is a separate assembly manual which I didn't receive but a phone call to Map soon solved the problem.

There are 22 links on the board, some of which are not needed for standard configurations and a 20 way link header to be wired up if one wants the SIO and/or CTC facilities. Details of the link functions and also the changes necessary to accommodate the 2793 controller, if the board is to be used as a Lucas/Nascom compatible FDC, are provided.

A slight disadvantage is that the instructions are not very clear on which links are needed for particular uses - my boards have links 2 (a-b),3 ,5 (a-b),6 (a-m), 13 (a-m), 14 (a- 13n) 15 (a-c), 16 (COM-E), 17 (b-c),18 (c-6, d-5), and 21 (a-Clk). Map will supply details of modifications required to read odd disk formats with bad side 2 flags, which might be necessary if you wish to use their Format transfer programs. No details are provided of the crystal frequencies or types required for on-board frequency generation.

The SIO and CTC seem to be standard implementations using the Z80A CTC and SIO chips. The circuitry around the 2797 seems to be almost identical to that used on the VFC and the RS485 interface is provided by differential line driver and receiver (75174/75175), in contrast to the RS232 interface which uses the conventional 75188 and 75189 types.

My main use for the MPI is as a Floppy/Winchester controller and both the ready-built and kit versions have performed without trouble with a wide range of other boards from Nascom, Map and Gemini. In particular, the combination of the MPI with TEAC compatible drives and the Map version of CP/M Plus provides fast, quiet disk access - I hate noisy computers!

As I sometimes need to copy disks with peculiar formats, the small modifications to the board for this purpose are easily made. I don't need to use either the SIO or CTC functions (although they are implemented on one of the boards) - I use the Serial o/p on the Map or Gemini CPU board for this.

The fully-assembled version with FDC, CTC and SIO options is possibly a bit pricey at £195 + VAT but money can be saved by building the kit yourself - not a long job - possibly a couple of evenings. Most users need only the FDC/SASI version and this is quite a bit cheaper. I haven't seen an up-to-date price list from Gemini or Map but suspect that the ready built FDC/SASI is about the same price as the GM829 or GM849 (£145, in the last price list I saw from Gemini). It would be worth contacting Map to see what their current prices are like.

I can thoroughly recommend this product for anyone contemplating upgrading an 80-BUS system even if they already have the Gemini 809 FDC, since it will enable them to take full advantage of higher access speeds and quieter operation if the newer TEAC-compatible drives are used. [Ed. - alternatively readers may like to make the GM809 modifications shown elsewhere in this issue.]

GM809 Fast Stepper by R. Mohamed

This article describes a simple hardware modification for the Gemini GM809 floppy disk controller board to allow full speed stepping of modern 5 1/4" disk drives.

About two years ago I upgraded my Nascom 2 by replacing the existing drives with a couple of TEAC FD-55s. They provided my system with a very worthwhile boost in performance and disk capacity (from 80K to 800K). Unfortunately they sounded like a couple of machine guns every time the disk heads moved. Altering the stepping rate from 20 to 6 mSec (still twice the minimum for the TEACs) reduced the noise to a slightly less objectionable burrrr.

DESCRIPTION

At about this time, I learned that the newer GM829 FDC board could support the faster 3 mSec stepping rate of modern floppy disk drives by doubling the clock frequency into the FDC controller chip. The GM829 board controls the FDC clock frequency, amongst other things, by using the upper most bit (bit 5) of a six bit resetting latch. Examining the GM809 circuit diagram revealed that this same bit is present and uncommitted on the board. LK 1 on the GM809 provides a selection of 3 clock frequencies, one of which is selected by a soldered link. Thus all that is needed is to add a simple (digital) 2 way switch in place of the soldered link.

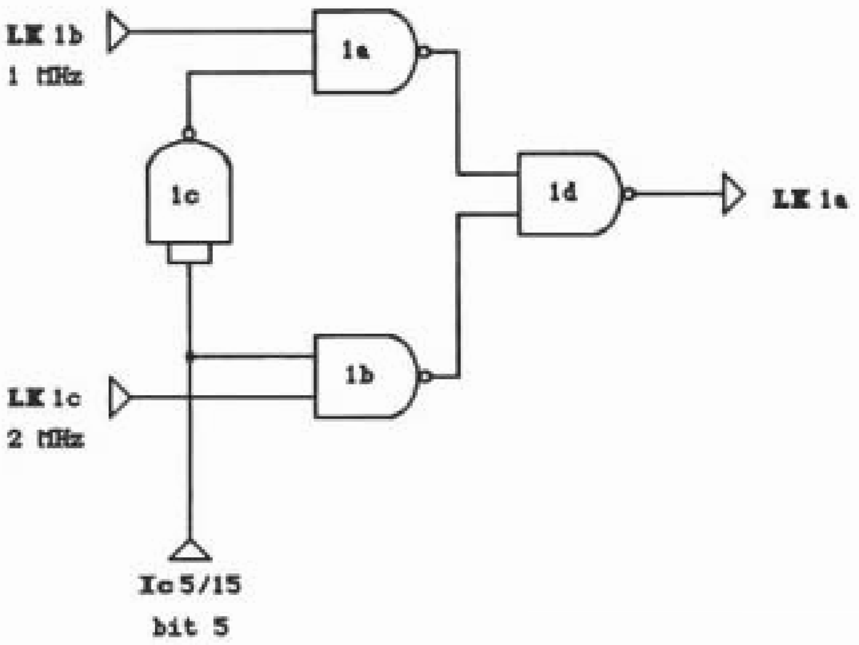


figure 1:74LS00 digital switch

The simple circuit in figure 1 can switch through one of 2 clocks under control of a single input line. The two clock frequencies (1 & 2 MHz) are each connected to one input of a pair of NAND gates which are functioning as digital switches. The other input to the gates are connected to the clock select line. Logic 0 (low) on the clock select line opens the switch and logic 1 (high) will close the switch letting the clock pulses through to the output of the relevant NAND gate. An inverter on the select line (made from another NAND gate) ensures that only one switch is open and the other closed at all times. A fourth NAND gate combines the outputs from the two digital switches into a single output representing the software selected clock frequency. After a reset all six bits of the latch (IC5) are set to logic 0 (low). By suitably arranging the two clocks, the standard 1MHz clock is selected on reset. This provision should leave all well behaved disk routines unaffected by the modification.

WIRING

The wiring is quite straight forward and should present no major problems to anyone. It follows my usual practice of NOT cutting any pcb tracks. The instructions assume that the board is component side up and that the 'BUS' end is nearest to you. Readers who decide to make this modification do so at their own risk, I must disclaim any responsibility for any loss or damage. Having said this, the modification really is very easy to do and I encourage you to try it.

1) First remove the permanent wire link on LK1, noting the original position which in my case was A-B. This information will be used later to route through the original 1MHz clock to the correct gate.

2) With the aid of figure 2, locate and mark out the positions for the 14 pins of the 74LS00 chip. You can use a small piece of 0.1" vero board as a template.

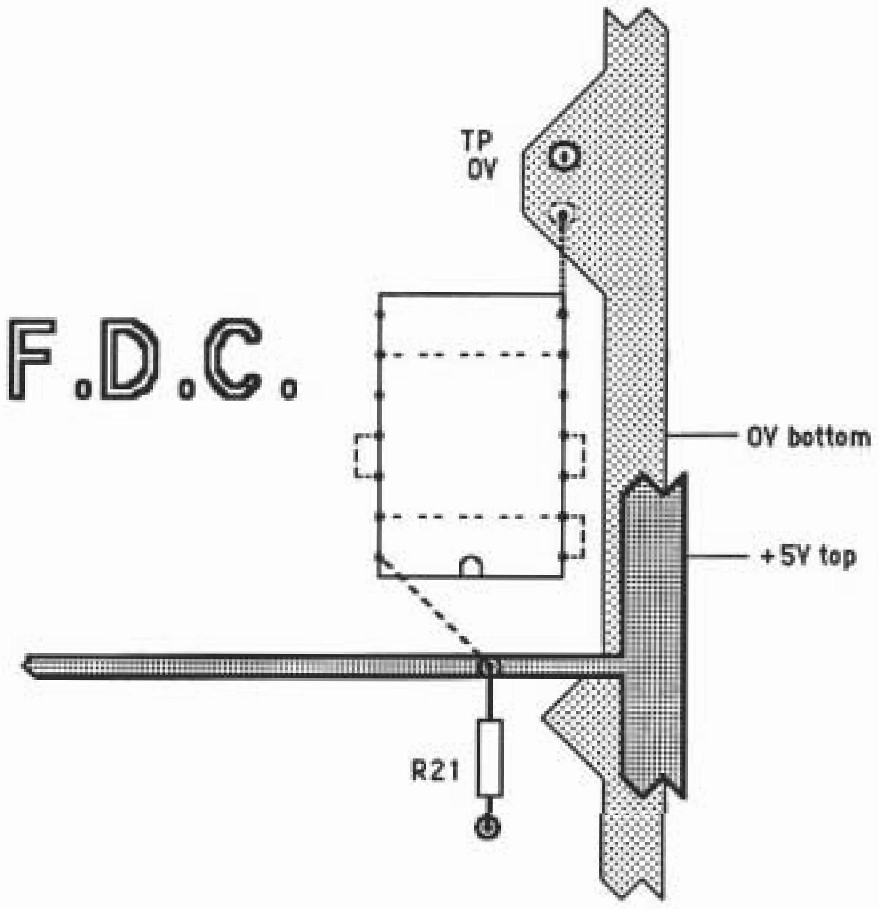


figure 2: suggested chip placement

3) Double check that no tracks are underneath before drilling out the holes with a 1mm drill. Depending on the exact position of the holes, some may straddle the etched lettering on the underside of the board. The affected lettering should be removed. Place a hot soldering iron on the letters for 5-10 seconds and then firmly push the foil off with the edge of the iron. Alternatively, if you don't want to drill holes in the pcb, you could splay out the legs of the 74LS00 and secured it to the underside of the board with glue or double sided tape.

4) Insert the 74LS00, see figure 2, notch or dot towards you. Turn the board over and solder pin 7 to 0V and pin 14 to the solder pad of R21 with some thickish tinned copper wire. The two wires have now mechanically secured the chip to the board.

5) Interconnect the pins of the 74LS00 as shown in figure 3, using some thin connecting wire, single core preferably.

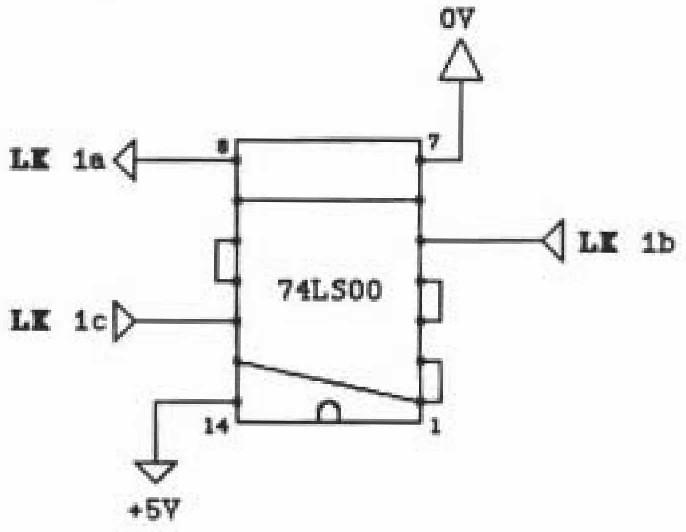


figure 3:Wiring diagram, viewed from component side.

6) Now connect the four additional links as follows ..

LK1/b	to 74LS00 pin 5	(your original 1MHz clock)
LK1/c	" " " 12	(your 2MHz clock)
LK1/a	" " " 8	(output clock)
IC5/15	" " " 1 & 2	(clock select line)
(i.e. IC 5 pin 15)		

7) Finally double check all the wiring for any mistakes or shorts.

TESTING

Re-install the FDC card into your system and reconnect the drive cabling. Power up and boot up in your normal disk operating system. Everything should work as before, if not check the voltage level on IC5 pin 15 which should be low. If this is ok then check that the two clock lines are not swapped. I had no problems, everything worked first time when I booted up the disk system.

THE SOFTWARE

Modifying existing software should be quite straight forward. Changes only need to be made whenever commands that (potentially) move the disk drive's head are issued to the WD1797 disk controller chip. I include details of 2 versions of software, one where primitive disk routines communicate directly with the WD1797 FDC chip and secondly where the routines communicate indirectly via a single call (i.e. SYS users).

Firstly, for routines that send commands to FDC directly, listing 1 shows the suggested alterations (in lower case) to the existing code. The existing code is presented in a hypothetical form, and you may have to alter it slightly in light of your own circumstances. The 'FDCBUSY' subroutine call may in fact be a jump to some common code shared by 'HOME:', 'SEEK:' and maybe other routines as well. In this case just append the source of the 'stepslow:' at the end for your 'FDCBUSY' routine.

Users of the SYS bios overlays implementing the standard Gemini disk routines (version 14 and later) need do nothing as the GM829 code will now become effective. For those running SYS under the mixed disk formats option (rdisk is true), listing 2 shows how multiple stepping rates can be supported in the SYS bios. The code reflects my current situation, that is, with drives A,B and C stepping at 3mSec and drive D stepping at 30mSec. If you require 3 (or more !!) different rates then just duplicate the code for 'doslow' using a different rate mask for each drive. After testing to see if that drive is selected ('bit ??,a') jump to its particular piece of code and then on to 'dotyp'.

LISTING 1

```
; First some equates

PCMD    equ     0e0h            ; FDC command register
pstat   equ     0e4h            ; Latch output port
clkmsk  equ     020h            ; mask for bit 5 only

; drivep : ram copy of IC5 latch outputs

home:   ...
        call    stepfast        ; double the clock frequency
        LD      A,HOMECMD       ; home the head
        OUT     (PCMD),A        ; do it
        CALL    FDCBUSY         ; wait till done
        call    stepslow        ; normal 1Mhz clock
        ...

seek:
        ...
        call    stepfast        ; double the clock frequency
        LD      A,SEEKCMD       ; move the head
        OUT     (PCMD),A        ; now do it
        CALL    FDCBUSY         ; wait till done
        call    stepslow        ; normal 1Mhz clock
        ...

stepfast:
        ld      a,(drivep)      ; get latch status
        or      clkmsk          ; set bit 5 high
        out     (pstat),a       ; do it
        ret

stepslow:
        ld      a,(drivep)      ; get latch status
        out     (pstat),a       ; reset clock to normal
        ret
```

LISTING 2

```
;SYSB6.MAC - ammendments for multiple stepping rates

; Drive characteristics
hldel   equ 50              ; Teac head load delay
rate    equ 0               ; Step rate 3ms for TEACs ***
slowrt  equ 3               ; Step rate 30mS for Pertec *
;*******************************
;       Disk Drivers           *
;*******************************
; drives stepped @ independant rates
;       Do a Type I command
dotypi: bit 7,a             ;? seek or home
        jr z,dofast         ;yes, head moves
;       process any other commands unaltered
dotyp:  out (cmdreg),a      ;Issue command
        ld a,10             ;Short delay
doid:   dec a
        jr nz,doid
doiw:   in a,(statrg)       ;Read status
        bit 0,a             ;Busy?
        jr nz,doiw          ;Yes,loop
        ret                 ;Else done
dofast: push hl             ;save contents
        ld l,a              ;store FDC command in L
        ld a,(drivea)       ;check which drive selected
        r
        ld h,a              ;and store for later use
        bit 3,a             ;? drive D selected
        jr nz,doslow        ;yes, slow step for Pertec
        or 20h              ;double clock to 2MHz
        out (drivep),a      ;do it now
        ld  a,l             ;get FDC command
        call dotyp          ;and issue it
        r
        ld l,a              ;store error status in L
        ld a,h              ;original latch contents
        out (drivep),a      ;and reset clock to 1MHz
        ld a,l              ;return status register contents
        pop hl              ;restore old contents
        ret                 ;return finally
doslow: ld  a,l             ;restore FDC command
        or  slowrt          ;add slow step rate
        pop  hl             ;restore old contents
        jr dotyp            ;issue modified command

          *******************
```

NEWBURN

NEWBURN ELECTRONICS

58 MANSE ROAD, BALLYCARRY
Co. ANTRIM BT38 9LF
Tel: 09603-78330

NE871 80-BUS 32 OPTO-INPUT BOARD. £350

Provides 32 opto isolated inputs to the 80-BUS. 5V to 125V sensitivity. On-board led indication for all 32 inputs.

NE874 80-BUS 16 CHANNEL 8 BIT A/D £275

May be configured as 16 balanced or 8 unbalanced + 8 balanced inputs. Input sensitivity between 100mV and 10V.

NE875 80-BUS 16 CHANNEL 8 BIT D/A £330

Provides 16 analogue voltage outputs 0-10V. Current supply on each output up to 40 mA. Short-circuit protection.

NE876 80-BUS 64 CHANNEL DIGITAL INPUT £195

Provides 64 digital inputs to 80-BUS. Inputs are protected up to +/-50V. Any input voltage between -50 and +50V may be used to switch inputs, threshold is +2V.

NE877 80-BUS 64 CHANNEL DIGITAL OUTPUT £215

Provides 64 digital outputs from the 80-BUS. Each output is short-circuit protected and capable of output of 50V at 500mA. Ideal for driving relays & lamps etc.

Industrial klippon block can be supplied for direct plant connections for all the above boards.

NE840 14 SLOT BACKPLANE £67

Correctly terminated for 80-BUS. May be reduced in size. Supplied complete, less connectors.

GM833/2M 2 MEGABYTE UPGRADE KIT £290

Expands a standard GM833 (512K) to 2 Megabytes. No changes are required to the standard Gemini bios to support this board. The board may be upgraded by us for £350.

CA802/1M 1 MEGABYTE COSTGOLD UPGRADE KIT £250

Expands a standard Costgold (256K or 128K) board (CA856) to 1 Megabyte capacity. No changes are required for MS-Dos or CP/M86. The board may be upgraded by us at a total cost of £300.

Please add 15% Vat and £4 p/p per order

NEWBURN

FML

Finlay Microfilm Limited

Finlay

Computer Aided Retrieval

Contact DAVE HUNT
for Gemini, Olivetti
and IBM* Clone
Components & Accessories

18 Woodside Road
Amersham
Bucks. HP6 6PA
Tel: 02403 22126-7

*IBM is the Trademark of IBM Corp.

ARCTIC

ARCTIC COMPUTERS LTD.

6 Church Street
Wetherby
West Yorkshire
LS22 4LP
Tel: (0937) 61644

Pluto Computer Graphics specialist sales and bureau services.

Gemini 80-BUS cards and systems.

Demonstrations by appointment.

KENILWORTH

Computers Limited

19 Talisman Square, Kenilworth, Warwickshire, CV8 1JB
Telephone (0926) 512348/512127 Telecom Gold 84:DDS132

INDUSTRIAL CONTROL SYSTEMS

You will save TIME and MONEY if you talk to us about Process Control, Data Collection, Production Monitoring.

YOU need expertise in Engineering, Computer Hardware and Control Software.

WE HAVE IT – with a little help from our friends GEMINI! Contact us NOW.

Scorpio Systems

P.O. Box 286, Aylesbury, Bucks. HP22 6PU

Publishers of Scorpio News.
Disk copying and file transfer service.
Laser printing of your disk files.
Technical Documentation our speciality.

Private Advertisements

FOR SALE - GM870 MODEM board (boxed) £140, two MAP 80 256K boards £185 each, and a Nascom 2 with a RAM board and power supply etc. All boards are in good working condition with all documentation etc. Open to all reasonable offers. Telephone St. Albans (0727) 73057.

Gemini Galaxy 2 system for sale. Excellent condition. SVC graphics, 512K RAM-DISK, 2 x 800K disks and amber monitor. Plus much software including Pro-Pascal, Aztec C, BDS C, COMAL and Devpac. Any reasonable offers for whole or parts? Phone Peter Kenny 0904 430000 X5538 daytime only.

FOR SALE - Due to upgrade we have the following for sale. Galaxy 1 with keyboard and screen. Also Galaxy 4 Fileserver 10MB with keyboard and screen. Will throw in accounts package, Wordstar Multiplan plus Mailmerge F.O.C. Open to all reasonable offers. Ring 0522-38525 (DAY) 0522-751769 (EVENINGS)

WANTED - to purchase, or borrow and return. Manuals and circuit diagrams for the following: Nascom Series 1 RAM board (RAM A), Nascom Buffer Board, Gemini GM809 Floppy Disk Controller. Circuit diagram for GM813 CPU. Please contact: Dr P. Coker, 23 Darwin Close, Orpington, Kent, BR6 7EP. Tel: 0689 58510.

EV COMPUTING LTD 700 BURNAGE LANE BURNAGE MANCHESTER M19 1NA
(Registered office) Telephone: (061) 431 4866

EV Computing Ltd. Forthcoming Products for 1987

-- PIO Interface boards --
-- For Industrial control applications --

All the below boards connect to either the Gemini or Nascom 26 way PIO ports. The software is supplied on disk to be re-configurable depending on PIO port addresses. A lead to plug into the 26 way PIO connector is provided with all boards.

If you require more PIO's then use a Nascom I/O board or a Gemini GM816. Both give 3 more PIO's.

LCD Interface panel £ 35.00

Supports Alphanumeric LCD modules from 16x1 to 40x2
Software to run in ROM or under CP/M supports most standard terminal control codes.
Can replace CP/M console output device
On board bleeper and viewing angle control

LCDs can be supplied from £ 35.00

3 channel A/D convertor £ 80.00

Three multiplexed 8 bit channels with optional sample & hold
Software supplied will run under CP/M or in ROM
On board gain & offset trimmers for each channel
On board -5 Volt convertor allowing AC operation

Keyboard Interface panel £ 25.00

Supports matrix type keyboards from 3x4 to 8x8 key locations
Software to run in ROM or under CP/M
Can replace CP/M console input device
Easy User configurable keyboard type & layout
Shift/Control/Function key features
Several standard configuration files supplied

Keypads available from £ 5.00 for 6x4 matrix

CMOS RAM file £ 75.00/64K, £195.00/256K

64K to 256K Battery backed CMOS RAM
Software to emulate disk drive under CP/M as well as low level driving software.
Designed to solve industrial storage needs, systems available to store upto 100 Mbyte if required.

Note: All software supplied is written under CP/M and uses the M80 assembler & L80 link loader which must be provided by the user.

All prices exclude VAT, Please specify CP/M disk format required.