80-BUS NEWS

NOVEMBER-DECEMBER 1983 VOL.2 ISSUE 6

- NASPAS REVIEW
- ACCESSING PRESTEL
- NUMBER CRUNCHER BENCHMARKS

£1.50

The Magazine for NASCOM & GEMINI USERS

November – December 1983 **80-BUS NEWS** Volume 2. Issue 6.

CONTENTS

SUBSCRIPTIONS

Annual Rates (6 issues)				
	UK	£9	Rest of World Surface	£12
	Europe	£12	Rest of World Air Mail	£20

Subscriptions to 'Subscriptions' at the address below.

EDITORIAL

Editor : Paul Greenhalgh Associate Editor : David Hunt

Material for consideration to 'The Editor' at the address below.

ADVERTISING

Rates on application to 'The Advertising Manager' at the address below.

ADDRESS: 80-BUS News,
Interface Data Ltd.,
Woodside Road,
Amersham, Bucks, HP6 5EW.

EDITORIAL

An apology

Sorry; a little late again. My fault. Too much working and socialising and not enough editing. Anyway, this is the latest effort and I trust that you enjoy reading it and find a number of useful items in it.

Future issues

I have recently received a number of conflicting letters concerning the type of article that we publish. Some say "I now find very little of interest in the magazine" whilst others say "80-BUS News is invaluable - there is so much useful information in every issue that is just not available elsewhere" - both of these are actual quotes, both are probably right, but I'm pleased to say that the latter is the more frequent. It obviously depends on your individual interests, and there is no doubt that there is a fairly wide cross-section of readership. Really all I can do is repeat, for newcomers, what has been said before. All articles published in 80-BUS News are non-commissioned items received from readers. From these potential articles I select a cross-section of items to cover as wide a range of topics that I believe will be of reasonably wide interest. In the main this seems to be very successful, but starting with the next issue I will also start to include the occasional article that, in my own opinion, is aimed at a much more specialised area than normal. I await any feedback with interest. Incidentally, this means that if you have submitted an article that has not yet materialised, you are now in with a chance!

Money

Owing to a little internal problem there are a couple of contributors to this esteemed publication who have not been remunerated for their efforts, and with whom we have lost contact. If this is you then please drop us a line stating the article for which payment is required. We will then try to oblige!

And finally

A belated Happy New Year to all our readers!

CLASSIFIED ADS

Nascom 1, 3 amp PSU, Cottis Blandford cassette interface, NAS-GRA graphics ROM, Space Invaders/Defender graphics ROM, NAS A/N in EPROM, NAS-SYS 3, Assemblers, Disassemblers, Debug, BASIC, 9 slot motherboard (4 fitted), also Nascom bufferboard and Micropower 64K RAM/buffer/programmable graphics (RAM and both buffers faulty but diagnosed). All documentation, many other goodies. Purchase of N2 forces sale. £130. Phone Canterbury (0227) 710720.

Nascom 1, 48K, ZEAP, Nas-Sys 3, Hobbit micro-cassette drive and tapes, EPROM programmer, I/O board. Make me an offer. Phone Paul on Macclesfield (0625) 72988 evenings.

Nascom 1 with 32K RAM, B-BUG monitor in ROM. High speed cassette interface, teletype interface, floppy disk control card and drive unit. All built into a Harris VDU terminal, plus Teletype printer. Software - BASIC & assembler on disk and all documentation. £400 the lot, or prepared to split up. Tel. Mr Taylor, Boldon 362165.

Nascom 2 with 48K RAM, Nas-Sys 3 Monitor, BASIC Toolkit, Graphics EPROM, assembled in case with manuals and 9 spare 416 -2 memories. Offers around £100. D. Parkinson, 104 Sompting Road, Lancing, West Sussex. (0903) 765714.

Nascom 2 with 48K RAM B board in Vero rack. Manuals, games, 80-BUS News and Tandy monitor. £130 ono. AVT Monitor, as new. £65. 08675 3750 (Oxford).

Nascom 2, 48K, spare RAM A board, Holmes colour graphics PCB + ROMs (unpopulated), all documentation, a few games. £220 or offers. Tel. Eastleigh (Hants) 617214.

Nascom 2, 48K, Nas-Sys 3, ZEAP, Nas-Dis & Debug, BASIC toolkit, 6800 cross compiler, parallel printer cable etc. £250. Ian Doble, St. Agnes (087255) 2121

Nascom 2 in Kenilworth case with fan; 40K RAM, I/O board with Uart + 1 PIO, display monitor. Nas-Sys 3, BASIC toolkit, Hullforth, Wordease, V&T Assembler and games. Manuals and large number of Nascom related magazines. All above + IMP printer & IMPRINT that needs attention for £350. Taylor - 029668 651 evngs

Nascom 2, RAM B 48K, Sound board and sound programs, 12K Pascal, ZEAP, Nas-Dis, Debug, Naspen & lots of games. Monitors Nas-Sys 1 & 3. Including all manuals & newsletters - £400. Also RAM A memory board £25. U. Soni, 32 Downs Rd., Lower Clapton, London E5.

Nascom 2 + RAM B board giving 48K RAM, Toolkit, Nas-Gra, Soundbox, Castle Interface, IMP printer, Naspen, Imprint, spare printer ribbon, tape recorder, all cased as a compact system. User manuals. £500 the lot. T. R. Kirby-Smith, San Cass, 22 Pantiles Close, St Johns, Woking, Surrey, GU21 1PT. Woking 72739.

Nascom 2 cased with 48K RAM B, AVC, ZEAP, Naspen, Nasprint 80, Nas-Dis/Debug, Pascal. Offers over £550. Waterlooville (Hants) 52314.

Nascom 2, 32K RAM, I/O board, Kenilworth case, GM805 disk system, software - Pascal, ZEAP, Naspen, Nas-Dis/Debug. £650. Ring Steve for details 0969 23459.

Micropolis 400K floppy disk drive, second drive, hardly used, upgrading to a RAM-DISK system. £180 ono. Roy Ward, Macclesfield 610678.

Nas-Sys 1 (ROM) - £6; ZEAP (4x2708) - £22; Bits & PCs Programmers Aid (Nas-Sys 1) - £10; Naspen - £10. Also: Easicomp sound generator (AY-3-8910 type) on 80-BUS board - £30, and IMP printer (+ IMPRINT) - £140 ono. Ring Plymouth 709722 (evenings).

O level standard multiple choice examination program. Basic program - insert your own data £3.00. Full program with data £5.00. SAE for details of subjects. Robert Wood, The Limes, Druidstone Road, St. Mellows, Cardiff, CF3 9XD. Phone (0222) 791435

WANTED - The buffer board section of a Micropower 64K RAM, P.C.G., buffer board. Any reasonable price considered. D.G.Richards, 29 Martin Crescent, Tonyrefail, Mid. Glam., CF39 8NT. Tel. Tonyrefail (0443) 676676 (after 12 noon).

DH's ODD MUSINGS AND LETTERS

By D. R. Hunt

Disk reliability

Now all those with disk systems will be aware that those square 5" (and 8") black plastic things known as disks are relatively expensive, and that even though disks are fast, prolific code or text writers, or itinerant software lifters will be fully aware that their capacity is limited and that new disks will be required sooner or later. Those blessed with high capacity disk drives, 400K - 800K per drive, et al, will be in the fortunate position of having to purchase fewer disks than those who only have, say, 70K per drive. However, if you believe the propaganda, those with high capacity drives will have to pay more for their disks anyway. Well, I propose to have a brief look at disks, and from my own experience, dispel a few old wives tales, and probably create a few heresies in the eyes of the manufacturers.

Firstly, let's have a look at the different breeds of disks, they are available from about £15.00 per ten to about £50.00 per ten, so what has one manufacturer got over another for the price? Well firstly, there are a lot fewer disk manufacturers than there are brands of disk. A lot of badge engineering goes on, not only sticking a particular computer manufacturers' name on the disk and shoving the price up accordingly, but also producing the same disk with different labels to sell in different markets at different prices.

Disk Oxides

Now on to the oxides used on the disks: once upon a time I used to read everything I could on the various types of chemistry that went into the manufacture of magnetic oxides for recording purposes, and the ground rules laid down for audio recording are in many ways the same for digital data recording.

Without being in the least technical, and committing my first heresy by making the the broadest of generalisations, there are two types of oxide, the dull brown stuff and the shiny black stuff. In general the brown stuff is softer (physically) than the black stuff and slightly more prone to wear than the black stuff. However, the brown stuff requires considerably less recording head current to reach magnetic saturation, and so is more suited to drives which have low head currents. The reverse is true of the black stuff, it is (physically) harder and requires considerably more head current to saturate the media.

Ignoring the wearing capabilities of the disks, which in normal home use are minimal anyway, it seems the choice of magnetic material is down to the head recording currents of the drives, in other words, those drives with lower head currents would prefer disks coated in the brown pudding, whilst those with higher head currents would prefer those with the black pudding but will work equally as well on the brown pudding. So how do you know which type of drive you have? Unfortunately this is the sort of information that manufacturers of drives don't tell you as the information is contained in the OEM (Original Equipment Manufacturers) technical manual, and you usually aren't privy to these. Apart from that, how would you know whether the head current given in the manual to create a saturation of X mWebers is high, low or indifferent without samples of all the drives around? Fortunately, the solution is easier than this, if somewhat empirical. If the drive was designed (designed, not necessarily manufactured) more than about 5 years ago, it is almost certainly more suited to the brown pudding. More recent designs keep step with improving media technology (lovely buzz phrase that, media technology) and would be suitable for both the brown and black puddings.

Testing Disk Suitability

One way to test the suitablity of a disk is to do a bit of software twiddling, not so easy as you have to understand the disk drive primitives, but consists of counting disk read failures. Not may people will be prepared to try this as an idea, and it does require a lot of disks to produce a valid statistical sample, so the only way is to gain experience over a period of time. I've noted, long term, that certain make of drive X will always be less reliable with disk brand Y than with disk brand Z.

Temperature and Humidity

Finding media errors this way may suggest that it is the magnetic coating which is unsuitable, again not necessarily true, as this assumes that all other things are equal. We have discounted temperature and humidity. Temperature first. If we take the worst case, double sided 96 track per inch drives, the track width is about 7 thou. Now at the outer edge of the disk this means that the head mechanism has to move the head to a known position +/- about 2 thou, not a difficult engineering job provided expansion is ignored. Taking expansion into account lifts drive and disk design from the mundane to the extremely difficult. The base material of the disk is usually mylar, a plastic of well defined characteristics, but one which non-the-less expands with temperature. Likewise, the drive head cantilever (or whatever type of support mechanism is used) also expands with temperature, and being made of different materials, like as not at a different rate. Shugart, in some of their earlier drives took the logical step of making the head support mechanism of the same material as the disk, unfortunately, one of the properties of mylar is flexibility so what they gained by matching the coefficient of expansion, they lost by the head wobbling all over the place. These days nylon head supports are usual, and the shape and length of the support, coupled with the length and material of the lead screw or cam used to move the head are combined to ensure that the head moves with expansion at a uniform and equal rate to the expansion of the disk, OVER A LIMITED TEMPERATURE RANGE. So knowing this temperature range is important, it's certainly between 15 and 35 C, and most drives are better. If the computer is cased and ventilation is inadequate, it's surprising how high the temperature can rise within the box. This wouldn't be the first time I've heard of obstinate media/drive faults causing disk misreads being entirely cured by cleaning out six months accumulated muck from the fan filter!!! Another 'aside' on the subject of temperature, why are diskettes always black? I'll let you work that one out yourselves.

Humidity. Now that's one I didn't expect. During the hottest and stickiest two weeks of summer a couple of years ago, we experienced a complete and utter failure of the works Gemini Galaxy to 'boot'. Once we'd got it booted, then the number of disk errors which were recovered by the retry logic were out of all proportion to those previously experienced or experienced later. We tried the lot, cleaned the filters, ran the thing with the lid off to keep it cooler, changed the drives, everything. Still no better. We noted that the disks read more easily towards the middle of the disk than the outer tracks, and if we PIPed to a newly formatted disk (preferably an old one) everthing was Ok. It all pointed to the disks being in trouble, but why? This situation went on for some days until a wandering disk rep. happened to drop by. After greeting him with my usual banter specially reserved for getting rid of people trying to sell me things, I told him his blankety-blank disks were useless anyway as we couldn't get them to boot. His cool reply was that, 'What did I expect when the temperature was in the eighties, and the humidity was 99%!!' It turns out (quite logically) that disks absorb atmospheric moisture

and swell. The mechanical temperature compensation of the drives was coping quite well with the high temperatures as proved by the fact that disks could be formatted and used normally; what it couldn't cope with was disks, which having drunk most of the atmospheric moisture around them and swollen up, became a few thousandsth of an inch larger in diameter than usual. Of course the greatest affect of this swelling would be at the outer edges of the disk, and guess where the boot and system tracks are.

So what was the cure. Well we did nothing, we managed to copy the old disks to new disks prepared under the elevated temperature and humidity and used those for the duration. We didn't chuck the old disks, because as soon as the weather returned to normal (sub-zero and rain), we updated the old disks from the new, and went back to using the old disks with no further trouble. However, one fact did emerge from that episode, and that was that Dysan disks were not affected anything like as badly as the others. I have since learned that Dysan lacquer their disks to stop them absorbing moisture and to improve their wearing capabilities, the only trouble with Dysan is that they are twice the price of the competition.

Different disk prices

Now on to the differences in price of double and single sided disks, and quad, double and single density disks. It is my opinion that all disks (except those coated on one side only for single sided use) are created equal, that is double sided, double density, 80 track (96 t.p.i.) disks. In the process of manufacture, they are tested and those which pass the 80 track test become quad disks, those which fail the 80 track test are rechecked at 40 track (48 t.p.i), those which pass becoming double density disks, and then the failures are checked as single density disks. Those that fail then end up in the bin. By this time most, if not all, of the disks have been accepted. Now the vagaries of disk production are not likely to bow to the dictates of the marketing boys whose market profile shows that the majority of disks required are the 40 track double sided breed. In fact, if the production process is going well and not too many lumps are in the pudding before coating, then the majority of disks that fall off the end of the production line are going to be of the double sided 80 track breed and not what is required for sale. So I guess a fair proportion of 80 track disks are diverted and labelled as 40 track disks, simply because the disks being made are better than specification.

So what is this all leading up to? Well we've stopped selling 80 track verified disks for a start. We sell (and use ourselves) only 40 track disks for the 80 track machines. This saves about 70p per disk with no apparent difference of performance whatever. We still get about 1% of duff disks, be they 80 track or 40 track, so as duff disks are sent back for exchange I know we would rather have the 70p's and so would you. All this is borne out by the fact that the original box of 'single sided single density' disks I bought for testing the original Henelec disk controller at the time when it was fitted with single sided Shugart SA400's are still seeing useful daily life on my present machine as 'quad density double sided'; and not one of them shows any signs of either wearing out or giving verify errors.

My Advice

So my advice? Find out what brands of disk suit your machine, and when you've found that out, find out how cheap you can get them. Never buy 80 track verified disks just because you've got an 80 track machine, it seems a waste of money to me. The only exception I make is Dysan, they certainly behave better in high humidity environments and because of the lacquer, should wear

better, so if the disks are being used for business and the firm is paying, then use them, otherwise, anything that goes will do. Personally, I keep my eyes open for secondhand disks, they're usually the best bet and dirt cheap (if not actually given away).

Stubborn Disks

Now on to another couple of tips. If you've got a disk that resolutely refuses to format in one or two places, don't necessarily assume that it is scratched and therefore useless. Before you ditch it (or take it back from whence it came), try an old 'speaker magnet on it. Needless to say, keep the magnet away from your other disks, but by giving the disk a good stir up with a powerful magnet, even the crudiest disks seem to spring into life. Don't ask me why, but they do!!!

Another point is formatting disks. Always allow time for the disk temperature to stabilize within the drive. It's no good formatting a stone cold disk and then expecting it to work on a hot day when the machine has been on for a couple of hours, as the originally formatted tracks are likely to be in entirely different positions under warms conditions. Shove the disk in the hole and wait for 30 - 40 seconds to allow the disk to warm up before formatting.

The final answer with Verbatim disks was revealed to me in a letter from Rory O'Farrell. A little while ago I gave him a disk on which he found no less than 65 duff sectors, Rory concludes as follows, "It might be due to some unkind person having put fingerprints all over the back of it, but as it was a Verbatim disk, made in Ireland, it is more probably due to their cutting down the Fairy Thorn tree that once stood on their site at Raheen, Co. Limerick. As I'm sure you realise, immeasurable bad luck always accompanies interference with the fairies." So if all else fails, go to the bottom of your garden and talk to the fairies.

Drive care (or lack of)

I hear a lot of waffle about cleaning disk drive heads. Again, if the propaganda is to be believed, disk drive heads should be cleaned weekly with the brand X disk cleaning kit for only £25.00. Now I can't answer for the public as a whole, but I've found that our customers fall over backwards with mirth at the thought that I was daft enough to buy some of these kits for stock in the hope that some customer would be equally as daft as to pay the asking price. In fact, the couple of disk cleaning kits the glib, smooth tongued rep. managed to flog me remain untouched, not even I can afford to use them.

Now I have had access to disk based machines for about 3 years now, and during that time I have dismantled and examined a number of drives for various reasons. (Don't you try it or you'll be along for the rip-off £25.00 disk drive alignment disk the same rep. talked me into. Realigning drive heads is something for masochists and loonies only, I know I qualify.) Anyway, to the point, I have not yet seen a drive head that warranted cleaning. Not even the drives out of the Galaxy we had in recently which seemed to live in the bottom of a cement works. This instance doesn't say much for the fan filters, as the whole thing was potted under a quarter inch layer of cement dust, but at least the heads were pristine clean.

It's not that I'm saying that drive heads never need cleaning, all I'm saying is that I have yet to see one that DOES need cleaning. So don't get talked into expensive disk head cleaning kits when any faults are more likely down to something else.

A Strange Drive Fault

Thinking about faults you don't see, try this one. A brand new Galaxy straight out of its box was being fitted with the Winchester Technology colour card for Prestel compatibility. There it was on the bench with its lid off, monitor alongside, being tested. At this stage, had we put the lid back on, then, of course, a certain immutable law would have guaranteed that the colour card would not to work. Anyway, everything worked fine except, try as we might, we could not get drive B: to give us anything else but disk read errors. Of course the obvious ocurred to us - Gemini - God bless their little cotton socks - had supplied a duff drive. It could happen, especially if it was made on a Friday and if someone had had a particularly liquid lunch that day. So we put in a new drive, and guess what, it didn't work either. Well I have this little theory, it's been printed in these annuls before, but repetition is good for the soul, so here goes:

One dead one is tough
Two dead ones is coincedence
Three dead ones and something else is wrong

So after the third drive failure, the drive card was changed - no difference. So the colour card was removed - no difference. So the processor card was changed (for no good reason, except desparation) - no difference. We even thought about changing the box, but even Graham at his most pessimistic could not advance any valid theory as to how the box could affect drive B:. So what was it? Well I gave you the clue at the beginning, have you spotted it. "The lid was off and the monitor was along side." Now monitors have fairly powerful line timebase stages (after all that's how the man in the TV detector van knows you have the box on but no TV licence), and line timebases chuck out an awful lot of low frequency RF muck, about the right sort of frequencies disk drives are wanting to look at, so guess what. Move the monitor, put the steel lid on the Galaxy, and that was the answer to another wasted afternoon.

Keyboard reliability

And so on to another tale of woe concerning, this time, the Rotec function keyboards used on the Gemini Galaxy machines. Apart from my not liking the layout of the cursor keys, a certain amount of trouble has been experienced with the earlier ones. To put it baldly, the early keyboards are unreliable.

One of the causes of unreliability is the egress of muck, and the failures all stem from the way in which the keyboard works. Now I haven't seen the circuits but it is obvious that the keyboard uses capacitive keys. The pcb has areas of track in the form of two pads under each key, which are separated by about 2 - 3mm. When a key is pushed a metal pad on the bottom of the key is pushed down so that the pad on the key covers the two pads on the pcb. The pcb is heavily lacquered so that there can be no electrical connection between the pads when the key is pushed. The coupling between the two pads is therefore affected capacitively, the keypad joining the two pads on the pcb with a capacitance of a few picoFarads. I understand that the key sense lines have 2pF capacitors in series so that the few additional pF's caused by the key closing cause a change of between 0 and something a bit less than 2pF. The 2pF capacitors are there to minimize the reduction in capacitance caused by muck under the keys, but read on. Now I guess the on-board processor (yes the keyboard has a processor all to itself) must be scanning the pads by sending sharp pulses across the keys to check for coupling between them. When it finds a couple of pads joined capacitively, it does the rest, works out what key it

is and generates the appropriate ASCII code and a strobe. Fine, it sounds good in theory, and given good close proximity of the pad on the key and the pads on the pcb, it works.

How does muck affect the keyboard? Where does it go wrong? The lacquer on the pcb is quite thick as far as pcb lacquers go, but is still only about 0.05mm. I haven't worked out the capacitance involved, but the capacitance is proportional to the area of the pads, and inversely proportional to the distance between them. This is also multiplied by the permittivity (dielectric constant) of the lacquer, which I will assume to be about 3 - 4 (air has a dielectric constant of 1). Muck under the keys holds the keypad off the pad, and the chunks of stuff I've found under the keypads is at least as thick as the laquer. So the muck halves the coupling capacitance for a start, next, because the keypad no longer makes contact with the lacquer (except for a small area) the dielectric is now air and not the lacquer, let's be generous and assume that this only reduces the permittivity by half. The capacitive coupling between the keys and the pads is now only a quarter of what it was, so the capacitive change detected by the CPU is now between 0 and some rather less than the 2pF (minus) which was present before.

Now this has puzzled Rotec, as the keyboards were designed with a certain amount of muck in mind, after all that's why the sense lines have 2pF in series, to minimize the affects of the muck. After much poking around, they have discovered that a whole batch of keyboards were made with a 10nF ceramic capacitor which was marked as 10% tolerence component (to their spec) which was in fact a wrongly marked -40/+100% component intended for use as a decoupling capacitor. This of course degraded the sharpness of the scanning pulses, with the predictable result that when the coupling capacitance was reduced by muck, then the key failed to work.

Two further things compound the liability of key failure through muck egress, the first batch of keyboards made no attempt to keep muck out, the next and susbsequent batches have rubber strips round the edges between the pcb and the keyframe to stop muck getting in, and between the keys to stop any muck that does get in from moving around (and also to cut down noise). All well and good, except there are 16 unfilled keyholes on the top of the keyboard frame covered with sticky tape (not really tape, it's quite thick). Sad to say, this can peel off after a few months use leaving half inch square holes on the top of the keyboard for the odd passing brick to drop through. Secondly, the pads on the bottoms of the keys seem to be slightly sticky and any muck that gets under a key promptly attaches itself to the keypad and stays put.

So onto the cure. Sling the thing back at your Gemini dealer, who in turn will sling it back at Rotec, via Gemini, and shout at them. Actually, I understand Rotec have been very good over this and all faulty keyboards returned have been sorted out. That of course is the legal recourse and you would be entitled to do just that, (provided that it is till under warranty) unfortunately, it's not a lot of help if you have just written a couple of hours rubbish using PEN (not having backed up of course) and then discover a passing brick has got under the CONTROL key, and all PEN does is print ZZZZZZ instead of breaking out of the insert mode (^Z, for those who don't know PEN). That's what happened to me once, which is the reason for this bit. The well tried cure is to turn the keyboard on its end and 'thumping it one' (known in the trade as a technical tap) in the hope that the muck will get dislodged and the key will become functional. Needless to say this didn't work. So in my wisdom I decided to take the keyboard apart. When I'd fixed the keyboard, I found that the action of removing or replacing the keyboard had crashed the video card, and using RESET to clear it lost about half of what I had written.

What did I do to the keyboard whilst I had it apart? I'm sure what follows voids any guarantees, so you have been warned. I removed the keyboard from the case. It was then I saw that the sticky over the unused keys had curled and peeled off and that was where the muck was getting in and, to not overemphasise it, the sticky side of the tape was filthy. I undid the 8 or so screws on the pcb side of the keyboard, noting that 3 of them had nylon insulating washers under them although I couldn't see why, and separated the pcb from the keyframe, paying careful attention to the positions of the rubber sealing strips. The true horror of what had been going on was then revealed. The rubber strips are also slightly sticky and were far from clean, the keypads had a lot of little bits sticking to them, but the pcb appeared clean. The topside of the pcb was cleaned with a soft cloth dampened with pure acetone (chemist, 20p) and the end of each keypad was also cleaned using a cotton wool bud, again just dampened with acetone. When all was bright and clean, the rubber strips were wiped to remove the adhereing muck and the keyframe blown out to remove any more muck floating about. The keyboard was then reassembled being very careful to see that the rubber strips were seated square. What was left of the sticky patches over the spare key holes was removed and replaced with heavy duty packing tape which has both a stronger adhesive and being thinner, less tendency to curl. Everthing was replaced in the case and tested, all keys were working, and all seems well. Sometime, when I find out which is the offending 10nF, I'll change it. In the mean time it remains to be seen how well it will continue to work, I am now happy that the only place muck can get in is down the key stems, so we shall see what we shall see. The whole job took about 45 minutes, but didn't improve my temper any. A further late extra from Gemini is that a certain manufacture of chips has been found to less satisfactory than others, being loaded capacitively by the nice curly cable now fitted to the Galaxy machines. Don't try to change this chip yourself. Return the keyboard to your dealer.

The letters

Some time ago Paul passed a large envelope to me and said, "That's your lot, see if anything is useful.", so I did as I usually do, shoved the envelope in my cupboard to see light next time I got round to sorting out letters. Well they've just reemerged, and I've found three letters and five articles. The articles I'll read tonight, the letters, well here goes.

Polydos

The first from Mr. Toler of Cheshire, admonishes me for not having heard of Polydos, and for mucking Mr. Trim's database about (see volume 2 issue 1). Well as I said before, sorry, we can't know everything, can we? Anyway, Mr. Toler goes on to praise the virtues of Polydos, and suggests to Chris Blackmore (Dr. Dark) that, "He includes Polydos in his 'Circle of Iron'". I hope Mr. Toler means to include programs for use under Polydos, rather than Polydos on the disks, as Polydos is copyright of the boys in Denmark, and any infringements would probably mean a reinvasion by the Vikings. Anyway, there you are Chris, I don't know whether you could cater for it in the 'Circle of Iron', but perhaps you should think about it as Polydos has quite a following.

Naspen

Naspen is still alive and kicking although I know that Lucas only managed to sell a grand total of three last quarter. Anyway Steve Stubbs of Inverurie has come up with a little mod. I quote:
"One of the more obvious omissions in Naspen is the ability to print multiple copies of the same thing without having to press 'P' each time. I use a small

piece of relocatable code which loads the B register with the number of copies to be printed, calls the Naspen print command, and loops until the B register is zero. The code can go anywhere except the Naspen workspace. I use 0C80H.

```
0C80  06 3A          LD      B,3AH          ; 58 copies wanted
0C82  CD BAC2        CALL    BAC2H          ; Naspen print routine
0C85  10 FB          DJNZ    0C82           ; Loop 58 times
0C87  C3 B806        JP      B806           ; Return to Naspen
```

Load the text into Naspen and leave the cursor pointing at the first character to be printed. Exit from Naspen using the 'N' command. Enter the above code, setting the second byte for the number of copies required (in HEX) and execute at the address loaded. As Naspen prints each copy, the word 'complete' will be displayed on the screen, this is because this is in the Naspen print command."

SYS

Lastly, from Mr Hill of Newhaven a little tweek to SYSN7, the last of the series of SYS for the Henelec/Gemini G805 disk sysetm. Mr. Hill uses the system under CP/M 1.4 with a Gemini GM812 video card:
"The modification allows me to use ^X to switch between the two keyboard options. Each successive use of ^X flips between the alternative options. The control can be used inside all the CP/M packages that I possess, although there may be some exceptions to this. The listing below shows my modification to the CP/M 1.4 version of SYS.

```
CICRT1          LD      HL,CHCD         ; Existing program
                CP      (HL)
                JR      NZ,CICRTX       ; Jump to new section
                LD      A,BS            ; Existing program
                LD      (BOPT),A
                JR      CONIN9
                                        ; NEW CODE
CICRTX          LD      HL,KOPT         ; Point to new keyboard option flag
                CP      18H             ; Is it ^X
                JR      NZ,CICRT2       ; Skip if not
                LD      A,1
                XOR     1               ; Flip the keyboard option flag ..
                LD      (HL),A           ; .. and put it back
                JR      CONIN9          ; Done
CICRT2          .....                  ; Continue with SYS"
```

Looking at the code, something seems a little wrong, where does the contents of register A come from at CICRTX? I've just rechecked what I have typed in and that's correct, but as I don't have a copy of SYSN7 to hand I hope the code works.

NASPAS Review

Here endeth the letters and, lastly, on to a mini review of the Nascom NASPAS sent in some time ago by Mr. Pennell who doesn't live a million miles away from me, in Pinner Middlesex, although we've never met except perhaps unknowingly over a pile of bean cans in the local Safeways.

"I first purchased the tape version of NASPAS, but later exchanged it for the EPROM version as my Nascom 1 has acres of EPROM space being fitted with two EPROM cards. My Nascom is also fitted with 64K of RAM, an I/O board and an external CUTS tape interface.

Both tape and EPROM versions of NASPAS are supplied with two booklets, a programming manual and an operating manual. The operating manual contains information on getting the system going and details on running the editor, the compiler and other miscellaneous information. The details in both manuals are quite sufficient and well written, but the programming manual, whilst accurate, is rather formal and could be rather heavy going for a novice to Pascal. The programming manual describes the format of Pascal, including data types, declaration expressions, statements, procedures, functions, parameters, etc, all very carefully. But I found the lack of examples made it somewhat difficult to define problems and sort out syntax errors when I got going. A good book on Pascal programming would seem to be a great help as well as the information supplied.

The tape version was supplied on a high quality cassette and loaded easily at 1200 BAUD. The EPROM version is supplied as six 2716 and runs from DOOOH to FFFFH, and is started in the same way as the Nascom Basic, that is, typing 'J' for cold start, plus the memory space to be allocated, otherwise NASPAS defaults to all available memory allocated. 'Z' warm starts NASPAS.

I had some difficulty with the EPROM version, as the title on the screen and the keyboard input routines appeared to be corrupt. As a listing is not supplied, I was unable to check the code myself, so the chips were returned, only to be sent back with 'no fault found'. As the fault was still apparent, I resorted to checksumming the EPROMS and found that one would occasionally give a different result. I made a copy of the chip to tape, and then erased and reblew the EPROM. After that my troubles disappeared. The fault seeming to be poor programming of the chip in question.

A good selection of functions are available including trig. functions, string functions, some graphics functions and port I/O. Also external machine code routines can be called, and external printer patches are provided to allow listings to be made from the editor.

In use, programs are written with the NASPAS editor which uses screen editing facilites similar to NAS-SYS with the exception that a line may be up to 80 characters long, the text scrolling sideways to the left off the screen as text is added to the right. When first used, the effect can be a little disconcerting. Editing of lines which 'wrap round' the screen can also be achieved by moving the cursor to the correct position on the screen, editing, and then pressing the ENTER key.

Once a program is written, it may be compiled and the error codes are displayed one at a time on the top line of the screen with the cursor pointing to the approximate place of error within the text. The meanings of the error codes are listed in the back of the manual. When a successful compile is achieved, then a message is displayed giving the memory locations of the source code, and the also of the machine code program produced.

Both the source code and the machine code produced can be loaded to and from tape, with or without file names. The machine code program produced can be reloaded and run 'stand alone' so long as the run-time program in NASPAS is present at the same time.

It appears to be very versatile and very fast. A comparison drawn with Nascom Basic indicates truly remarkable speed, and programs containing masses of 'IF' statements and the like cause no problems at all. Most of my programs are written for amateur RTTY operation and in the past have been mostly machine code; the Pascal versions appear to run with almost machine code speed.

Overall NASPAS appears to be a very versatile package as supplied, and is an excellent addition and alternative to the Nascom ROM Basic."

My Drive TEAC FD 50E-22

DISK DRIVE COMPATIBILITY

By Richard Beal

As floppy disk technology has developed, more and more data can be stored on each mini-floppy disk. One of the biggest steps was the change from 48 tracks per inch (tpi) to 96 tpi. This was achieved first by Micropolis, who developed a narrower disk head. Other manufacturers such as Teac and Mitsubishi have followed, and the 48 tpi units can be regarded as obsolete, as they offer poorer value for money. Of course the advantage of obsolete items is that it may be possible to get hold of them at a low price, so 48 tpi units are still useful. Incidentally, the IBM Personal Computer at present uses 48 tpi drives!

This article answers the following question:-
"If you have 96 tpi drives, is it possible to process disks from a 48 tpi system?"

The first problem is to make the 96 tpi drives double step, so that only alternate tracks are processed. This can be done through software in the BIOS, or the drive manufacturer may have incorporated this in the design. For example a switch can be added to the Teac 55F which selects normal or double stepping. I believe the Mitsubishi drives can be sent a command by the disk controller specifying this option. In some cases drives may have this feature but it is undocumented.

If you try this you will find that 96 tpi drives have no trouble reading or rewriting disks from a 48 tpi drive, and I have found that rewritten tracks can be read without difficulty on a 48 tpi drive. This is lucky, as the 48 tpi drive is picking up noise as it is reading a wider track than that written. This works with both single and double density.

However I have also tried formatting disks for 48 tpi use using a 96 tpi drive, and this does not work reliably. The 48 tpi drive has trouble finding sector headers and RNF and CRC errors are likely. I have tried erasing the disk to be formatted using a large magnet, then formatting and writing using a 96 tpi drive. This seemed to considerably improve the ability of a 48 tpi drive to read the disk, because the inter-track noise is reduced.

So the answer is that disks can easily be exchanged between users with 96 and 48 tpi drives, provided that the disks are always formatted by the person with the 48 tpi drives. In emergency the 96 tpi drives can be used for formatting, using the magnet method. If you use a magnet, keep it far away from other disks, disk drives, and televisions or monitors, since all of these can suffer damage from magnetic fields. A bulk eraser would be an alternative to a magnet.

It is beyond the scope of this article to describe how to write a BIOS which can process both 48 and 96 tpi disks. This is left as an exercise for the reader!

DOCTOR DARK'S DIARY – EPISODE 19.

By C. Blackmore

My regular reader has probably already noticed that I am constantly extending and improving the noble Marvin, in an effort to build up the biggest and best computer in town. After all, it is very well known that without such a machine available to flash lights and spin great big empty tape reels round and round, the best laid world domination plans are doomed to failure...

My latest addition to the amazing Marvin is the HSA-88B arithmetic processor board from Belectra. The board was originally advertised in this magazine, at £199. However, you will be sad to hear, this price no longer applies. I quote from a letter Belectra sent me - "Since our advertisement in 80-BUS News, we have reviewed our marketing policy (which used to be direct sales to the public) and are now selling also via computer retailers. The price of the HSA-88B has therefore been increased to include dealer margins and other factors associated with selling to trade customers, to £253 plus VAT." So, if you rushed out and ordered one as soon as you saw the advertisement, you are now feeling pretty pleased with yourself, I expect.

There was going to be a great deal of talk about the hardware you get for the money, here, but a certain David Parkinson stole my thunder in the last issue, so I am frantically re-editing this article to save wasting space by going over old ground. So if it reads as though I have done a rush job, you will know why. I don't really feel bitter about this, David, honest! (Your article doesn't say how much you paid for your board, was it a freebie?) [Ed.- it was on loan for review and has now been returned to Belectra.]

As processor chips go, the 9511 is fairly odd (though it is by no means as unbelievably bizarre as the old SC/MP chip in Sinclair's original computer, the amazing MK14) as it does not fetch opcodes and data from memory for itself. Instead, your main processor loads data into the chip, followed by an instruction opcode. When the 9511's status line indicates that the processing has been done, the main processor can read out the answer. The 9511 has an eight bit bus system for communication with the outside world. Since it deals with either 16 or 32 bit quantities, data has to be sent in two or four byte blocks, and the fast Z80 block output instructions can be used. The chip has five distinct groups of instruction codes, which are for 16 bit fixed point operations (integers), 32 bit fixed point operations (enormous integers!), 32 bit floating point primary and derived operations, and data and stack manipulation operations. The 16 bit and 32 bit fixed point operations that can be performed are addition, subtraction, multiplication and division. The same four operations can also be done in 32 bit floating point format, and these are referred to as the primary operations. The derived operations consist of those that can be built up using the primary operations in various combinations, and are square roots, sines, cosines, tangents, inverse sines, inverse cosines, inverse tangents, common logarithms, natural logarithms, "e" raised to a given power, and any number raised to a given power. If you didn't understand that sentence, I suggest you try the Open University course, M101, that I recently finished (I am not claiming to have passed, as I don't believe in tempting providence!) The data and stack manipulation commands are mainly concerned with matters such as changing to and from floating point format, changing sign and so on, but also allow you to load pi into the processor in an amazing 16 clock cycles. If you are into that sort of thing, you might like to work out how many clock cycles it would take to load pi from the main processor to the chip, assuming you know the 32 bit floating point value of pi, which I seem to have forgotten...

Some of the above I got from Belectra's part of the documentation, which is clear, if rather brief; much of the rest of the information about the 9511 came from "Modern Microprocessor System Design" by Daniel R McGlynn. Also included in the information Belectra give you are details of the ports used by the board, but there is no need to repeat them here. If you do reconfigure the board to other ports, you can easily patch the Pascal compiler's run time routines to take account of this, as the address of the data port is stored at £0103, and the address of the control port is at £0104. (Quoted from Hisoft's "Functional Differences between Hisoft Pascal 5D and Hisoft Pascal 4D", which comes with the Pascal manual.)

Belectra's documentation gives brief but very clear examples of how to program the board using Z80 assembly language, and it is clear that this is quite an easy job, when compared with writing your own set of floating point routines. The thing is, the rest of the program is going to be a bit of a pig, if it uses the calculations the board can do, to any great extent. I once wrote a biorhythm program in 1K of Z80 code (and I hasten to point out that I do not believe in biorhythms!), and it was no fun at all, especially the bit that drew the sine waves. So I will move hurriedly on to the sensible way of doing things, namely with high level languages, pausing only to note that if you do program the board in assembler, then there is no reason why the Z80 can not get on with something else while the HSA-88B is working. For instance, it could be preparing the next data to be worked on. It could just as easily be controlling another HSA-88B, if you have the money to run more than one of them! Then you would need to write some nice queue handling routines, especially if there was another board that also ran independantly, like the Pluto. I think you had better forget that sort of programming style, unless you are a real masochist. After all, you are supplied with Hisoft Pascal 5 along with the board, and Hisoft's compilers are renowned for the speed of the code they produce. HP5 uses the HSA-88B for all its arithmetical work, which means that as well as compiling programs that run faster, it also needs a lot less memory space for its completely rewritten run-time routines. (Sometimes you can have your cake and eat it too!) If you want to use some other compiler with the board, or even an interpreter, this is possible, as long as you know how to do things like modifying the run-time routines the compiler churns out, or finding the bit of the interpreter you need to change. Belectra give brief notes about what is required, but also say "The only remaining consideration is whether a particular compiler or interpreter is worth modifying; some generate code which is so inefficient that even if the HSA-88B performed its function in zero time there would not be any noticeable difference in overall program execution speed." I haven't tried the patch to MBASIC from the last issue, as I can't afford MBASIC...

Hisoft Pascal 5D is supplied with the manual for the earlier 4D version, and six pages of notes concerning the differences between the two compilers, marked "provisional" on my copy. The major change, obviously, is the use of the HSA-88B for all arithmetic. This includes integers, all of which are now to 32 bit precision. As a result of this, MAXINT is now 2,147,483,647, which is a big number! I am not at all bothered by the compiler's inability to handle recursive WITH statements, though there is bound to be someone, somewhere who thinks this is a fatal flaw! Instead of RANDOM, you now put RANDOM(X), where X is either zero, or a seed for the random number routine. If you use 0, the value returned is between 0 and 65535 inclusive, rather than the old 0 to 255. GOTO (whatever that is) can not transfer execution to another block. I haven't written a GOTO in Pascal yet, and if I ever need to, I will be quite surprised. The provisional manual gives the addresses in the

run-time routines which have to be changed (presumably using POKE once your program is running) if the HSA-88B is not at ports 80,81H. Another change allows underline characters to be used in identifiers, to help with legibility. In order to comply with the ISO standard, a variable used as a loop control variable must be declared in the block the loop is in, rather than globally. This seems quite a good idea. If you are using my "explain" utility, you will need to add a line to the CASE statement, to add error message number 74, which says "FOR statement control variable must be declared in the closest enclosing block." If you are not using "explain", I bet you spend a lot of time looking in the manual! The new procedures PRON and PROFF have the effect of switching output between the printer and the screen, which is handy, but far from ISO standard, as are the new functions ARCSIN(X), ARCCOS(X) and POWER(X,Y). I expect you can tell what they do, and won't use them in programs you want to be portable...

That brings us to the question I always used to ask when confronted with a new sports car or huge motor cycle, namely "Wot'll it do, mister?" (Nowadays, I don't ask about cars at all, and given the bike, will find out by experimental means!) Just for you, I looked up, typed in, compiled and ran the Pascal benchmarks suggested by Personal Computer World. The timings were done with an aged Post Office hand operated stop watch, so the accuracy is very much dependant on the accuracy or otherwise of my own reaction times. I was sadly out of home-brew at the time, so the results should be accurate to within say half a second (I'm older than the stop watch!) As you will see, I compiled them with HP4, HP5 and Compas. With HP4, I used the compiler options O-,C-,S-,A-,I-to cut out all the checks that slow down a program, and with HP5 I used O-,C-,S-,A-. (HP5 has no "I-"). I couldn't find the page about compiler options in the Compas manual, so I compiled the programs without the benefit of any such options. Here are the results, which are not entirely as I would have expected them to be, at first sight.

Program	HP4	HP5	Compas	
Magnifier	.2	.6	.6	
Forloop	2.4	8.8	8.8	
Whileloop	4.2	16.4	7.4	
Repeatloop	3.6	15.2	6.6	
Literalassign	3.4	10.2	3.6	
Memoryaccess	3.6	10.6	3.8	
Realarithmetic	14.6	11.2	52.2	
Realalgebra	15.0	16.8	41.6	
Vector	8.0	22.0	14.8	
Equalif	6.0	19.2	8.2	
Unequalif	5.6	19.6	8.0	
Noparameters	3.0	3.6	.8	*
Value	3.4	5.0	7.0	
Reference	3.6	4.2	7.2	
Maths	6.2	5.2	47.8	

You should be able to spot some obvious and some surprising trends in the table. The surprise is that the programs sometimes take longer in HP5 than in HP4. This is in fact not so very surprising, because the integers used by HP5 are twice the length, which probably more than doubles the time taken to handle them, even though actually working things out with them will be faster. Thus, Program Magnifier, which is just an empty loop, takes .6 seconds instead of .2 seconds (bearing in mind what I said about timing accuracy). Compas does

```
PROGRAM MAGNIFIER;
VAR
  K : INTEGER;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO ;
  WRITELN('E')
END.

PROGRAM FORLOOP;
VAR
  J, K : INTEGER;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO
    FOR J := 1 TO 10 DO ;
  WRITELN('E')
END.

PROGRAM WHILELOOP;
VAR
  J, K : INTEGER;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO
  BEGIN
    J := 1;
    WHILE J <= 10 DO J := J+1
  END;
  WRITELN('E')
END.

PROGRAM REPEATLOOP;
VAR
  J, K : INTEGER;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO
  BEGIN
    J := 1;
    REPEAT
      J := J+1
    UNTIL J > 10
  END;
  WRITELN('E')
END.

PROGRAM LITERALASSIGN;
VAR
  J, K, L : INTEGER;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO
    FOR J := 1 TO 10 DO L := 0;
  WRITELN('E')
END.
```

```
PROGRAM MEMORYACCESS;
VAR
  J, K, L : INTEGER;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO
    FOR J := 1 TO 10 DO L := J;
  WRITELN('E')
END.

PROGRAM REALARITHMETIC;
VAR
  K : INTEGER;
  X : REAL;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO
    X := K/2*3+4-5;
  WRITELN('E')
END.

PROGRAM REALALGEBRA;
VAR
  K : INTEGER;
  X : REAL;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO
    X := K/K*K+K-K;
  WRITELN('E')
END.

PROGRAM VECTOR;
VAR
  K, J : INTEGER;
  MATRIX : ARRAY [0..10] OF INTEGER;
BEGIN
  WRITELN('S');
  MATRIX[0] := 1;
  FOR K := 1 TO 10000 DO
    FOR J := 1 TO 10 DO
      MATRIX[J] := MATRIX[J-1];
  WRITELN('E')
END.

PROGRAM EQUALIF;
VAR
  J, K, L : INTEGER;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO
    FOR J := 1 TO 10 DO
      IF J < 6
      THEN L := 1
      ELSE L := 0;
  WRITELN('E')
END.
```

```
PROGRAM UNEQUALIF;
VAR
  J, K, L : INTEGER;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO
    FOR J := 1 TO 10 DO
      IF J < 2
      THEN L := 1
      ELSE L := 0;
  WRITELN('E')
END.

PROGRAM NOPARAMETERS;
VAR
  J, K : INTEGER;
PROCEDURE NONE5;
BEGIN
  J := 1
END;
PROCEDURE NONE4;
BEGIN
  NONE5
END;
PROCEDURE NONE3;
BEGIN
  NONE4
END;
PROCEDURE NONE2;
BEGIN
  NONE3
END;
PROCEDURE NONE1;
BEGIN
  NONE2
END;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 10000 DO NONE1;
  WRITELN('E')
END.

PROGRAM VALUE;
VAR
  J, K : INTEGER;
PROCEDURE VALUE5(I : INTEGER);
BEGIN
  I := 1
END;
PROCEDURE VALUE4(I : INTEGER);
BEGIN
  VALUE5(I)
END;
PROCEDURE VALUE3(I : INTEGER);
BEGIN
  VALUE4(I)
END;
```

```
PROCEDURE VALUE2(I : INTEGER);
BEGIN
  VALUE3(I)
END;
PROCEDURE VALUE1(I : INTEGER);
BEGIN
  VALUE2(I)
END;
BEGIN
  WRITELN('S');
  J := 0;
  FOR K := 1 TO 10000 DO
    VALUE1(J);
  WRITELN('E')
END.

PROGRAM REFERENCE;
VAR
  J, K : INTEGER;
PROCEDURE REFER5(VAR I : INTEGER);
BEGIN
  I := 1
END;
PROCEDURE REFER4(VAR I : INTEGER);
BEGIN
  REFER5(I)
END;
PROCEDURE REFER3(VAR I : INTEGER);
BEGIN
  REFER4(I)
END;
PROCEDURE REFER2(VAR I : INTEGER);
BEGIN
  REFER3(I)
END;
PROCEDURE REFER1(VAR I : INTEGER);
BEGIN
  REFER2(I)
END;
BEGIN
  WRITELN('S');
  J := 0;
  FOR K := 1 TO 10000 DO
    REFER1(J);
  WRITELN('E')
END.

PROGRAM MATHS;
VAR
  K : INTEGER;
  X, Y : REAL;
BEGIN
  WRITELN('S');
  FOR K := 1 TO 1000 DO
    BEGIN
      X := SIN(K);
      Y := EXP(X)
    END;
  WRITELN('E')
END.
```

not use quite such big integers, so it seems to be producing less efficient code for this benchmark. It is not easy to be sure about this, however, on account of the stopwatch and its operator... All the routines using just integers suffer from the same "negative improvement", but look what happens with Realarithmetic! HP5 cuts the time down quite well, while Compas is nowhere near even HP4. And when it comes to the (number) crunch, in Program Maths, HP5 is probably about 20% faster than HP4, even though it is doing a lot more work in the loop counting part of the program. The "*" mark indicates that I didn't believe the timing I got for Program Value, using Compas. Either the compiler is very efficient at calling procedures when there are no parameters, or this is a result of the fact that Compas procedures are not recursive unless you tell it they should be. (In other words, it does not normally set up a new local workspace every time a procedure begins, which proper Pascal compilers have to do.) Or perhaps I compiled it wrong...

Since writing that paragraph, I have read Polydata's remarks in the last issue, and appreciate that much of what I have just said about their compiler may well be hopelessly wrong, when applied to version 2. It would be nice if they could let us know how fast the new version runs the benchmarks. I was pleased to see that they are now taking a much more sensible line over royalties on your programs, too...

Anyway, the bottom line of the table is the one you are buying, and the HSA-88B is clearly pulling its weight there. By comparison, the Sage II, with its 68000 processor, using the well known p-system, and costing rather a lot more than my system, takes 7.6 seconds to run Program Maths. And if you run this:

```
1000 PRINT "S"
1010 FOR K% = 1 TO 1000
1020 X = SIN(K%)
1030 Y = EXP(X)
1040 NEXT K%
1050 PRINT "E"
1060 END
```

using MBASIC, you will find it takes about 25 seconds, running at 4MHz, even though it is using K%, which is faster than plain K. I have also written some other test programs which lean more heavily on the maths processor than the benchmarks do, because they don't really probe very deeply into the realms of number crunching, and there is an even more noticeable improvement, but space does not permit the inclusion of these programs. They don't actually do anything useful, anyway, unless anyone wants a list of all the prime numbers lower than MAXINT...

If you want the answers to your sums faster, or just want to be the envy of all the people with slow computers, then the HSA-88B has to be a good buy. As soon as I have taken my Open University exam for this year, and have some time to myself and Marvin, I am hoping to write some fancy stuff, using Pluto, HSA-88B (pity they couldn't have given it a nice name - what does the 88B bit refer to?) and HP5. Real 3D programs need fast matrix calculating routines to make them work. See in particular the article in December PCW for an idea of what can be done. Perhaps I need an array of HSA-88B's, say four of them, running in parallel...

References:

Pascal Benchmark programs from Personal Computer World, Dec 1980, pages 59-61.

Corrections to the above, Personal Computer World, March 1981, pages 116-117.

Sage II benchmarks, Personal Computer World, February 1983, page 174.

"Modern Microprocessor System Design" by Daniel R McGlynn, published by John Wiley and Sons, ISBN 0-471-06492-0.

Late extra bits!

Prompted by my unusually rude remarks, several people have stopped being apathetic, and volunteered to set up and organise software circulation systems for systems like theirs. Incidentally, would whoever is hanging onto the original Circle disc please put it back into circulation. It is an Inmac lifetime one, and the owner must be wishing he had used a cheap one!

Nascom users running Nas-Dos with Nascom double density drives should contact Roger Dowling, at 11 Westbrooke Road, Welling, Kent, DA16 1PR.

Users compatible with G805 using DCS-DOS and/or CP/M (but I don't know what drives - Pertec, I assume?) should write to Ray Cutler, 6 Bearmore Road, Cradley Heath, Warley, West Midlands, B64 6DX. He says if you have a text editor to spare, it would help, as he is writing letters in the form of BASIC programs at present! No piracy, please...

I have been sent (by our generous editor) a huge pile of Nascom related items, which I hope to be able to get to work on soon. If you have been wondering what happened to your masterpiece for ages, I think I have it. Don't worry, if it gets printed, you get paid, not me!

END

Letter to the Editor

Belectra Arithmetic Board

Dear Sir,

We wish to clarify the following points regarding the review of our Arithmetic processor board in Vol. 2 Issue 5 of 80-BUS News:

1) The Programming Manual is now printed on a printer with desenders.

2) Because Mr Parkinson received our review board directly from Gemini Microcomputers, who borrowed it for an exhibition, while some parts of the documentation came directly from us, he did not receive the complete documentation set. The HP5 Pascal compiler is a substantially enhanced version of HP4 and the Manual sent out with each board clearly lists all the differences and enhancements.

3) We have not tested our board with the Am9512 processor but we can see no reason why it should not work. Anyone requiring this alternative processor is welcome to contact us.

Yours faithfully
P.Holy, Director, Belectra Ltd.

ACCESSING PRESTEL with a NASCOM 2

By Robin Luxford

Although Prestel was originally designed to reach the home via the television receiver and an adaptor with a numeric keypad, this method seems to have severe limitations; unless you buy a special printer you cannot have hard copy and any form of magnetic storage comes even more expensive.

However, with the availability of surplus Post Office approved modem cards, and the 300 baud service now being run by Prestel, it becomes a fairly straight forward matter to access Prestel with your home computer.

The three programs PREST, SAVEPRES and PRINPRES were written for a Nascom 2 running CP/M 2.2 with a Gemini IVC card, using the N2 keyboard, although the principles are applicable to any system.

PREST makes your computer run as a dumb terminal storing incoming data until all the RAM up to the start of the BDOS is full. When you are up to your last page of RAM the screen inverts as a warning but in practice you have to be connected to Prestel for a very long time to get that far. When you hit CTL/C to exit, the program sends the sign-off string to Prestel then waits a few seconds to see if there is a mailbox message for you before warm-booting. Because the mailbox facility is only avaiable at present on the Enterprise computer and the 300 baud service is only available on the Kipling and Dryden computers, I have not yet been able to develop any routines to handle mailbox messages. At the start of each new page Prestel sends a string of 40 DELs; in order to save buffer space and also to simplify subsequent handling of files by GEMPEN, the first DEL is converted to 07H and the rest of the string is suppressed.

After exiting from PREST you can make a disk file fom the buffer by running SAVEPRES or print it on CP/M List device by running PRINPRES.

If you are connecting to a telephone jack via a unit such as the GEC LTU 11 line coupler (also available on the surplus market), you can have the added convenience of being able to dial the Prestel number automatically. The PREST program has routines for doing this, also for sending your Customer Identity number automatically when the Prestel computer replies.

The only hardware mod. to the Nascom required is to connect the uart rx clock to the tx clock so that when the tx clock is switched to 300 baud, the rx clock will also be switched to it. The 300 baud "cassette" position of the rx clock will not work. The simplest way of achieving this is to connect the end of R19 nearest to IC29 to TP5 and switch LSW2/6 to EXT. LSW2/5 must be in th EXT/TTY position of couse and LSW2/7 must be in the TTY position to receive from the modem.

Acknowledgement is due to tha author of the DUMBTERM program published in this magazine some time ago, who I think was David Parkinson. Some of his program appears in the source code of PREST, mostly the upper case lines.

```
                        title PRESTEL for Nascom 2 & Gemini IVC
                        subttl PREST V8, PRESTEL terminal program. RL.
                              22/11/83
                        .z80

0000                    false   equ 0
FFFF                    true    equ not false

                        ; Autodial, set true if connecting to line via barrier
                        ; unit such as LTU 11, false if using acoustic coupler
0000                    autodl  equ false

0000'                           ASEG
                                ORG 100H


                        ; UART type - 6402

                        ; Ports for UART to host/modem
0001                    uartd   equ 01H         ; Data
0002                    uarts   equ uartd+1     ; Status

                        ; IVC Ports
00B1                    IVCD    EQU 0B1H
00B2                    IVCS    EQU 0B2H

                        ; Port for dialling, (Skt. A)
0004                    dialpd  equ 04h         ; Data
0006                    dialps  equ dialpd+2    ; Status

                        ; Socket A bit allocation
                        ; b4    Select line relay
                        ;  5    Dial relay

                        ; Characters
0003                    conc    equ 03h
000D                    cr      equ 0dh
0007                    bell    equ 07h

                        ; CP/M routines
0006                    condir  equ 6
0009                    pstrng  equ 9
0000                    jwboot  equ 0
0005                    jbdos   equ 0005H

                        ; Buffer - also used by utilities subsequently loaded
0500                    buffer  equ 500h

0100  C3 0324                   JP START

                        ; Insert phone number & Customer identity no. which
                        ; Prestel supply when you enrol

0103  31 32 33 34       pnum:   defb "1234567",00       ; may be more than 7
0107  35 36 37 00                                       ; digits, must be
                                                        ; followed by null
```



```
                        ; Customer identity number
010B  31 32 33 34       cinum:  defb "1234567890"
010F  35 36 37 38
0113  39 30

                        ; Sign off string
0115  2A 39 30 23       sigoff: defb "*90£ *90£"
0119  20 2A 39 30
011D  23

                        ; Exit jump, can be patched
011E  C3                jconc:  defb 0c3h
011F  0000                      defw jwboot

                        ; Working storage
0121  00                delflg: defb 0          ; Flag for suppressing DEL
                                                ;       strings
                        ; Messages output to screen
0122  1A                msgint: defb 1ah
0123  09 50 52 45               defb 09h,"PRESTEL COMPUTER CALLING PROGRAM"
0127  53 54 45 4C
012B  20 43 4F 4D
012F  50 55 54 45
0133  52 20 43 41
0137  4C 4C 49 4E
013B  47 20 50 52
013F  4F 47 52 41
0143  4D
0144  0A 0A 0A 0D               defb 0ah,0ah,0ah,0dh
0148  43 68 65 63               defb "Check that modem is switched ON,"
014C  6B 20 74 68
0150  61 74 20 6D
0154  6F 64 65 6D
0158  20 69 73 20
015C  73 77 69 74
0160  63 68 65 64
0164  20 4F 4E 2C
0168  20 69 6E 20               defb " in Normal,",0DH,0AH
016C  4E 6F 72 6D
0170  61 6C 2C 0D
0174  0A
0175  4F 72 69 67               defb "Originate and Full duplex mode and baud-"
0179  69 6E 61 74
017D  65 20 61 6E
0181  64 20 46 75
0185  6C 6C 20 64
0189  75 70 6C 65
018D  78 20 6D 6F
0191  64 65 20 61
0195  6E 64 20 62
0199  61 75 64 2D
019D  0A 0D                     defb 0ah,0dh
019F  72 61 74 65               defb "rate switches in computer are set as"
01A3  20 73 77 69
01A7  74 63 68 65
01AB  73 20 69 6E
01AF  20 63 6F 6D
```

```
01B3  70 75 74 65
01B7  72 20 61 72
01BB  65 20 73 65
01BF  74 20 61 73
01C3  20 66 6F 6C          defb " follows:-",0ah,0ah,0dh
01C7  6C 6F 77 73
01CB  3A 2D 0A 0A
01CF  0D
01D0  09 31 20 3C          defb 09h,"1 <--",0ah,0dh
01D4  2D 2D 0A 0D
01D8  09 32 20 3C          defb 09h,"2 <--",0ah,0dh
01DC  2D 2D 0A 0D
01E0  09 33 20 3C          defb 09h,"3 <--",0ah,0dh
01E4  2D 2D 0A 0D
01E8  09 34 20 3C          defb 09h,"4 <--",0ah,0dh
01EC  2D 2D 0A 0D
01F0  09 35 20 3C          defb 09h,"5 <--",0ah,0dh
01F4  2D 2D 0A 0D
01F8  09 36 20 20          defb 09h,"6  -->",09h,"any key "
01FC  2D 2D 3E 09
0200  61 6E 79 20
0204  6B 65 79 20
0208  74 6F 20 70          defb "to proceed,",0AH,0DH
020C  72 6F 63 65
0210  65 64 2C 0A
0214  0D
0215  09 37 20 20          defb 09h,"7  -->"
0219  2D 2D 3E
                           if autodl
                           defb 09h,"but T omits auto dialling & CI no."
                           else
021C  09 61 66 74          09h,"after dialling PRESTEL no."
0220  65 72 20 64
0224  69 61 6C 6C
0228  69 6E 67 20
022C  50 52 45 53
0230  54 45 4C 20
0234  6E 6F 2E
                           endif
0237  0A 0A 0D 24          defb 0ah,0ah,0dh,"$"

023B  0A 0D 57 61   msgwdt: defb 0ah,0dh,"Waiting for dial tone$"
023F  69 74 69 6E
0243  67 20 66 6F
0247  72 20 64 69
024B  61 6C 20 74
024F  6F 6E 65 24
0253  0A 0D 44 69   msgdlg: defb 0ah,0dh,"Dialling now$"
0257  61 6C 6C 69
025B  6E 67 20 6E
025F  6F 77 24
0262  1C 1C 1C 63   msgdc:  defb 1ch,1ch,1ch,"completed$"
0266  6F 6D 70 6C
026A  65 74 65 64
026E  24
```



```
026F  07 0D 0A 2A   msgcf:  defb bell,0dh,0ah,"** Call failed, aborting "
0273  2A 20 43 61
0277  6C 6C 20 66
027B  61 69 6C 65
027F  64 2C 20 61
0283  62 6F 72 74
0287  69 6E 67 20
028B  2A 2A 24              defb "**$"
028E  0D 0A 0A 43   msgclr: defb 0dh,0ah,0ah,"CALL DISCONNECTED: After "
0292  41 4C 4C 20
0296  44 49 53 43
029A  4F 4E 4E 45
029E  43 54 45 44
02A2  3A 20 41 66
02A6  74 65 72 20
02AA  3E 20 70 72           defb "> prompt appears -"
02AE  6F 6D 70 74
02B2  20 61 70 70
02B6  65 61 72 73
02BA  20 2D
02BC  0D 0A 0A 54           defb 0dh,0ah,0ah,"To print,"
02C0  6F 20 70 72
02C4  69 6E 74 2C
02C8  20 72 75 6E           defb " run `PRINPRES' ",0dh,0ah
02CC  20 60 50 52
02D0  49 4E 50 52
02D4  45 53 27 20
02D8  0D 0A
02DA  54 6F 20 73           defb "To save, run `SAVEPRES' "
02DE  61 76 65 2C
02E2  20 72 75 6E
02E6  20 60 53 41
02EA  56 45 50 52
02EE  45 53 27 20
02F2  5B 64 3A 5D           defb "[d:] filename.typ$"
02F6  20 66 69 6C
02FA  65 6E 61 6D
02FE  65 2E 74 79
0302  70 24

0304                        defs 32         ; Stack
0324                stack:

                    ; Start
0324  31 0324       start:  ld sp,stack

                    if autodl
                    ; Initialise dialling port to output mode
                            ld a,0fh
                            out (dialps),a
                    endif

                    ; Output initial  message to screen & wait for key
0327  11 0122       init:   ld de,msgint
032A  0E 09                 ld c,pstrng
032C  CD 0005               call jbdos
032F  0E 01         init1:  ld c,1          ; Console input
```

```
0331  CD 0005          call jbdos
0334  FE 03            cp conc
0336  CA 011E          jp z,jconc
0339  FE 54            cp "T"
033B  CA 03E3          jp z,conlin      ; connect line & on to term
033E  1E 1A            ld e,1ah         ; Clear screen
0340  0E 02            ld c,2           ; Console output
0342  CD 0005          call jbdos

                  if autodl
                  dial1: ld de,msgwdt
                         ld c,pstrng
                         call jbdos
                         ld a,30H
                         out (dialpd),a  ; close both relays
                         ld b,20         ; wait 5 secs, increase if..
                         call dly255     ; ..your exchange needs longer
                         djnz $-3
                         ld de,msgdlg
                         ld c,pstrng
                         call jbdos
                         ld hl,pnum

                  dial2: ld a,(hl)
                         or a            ; finished if null
                         jr z,dial4
                         sub 30h
                         jr nz,dial3
                         ld a,10         ; dial "0"
                  dial3: ld b,a
                         call pulse
                         djnz $-3
                         push bc
                         ld b,4          ; 1 sec post digit pause
                         call dly255
                         djnz $-3
                         pop bc
                         inc hl
                         jr dial2

                  dial4: ld de,msgdc
                         ld c,pstrng
                         call jbdos
                  endif

                  ; Wait for LF then send Customer identity number
0345  06 B0       cick:  ld b,0B0H        ; try for 45 secs (80H= 20")
0347  C5          cick1: push bc
0348  48                ld c,b
0349  DB 02       cick2: in a,(uarts)
034B  17                 rla
034C  30 06              jr nc,cick3
034E  DB 01              in a,(uartd)
0350  FE 0A              cp 0ah
0352  28 0D              jr z,cisnd
```



```
0354  0D          cick3: dec c
0355  20 F2              jr nz,cick2
0357  10 F0              djnz cick2
0359  C1                pop bc
035A  10 EB              djnz cick1
035C  11 026F            ld de,msgcf
035F  18 3E              jr clear

0361  C1          cisnd: pop bc
0362  21 010B            ld hl,cinum
0365  06 0A              ld b,10         ; Send 10 digits
0367  CD 036C            call stgsnd
036A  18 7B              jr DTERM        ; on to terminal routines

                  ; Send <HL> string to serial port, <B> chars.
036C  7E          stgsnd: ld a,(hl)
036D  CD 044E            call serout
0370  23                 inc hl
0371  10 F9              djnz stgsnd
0373  C9                 ret

                  ; Exit routines
0374  D9          exit:  exx
0375  21 0115            ld hl,sigoff    ; send sign off (*90£) twice
0378  06 09              ld b,9
037A  CD 036C            call stgsnd
037D  1E 07              ld e,bell
037F  0E 06              ld c,condir     ; sound bell to acknowledge ^C
0381  CD 0005            call jbdos
0384  D9                 exx
0385  3E 1B              ld a,1bh        ; Restore screen if inverted
0387  CD 045F            call putvid
038A  3E 4A              ld a,"J"
038C  CD 045F            call putvid
038F  06 14              ld b,20         ; 5 sec delay, in case..
0391  CD 03AA            call dly255     ; ..mailbox message
0394  10 FB              djnz $-3
0396  21 0500            ld hl,buffer    ; Store <DE> for print..
0399  73                 ld (hl),e       ; ..or save prog
039A  23                 inc hl
039B  72                 ld (hl),d
039C  11 028E            ld de,msgclr
039F  AF          clear: xor a
03A0  D3 04              out (dialpd),a  ; open both relays
03A2  0E 09              ld c,pstrng     ; print message
03A4  CD 0005            call jbdos
03A7  C3 011E            jp jconc

                  ; Dialling utillities
03AA  3E FF       dly255: ld a,255
03AC  18 06              jr delay
03AE  3E 4A       dly66:  ld a,74         ; Nominal 66 millisecs...
03B0  18 02              jr delay        ;... AOTd to allow for relay
03B2  3E 21       dly33:  ld a,33
03B4  C5          delay:  push bc         ; as "delay:" in SYS
03B5  47                  ld b,a  ; delay = <A> msecs
```

```
03B6  D9                exx
03B7  1E FF             ld e,0ffh       ; poll for ^C
03B9  0E 06             ld c,condir
03BB  CD 0005           call jbdos
03BE  D9                exx
03BF  B7                or a
03C0  28 05             jr z,del0
03C2  FE 03             cp conc
03C4  CA 0374           jp z,exit
03C7  3A 03D3     del0: ld a,(ms)
03CA  E3          del1: ex (sp),hl
03CB  E3                ex (sp),hl
03CC  3D                dec a
03CD  20 FB             jr nz,del1
03CF  10 F6             djnz del0
03D1  C1                pop bc
03D2  C9                ret
03D3  4B          ms:   defb 75         ; for 4MHz, no waits

03D4  3E 10       pulse: ld a,10H       ; open dial relay
03D6  D3 04              out (dialpd),a
03D8  CD 03AE            call dly66
03DB  3E 30              ld a,30H       ; close dial relay
03DD  D3 04              out (dialpd),a
03DF  CD 03B2            call dly33
03E2  C9                 ret

                  ; Connect line and on to term routine
                  ; - from "T" command
03E3  3E 30       conlin: ld a,30h
03E5  D3 04               out (dialpd),a

                  ; Clear screen, and set limit

03E7  3E 1A       DTERM:  ld a,1ah
03E9  CD 045F             call putvid
03EC  3A 0007             LD      A,(7)
03EF  3D                  DEC     A
03F0  32 0437             LD      (LIMIT),A

                  ; Start of main processing loop
                  ; Uses a FIFO buffer to handle the data
                  ; <DE> = Input pointer  <HL> = Output pointer

03F3  11 0502     LOOPO:  LD      DE,BUFFER+2
03F6  21 0502             LD      HL,BUFFER+2
03F9  CD 046C     LOOP1:  CALL    CONST
03FC  C4 044E             CALL    NZ,SEROUT
03FF  E5                  PUSH    HL
0400  B7                  OR      A
0401  ED 52               SBC     HL,DE
0403  E1                  POP     HL
0404  28 F3               JR      Z,LOOP1 ; original (non-store)..
0406  7E                  LD      A,(HL)  ; version is jr z,loop0
0407  23                  INC     HL
0408  CD 045F             CALL    PUTVID
040B  18 EC               JR      LOOP1
```



```
                  ; POLLU - Polls the UART and moves any
                  ;         waiting character to (DE)
                  ; 1st of a DEL string is converted to 07h and
                  ; rest of string is suppressed. Simplifies subsequent
                  ; handling by GEMPEN and saves buffer space.

040D  DB 02       POLLU:  IN      A,(UARTS)
040F  17                  RLA
0410  D0                  RET     NC
0411  3A 0121             ld a,(delflg)
0414  B7                  or a
0415  28 0F               jr z,pollu1
0417  DB 01               in a,(uartd)    ; flg is set
0419  E6 7F               and 7fh
041B  FE 7F               cp 7fh          ; suppress if DEL
041D  C8                  ret z
041E  F5                  push af
041F  AF                  xor a
0420  32 0121             ld (delflg),a   ; reset flg if not
0423  F1                  pop af
0424  18 0D              jr pollu2

0426  DB 01       POLLU1: IN      A,(UARTD)
0428  E6 7F               AND     7FH
042A  FE 7F               cp 7fh
042C  20 05               jr nz,pollu2
042E  32 0121             ld (delflg),a   ; set flg & convert..
0431  3E 07               ld a,07h        ;..to bell symbol
0433  12          POLLU2: LD      (DE),A
0434  13                  INC     DE
0435  7A                  LD      A,D
0436  FE 02               CP      2
0437              LIMIT   EQU     $-1     ; modified on init
0438  D8                  RET C
0439  28 02               JR Z,SWITCH
043B  1B                  DEC     DE
043C  C9                  RET

043D  00          switch: nop             ; only pass once
043E  3E C9               ld a,0c9h
0440  32 043D             ld (switch),a
0443  3E 1B               ld a,1bh
0445  CD 045F             call putvid
0448  3E 49               ld a,"I"        ; Invert screen as..
044A  CD 045F             call putvid     ;..buffer full warniing
044D  C9                  ret

                  ; SEROUT - Outputs the character in <A> to
                  ;          the serial port.

044E  F5          SEROUT: PUSH    AF
044F  DB 02       SEROO:  IN      A,(UARTS)
0451  E6 40               AND     40H
0453  28 FA               JR      Z,SEROO
```

```
                        Title PRINPRES
                        subttl V1 1357 23/11/83

                        ; Prints PRESTEL buffer to CP/M List device

                                .z80
0000'                           aseg
                                ORG 100H

                        ; CP/M equates
0005                    jbdos   equu 5

                        ; PRES equates
0500                    buffer  equ 500H

0100    C3 017A                 jp start
0103    50 52 49 4E             defb "PRINPRES V1 23/11/83"
0107    50 52 45 53
010B    20 56 31 20
010F    32 33 2F 31
0113    31 2F 38 33
0117                    stack:                  ; overwrites title
0117    1A              mesbr:  defb 1ah        ; Clear screen
0118    2A 2A 20 43             defb "** Check that baud-rate switches"
011C    68 65 63 6B
0120    20 74 68 61
0124    74 20 62 61
0128    75 64 2D 72
012C    61 74 65 20
0130    73 77 69 74
0134    63 68 65 73
0138    20 61 72 65             defb " are set for printer **"
013C    20 73 65 74
0140    20 66 6F 72
0144    20 70 72 69
0148    6E 74 65 72
014C    20 2A 2A
014F    0D 0A 20 2D             defb 0dh,0ah," - any key to proceed$"
0153    20 61 6E 79
0157    20 6B 65 79
015B    20 74 6F 20
015F    70 72 6F 63
0163    65 65 64 24
0167    0D 0A 42 75     mesbc:  defb 0dh,0ah,"Buffer corrupted$"
016B    66 66 65 72
016F    20 63 6F 72
0173    72 75 70 74
0177    65 64 24

017A    31 0117         start:  ld sp,stack
017D    11 0117                 ld de,mesbr     ; Reminder to check..
0180    0E 09                   ld c,9          ;..baud rate switches
0182    CD 0005                 call jbdos
0185    0E 01                   ld c,1          ; Wait for keypress
0187    CD 0005                 call jbdos
018A    FE 03                   cp 03h          ; Check for ^C
```



```
0455    F1                      POP     AF
0456    B7                      OR      A
0457    EA 045C                 JP      PE,SER01
045A    F6 80                   OR      80H
045C    D3 01           SER01:  OUT     (UARTD),A
045E    C9                      RET

                        ; PUTVID - sends a character in <A> to the IVC
                        ;          Polls the serial port whilst waiting.

045F    F5              PUTVID: PUSH    AF
0460    CD 040D         PVO:    CALL    POLLU
0463    DB B2                   IN      A,(IVCS)
0465    0F                      RRCA
0466    38 F8                   JR      C,PVO
0468    F1                      POP     AF
0469    D3 B1                   OUT     (IVCD),A
046B    C9                      RET

                        ; Get a character from the keyboard
                        ; Returns NZ if a character is read.

046C    CD 040D         CONST:  CALL POLLU
046F    D9                      exx
0470    1E FF                   ld e,0ffh
0472    0E 06                   ld c,condir
0474    CD 0005                 call jbdos
0477    D9                      exx
0478    B7                      OR A
0479    C8                      RET Z
047A    FE 03                   CP CONC
047C    CA 0374                 JP Z,exit
047F    C9                      ret

                                END
```

```
SAVEPRES        M-80            5 Dec 1983      23:33   PAGE    1

                        Title SAVEPRES
                        subttl V 1 1511 18/11/83

                        ; Makes file from PRESTEL buffer.

                                .z80
0000'                           aseg
                                ORG 100H

0100    C3 0157                 jp start
0103    53 41 56 45             defb "SAVEPRES V1 18/11/83"
0107    50 52 45 53
010B    20 56 31 20
010F    31 38 2F 31
0113    31 2F 38 33
0117                    stack:                  ; overwrites title

                        ; Workspace
0117    0502            from:   defw buffer+2
0119                    ebuf:   defs 2

011B    0D 0A 46 69     mesfs:  defb 0dh,0ah,"File saved$"
011F    6C 65 20 73
0123    61 76 65 64
0127    24
0128    0D 0A 4E 6F     mesns:  defb 0dh,0ah,"No disk space$"
012C    20 64 69 73
0130    6B 20 73 70
0134    61 63 65 24
0138    0D 0A 42 75     msgbc:  defb 0dh,0ah,"Buffer corrupted$"
013C    66 66 65 72
0140    20 63 6F 72
0144    72 75 70 74
0148    65 64 24
014B    00 24 24 24     deflt:  defb 0,"$$$     PRS"
014F    20 20 20 20
0153    20 50 52 53

                        ; CP/M equates
0005                    jbdos   equ 5
005C                    fcb     equ 5ch
0080                    tbuff   equ 80h

                        ; PRES equates
0500                    buffer  equ 500H

0157    31 0117         start:  ld sp,stack
015A    21 0500                 ld hl,buffer
015D    5E                      ld e,(hl)
015E    23                      inc hl
015F    56                      ld d,(hl)
0160    EB                      ex de,hl
0161    E5                      push hl         ; <HL> buffer end
0162    B7                      or a
0163    ED 52                   sbc hl,de
0165    E1                      pop hl
```



```
PRINPRES        M-80            5 Dec 1983      23:18   PAGE    1-1
V1 1357 23/11/83

018C    CA 0000                 jp z,0
018F    21 0500                 ld hl,buffer
0192    5E                      ld e,(hl)
0193    23                      inc hl
0194    56                      ld d,(hl)
0195    EB                      ex de,hl
0196    E5                      push hl         ; <HL> = buffer end
0197    B7                      or a
0198    13                      inc de
0199    13                      inc de
019A    ED 52                   sbc hl,de       ; <DE> = buffer start
019C    44                      ld b,h
019D    4D                      ld c,l          ; <BC> = length
019E    E1                      pop hl
019F    30 09                   jr nc,prn
01A1    11 0167                 ld de,mesbc
01A4    0E 09                   ld c,9
01A6    CD 0005                 call jbdos
01A9    C7                      rst 0           ; warm boot

01AA    D9              prn:    exx
01AB    0E 06                   ld c,6
01AD    1E FF                   ld e,0ffh
01AF    CD 0005                 call jbdos
01B2    FE 03                   cp 03h          ; Check for ^C
01B4    CA 0000                 jp z,0
01B7    D9                      exx
01B8    1A                      ld a,(de)
01B9    D9                      exx
01BA    0E 05                   ld c,5
01BC    5F                      ld e,a
01BD    CD 0005                 call jbdos
01C0    D9                      exx
01C1    13                      inc de
01C2    0B                      dec bc
01C3    78                      ld a,b
01C4    B1                      or c
01C5    20 E3                   jr nz,prn
01C7    C7                      rst 0           ; warm boot

                                end
```

```
0166  30 09              jr nc,chfn
0168  11 0138            ld de,msgbc
016B  0E 09              ld c,9
016D  CD 0005            call jbdos
0170  C7                 rst 0           ; warm boot

0171  23           chfn: inc hl
0172  36 1A              ld (hl),1ah     ; add EOF
0174  22 0119            ld (ebuf),hl
0177  21 005D            ld hl,fcb+1     ; Filename present?
017A  3E 20              ld a,20h
017C  BE                 cp (hl)
017D  20 12              jr nz,open      ; yes
017F  21 014B            ld hl,deflt
0182  11 005C            ld de,fcb
0185  01 000C            ld bc,12
0188  ED B0              ldir
018A  06 15              ld b,21         ; Fill up fcb with nulls
018C  AF                 xor a
018D  12                 ld (de),a
018E  13                 inc de
018F  10 FC              djnz $-2
0191  11 005C      open: ld de,fcb
0194  0E 0F              ld c,15         ; open file
0196  CD 0005            call jbdos
0199  FE FF              cp -1
019B  28 08              jr z,make
019D  11 005C            ld de,fcb
01A0  0E 13              ld c,19         ; delete file
01A2  CD 0005            call jbdos
01A5  11 005C      make: ld de,fcb
01A8  0E 16              ld c,22         ; make file
01AA  CD 0005            call jbdos
01AD  FE FF              cp -1
01AF  28 31              jr z,nods
01B1  11 005C     open2: ld de,fcb
01B4  0E 0F              ld c,15
01B6  CD 0005            call jbdos
01B9  FE FF              cp -1
01BB  28 25              jr z,nods

01BD  2A 0117     chend: ld hl,(from)    ; check end
01C0  E5                 push hl
01C1  ED 5B 0119         ld de,(ebuf)
01C5  AF                 xor a
01C6  EB                 ex de,hl
01C7  ED 52              sbc hl,de
01C9  E1                 pop hl
01CA  38 1E              jr c,exit
01CC  11 0080            ld de,tbuff
01CF  01 0080            ld bc,80h
01D2  ED B0              ldir
01D4  22 0117            ld (from),hl
01D7  11 005C            ld de,fcb
01DA  0E 15              ld c,21         ; write next rec
01DC  CD 0005            call jbdos
```



```
01DF  B7                 or a
01E0  28 DB              jr z,chend
01E2  11 0128     nods:  ld de,mesns     ; "No disk space"
01E5  0E 09              ld c,9
01E7  CD 0005            call jbdos
01EA  11 005C     exit:  ld de,fcb
01ED  0E 10              ld c,16         ; close file
01EF  CD 0005            call jbdos
01F2  11 011B            ld de,mesfs     ; "File saved"
01F5  0E 09              ld c,9
01F7  CD 0005            call jbdos
01FA  C7                 rst 0           ; warm boot

                         end
```

INSTALLING AN EIGHT INCH DISK DRIVE

By Richard Beal

I recently added an eight inch drive to my Gemini system. This article describes the various problems I encountered and how to overcome them. My system has a Gemini GM829 disk controller card, with Teac 5.25 inch drives which have the same interface as the 48 tpi Pertec drives that Gemini used to supply. The eight inch drive is a double density double sided Pertec FD650, but experiences with any other eight inch drives would probably be similar.

I powered up the drive with a separate power supply and obtained a cable to connect it to the disk card from Henry's Radio. I generated a version of SYS to support the drive (as a standard single density single sided drive) and booted up the system. To my satisfaction it worked first time, and I started to use it. [Ed. - certain Gemini CP/Ms also have 8" support already incorporated. If the CP/M sign-on message has an 'E' in its version number (e.g. BIOS Vers 2.4 2-SME) then it means that the drive following the 2 Single sided Micropolis drives is an Eight inch one - SSSD. If you want 8" support and have not got the support contact your dealer for BIOS upgrade details.]

I started doing some text processing work and as the drive and fan were a bit noisy I turned it off when I wasn't using it. When I started trying to read the files I had written on the Teac drives, I found that whole tracks had CRC errors that were permanent, although other parts of the disk were all right. Since this happened on both Teacs, I realised that the problem must lie not with the disks or the controller, or the software, but must be something to do with the 8" drive, which wasn't even switched on.

The problem is caused by noise of some sort coming from the cable leading to the 8" drive. This happens only when there is no power to the PCB in the 8" drive, and is also intermittent. This explained why the errors were CRC errors not RNF, as the data written was corrupted and could never be read back, but the sector headers were unaffected. The solution was to rewire the power supply connections and the switch in the 8" drive case so that the switch controlled the drive motor and fan, while the transformer and drive PCB are always live when the entire unit is plugged in.

While doing the rewiring I was uncertain about one of the connections and tried to pull off the insulating cover with a pair of pliers. The next moment there was a terrible sensation through my whole body and then I found myself at the other side of the room, with bits of furniture hurled around. The metal connector had been bent through 90 degrees, which I would not normally have had the strength to do. When I had recovered I found that the 3 amp mains fuse had blown, which probably saved me. Remember that 250V is rather more than the usual 5V we get used to handling, and that it is sensible to unplug everything from the wall before starting work!

The system then worked reliably, but still had one annoying problem. If no disk was in the drive, or if the door wasn't closed, the system would simply hang up indefinitely. This was disastrous when there was no disk in the drive but the door was shut, as the door locks when the drive is selected. Since I couldn't open the door to insert a disk, the only answer was to reset the system. The software is designed to detect the status of the Ready line, but this was not connected. On the 5.25 inch drives, the software times out when the motor stops, but on 8" drives the motor runs continuously.

On studying the 829 manual, it became apparent that it was impossible to configure the card for both 8" with Ready line, and 5.25 inch without Ready line. If the 5.25 inch drives had been Micropolis there would not have been a problem. Therefore I consulted the designer of the 829, who kindly advised a simple modification to the card which enhances it so that it can support this combination of drives.

For Pertec interface 5.25 inch drives, link 3 is normally removed, and this disables the Ready line input. This link should now be reinserted. Link 1 is left as before, linking B to C only. At this point the system will not work, since the 1797 will not operate without a Ready signal, and when the 5.25 inch drives are selected, no signal is provided. Therefore two diodes are added so that when drive 0 or 1 is selected, a Ready signal is provided. When drive 2 is selected, the 8 inch drive Ready signal operates as normal. Insert a diode between pin 22 and pin 26, also pin 22 to pin 28 underneath the 8 inch connector on the card. A hole in the card running from pin 22 to link 1 (A) can conveniently be used for one end of the diodes. The marked end (the cathode) must be at the pin 26 and pin 28 ends. In addition it is essential to alter the value of R18 from 150 to 3K3 (or rather more if you prefer) as otherwise the diodes cannot pull down the Ready line enough to activate it.

Summary of Modification

Applies to GM829 with Pertec or Teac 5.25 inch drives at addresses 0 and 1, and 8 inch drive(s) at addresses 2 and optionally 3.

Link 1: B to C only.
Link 3: Inserted.
Diode from PL3 pin 22 to PL3 pin 26 (cathode end at pin 26).
Diode from PL3 pin 22 to PL3 pin 28 (cathode end at pin 28).
R18 changed from 150 to 3K3 or rather more.

Further Difficulty

One further problem was encountered. When PIPing files to the 8 inch drive, it would sometimes fail to give the drive ready signal. It was being used as a single sided drive with a single sided disk. The drive can tell whether a single or double sided disk has been inserted, because 8 inch disks have the index hole in different places to allow this to be distinguished. Although the software was attempting only to use the first side (side 0), since the Teac drives were double sided and the file was on the second side, the BIOS was first selecting the drive, then selecting the side and doing the I/O operation. Therefore the drive was selected with the second side active. The drive immediately signalled that it was not ready (showing its lack of intelligence). This could be corrected by modifying the BIOS to select the side before the disk, but an easier method was to disable this option on the drive itself. This modification is documented in the Pertec FD650 manual, and is easily carried out. Another advantage of making this modification is that single sided disks may be used as if they are double sided, without the drive objecting. This generally works perfectly, just as 5.25 inch single sided diskettes generally work perfectly when used as double sided.

NASCOM ROM BASIC DIS–ASSEMBLED

PART 4

BY CARL LLOYD–PARKER


```
EB1F 2B       MRPRNT: DEC     HL              ; DEC 'cos GETCHR INCs
EB20 CD36E8           CALL    GETCHR          ; Get next character
EB23 CA81EB   PRINT:  JP      Z,PRNTCR        ; CRLF if just PRINT
EB26 C8       PRNTLP: RET     Z               ; End of list - Exit
EB27 FEA5             CP      ZTAB            ; "TAB(" token?
EB29 CAAFEB           JP      Z,DOTAB         ; Yes - Do TAB routine
EB2C FEA8             CP      ZSPC            ; "SPC(" token?
EB2E CAAFEB           JP      Z,DOTAB         ; Yes - Do SPC routine
EB31 E5               PUSH    HL              ; Save code string address
EB32 FE2C             CP      ","             ; Comma?
EB34 CA98EB           JP      Z,DOCOM         ; Yes - Move to next zone
EB37 FE3B             CP      ";"             ; Semi-colon?
EB39 CAD2EB           JP      Z,NEXITM        ; Do semi-colon routine
EB3C C1               POP     BC              ; Code string address to BC
EB3D CD5AED           CALL    EVAL            ; Evaluate expression
EB40 E5               PUSH    HL              ; Save code string address
EB41 3AAD10           LD      A,(TYPE)        ; Get variable type
EB44 B7               OR      A               ; Is it a string variable?
EB45 C26DEB           JP      NZ,PRNTST       ; Yes - Output string contents
EB48 CDB8F9           CALL    NUMASC          ; Convert number to text
EB4B CDCEF1           CALL    CRTST           ; Create temporary string
EB4E 3620             LD      (HL)," "        ; Followed by a space
EB50 2AE410           LD      HL,(FPREG)      ; Get length of output
EB53 34               INC     (HL)            ; Plus 1 for the space
EB54 2AE410           LD      HL,(FPREG)      ; < Not needed >
EB57 3A4210           LD      A,(LWIDTH)      ; Get width of line
EB5A 47               LD      B,A             ; To B
EB5B 04               INC     B               ; Width 255 (No limit)?
EB5C CA69EB           JP      Z,PRNTNB        ; Yes - Output number string
EB5F 04               INC     B               ; Adjust it
EB60 3AAB10           LD      A,(CURPOS)      ; Get cursor position
EB63 86               ADD     A,(HL)          ; Add length of string
EB64 3D               DEC     A               ; Adjust it
EB65 B8               CP      B               ; Will output fit on this line?
EB66 D481EB           CALL    NC,PRNTCR       ; No - CRLF first
EB69 CD13F2   PRNTNB: CALL    PRS1            ; Output string at (HL)
EB6C AF               XOR     A               ; Skip CALL by setting "Z" flag
EB6D C413F2   PRNTST: CALL    NZ,PRS1         ; Output string at (HL)
EB70 E1               POP     HL              ; Restore code string address
EB71 C31FEB           JP      MRPRNT          ; See if more to PRINT

EB74 3AAB10   STTLIN: LD      A,(CURPOS)      ; Make sure on new line
EB77 B7               OR      A               ; Already at start?
EB78 C8               RET     Z               ; Yes - Do nothing
EB79 C381EB           JP      PRNTCR          ; Start a new line
```

```
EB7C 3600.      ENDINP: LD      (HL),0          ; Mark end of buffer
EB7E 216010             LD      HL,BUFFER-1     ; Point to buffer
EB81 3E0D       PRNTCR: LD      A,CR            ; Load a CR
EB83 CD9BE6             CALL    OUTC            ; Output character
EB86 AF         DONULL: XOR     A               ; Set to position 0
EB87 32AB10             LD      (CURPOS),A      ; Store it
EB8A 3A4110             LD      A,(NULLS)       ; Get number of nulls
EB8D 3D         NULLP:  DEC     A               ; Count them
EB8E C8                 RET     Z               ; Return if done
EB8F F5                 PUSH    AF              ; Save count
EB90 AF                 XOR     A               ; Load a null
EB91 CD9BE6             CALL    OUTC            ; Output it
EB94 F1                 POP     AF              ; Restore count
EB95 C38DEB             JP      NULLP           ; Keep counting

EB98 3A4310     DOCOM:  LD      A,(COMMAN)      ; Get comma width
EB9B 47                 LD      B,A             ; Save in B
EB9C 3AAB10             LD      A,(CURPOS)      ; Get current position
EB9F B8                 CP      B               ; Within the limit?
EBA0 D481EB             CALL    NC,PRNTCR       ; No - output CRLF
EBA3 D2D2EB             JP      NC,NEXITM       ; Get next item
EBA6 D60E       ZONELP: SUB     14              ; Next zone of 14 characters
EBA8 D2A6EB             JP      NC,ZONELP       ; Repeat if more zones
EBAB 2F                 CPL                     ; Number of spaces to output
EBAC C3C7EB             JP      ASPCS           ; Output them

EBAF F5         DOTAB:  PUSH    AF              ; Save token
EBB0 CD81F4             CALL    FNDNUM          ; Evaluate expression
EBB3 CD90E6             CALL    CHKSYN          ; Make sure ")" follows
EBB6 29                 DEFB    ")"
EBB7 2B                 DEC     HL              ; Back space on to ")"
EBB8 F1                 POP     AF              ; Restore token
EBB9 D6A8               SUB     ZSPC            ; Was it "SPC(" ?
EBBB E5                 PUSH    HL              ; Save code string address
EBBC CAC2EB             JP      Z,DOSPC         ; Yes - Do "E" spaces
EBBF 3AAB10             LD      A,(CURPOS)      ; Get current position
EBC2 2F         DOSPC:  CPL                     ; Number of spaces to print to
EBC3 83                 ADD     A,E             ; Total number to print
EBC4 D2D2EB             JP      NC,NEXITM       ; TAB < Current POS(X)
EBC7 3C         ASPCS:  INC     A               ; Output A spaces
EBC8 47                 LD      B,A             ; Save number to print
EBC9 3E20               LD      A," "           ; Space
EBCB CD9BE6     SPCLP:  CALL    OUTC            ; Output character in A
EBCE 05                 DEC     B               ; Count them
EBCF C2CBEB             JP      NZ,SPCLP        ; Repeat if more
EBD2 E1         NEXITM: POP     HL              ; Restore code string address
EBD3 CD36E8             CALL    GETCHR          ; Get next character
EBD6 C326EB             JP      PRNTLP          ; More to print
```



```
EBD9 3F526564   REDO:   DEFB    "?Redo from start",CR,LF,0

EBEC 3ACD10     BADINP: LD      A,(READFG)      ; READ or INPUT?
EBEF B7                 OR      A
EBF0 C2A7E3             JP      NZ,DATSNR       ; READ - ?SN Error
EBF3 C1                 POP     BC              ; Throw away code string addr
EBF4 21D9EB             LD      HL,REDO         ; "Redo from start" message
EBF7 CD10F2             CALL    PRS             ; Output string
EBFA C3F8E4             JP      DOAGN           ; Do last INPUT again

EBFD CD7BF1     INPUT:  CALL    IDTEST          ; Test for illegal direct
EC00 7E                 LD      A,(HL)          ; Get character after "INPUT"
EC01 FE22               CP      '"'             ; Is there a prompt string?
EC03 3E00               LD      A,0             ; Clear A and leave flags
EC05 324510             LD      (CTLOFG),A      ; Enable output
EC08 C217EC             JP      NZ,NOPMPT       ; No prompt - get input
EC0B CDCFF1             CALL    QTSTR           ; Get string terminated by '"'
EC0E CD90E6             CALL    CHKSYN          ; Check for ";" after prompt
EC11 3B                 DEFB    ";"
EC12 E5                 PUSH    HL              ; Save code string address
EC13 CD13F2             CALL    PRS1            ; Output prompt string
EC16 3E                 DEFB    (LD A,n)        ; Skip "PUSH HL"
EC17 E5         NOPMPT: PUSH    HL              ; Save code string address
EC18 CDFCE4             CALL    PROMPT          ; Get input with "? " prompt
EC1B C1                 POP     BC              ; Restore code string address
EC1C DA77E8             JP      C,INPBRK        ; Break pressed - Exit
EC1F 23                 INC     HL              ; Next byte
EC20 7E                 LD      A,(HL)          ; Get it
EC21 B7                 OR      A               ; End of line?
EC22 2B                 DEC     HL              ; Back again
EC23 C5                 PUSH    BC              ; Re-save code string address
EC24 CA6FEA             JP      Z,NXTDTA        ; Yes - Find next DATA stmt
EC27 362C               LD      (HL),","        ; Store comma as separator
EC29 C331EC             JP      NXTITM          ; Get next item

EC2C E5         READ:   PUSH    HL              ; Save code string address
EC2D 2ADC10             LD      HL,(NXTDAT)     ; Next DATA statement
EC30 F6                 DEFB    (OR n)          ; Flag "READ"
EC31 AF         NXTITM: XOR     A               ; Flag "INPUT"
EC32 32CD10             LD      (READFG),A      ; Save "READ"/"INPUT" flag
EC35 E3                 EX      (SP),HL         ; Get code str' , Save pointer
EC36 C33DEC             JP      GTVLUS          ; Get values
```

```
EC39 CD90E6     NEDMOR: CALL    CHKSYN          ; Check for comma between items
EC3C 2C                 DEFB    ","
EC3D CD2DEF     GTVLUS: CALL    GETVAR          ; Get variable name
EC40 E3                 EX      (SP),HL         ; Save code str" , Get pointer
EC41 D5                 PUSH    DE              ; Save variable address
EC42 7E                 LD      A,(HL)          ; Get next "INPUT"/"DATA" byte
EC43 FE2C               CP      ","             ; Comma?
EC45 CA65EC             JP      Z,ANTVLU        ; Yes - Get another value
EC48 3ACD10             LD      A,(READFG)      ; Is it READ?
EC4B B7                 OR      A
EC4C C2D2EC             JP      NZ,FDTLP        ; Yes - Find next DATA stmt
EC4F 3E3F               LD      A,"?"           ; More INPUT needed
EC51 CD9BE6             CALL    OUTC            ; Output character
EC54 CDFCE4             CALL    PROMPT          ; Get INPUT with prompt
EC57 D1                 POP     DE              ; Variable address
EC58 C1                 POP     BC              ; Code string address
EC59 DA77E8             JP      C,INPBRK        ; Break pressed
EC5C 23                 INC     HL              ; Point to next DATA byte
EC5D 7E                 LD      A,(HL)          ; Get byte
EC5E B7                 OR      A               ; Is it zero (No input) ?
EC5F 2B                 DEC     HL              ; Back space INPUT pointer
EC60 C5                 PUSH    BC              ; Save code string address
EC61 CA6FEA             JP      Z,NXTDTA        ; Find end of buffer
EC64 D5                 PUSH    DE              ; Save variable address
EC65 3AAD10     ANTVLU: LD      A,(TYPE)        ; Check data type
EC68 B7                 OR      A               ; Is it numeric?
EC69 CA8FEC             JP      Z,INPBIN        ; Yes - Convert to binary
EC6C CD36E8             CALL    GETCHR          ; Get next character
EC6F 57                 LD      D,A             ; Save input character
EC70 47                 LD      B,A             ; Again
EC71 FE22               CP      '"'             ; Start of literal sting?
EC73 CA83EC             JP      Z,STRENT        ; Yes - Create string entry
EC76 3ACD10             LD      A,(READFG)      ; "READ" or "INPUT" ?
EC79 B7                 OR      A
EC7A 57                 LD      D,A             ; Save 00 if "INPUT"
EC7B CA80EC             JP      Z,ITMSEP        ; "INPUT" - End with 00
EC7E 163A               LD      D,":"           ; "DATA"  - End with 00 or ":"
EC80 062C       ITMSEP: LD      B,","           ; Item separator
EC82 2B                 DEC     HL              ; Back space for DTSTR
EC83 CDD2F1     STRENT: CALL    DTSTR           ; Get string terminated by D
EC86 EB                 EX      DE,HL           ; String address to DE
EC87 219AEC             LD      HL,LTSTND       ; Where to go after LETSTR
EC8A E3                 EX      (SP),HL         ; Save HL , get input pointer
EC8B D5                 PUSH    DE              ; Save address of string
EC8C C3A2EA             JP      LETSTR          ; Assign string to variable
```



```
EC8F CD36E8     INPBIN: CALL    GETCHR          ; Get next character
EC92 CD1AF9             CALL    ASCTFP          ; Convert ASCII to FP number
EC95 E3                 EX      (SP),HL         ; Save input ptr,Get var addr
EC96 CD6BF8             CALL    FPTHL           ; Move FPREG to variable
EC99 E1                 POP     HL              ; Restore input pointer
EC9A 2B         LTSTND: DEC     HL              ; DEC 'cos GETCHR INCs
EC9B CD36E8             CALL    GETCHR          ; Get next character
EC9E CAA6EC             JP      Z,MORDT         ; End of line - More needed?
ECA1 FE2C               CP      ","             ; Another value?
ECA3 C2ECEB             JP      NZ,BADINP       ; No - Bad input
ECA6 E3         MORDT:  EX      (SP),HL         ; Get code string address
ECA7 2B                 DEC     HL              ; DEC 'cos GETCHR INCs
ECA8 CD36E8             CALL    GETCHR          ; Get next character
ECAB C239EC             JP      NZ,NEDMOR       ; More needed - Get it
ECAE D1                 POP     DE              ; Restore DATA pointer
ECAF 3ACD10             LD      A,(READFG)      ; "READ" or "INPUT" ?
ECB2 B7                 OR      A
ECB3 EB                 EX      DE,HL           ; DATA pointer to HL
ECB4 C25CE8             JP      NZ,UPDATA       ; Update DATA pointer if "READ"
ECB7 D5                 PUSH    DE              ; Save code string address
ECB8 B6                 OR      (HL)            ; More input given?
ECB9 21C1EC             LD      HL,EXTIG        ; "?Extra ignored" message
ECBC C410F2             CALL    NZ,PRS          ; Output string if extra given
ECBF E1                 POP     HL              ; Restore code string address
ECC0 C9                 RET

ECC1 3F457874   EXTIG:  DEFB    "?Extra ignored",CR,LF,0

ECD2 CD7OEA     FDTLP:  CALL    DATA            ; Get next statement
ECD5 B7                 OR      A               ; End of line?
ECD6 C2EBEC             JP      NZ,FANDT        ; No - See if DATA statement
ECD9 23                 INC     HL
ECDA 7E                 LD      A,(HL)          ; End of program?
ECDB 23                 INC     HL
ECDC B6                 OR      (HL)            ; 00 00 Ends program
ECDD 1E06               LD      E,0D            ; ?0D Error
ECDF CAC1E3             JP      Z,ERROR         ; Yes - Out of DATA
ECE2 23                 INC     HL
ECE3 5E                 LD      E,(HL)          ; LSB of line number
ECE4 23                 INC     HL
ECE5 56                 LD      D,(HL)          ; MSB of line number
ECE6 EB                 EX      DE,HL
ECE7 22C910             LD      (DATLIN),HL     ; Set line of current DATA item
ECEA EB                 EX      DE,HL
ECEB CD36E8     FANDT:  CALL    GETCHR          ; Get next character
ECEE FE83               CP      ZDATA           ; "DATA" token
ECF0 C2D2EC             JP      NZ,FDTLP        ; No "DATA" - Keep looking
ECF3 C365EC             JP      ANTVLU          ; Found - Convert input
```

```
ECF6 110000     NEXT:   LD      DE,0            ; In case no index given
ECF9 C42DEF     NEXT1:  CALL    NZ,GETVAR       ; Get index address
ECFC 22CE10             LD      (BRKLIN),HL     ; Save code string address
ECFF CD56E3             CALL    BAKSTK          ; Look for "FOR" block
ED02 C2B3E3             JP      NZ,NFERR        ; No "FOR" - ?NF Error
ED05 F9                 LD      SP,HL           ; Clear nested loops
ED06 D5                 PUSH    DE              ; Save index address
ED07 7E                 LD      A,(HL)          ; Get sign of STEP
ED08 23                 INC     HL
ED09 F5                 PUSH    AF              ; Save sign of STEP
ED0A D5                 PUSH    DE              ; Save index address
ED0B CD51F8             CALL    PHLTFP          ; Move index value to FPREG
ED0E E3                 EX      (SP),HL         ; Save address of TO value
ED0F E5                 PUSH    HL              ; Save address of index
ED10 CDBEF5             CALL    ADDPHL          ; Add STEP to index value
ED13 E1                 POP     HL              ; Restore address of index
ED14 CD6BF8             CALL    FPTHL           ; Move value to index variable
ED17 E1                 POP     HL              ; Restore address of TO value
ED18 CD62F8             CALL    LOADFP          ; Move TO value to BCDE
ED1B E5                 PUSH    HL              ; Save address of line of FOR
ED1C CD8EF8             CALL    CMPNUM          ; Compare index with TO value
ED1F E1                 POP     HL              ; Restore address of line num
ED20 C1                 POP     BC              ; Address of sign of STEP
ED21 90                 SUB     B               ; Compare with expected sign
ED22 CD62F8             CALL    LOADFP          ; BC = Loop stmt,DE = Line num
ED25 CA31ED             JP      Z,KILFOR        ; Loop finished - Terminate it
ED28 EB                 EX      DE,HL           ; Loop statement line number
ED29 225C10             LD      (LINEAT),HL     ; Set loop line number
ED2C 69                 LD      L,C             ; Set code string to loop
ED2D 60                 LD      H,B
ED2E C3EEE7             JP      PUTFID          ; Put back "FOR" and continue

ED31 F9         KILFOR: LD      SP,HL           ; Remove "FOR" block
ED32 2ACE10             LD      HL,(BRKLIN)     ; Code string after "NEXT"
ED35 7E                 LD      A,(HL)          ; Get next byte in code string
ED36 FE2C               CP      ","             ; More NEXTs ?
ED38 C2F2E7             JP      NZ,RUNCNT       ; No - Do next statement
ED3B CD36E8             CALL    GETCHR          ; Position to index name
ED3E CDF9EC             CALL    NEXT1           ; Re-enter NEXT routine
                        ; < will not RETurn to here , Exit to RUNCNT or Loop >
```



```
ED41 CD5AED     GETNUM: CALL    EVAL            ; Get a numeric expression
ED44 F6         TSTNUM: DEFB    (OR n)          ; Clear carry (numeric)
ED45 37         TSTSTR: SCF                     ; Set carry (string)
ED46 3AAD10     CHKTYP: LD      A,(TYPE)        ; Check types match
ED49 8F                 ADC     A,A             ; Expected + actual
ED4A B7                 OR      A               ; Clear carry , set parity
ED4B E8                 RET     PE              ; Even parity - Types match
ED4C C3BFE3             JP      TMERR           ; Different types - Error

                        ; <<< NO REFERENCE TO HERE >>>

ED4F CD90E6             CALL    CHKSYN          ; Make sure "=" follows
ED52 B4                 DEFB    ZEQUAL          ; "="
ED53 C35AED             JP      EVAL            ; Evaluate expression

ED56 CD90E6     OPNPAR: CALL    CHKSYN          ; Make sure "(" follows
ED59 28                 DEFB    "("
ED5A 2B         EVAL:   DEC     HL              ; Evaluate expression & save
ED5B 1600               LD      D,0             ; Precedence value
ED5D D5         EVAL1:  PUSH    DE              ; Save precedence
ED5E 0E01               LD      C,1
ED60 CD8AE3             CALL    CHKSTK          ; Check for 1 level of stack
ED63 CDD1ED             CALL    OPRND           ; Get next expression value
ED66 22D010     EVAL2:  LD      (NXTOPR),HL     ; Save address of next operator
ED69 2AD010     EVAL3:  LD      HL,(NXTOPR)     ; Restore address of next opr
ED6C C1                 POP     BC              ; Precedence value and operator
ED6D 78                 LD      A,B             ; Get precedence value
ED6E FE78               CP      78H             ; "AND" or "OR" ?
ED70 D444ED             CALL    NC,TSTNUM       ; No - Make sure it's a number
ED73 7E                 LD      A,(HL)          ; Get next operator / function
ED74 1600               LD      D,0             ; Clear Last relation
ED76 D6B3       RLTLP:  SUB     ZGTR            ; ">" Token
ED78 DA92ED             JP      C,FOPRND        ; + - * / ^ AND OR - Test it
ED7B FE03               CP      ZLTH+1-ZGTR     ; < = >
ED7D D292ED             JP      NC,FOPRND       ; Function - Call it
ED80 FE01               CP      ZEQUAL-ZGTR     ; "="
ED82 17                 RLA                     ; <- Test for legal
ED83 AA                 XOR     D               ; <- combinations of < = >
ED84 BA                 CP      D               ; <- by combining last token
ED85 57                 LD      D,A             ; <- with current one
ED86 DAADE3             JP      C,SNERR         ; Error if "<<" "==" or ">>"
ED89 22C510             LD      (CUROPR),HL     ; Save address of current token
ED8C CD36E8             CALL    GETCHR          ; Get next character
ED8F C376ED             JP      RLTLP           ; Treat the two as one
```

```
EDD1 AF        OPRND:  XOR     A               ; Get operand routine
EDD2 32AD10            LD      (TYPE),A        ; Set numeric expected
EDD5 CD36E8            CALL    GETCHR          ; Get next character
EDD8 1E24              LD      E,MO            ; ?MO Error
EDDA CAC1E3            JP      Z,ERROR         ; No operand - Error
EDDD DA1AF9            JP      C,ASCTFP        ; Number - Get value
EDE0 CD77E9            CALL    CHKLTR          ; See if a letter
EDE3 D222EE            JP      NC,CONVAR       ; Letter - Find variable
EDE6 FEAC              CP      ZPLUS           ; "+" Token ?
EDE9 CAD1ED            JP      Z,OPRND         ; Yes - Look for operand
EDEB FE2E              CP      "."             ; "." ?
EDED CA1AF9            JP      Z,ASCTFP        ; Yes - Create FP number
EDF0 FEAD              CP      ZMINUS          ; "-" Token ?
EDF2 CA11EE            JP      Z,MINUS         ; Yes - Do minus
EDF5 FE22              CP      '"'             ; Literal string ?
EDF7 CACFF1            JP      Z,QTSTR         ; Get string terminated by '"'
EDFA FEAA              CP      ZNOT            ; "NOT" Token ?
EDFC CA08EF            JP      Z,EVNOT         ; Yes - Eval NOT expression
EDFF FEA7              CP      ZFN             ; "FN" Token ?
EE01 CA33F1            JP      Z,DOFN          ; Yes - Do FN routine
EE04 D6B6              SUB     ZSGN            ; Is it a function?
EE06 D233EE            JP      NC,FNOFST       ; Yes - Evaluate function
EE09 CD56ED    EVLPAR: CALL    OPNPAR          ; Evaluate expression in "()"
EE0C CD90E6            CALL    CHKSYN          ; Make sure ")" follows
EE0F 29                DEFB    ")"
EE10 C9                RET

EE11 167D      MINUS:  LD      D,7DH           ; "-" precedence
EE13 CD5DED            CALL    EVAL1           ; Evaluate until prec' break
EE16 2AD010            LD      HL,(NXTOPR)     ; Get next operator address
EE19 E5                PUSH    HL              ; Save next operator address
EE1A CD3CF8            CALL    INVSGN          ; Negate value
EE1D CD44ED            CALL    TSTNUM          ; Make sure it's a number
EE20 E1                POP     HL              ; Restore next operator address
EE21 C9                RET

EE22 CD2DEF    CONVAR: CALL    GETVAR          ; Get variable address to DE
EE25 E5                PUSH    HL              ; Save code string address
EE26 EB                EX      DE,HL           ; Variable address to HL
EE27 22E410            LD      (FPREG),HL      ; Save address of variable
EE2A 3AAD10            LD      A,(TYPE)        ; Get type
EE2D B7                OR      A               ; Numeric?
EE2E CC51F8            CALL    Z,PHLTFP        ; Yes - Move contents to FPREG
EE31 E1                POP     HL              ; Restore code string address
EE32 C9                RET
```



```
ED92 7A        FOPRND: LD      A,D             ; < = > found ?
ED93 B7                OR      A
ED94 C2A8EE            JP      NZ,TSTRED       ; Yes - Test for reduction
ED97 7E                LD      A,(HL)          ; Get operator token
ED98 22C510            LD      (CUROPR),HL     ; Save operator address
ED9B D6AC              SUB     ZPLUS           ; Operator or function?
ED9D D8                RET     C               ; Neither - Exit
ED9E FE07              CP      ZOR+1-ZPLUS     ; Is it + - * / ^ AND OR ?
EDA0 D0                RET     NC              ; No - Exit
EDA1 5F                LD      E,A             ; Coded operator
EDA2 3AAD10            LD      A,(TYPE)        ; Get data type
EDA5 3D                DEC     A               ; FF = numeric , 00 = string
EDA6 B3                OR      E               ; Combine with coded operator
EDA7 7B                LD      A,E             ; Get coded operator
EDA8 CA06F3            JP      Z,CONCAT        ; String concatenation
EDAB 07                RLCA                    ; Times 2
EDAC 83                ADD     A,E             ; Times 3
EDAD 5F                LD      E,A             ; To DE (D is 0)
EDAE 21A4E2            LD      HL,PRITAB       ; Precedence table
EDB1 19                ADD     HL,DE           ; To the operator concerned
EDB2 78                LD      A,B             ; Last operator precedence
EDB3 56                LD      D,(HL)          ; Get evaluation precedence
EDB4 BA                CP      D               ; Compare with eval precedence
EDB5 D0                RET     NC              ; Exit if higher precedence
EDB6 23                INC     HL              ; Point to routine address
EDB7 CD44ED            CALL    TSTNUM          ; Make sure it's a number

EDBA C5        STKTHS: PUSH    BC              ; Save last precedence & token
EDBB 0169ED            LD      BC,EVAL3        ; Where to go on prec' break
EDBE C5                PUSH    BC              ; Save on stack for return
EDBF 43                LD      B,E             ; Save operator
EDC0 4A                LD      C,D             ; Save precedence
EDC1 CD44F8            CALL    STAKFP          ; Move value to stack
EDC4 5B                LD      E,B             ; Restore operator
EDC5 51                LD      D,C             ; Restore precedence
EDC6 4E                LD      C,(HL)          ; Get LSB of routine address
EDC7 23                INC     HL
EDC8 46                LD      B,(HL)          ; Get MSB of routine address
EDC9 23                INC     HL
EDCA C5                PUSH    BC              ; Save routine address
EDCB 2AC510            LD      HL,(CUROPR)     ; Address of current operator
EDCE C35DED            JP      EVAL1           ; Loop until prec' break
```

```
EE33 0600      FNOFST: LD      B,0             ; Get address of function
EE35 07                RLCA                    ; Double function offset
EE36 4F                LD      C,A             ; BC = Offset in function table
EE37 C5                PUSH    BC              ; Save adjusted token value
EE38 CD36E8            CALL    GETCHR          ; Get next character
EE3B 79                LD      A,C             ; Get adjusted token value
EE3C FE22              CP      2*(ZPOINT-ZSGN) ; Adjusted "POINT" token?
EE3E CA79FF            JP      Z,POINTB        ; Yes - Do "POINT" (not POINTB)
EE41 FE2D              CP      2*(ZLEFT-ZSGN)-1; Adj' LEFT$,RIGHT$ or MID$ ?
EE43 DA5FEE            JP      C,FNVAL         ; No - Do function
EE46 CD56ED            CALL    OPNPAR          ; Evaluate expression  (X,...
EE49 CD90E6            CALL    CHKSYN          ; Make sure "," follows
EE4C 2C                DEFB    ","
EE4D CD45ED            CALL    TSTSTR          ; Make sure it's a string
EE50 EB                EX      DE,HL           ; Save code string address
EE51 2AE410            LD      HL,(FPREG)      ; Get address of string
EE54 E3                EX      (SP),HL         ; Save address of string
EE55 E5                PUSH    HL              ; Save adjusted token value
EE56 EB                EX      DE,HL           ; Restore code string address
EE57 CD84F4            CALL    GETINT          ; Get integer 0-255
EE5A EB                EX      DE,HL           ; Save code string address
EE5B E3                EX      (SP),HL         ; Save integer,HL = adj' token
EE5C C367EE            JP      GOFUNC          ; Jump to string function

EE5F CD09EE    FNVAL:  CALL    EVLPAR          ; Evaluate expression
EE62 E3                EX      (SP),HL         ; HL = Adjusted token value
EE63 111DEE            LD      DE,RETNUM       ; Return number from function
EE66 D5                PUSH    DE              ; Save on stack
EE67 010FE1    GOFUNC: LD      BC,FNCTAB       ; Function routine addresses
EE6A 09                ADD     HL,BC           ; Point to right address
EE6B 4E                LD      C,(HL)          ; Get LSB of address
EE6C 23                INC     HL
EE6D 66                LD      H,(HL)          ; Get MSB of address
EE6E 69                LD      L,C             ; Address to HL
EE6F E9                JP      (HL)            ; Jump to function

EE70 15        SGNEXP: DEC     D               ; Dec to flag negative exponent
EE71 FEAD              CP      ZMINUS          ; "-" token ?
EE73 C8                RET     Z               ; Yes - Return
EE74 FE2D              CP      "-"             ; "-" ASCII ?
EE76 C8                RET     Z               ; Yes - Return
EE77 14                INC     D               ; Inc to flag positive exponent
EE78 FE2B              CP      "+"             ; "+" ASCII ?
EE7A C8                RET     Z               ; Yes - Return
EE7B FEAC              CP      ZPLUS           ; "+" token ?
EE7D C8                RET     Z               ; Yes - Return
EE7E 2B                DEC     HL              ; DEC 'cos GETCHR INCs
EE7F C9                RET                     ; Return "NZ"
```



```
EE80 F6        POR:    DEFB    (OR n)          ; Flag "OR"
EE81 AF        PAND:   XOR     A               ; Flag "AND"
EE82 F5                PUSH    AF              ; Save "AND" / "OR" flag
EE83 CD44ED            CALL    TSTNUM          ; Make sure it's a number
EE86 CD8BE9            CALL    DEINT           ; Get integer -32768 to 32767
EE89 F1                POP     AF              ; Restore "AND" / "OR" flag
EE8A EB                EX      DE,HL           ; <- Get last
EE8B C1                POP     BC              ; <-  value
EE8C E3                EX      (SP),HL         ; <-  from
EE8D EB                EX      DE,HL           ; <-  stack
EE8E CD54F8            CALL    FPBCDE          ; Move last value to FPREG
EE91 F5                PUSH    AF              ; Save "AND" / "OR" flag
EE92 CD8BE9            CALL    DEINT           ; Get integer -32768 to 32767
EE95 F1                POP     AF              ; Restore "AND" / "OR" flag
EE96 C1                POP     BC              ; Get value
EE97 79                LD      A,C             ; Get LSB
EE98 21F1F0            LD      HL,ACPASS       ; Address of save AC as current
EE9B C2A3EE            JP      NZ,POR1         ; Jump if OR
EE9E A3                AND     E               ; "AND" LSBs
EE9F 4F                LD      C,A             ; Save LSB
EEA0 78                LD      A,B             ; Get MBS
EEA1 A2                AND     D               ; "AND" MSBs
EEA2 E9                JP      (HL)            ; Save AC as current (ACPASS)

EEA3 B3        POR1:   OR      E               ; "OR" LSBs
EEA4 4F                LD      C,A             ; Save LSB
EEA5 78                LD      A,B             ; Get MSB
EEA6 B2                OR      D               ; "OR" MSBs
EEA7 E9                JP      (HL)            ; Save AC as current (ACPASS)

EEA8 21BAEE    TSTRED: LD      HL,CMPLOG       ; Logical compare routine
EEAB 3AAD10            LD      A,(TYPE)        ; Get data type
EEAE 1F                RRA                     ; Carry set = string
EEAF 7A                LD      A,D             ; Get last precedence value
EEB0 17                RLA                     ; Times 2 plus carry
EEB1 5F                LD      E,A             ; To E
EEB2 1664              LD      D,64H           ; Relational precedence
EEB4 78                LD      A,B             ; Get current precedence
EEB5 BA                CP      D               ; Compare with last
EEB6 D0                RET     NC              ; Eval if last was rel' or log'
EEB7 C3BAED            JP      STKTHS          ; Stack this one and get next
```

```
EEBA BCEE       CMPLOG: DEFW    CMPLG1          ; Compare two values / strings
EEBC 79         CMPLG1: LD      A,C             ; Get data type
EEBD B7                 OR      A
EEBE 1F                 RRA
EEBF C1                 POP     BC              ; Get last expression to BCDE
EEC0 D1                 POP     DE
EEC1 F5                 PUSH    AF              ; Save status
EEC2 CD46ED             CALL    CHKTYP          ; Check that types match
EEC5 21FEEE             LD      HL,CMPRES       ; Result to comparison
EEC8 E5                 PUSH    HL              ; Save for RETurn
EEC9 CA8EF8             JP      Z,CMPNUM        ; Compare values if numeric
EECC AF                 XOR     A               ; Compare two strings
EECD 32AD10             LD      (TYPE),A        ; Set type to numeric
EED0 D5                 PUSH    DE              ; Save string name
EED1 CD53F3             CALL    GSTRCU          ; Get current string
EED4 7E                 LD      A,(HL)          ; Get length of string
EED5 23                 INC     HL
EED6 23                 INC     HL
EED7 4E                 LD      C,(HL)          ; Get LSB of address
EED8 23                 INC     HL
EED9 46                 LD      B,(HL)          ; Get MSB of address
EEDA D1                 POP     DE              ; Restore string name
EEDB C5                 PUSH    BC              ; Save address of string
EEDC F5                 PUSH    AF              ; Save length of string
EEDD CD57F3             CALL    GSTRDE          ; Get second string
EEE0 CD62F8             CALL    LOADFP          ; Get address of second string
EEE3 F1                 POP     AF              ; Restore length of string 1
EEE4 57                 LD      D,A             ; Length to D
EEE5 E1                 POP     HL              ; Restore address of string 1
EEE6 7B         CMPSTR: LD      A,E             ; Bytes of string 2 to do
EEE7 B2                 OR      D               ; Bytes of string 1 to do
EEE8 C8                 RET     Z               ; Exit if all bytes compared
EEE9 7A                 LD      A,D             ; Get bytes of string 1 to do
EEEA D601               SUB     1
EEEC D8                 RET     C               ; Exit if end of string 1
EEED AF                 XOR     A
EEEE BB                 CP      E               ; Bytes of string 2 to do
EEEF 3C                 INC     A
EEF0 D0                 RET     NC              ; Exit if end of string 2
EEF1 15                 DEC     D               ; Count bytes in string 1
EEF2 1D                 DEC     E               ; Count bytes in string 2
EEF3 0A                 LD      A,(BC)          ; Byte in string 2
EEF4 BE                 CP      (HL)            ; Compare to byte in string 1
EEF5 23                 INC     HL              ; Move up string 1
EEF6 03                 INC     BC              ; Move up string 2
EEF7 CAE6EE             JP      Z,CMPSTR        ; Same - Try next bytes
EEFA 3F                 CCF                     ; Flag difference (">" or "<")
EEFB C31EF8             JP      FLGDIF          ; "<" gives -1 , ">" gives +1

EEFE 3C         CMPRES: INC     A               ; Increment current value
EEFF 8F                 ADC     A,A             ; Double plus carry
EF00 C1                 POP     BC              ; Get other value
EF01 A0                 AND     B               ; Combine them
EF02 C6FF               ADD     A,-1            ; Carry set if different
EF04 9F                 SBC     A,A             ; 00 - Equal , FF - Different
EF05 C325F8             JP      FLGREL          ; Set current value & continue
```



```
EF08 165A       EVNOT:  LD      D,5AH           ; Precedence value for "NOT"
EF0A CD5DED             CALL    EVAL1           ; Eval until precedence break
EF0D CD44ED             CALL    TSTNUM          ; Make sure it's a number
EF10 CD8BE9             CALL    DEINT           ; Get integer -32768 - 32767
EF13 7B                 LD      A,E             ; Get LSB
EF14 2F                 CPL                     ; Invert LSB
EF15 4F                 LD      C,A             ; Save "NOT" of LSB
EF16 7A                 LD      A,D             ; Get MSB
EF17 2F                 CPL                     ; Invert MSB
EF18 CDF1F0             CALL    ACPASS          ; Save AC as current
EF1B C1                 POP     BC              ; Clean up stack
EF1C C369ED             JP      EVAL3           ; Continue evaluation

EF1F 2B         DIMRET: DEC     HL              ; DEC 'cos GETCHR INCs
EF20 CD36E8             CALL    GETCHR          ; Get next character
EF23 C8                 RET     Z               ; End of DIM statement
EF24 CD90E6             CALL    CHKSYN          ; Make sure "," follows
EF27 2C                 DEFB    ","
EF28 011FEF     DIM:    LD      BC,DIMRET       ; Return to "DIMRET"
EF2B C5                 PUSH    BC              ; Save on stack
EF2C F6                 DEFB    (OR n)          ; Flag "Create" variable
EF2D AF         GETVAR: XOR     A               ; Find variable address,to DE
EF2E 32AC10             LD      (LCRFLG),A      ; Set locate / create flag
EF31 46                 LD      B,(HL)          ; Get First byte of name
EF32 CD77E9     GTFNAM: CALL    CHKLTR          ; See if a letter
EF35 DAADE3             JP      C,SNERR         ; ?SN Error if not a letter
EF38 AF                 XOR     A
EF39 4F                 LD      C,A             ; Clear second byte of name
EF3A 32AD10             LD      (TYPE),A        ; Set type to numeric
EF3D CD36E8             CALL    GETCHR          ; Get next character
EF40 DA49EF             JP      C,SVNAM2        ; Numeric - Save in name
EF43 CD77E9             CALL    CHKLTR          ; See if a letter
EF46 DA56EF             JP      C,CHARTY        ; Not a letter - Check type
EF49 4F         SVNAM2: LD      C,A             ; Save second byte of name
EF4A CD36E8     ENDNAM: CALL    GETCHR          ; Get next character
EF4D DA4AEF             JP      C,ENDNAM        ; Numeric - Get another
EF50 CD77E9             CALL    CHKLTR          ; See if a letter
EF53 D24AEF             JP      NC,ENDNAM       ; Letter - Get another
EF56 D624       CHARTY: SUB     "$"             ; String variable?
EF58 C265EF             JP      NZ,NOTSTR       ; No - Numeric variable
EF5B 3C                 INC     A               ; A = 1 (string type)
EF5C 32AD10             LD      (TYPE),A        ; Set type to string
EF5F 0F                 RRCA                    ; A = 80H , Flag for string
EF60 81                 ADD     A,C             ; 2nd byte of name has bit 7 on
EF61 4F                 LD      C,A             ; Resave second byte on name
EF62 CD36E8             CALL    GETCHR          ; Get next character
EF65 3ACB10     NOTSTR: LD      A,(FORFLG)      ; Array name needed ?
EF68 3D                 DEC     A
EF69 CA12F0             JP      Z,ARLDSV        ; Yes - Get array name
EF6C F275EF             JP      P,NSCFOR        ; No array with "FOR" or "FN"
EF6F 7E                 LD      A,(HL)          ; Get byte again
EF70 D628               SUB     "("             ; Subscripted variable?
EF72 CAEAEF             JP      Z,SBSCPT        ; Yes - Sort out subscript
```

```
EF75 AF       NSCFOR: XOR     A               ; Simple variable
EF76 32CB10           LD      (FORFLG),A      ; Clear "FOR" flag
EF79 E5               PUSH    HL              ; Save code string address
EF7A 50               LD      D,B             ; DE = Variable name to find
EF7B 59               LD      E,C
EF7C 2ADE10           LD      HL,(FNRGNM)     ; FN argument name
EF7F CD8AE6           CALL    CPDEHL          ; Is it the FN argument?
EF82 11E010           LD      DE,FNARG        ; Point to argument value
EF85 CA54F7           JP      Z,POPHRT        ; Yes - Return FN argument value
EF88 2AD810           LD      HL,(VAREND)     ; End of variables
EF8B EB               EX      DE,HL           ; Address of end of search
EF8C 2AD610           LD      HL,(PROGND)     ; Start of variables address
EF8F CD8AE6   FNDVAR: CALL    CPDEHL          ; End of variable list table?
EF92 CAA8EF           JP      Z,CFEVAL        ; Yes - Called from EVAL?
EF95 79               LD      A,C             ; Get second byte of name
EF96 96               SUB     (HL)            ; Compare with name in list
EF97 23               INC     HL              ; Move on to first byte
EF98 C29DEF           JP      NZ,FNTHR        ; Different - Find another
EF9B 78               LD      A,B             ; Get first byte of name
EF9C 96               SUB     (HL)            ; Compare with name in list
EF9D 23       FNTHR:  INC     HL              ; Move on to LSB of value
EF9E CADCEF           JP      Z,RETADR        ; Found - Return address
EFA1 23               INC     HL              ; <- Skip
EFA2 23               INC     HL              ; <- over
EFA3 23               INC     HL              ; <- F.P.
EFA4 23               INC     HL              ; <- value
EFA5 C38FEF           JP      FNDVAR          ; Keep looking

EFA8 E1       CFEVAL: POP     HL              ; Restore code string address
EFA9 E3               EX      (SP),HL         ; Get return address
EFAA D5               PUSH    DE              ; Save address of variable
EFAB 1125EE           LD      DE,FRMEVL       ; Return address in EVAL
EFAE CD8AE6           CALL    CPDEHL          ; Called from EVAL ?
EFB1 D1               POP     DE              ; Restore address of variable
EFB2 CADFEF           JP      Z,RETNUL        ; Yes - Return null variable
EFB5 E3               EX      (SP),HL         ; Put back return
EFB6 E5               PUSH    HL              ; Save code string address
EFB7 C5               PUSH    BC              ; Save variable name
EFB8 010600           LD      BC,6            ; 2 byte name plus 4 byte data
EFBB 2ADA10           LD      HL,(ARREND)     ; End of arrays
EFBE E5               PUSH    HL              ; Save end of arrays
EFBF 09               ADD     HL,BC           ; Move up 6 bytes
EFC0 C1               POP     BC              ; Source address in BC
EFC1 E5               PUSH    HL              ; Save new end address
EFC2 CD79E3           CALL    MOVUP           ; Move arrays up
EFC5 E1               POP     HL              ; Restore new end address
EFC6 22DA10           LD      (ARREND),HL     ; Set new end address
EFC9 60               LD      H,B             ; End of variables to HL
EFCA 69               LD      L,C
EFCB 22D810           LD      (VAREND),HL     ; Set new end address
```



```
EFCE 2B       ZEROLP: DEC     HL              ; Back through to zero variable
EFCF 3600             LD      (HL),0          ; Zero byte in variable
EFD1 CD8AE6           CALL    CPDEHL          ; Done them all?
EFD4 C2CEEF           JP      NZ,ZEROLP       ; No - Keep on going
EFD7 D1               POP     DE              ; Get variable name
EFD8 73               LD      (HL),E          ; Store second character
EFD9 23               INC     HL
EFDA 72               LD      (HL),D          ; Store first character
EFDB 23               INC     HL
EFDC EB       RETADR: EX      DE,HL           ; Address of variable in DE
EFDD E1               POP     HL              ; Restore code string address
EFDE C9               RET

EFDF 32E710   RETNUL: LD      (FPEXP),A       ; Set result to zero
EFE2 214AE3           LD      HL,ZERBYT       ; Also set a null string
EFE5 22E410           LD      (FPREG),HL      ; Save for EVAL
EFE8 E1               POP     HL              ; Restore code string address
EFE9 C9               RET

EFEA E5       SBSCPT: PUSH    HL              ; Save code string address
EFEB 2AAC10           LD      HL,(LCRFLG)     ; Locate/Create and Type
EFEE E3               EX      (SP),HL         ; Save and get code string
EFEF 57               LD      D,A             ; Zero number of dimensions
EFF0 D5       SCPTLP: PUSH    DE              ; Save number of dimensions
EFF1 C5               PUSH    BC              ; Save array name
EFF2 CD7FE9           CALL    FPSINT          ; Get subscript (0-32767)
EFF5 C1               POP     BC              ; Restore array name
EFF6 F1               POP     AF              ; Get number of dimensions
EFF7 EB               EX      DE,HL
EFF8 E3               EX      (SP),HL         ; Save subscript value
EFF9 E5               PUSH    HL              ; Save LCRFLG and TYPE
EFFA EB               EX      DE,HL
EFFB 3C               INC     A               ; Count dimensions
EFFC 57               LD      D,A             ; Save in D
EFFD 7E               LD      A,(HL)          ; Get next byte in code string
EFFE FE2C             CP      ","             ; Comma (more to come)?
F000 CAF0EF           JP      Z,SCPTLP        ; Yes - More subscripts
F003 CD90E6           CALL    CHKSYN          ; Make sure ")" follows
F006 29               DEFB    ")"
F007 22D010           LD      (NXTOPR),HL     ; Save code string address
F00A E1               POP     HL              ; Get LCRFLG and TYPE
F00B 22AC10           LD      (LCRFLG),HL     ; Restore Locate/create & type
F00E 1E00             LD      E,0             ; Flag not CSAVE* or CLOAD*
F010 D5               PUSH    DE              ; Save number of dimensions (D)
F011 11       DEFB    (LD DE,nn)              ; Skip "PUSH HL" and "PUSH AF"
```

BOOK REVIEWS

By Rory O'Farrell

The short interval between the arrival of the last two 80-BUS Newses, and the assumed deadline for the Christmas issue means that the following notes are thrown together in somewhat of a hurry to get the disc in the post. As most microcomputer enthusiasts, I now find it difficult to use an ordinary typewriter (even, dare I say it, an IBM). These notes are prepared using a text editor, in this case WordStar, and a disc with the files on it posted off to the Editor. The editorial discretion is exercised, the disc output is fed through his Qume, and camera ready copy is produced from what is substantially my original typing.

The tools to do this are Word processor or text Editor programs. Nearly every vendor of software can supply at least one of these, sometimes a number of different programs. Each such program has its adherents and as the numbers of adherents grow, so the book publishers reflect this by publishing books on particular word processing programs.

WordStar is one of the most widely used text editors. This can clearly be seen in the number of books available for it. Two of these have recently come to my attention.

WordStar Made Easy by Walter Ettlin, published Osborne/McGraw-Hill is an easy to read "how to use it" manual on WordStar. It is certainly streets ahead of the manual supplied with the program itself. [Ed. - I wonder if Rory is referring to WordStar 3.0, as the 3.3 manual is considerably better and is also typeset?] This book is typeset, and well laid out. The subject is broken up into 18 sections, which conduct the reader logically from turning on the computer and getting WS running to complicated formatting and printing using files of data.

An example of this might be the preparation of circular letters to clients, including a reference by name within the body of the letter to their wife. The phrase "and we look forward to seeing you and your wife nnnnn at our Christmas party" will force a line to different lengths, depending on the length of the name. The line with "Ann" in it will be much shorter than the line with "Theodora". If the body of the letter is right justified, then it will be rare for the right margin of the line with the insertion to come to the correct place on the paper. WS has an optional program called MailMerge, which will print such a letter, extracting the variables from a separate file. It will also reformat the body of the letter to accomodate different length insertions, so that the above discussed problem does not arise. Having read Ettlin, one would be well able to start work on such a project, even had one never used a Word processor program before.

Another WordStar book is **"WordStar and CP/M made Easy"** by Lee, published John Wiley and Sons.

This book has been written and typeset using WordStar, the typesetting having been done on what appears to be a daisy-wheel typewriter. In consequence, it is tiring to read, both as the typestyle is typewriterish, and also quite small (about 7pt. according to my measurement). The typewriter is all right for short documents, but most book publishers nowadays seem to think that it will do for longer ones as well. They loose sight of the vast amount of thought and effort that has gone into the design of typefaces, and the contribution to legibility that these make. As the source file of the book is on disk, why don't they feed it into a compositor rather than a typewriter?

That said, this book is more a advanced and fuller treatment of WordStar than the foregoing. For most users, it will not be relevant, but if you are trying to do something really tricky, then I think this is the book for you. I would not suggest that you set any great store by his treatment of CP/M. He deals with CP/M in three chapters at the end of the book, discussing the syntax of the commands, the use of the intrinsic commands and of the supplied transient commands. It is very much an introduction - aimed, I imagine at the person who wishes to use WordStar, and doesn't want to do much more on the computer than that.

So there they are: Ettlin if you have never used a text editor before, Lee if you have and want to get really tricky with WordStar.

If you are one of the many contemplating the plunge into CP/M, then you might do worse than invest in "**CP/M and the Personal Computer**" by Dwyer and Critchfield, published Addison Wesley.

This 490 page quarto book is an easy to read introduction to CP/M, both to the intricacies of CP/M itself and to many of the major programs which run on it. It gives a readable introduction to CP/M, ultimately getting deeply involved, writing on assembly language programming for interfacing with CP/M, and discussing the CP/M supplied transients in detail. It goes further, surveying some Word processing programs, a data base management system (dBASE II), an accounting package (Peachtree), and spreadsheets. Its survey of these might be sufficient justification for you to purchase it, although I stress that the surveys are not (and do not purport to be) encyclopediac. They consist of a general introduction to the area, and a more detailed discussion/example of the use of one of the typical programs of the area.

There is an interesting discussion on MBASIC and BASCOM, and another on C, which might whet your appetite to go further down that road. I for one intend to stick to Pascal! The section on Assembly language suffers from 8080 mnemonics, but is advanced enough to deal with adding customised I/O drivers to the BIOS. See it, particularly if you are not yet into CP/M - it might be just what you need.

Another book on the same lines is **CP/M - The Software Bus** by Clarke, Eaton and Powys-Lybbe, published Sigma Technical Press (dist. John Wiley).

This book is written by three long time members of the UK CP/M User Group. That is good and bad. Good in that they are well acquainted with CP/M and its intricacies, bad in that they often overlook, through utter familiarity, little matters which would prove awkward for the tyro. It surveys CP/M, comparing and contrasting versions 1.3 up to 3.0 and CP/M 86, dealing also with MP/M. It gives a good discussion of the usual CP/M transients, well illustrated with examples, but occasionally omitting a little nugget of information which I'm sure they knew, and which can be a lifesaver. For example, a SUBMIT file can have parameters $1....$9. Undocumented, but working nevertheless, is $0, which refers to the name of the SUBMITted file itself. This can save a lot of trouble, and I'm surprised that these authors did not draw our attention to it.

Well described are the different assemblers, both from Digital Research and other sources, including the various User Group contributions. The high level languages get a good treatment, CBASIC, MBASIC and BASCOM, Pascal MT, CIS COBOL and FORTRAN all being covered in survey. More space is given to the more popular of these, so MBASIC and BASCOM are fairly fully covered. Similarly, the various editors/Word processors, including ED. I have been asked "if ED is so bad, why does every book on CP/M treat on it in detail?".

There are two reasons. First of all, it comes free with CP/M, and as yet DRI have not seen fit to supply a free WordStar or GEMPEN. Secondly, it is not slow, and (rareity of rareities) can be driven from a SUBMIT file for most of its functions. I know no other editor which will allow that. Unimpressed cries of "So?". Let me tell you a story.

It was a dark and stormy night - sorry, I forgot I wasn't Snoopy. Some time ago, I needed to transfer a suite of programs and data from an Osborne. This was due to the ridiculous disc capacity of that machine, making it unable to cope with the amount of information without major revision of the program. So, as a simple expedient, we thought, "Send it all down to Rory's GM813, which has 800k drives attached, and let it get on with it". We (naturally) did not have a modem transfer program, and even if we had, we could not at that time access the status port on the Osborne RS232 line, which modem programs always require. So we broke the programs and data up into 4k blocks, converted to Intel Hex format, and pipped them out through the PTP:. The GM813 received them through the RDR:. 25 files! We did the entire transmission under a SUBMIT file. Similarly at the receiving end, we used another SUBMIT file. We discovered that PIP and the PTP: add 40 nulls to the start of each transmission, and we had to get these off before LOADing the files back to using the [H] option will clear the nulls), we added a few lines to the SUBMIT file to call up ED, strip off 40 nulls from the file in question, exit ED and LOAD the file. Including transmission, the whole job took about one hour, which we spent watching 'Yes Minister' on TV, keeping an eye on the computers every ten minutes or so.

Anyway, enough of a digression. Back to our muttons. This book also deals with assembly language and CP/M. I should stress that it doesn't do this as fully as Miller (reviewed 80-BUS News, Vol 2, No. 4), nor for that matter does the last book, but well enough (apart from 8080 mnemonics) to get you started in the right direction. It has an interesting final chapter giving patches to cure some of the bugs in the standard CP/M transients (the patch for PIP is already available in the HR Utilities disc). If you know a bit about CP/M, and are a computer enthusiast, then this is probably the book for you.

Z80 Assembly Language Subroutines by Leventhal and Saville, published Osborne/McGraw-Hill is of the usual high standard of Leventhal's works. It treats Assembly language programming on the assumption that the reader will have had experience in some form of assembly language - it is by way of a conversion manual to the Z80 from other machines. The authors deal with methods of getting the addressing modes not supported in the Z80, which makes for very interesting reading. They include a discussion of common programming errors, and 58 fully commented listings of useful subroutines such as HEX/ASCII conversion, array addressing etc. These are fully commented, with details of the calling conventions, registers used, and times taken. If only we had had this book four years ago! As an added bonus, it uses Z80 mnemonics!

SuperCalc! The Book by D.H. Beil, published Reston (distr. Prentice-Hall) is concerned with taking the reader through the facilities of SuperCalc. It is very clear and easy to follow, although you have to follow the route the author maps out, for easiest reading. It contains a full description of the SuperCalc commands, with detailed examples. I'm using it in conjunction with SC - SC has a good help level, but often one needs the explanation expanded somewhat. For this I refer to Beil. It also contains a full bibliography of articles and books on SuperCalc and other spreadsheets.

The editor, when he's not execising his discretion, keeps the mailing list for this learned Journal on dBASE II. How this was ever managed is a matter for considerable conjecture, as the dBASE II manual must win prizes for being THE MOST DIFFICULT manual to read of all application programs. It consists of some ten or eleven sections, all in indexed tabs. It starts off with a section telling you all the new facilities and alterations added to this version, completely forgetting that you have just purchased it and don't know anything about the old features. The actual start of the manual seems to be down some four or five sections. It needs a map, or better still, a large label saing "The Secrets of dBASE II revealed! START HERE"

If you look at the software pricelists, you will notice that there are two entries for dBASE II. One is approximately £20 dearer than the other. This is a version supplied with the standard manual, and a further manual called **Everyman's Database Primer** by Robert Byers, published Ashton-Tate (dist. Prentice-Hall, I think) at £12. This book is 300 pages, quarto sized, and typeset (thank God). It is a very readable book, taking all its examples from dBASE II usage. Using a fairly simple example, it proceeds to show how a database can access information, and manipulate it into a desired form. Having read it, I began to feel that I might achieve something with dBASE¢II at last.

A slightly more advanced book on the same subject is **dBASE II User's Guide** by Adam Green, published Prentice-Hall at £24.65.
This is another book in the "get your own daisywheel to do the typesetting" school. It is spiral bound, consisting of 150 pages, quarto. It is not as readable as Byers, though there is a slightly fuller treatment of dBASE II in it. Its layout does not help easy reading - it could do with the pages of type being shrunk some 10-15% before printing, to compact them slightly. Better yet, it could be properly typeset.

Of the two of these books, I feel that Byers is the easier to read, and the better value. Neither of these books does away with the need to consult the dBASE II manual for elaboration, and neither of them is the last word on this program. What it really needs is a patch to call up a well written HELP facility, such as exists in WordStar and SuperCalc. Why don't more software authors use such a method? [Ed. - again, I wonder if Rory has not got the latest dBASE as 2.4 does have a HELP facility.]

PASCAL

By Rory O'Farrell

Of recent months, my mantle as advocate of Pascal seems to have been assumed by Dr. Dark, who has been prolix in his advocacy of it. I've recently purchased a copy of ProPascal, a full native code compiler for the Z80, and have seen and used a copy of the JRT Pascal V3. In the next issue of the 80-BUS news, I hope to review the ProPascal in detail. I do not intend to review the JRT Pascal, as coming to terms with the full facilities of an ISO standard Pascal is quite sufficient work for anyone who has not previously used one, but I will attempt to persuade a friend who has the JRT Pascal to review it. Perhaps we will be able to work on a joint article to describe these two programs.

AUNT AGATHA'S AGONY COLUMN

By David Parkinson

EPROM PROGRAMMERS

This issue I start with a few comments about the Bits & PCs EPROM programmer that appeared as the Gemini G808. I bought one shortly after they first appeared, and have used it quite a bit since then. Some time ago I started to have trouble programming some of my 2716s, (I never used it for 2708s), and even after overprogramming them several times some bytes were still not programmed. I put this down to old-age on the part of the EPROMs, after all they had been through the program/erase cycle many times, and so I marked them as suspect and put them on one side. Some months later I happened to check the programming voltage on an EPROM as it was being programmed, and found it was only 21v rather than the required 25v. Off load the programmer produced the correct 25v. This made me dig out the 2716 data sheet, and a quick glance showed that a maximum current of 25ma could be drawn from the 25v supply during programming. I removed the 2716 from the programmer, and replaced it by a 1k resistor from pin 21 to ground. (25v across 1k gives a current of 25mA). I then started "programming" and measured the programming voltage that appeared across the 1k resistor - 18v. Bingo, there was my answer, the charge pump that generates the 25v supply for G808 was obviously rather short of breath, and unable to provide the necessary power. The 2716s I had had trouble with were obviously not programming because the programming voltage was too low. Perhaps with use the programming current actually taken by a 2716 slowly increases, and these had reached the stage where the on-board 25v supply was seriously overloaded.

The 25v is generated by a charge-pump circuit based round a 555 timer running as an oscillator. This runs off the +12v and -5v supplies, and theoretically adds another 17v onto the 12v line (ignoring all losses!), which is then regulated to 25v. I scrabbled in the proverbial junk box and tried changing the 555 to no effect. Altering the values of various components around the 555 didn't help either. So I went to the drawer containing "projects started but never finished" and dug out the last of my EPROM programmer designs that had failed to see the light of day (until now). This particular one used a Texas Instruments TL497 switching regulator in a step-up mode to generate 25v from a 5v supply. This I connected in parallel with the existing voltage generator on G808, and there-after I had my 25v. The 2716s previously labelled "defunct" now programmed perfectly. As this fix worked I progressed no further.

My solution (of geting the 25V from elsewhere), is one way out, but if anyone with the same problem can come up with a suitable simple modification to the G808 circuit - send it in. Either as a few words which I can include in a later column, or as a short article in its own right (in which case you get paid!).

Generous offer THAT NO-ONE SHOULD REFUSE

Talking of articles for the NEWS, if an article is submitted on a Gemini-format CP/M disk then the odds of it being published shorten considerably. So, in order to help the balance of articles in favour of non-disk (or non-CP/M) systems, I am offering to accept articles on cassette tape (Nascom 2 or Gemini format - not Nascom 1 tapes I'm afraid) or disk (5.25" or 8" - any format). These I will transcribe and pass on to the Editor. Note: this offer applies only to machine readable media - I'm not offering to type anything in from paper! All I ask is that you include two things:

1) A few words telling me what to expect (e.g. Naspen file % 1200 baud, or 48tpi Poly-Dos disk). and
2) a stamped addressed label (or envelope) for the return of your media if so required.

So put your fingers to the keyboard and send the results to 80-BUS News.

VIDEO OUTPUT STAGES

I was recently doing some work with a video output stage, and, having a reasonable 'scope to hand as a result, I decided to look at the ouput stage of Gemini IVC. There were two points I wanted to deal with, a) the low level of output (< standard 1V p-p), and b) 'The marching sands of time'. (Background interference patterns that are noticeable when using large areas of inverse video).

The IVC output stage is shown in Fig 1. It consists of an emitter follower driven by the video signal from IC3a (via R5), and the mixed syncs from IC5f (via R1). The ideal output waveform is shown in Fig 2. This consists of a three level signal whose peak-to-peak amplitude is 1V (when driving into a 75 ohm load). The three levels of the signal are white, black, and sync, and their relative amplitudes are also given in Fig 2. Assuming a perfect emitter follower with an output impedance of 75 ohms, we want to aim for a 2v p-p signal at the base of the transistor. I started by selecting arbitary voltage levels of White=3V, Black=1.6V, Sync=1V. Let us now consider how the three signals are generated: For the 'White' level the output of IC3a is high, (dot on), and the output of IC5f is high (no sync signal). For the 'Black' level we have the output of IC3a low (dot off), and the output of IC5f is high (no sync signal). Finally for the 'Sync' level the outputs of both IC3a and IC5f are low. Armed with this information it is only a case of applying Ohm's Law and Kirchoff's Law to the circuit...or is it? One problem is that the high level output voltage from a TTL gate is not very well defined, and it also has an appreciable output impedance. With the ICs fitted to my card the high level seemed to be around 3V, so, as I just happened to have picked 3V for my white level, I could conveniently ignore R1 and R5 and calculate a suitable value for R3. This then left me with two unknowns, and two equations (for the Black level and the Sync level). The result of the calculations (followed by a small adjustment after an actual trial) is shown in Fig 2. (Compare the values against those of the IVC circuit diagram). These values resulted in a reasonable video signal driving out from the card.

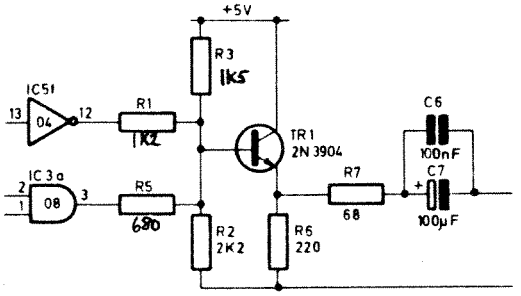


Fig 1

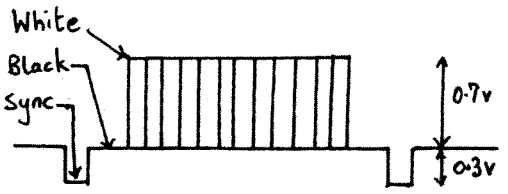


Fig 2

One other small change I would recommend, especially if you use inverse video or occasionally have large white areas on your screen, is to short out C7. This should not have any harmful effect, as all monitors I have encountered have their own DC blocking capacitor. (They normally have a 75 ohm terminating resistor, the top-end of which is coupled via a small capacitor, - 10uF, into a high impedance input stage. I note also that the BBC Micro has no

ouput coupling capacitor). If you do short out C7, and your monitor does provide a 75 ohm termination, then you may also improve matters by raising the value of the emitter resistor (R6) to 1k or more.

(N.B. Nascom 2 owners who use a monitor rather than a TV and the on-board modulator may like to make similar modifications. I assume the current drive levels on the N2 are set up for the modulator, rather than the direct video output).

Moving on to 'The sands of time', this interference is obviously being caused by crosstalk from the logic on the IVC. (The patterns change depending on what job the Host is asking the IVC to perform). I started by decoupling the collector of TR1, the top end of the Base bias chain (R3), and adding diodes in series with R1 and R5 (the 'bar' towards the output of each gate). The thinking behind the latter is that it isolates the 'Hi' output level from the video drive, each gate now looking like an open collector driver. As a result R5 was changed to 470 ohms, and R1 to 220 ohms to get back to the correct video and sync levels. This didn't produce any significant change to the visible interference.

So finally I resorted to butchery. I cut the thick power track running to the collector of TR1, and the corresponding track running to the top end of R3. I connected these two isolated points together, and decoupled them with a 47uF tantalum capacitor. I finally added a 68 ohm resistor between this point and the positive supply. (A few turns of wire on a ferrite toroid would have been better, but I had no toroid to hand). This certainly reduced the background noise, although a little was still visible if you looked for it. At this point I stopped. The next step to try, (if you are seeking perfection), is to build a new output stage on a separate board, taking care over the layout and postioning of the board. As for me - I'm sticking with what I've got!

Mixed 5.25" and 8" drives on GM829

I suggest you read Richard Beal's article elsewhere in this issue as an introduction to the rest of this section.

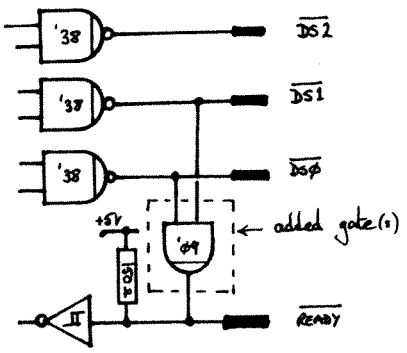


Fig 3.

In the case of the 5.25" drives the lack of a ready line, though inconvenient from a software point of view, is not a total disaster. This is because the motor-on monostable will eventually time out and produce a 'not ready' signal. This immediately causes any 'hung-up' access to a 5.25" drive to abort. However, when 8" drives are selected, all the circuitry associated with the motor-on monostable is disabled because virtually all 8" drives use mains powered motors which run continually. As a result there is no abort signal unless the 8" ready line is connected through. The answer is to insert link 3 - to connect the ready signal through - and to 'fake' ready signals from those drives that do not provide them directly.

The suggestion made to Richard was to try adding diodes to do this, but the same effect can be acheived by using an open-collector AND gate (e.g. 74LS09), or even an open-collector buffer (e.g. 7417). The input(s) to the gate(s) are connected to the drive select lines that do not provide a ready line, and the output(s) are connected to the Ready line (see fig 3). So, as soon as a drive select line connected to one of these added gates goes low, the Ready line is also pulled low. N.B. The gate used must be an open-collector gate, otherwise it will interfere with the 8" drive when it attempts to drive the ready line. In the approach suggested to Richard the diodes provide the required isolation, but as they are passive devices there may be problems with voltage drops across various components (as he discovered). The 74LS09 approach avoids this.

How to find your feet on a Nascom 2

A letter from Mr Mathison of West Germany has brought the N2 out again. He's been chasing the feet of the little men in the N2 character generator, but, despite following the suggestions published recently, has had no success. I'll start by describing the action of the N2 video circuitry before running through the solutions. That way any of you who attempt the modifications should have some idea of what you're up to. You will need a copy of the N2 circuit diagram if you want to follow the description below.

How it works

A 1MHz character clock emerges from IC49/13 (IC43 pin 13). This is divided down in the six-bit counter formed by IC51/IC52. The outputs of the six-bit counter form the address of a character within a display line, and address the video RAM via multiplexors IC62/IC63. The address is also decoded by IC55/IC60 to provide the blanking signal that frames the active 48 characters of the display, out of the total of the 64 characters that make up a line. (Remember that the Nascom video RAM has 16 unused characters between the end of one display line and the start of the next one). The output of IC52/13 triggers the monostable IC57 to provide the horizontal sync pulse, and also clocks the four-bit counter IC53, whose outputs form the 'row address' to the character generator. The character generator ROM address lines are also decoded by IC44c which, when the 625 line option is selected, resets the row address counter to zero everytime row 14 is reached. Thus only rows 0-13 of the character generator are displayed. Every time IC53/11 returns to 0, the five-bit counter formed by IC68/IC13b is clocked. The outputs of IC68 address the video ram via multiplexor IC64 and form the 'character line' address. Finally the IC68/IC13b counter is decoded in the N2V PROM (IC59). There are two outputs from this PROM, D1 provides the vertical blanking signal, and the D0 output is used to trigger the vertical sync signal (from IC57) and preset the five-bit counter IC68/IC13b to 11. (Check the hardwired inputs on IC68 to confirm this). The contents of the N2V PROM can be found in the Nascom 2 documentation, and from the listing you should be able to deduce the following, starting at the point where IC68/IC13b is reset to 11...

Counter	Action
11-14	Display is blanked.
15-30	Display is unblanked. Display lines go 15,0,1,2,3....13,14.
31	Display is blanked again
0	Display is blanked. (Counter wraps round to 0).
1	Vertical sync triggered & counter immediately reset to 11.

From this we can see that the weird order of the lines in the Nascom display was done with malice aforethought! (I've always assumed it was due to some quirk of the hardware implementation). By rotating the contents of the PROM by one location, and preloading IC68 with 12 instead of 11, the N2 can have a conventional video ram where the screen starts at the first address, and ends at the last address. If required, a non-scrolling top line can be implemented with a zero software overhead in this environment. Why did they do it the way they did? - I suppose an N1 circuit diagram may answer that.

However, returning to the matter in hand, it is a very easy matter to get all 16 lines out of the character generator. All that is necessary is to lift pin 1 of IC53. (If this is all you do, then this pin should be tied to +5v, either direct, or via a 1k resistor). This will prevent the row address counter being cleared on line 14, and it will now cycle round all the 16 lines of the character generator. There are two side-effects to this modification. i) The active display height will increase by 32 TV lines, which may require a picture height adjustment on your monitor or TV. ii) The frame rate will change. There will now be a total of 22x16 TV lines (=352) instead of the previous 308. (The 625 line standard actually requires 312.5). This will give a frame rate of 44Hz. When I made this modification to my N2 it did not disturb the frame hold on my monitor - you may not be so lucky.
NOTE The only effects of this modification will be an increase in screen height and a possible loss of vertical hold. The extra lines of the character generator can only appear in the correct place. Other symptoms may appear if you have disturbed dry joints, or other ICs.

Getting back into Sync

To get back to a 50Hz frame rate some more modifications are necessary. Somehow we need to take out about 40 TV lines from a frame. We can remove an integral number of 16 lines by changing the address intially loaded into IC68 at the start of every frame. However the effect of this is to move the display upwards on the screen. (The screen is unblanked sooner after the vertical sync pulse). It is also a simple matter to remove some lines from the end of the display, all you have to do is program a new N2V PROM(!). Ignoring the latter suggestion as it is impractical for most people, lift IC68/1 and connect it to IC68/8. Lift IC68/10 and connect it to IC68/16. This now arranges for the counter to start at a count of 13, rather than 11, and will thus remove 32 TV lines from the start of the frame.

This leaves us with a further 8 lines to get rid of if we want to hit a 50Hz frame rate. We can do this by including the row address counter IC53 in the 'frame reset' sequence, and preloading it to 8. Unfortunately the 'LS161 has a synchronous load although the clear is asynchronous, and the modification involves more than a simple strap. So, unless you're a perfectionist, I suggest you stop here.

(With the 'LS161 the load occurs synchronously with the clock. i.e. The load input has to be taken low, and held low until the clock input makes a low-to-high transition, at which point the data on the parallel inputs will be loaded into the counter. The clear input is asynchronous, which means that as soon as the clear input is taken low the counter is cleared immediately irrespective of the state of the input clock.)

Final tweak

For those who wish to persevere we are effectively going to invert the high order bit of the row address counter (IC53/11). This way we can use the asynchronous clear input as a 'set-to-eight' input rather than a 'set-to-zero'. Lift IC56/5 IC56/6 IC68/5 IC53/11. Connect IC53/11 to IC68/5. The row address counter now directly drives the line counter IC68 instead of being routed via the inverter IC56c. Connect IC53/11 to IC56/5, and IC56/6 to IC54/5 (or IC44/11 or IC66/5). This means that we have the inverse of the msb of IC53 driving RS3. Finally connect IC53/1, (after disconnecting it from +5V if you added that earlier), to the load line from the PROM. (IC59/1 or IC68/11 or IC13/13 or IC57/10). This connects the load signal to the clear of IC53. As the RS3 line is now inverted in IC56c, clearing IC53 is equivalent to setting RS0-RS3 to 8, thus removing a further 8 TV lines from the frame.

Try little steps

If you totally loose the top line(s) of your display, trying taking things a step at a time until you reach a point where you can lock your TV/Monitor. First try presetting IC68 to 11, then 12, then....

TRAILER

That's almost all for this issue, but just one final comment to confuse those of you who have read Richard Beal's article. I have a Shugart SA800 8" drive connected to my GM829, and currently it spends most of its time powered down. In this state it does not interfere in any way with my use of the 5.25" drives, so you may be lucky like me, or unlucky like RB, in which case I suggest you switch off and unplug before doing any rewiring!

Addendum to the Belectra HSA-88B Review

The above review just made it into the last issue by the narrowist of margins, and thus one correction and some updated information couldn't be included. First a minor point, the board size is 8"x4", not 8"x3" as stated. Secondly, as the review board and documentation reached me at different times and from different sources one document was left out. This was a seven-page addendum to the Hisoft Pascal manual, detailing the differences between versions 4 and 5. (Anyone who has bought the card will have received a copy).

As well as various enhancements to the compiler it points out that all integers in HP5 are 32-bit integers, giving a MAXINT of 2147483647! For the REAL precision and format it refers you to the AMD9511 documentation (see the review). The information on where to patch the run-time routines in order to use the HSA-88B at an alternative address is also given.

For a user's view of the HSA-88B see Dr Dark's column.

EVEN MORE RAMBLINGS

By David Hunt

Animal Antics

So Dr. Dark has had his moggie kipping in his computer; I can top that. In the spring some three or four years ago, I was building my Nascom 2 into a 19" rack, and making a decent job of it too. Proper laced wiring and all that. Well this was very boring for the rest of the family, who were entertained by the squirrels at the bottom of the garden collecting twigs and grass and stuff. Now what no-one bothered to tell me, was that the squirrels were running along the fence towards the house and disappearing in an upwards direction. The little so and so's were building a nice comfy nest in my chimneys. Incedental intelligence bit: squirrel nests are called dreys. Stuffed full of useless facts is this fellah!!

One morning, number two daughter runs in and says, "Daddy, daddy, there's a squirrel eating your computer!". Well at 7 a.m., what am I to think? Jokes I can take, but at that time of day I'm a little slow, and took the news very seriously. Half way down the stairs, the incongruity of the statement started to sink in. 'Squirrels?? Oh well, as I'm half way down stairs I might as well take a look see.' Sure enough, in the living room, sitting on the 19" rack is a small squirrel, grey variety, chomping through the Nascom wiring harness.

Now normally I'm the tolerant sort, squirrels are nice furry things which run around at the bottom of the garden doing me no harm and keeping the cat amused. Live and let live ... etc. But there's this one doing the Egon Ronay bit with my Nascom!!! Putting on my best impression of a Samuri, I entered the room. I'm not sure whether it was my entrance or the fact that I usually sleep in my birthday suit and had forgotten my dressing gown, which caused the strange look squirrel gave me. Anyway, a small grey flurry shot straight up the curtains and disappeared. At that moment it occurred to me that if squirrels like various thicknesses of pvc covered wire for breakfast, I was sadly unprotected against the teeth which could gnash their way through wires. I beat a hasty retreat and got dressed whilst considering the problem.

At last, armed with a small blanket borrowed from a dolls pram, I was ready to do battle. It wasn't easy, they move fast do squirrels, but in the end I caught it. If you've never caught a squirrel, I'll warn you. They can make the most unearthly screaming sound you've ever heard when caught, loud enough to make you drop it. But this lad is made of sterner stuff, wrapping the blanket firmly round the thing I dumped my squirming bundle at the bottom of the garden to be reclaimed by its parents. That, fortunately, is the last time any computer of mine has been attacked by anything other than me.

The missing DH bits.

Ardent fans of the DH series on databases (there is at least one fan, me) will notice the absence of an episode in this issue. Well this is not because I haven't written it, but that all the bits I was going to cover have been very nicely written for me by Clive Bowden in the last issue. His bit about random access methods using MBASIC just about sums up all I had to say on the matter, and so I must offer my thanks. However, I did find it annoying, from the point of view that I had already written half that bit for this issue and I didn't know that someone else had already done it. Never mind, I'll see to it that the boreing DH series starts again next issue and goes on and on .... and on ... [Ed. - ... and on ...]

The Spectre of Impending Doom

There are a lot of cowboys in the home computer business (if you hadn't already noticed) and these are about to get their 'come uppance'. Gazing into my crystal ball the other day, I noticed that over the months there has been a distinct downturn in the sales of the 'Mickey Mouse' computers (you don't expect me to name them do you?). Now a lot of this is due to the faddish buying profiles of the Great British public.

A few years back metal detectors were the thing, and a little later, a vast boom in dedicated TV games. Where are they now? I no longer know where to get a metal detector even if I wanted one, although in 1977 I could have had a choice of several in the local branch of one of the chain chemists. Last Christmas every kid, but every kid, expected a home computer in their Christmas stockings, and dad justified the expense by saying that he might as well get some use out of it and catch up on the 'New Technology' at the same time. How many of these computers are now kept at the back of a dark cupboard and are no longer in use because the users got bored with the games you could get, or 'Could never quite get the knack of this programming lark'.

No, it's my guess that this Christmas will ring the death knell for a number of familiar names; the Christmas spurt through newsagents, chemists, etc, will be the death throws of the home computer market. So where are the cowboys going to catch a cold? Well reliable information has it that there are thousands of Hong Kong and Taiwanese computers on the seas at this very moment, and the cowboys and GRQM's (Get Rich Quick Merchants, a mnemonic, courtesy of John Marshall) who have ordered them are going to be left with lots of machines and no market to sell them in. This means that those machines are going to be dumped at silly prices with no follow up support. This is bad news for those SERIOUSLY involved in the home computer business, as the word is going to spread that home computers are not worth touching as the: 'So called dealers can't support the machines, can't supply software, and are generally useless'. Whilst this may be true of many, the good machines and the good dealers (the minority in both instances) will get rolled in with the bad and that will do no one any good.

Which leads to an interesting development. Several of the manufacturers catering to the home computer market have been looking into the future and have seen the gloom approaching. So they think they will have a go at the economy end of the business machine market just to keep in business. Dragon have just introduced the Dragon 64 with a disk operating system called OS9, something I'm told is UNIX like. How far they expect to get with that with only three pieces of application software and a 57 x 24 screen I do not know. Lynx, because their machine is Z80 based have opted for CP/M. I've seen a sample with the most atrocious screen handling that can be imagined. Both manufacturers hope to sell systems at under a £1000. Best of luck in view of what follows.

The clouds are gathering over the business computer market. The big shake out is coming. Big names are going broke weekly and those who hoped to climb onto the bandwagon are having serious second thoughts. The 'el cheapo' business machines are in trouble, they seem to have saturated their own market, and it doesn't seem to make much difference how much free software you give away with the machine. IBM are now wagging the 'business machine' dog very firmly by the tail, so if you hope to sell a new business machine, it had better look like an IBM PC. All this despite the fact that large numbers of

IBMs and IBM 'look alikes' are being fitted with Z80 soft cards to enable the new 'all laughing dancing' 16 bit machines to run good old fashioned 8 bit software.

The same thing is happening with the computer industry magazines. Did you know that a recent advertising bureau audit showed over 160 computer magazines available to both the trade and the consumer. These same magazines are now scrambling all over each other for a shrinking advertising market and making 'four pages for the price of two' and similar type offers just to keep going.

Oh no, the computer market is in for one hell of a shake up, and I don't really need a crystal ball to predict the outcome, the pointers that I could see have been there since last March, and the mightier of the gloom pundits have probably seen this coming for far longer than I.

So what does this mean to the typical readership is this mag. Well the readership here is very mixed, still a large number of home users, a fair amount of lab and development types, and a lesser number of business users. Now we (I include myself) have never considered our machines as 'Mickey Mouse' machines, although my first Nascom 1 was definitely 'Mickey Mouse' compared with a Sinclair Spectrum (shudder). Most have graduated to more powerful systems which, whilst not all 'Bells and Whistles' as some of the offerings around, are used for the serious business of software development, hardware development, self education, or for running our businesses. Certainly not for entertainment, except of the most masochistic sort.

Fortunately, the few dealers who cater for our needs are not the 'cowboy' type. Nor, for that matter, are computers these companies only source of income. Some are general components suppliers, some are 'heavy' software suppliers. Either way, these companies seem well equipped to survive the forthcoming storm. Likewise, the machines which we use neither fall into the home computer category, nor the outright business machine end. The manufacturers supplying our needs look likely to survive because of the very diversity of the facilities offered by the machines we use. The sales profiles may change, more going to development and lab type people, less perhaps to the home user and, maybe, the business user, but companies we deal with will survive.

So amongst all this gloom, despondency, and more and more expensive and glossy advertising. Certain things are likely to remain, streamlined a little perhaps, a little more pricey perhaps, but I reckon our machines will remain around and supported this time next year. They'd better, 'I feel it in my bones, and my bones are never wrong', as Izzy Cohen said to the ships' captain, but then if you've never heard that story you'll never know how Izzy made his money.

Modems

I was intrigued by Dr. Dark's reason for buying a Sinclair Spectrum, to use it as modem for Micronet on the grounds that Marvin was too much of a heap of bits to gain the approval of British Telecom. Now this BT approval thing is something of an interesting problem, not that I claim to be an authority on the subject, but some of the bits I've read are nonsense and some of the things I've read make very sound sense.

Let's have a look at some of the things BT want, as I understand it. I'll only deal with a couple of the sensible things, I'm not sure about some of the things I've read, they seem daft, and I'm not sure whether they are serious or just 'hearsay'.

Firstly, they don't allow anything to be connected directly across the telephone line unless it's been approved. Now this is sound sense. Despite all the precautions a home constructor could take, a few are careless. I know, I've seen power supplies with no earths, I've seen power supplies with the neutral connected to the system ground (I admit I haven't seen one with the live connected to system ground yet). Now BT aren't interested in the ways in which you set out to kill yourself, but despite your personal opinions of BT engineers, imagine what would happen to an unsuspecting guy checking the wiring down at the exchange when he finds a very live wire. Live, not because of the red hot conversation taking place, but live because it's connected to the mains at your end. That's why any gear approved for direct connection is equipped with an isolating transformer.

Now a second thing that seems a bit odd at first sight is that you're not allowed to indirectly connect anything to the phone unless it's been approved. Indirect connection means amongst other things, acoustically. Now I was under the impression that acoustic couplers (home brew or otherwise) were Ok. Not so. The answer is signal levels. In this case BT are worried about excessive signals causing crosstalk between cables and so annoying other subscribers. (They're probably also a bit leary about excessive levels upsetting some of their tone controlled switching networks, but they don't say that.) Anyway that's my guess at the reason they don't like the relatively cheap kit modem supplied by a large company in Southend.

So it seems that legally you can't connect anything directly or indirectly to your telephone unless it's been approved; approval costs a bomb and takes a long time, and in the case of a kit modem, they'd want to approve each one individually before it's used.

Now about Micronet, it is a Prestel utility and contains a billboard and lots of useful software. But for Micronet you need access to Prestel. Prestel itself is odd in computer terms in that its data rates are peculiar, being 75 BAUD for transmit and 1200 BAUD for receive. Prestel modems are all directly connected, I don't know of any acoustic modems capable of going at those speeds. Most acoustic modems run bidirectionally at 300 BAUD. However, what is not widely publicised is that Prestel do have a 300 BAUD service on the Kipling computer, although only admittedly in the London area, try 01-248-5747. This service is primarily for business use, where business computers already have the usual 300 BAUD I/O.

So it seems from the guy at Micronet that Dr. Dark's Marvin couldn't be connected to the public telephone network, or could he? Well a new approved acoustic modem could be purchased for about twice the price of a Spectrum and its approved modem; or --- not that I would condone such a thing --- something else could be connected to the telephone network which satisfied the rules. In this instance, if BT actually detected it (indiscriminate phone tapping is still illegal, and a warrant is needed for discriminate phone tapping), it would appear to be an approved device and therefore wouldn't give rise to questions.

When using acoustic (or direct connected) 300 BAUD modems with Prestel, there have to be snags of course, firstly, the 300 BAUD service is restricted to only one computer, with a limited number of ports for its modems. This means that it is often engaged during the day, but evenings are a different story. I've no way of knowing, but I would guess that out of business hours, the 300 BAUD service is the quietest of the lot on the grounds that no-one knows it's there. Another nasty is that BT don't offer preferential rates for that number. If you live outside London, then dialing that number will get charged at trunk call rates, worse, because the data rate is one quarter of the normal Prestel data rate, everything takes four times as long to receive. The last flaw in the scheme is gaining access to Prestel, you require an individual access code which is many characters long. I'm not saying how many characters, just in case someone tries a bit of code cracking and hits my code by sheer good/bad luck; mind you that could be expensive, because if you get the access code wrong a couple of times, Prestel tells you you are an idiot, and disconnects the line. You get your access code when you apply for the Prestel service. They ask you what device you intend to use. You reply, "Personal computer". They ask you what. You say, "BBC", or "Spectrum", or "Oric" or some other approved device depending upon the computer you are using, or if you are using a 300 BAUD modem, you must make sure you get logged onto the Kipling computer, then you say "300 BAUD modem.". Typically the person taking the logging details knows nothing about the 300 BAUD service so you'll have to tell them. After a while, you get your access code, and given suitable software, away you go.

Now to suitable software, I'm writing a bit to make the Climax card look like a Prestel display, 'cos it's got colour and all that. But David Parkinson's dumb terminal routine works well enough to extract textual data and Ward Christenssen's MODEM 7 works well for textual data with the advantage that you can store the incoming data and use it for testing Climax colour routines, etc. If you've got a Winchester Technology colour card collecting dust, you're laughing, just plug it in and with the simplest of dumb terminal type software, you're away.

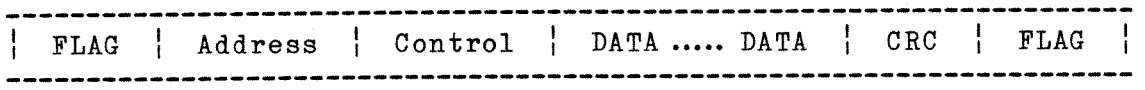
Apart from Prestel with Micronet, which costs, there are a lot of telephone numbers with modems and active computers on the end. There are billboards, companies advertising, all sorts of things, most of which allow access for nothing. A whole bunch were published in PCW a couple of months back and some of these can provide hours of endless fun (providing someone else is paying the phone bill). One person I know tried the Swedish number published, hoping that the Swedes with their liberated views had something interesting to say. Unfortunately the computer replied in Swedish, so he's none the wiser.

HDLC and Packet Radio

On the subject of serial communucations, I have recently come across references to a subject called 'Packet Radio' which has raised some interest in the States. On first reading this was pretty meaningless due to the sketchy nature of the odd paragraphs that appeared over here. However, Practical Wireless has seen fit to reprint the whole of one American explanatory article, and with a sigh (Oh no, not another protocol!), I sat down to read it. Now it turns out that 'Packet Radio' is not 'just another protocol', but a radio adaptation of an old enemy, HDLC. Now I say old enemy, as not so long ago one of the 80BUS board manufacturers asked me to look into high speed synchronous serial communications, in other words, making micros talk to

mainframes at speeds that make the mind boggle, like 250K to 1M BAUD. Not a bad thing in itself, but it made my brain ache by having to learn about things like HDLC. Anyway HDLC, which stands for High-level Data Link Communication, although a bit nasty to get to grips with, is rather clever when it clicks. Not only is it efficient, but at can be used at very high speeds.

The real sneaky with HDLC is in the way the data is packaged up (hence the term Packet Radio). Instead of sending serial data in the form we are familiar with, that is, a start bit, a byte of data and a stop bit, each byte being sent and received asynchronously with separate clocks; whole chunks of stuff are sent at once (typically 128 or 256 bytes at a time) with the clock signal as part of the data, in other words synchronously. To make this possible an HDLC packet consists of six fields like this:



The first flag is a unique signal, usually 01111110, now thats not unique you will think, but we'll come to the way it is unique in a moment. Anyway, the flag says 'Get ready, here comes the packet'.

The next thing to follow is the address, now the address length is not defined by the protocol, except that it must not be less than one byte long. Typically the address is two bytes long, and in that 16 bits, it may contain the coded address of the sending device within a network, and the coded address of the receiver within the same network; it may also contain routing instructions through the network. Overall, the address is the code which allows the packet to reach its correct destination.

The control is again a minimum one byte code, typically two bytes, and contains supervisory information, things like the current packet number, the length of the following packet, an acknowledgement of the correct receipt of the last packet, or a request to resend the last packet, or end of message, or anything else of a control nature defined and understood by the other end.

Next comes the data. This may be as many bytes as has been decided, it could be a standard number, 128 or 256, or if could be a number that has been decided by the control byte. Again the minimum is one byte

The CRC is a two byte Cyclic Redundancy Check (or checksum) which is computed from all the preceding data bits back to the first flag, so it encompasses the address and the control as well as the data. CRC's are lot better than simple checksums, but whatever it is, it forms a check on the validity of the preceding data.

Lastly comes the end of packet flag, again the unique code 01111110. This signals the end of the packet and also that the preceding two bytes were the CRC.

So to the uniqueness of the flags. All data is sent as a continual bit stream, it starts 01111110.....and finishes.....01111110. There are no start bits, no stop bits, nothing to indicate the bytes at all, just a continual bit stream from first to last. So what happens if we send a stream of FF's or 00's, apart from the flags there will be no low-high transitions and no high-

low transitions, just a continual high or low to the end of the data block. This is an invitation to lost sync, particularly if the clock is an integral part of the signal, how can this be overcome? The answer is what is called bit stuffing, which gives rise to a highly efficient data transmission system known as NRZI.

First a look at NRZI, it stands for Non Return to Zero Inverted (it is actually used on our disk systems but no-one shouts about it). It's not a case of high means a 1 and low means a 0. It is a case of change of input state means a change of output data state. Lets assume that the input signal is low, and the output data state is currently 0. The next input signal transition from low to high will cause a change of output state from 0 to 1. No different from what we already know, but if the output data state were already high when the input low to high transition occurred then the data state would change from 1 to 0. It is the input transition edges associated with the transition which causes the output to change from one data state to another. Thats where the Non Return to Zero bit comes in, an input transition to zero does not necessarily mean that the data state returns to 0, it could go to 1. None of this matters so long as the decoder knows what is to happen, and knows its starting states.

It should be obvious that to be a synchronous data transmission system, NRZI must have some sort of sychronising mechanism built in to allow clock recovery. It must know how many bits have passed between one input transition and the next. A continual steam of 1's or 0's is an invite to lost sync. If we sent a stream of 010101010.... then it would be no problem to recover the clock from the data itself, a simple phase lock would do it. Given the proven 'sample and hold' characteristics of phase locks, we could send quite a lot of 0's before we needed to send a 1 to regain the sync, likewise we could send a lot of 1's before we needed to send a 0 to regain sync. This is where the 'bit stuffing' technique comes in. The HDLC logic counts the number of similar bits sent. If five consecutive 1's are sent, then the logic will 'stuff in' an extra 0 in the sixth position to regain the sync. Likewise if five consecutive 0's are sent, then the logic 'stuffs in' an extra 1 in the sixth position. The receiving end knows all about this, and if it sees a transition in the sixth position it knows to throw that bit away. Very simple.

The only time this 'bit stuffing' does not take place is with the flags. So, if there are six consecutive 1's received without the necessary 'stuffed bit', then one of two things has happened. It must be that start or end of a packet. If it's the start, then no preceding data is of consequence, already having been dealt with. If it's the end of a packet then either a data error occurred, in which case the preceding two bytes won't add up to the CRC, making the packet invalid. Or the extra bit did indicate the end of the packet, in which case the CRC does add up and the packet is valid.

Of course with a high speed interface all the HDLC logic must be in hardware, no way can you do the 'bit stuffing', phase lock clock recovery and CRC checking in software when the data rate is approaching the cycle time of the processors involved. Special chips have been developed to handle this and although expensive, they can really shift. Another criteria with very high speed links is the clock recovery itself. Phase shifts in the connecting circuits can cause delays in the arrival of the transitions. These delays can be a substantial part of a 'bit time' when high speeds are considered. This one of the advantages of 'bit stuffing, the clock signal is an integral part

of the data, so that any delays in the data will cause similar delays in the recovered clock signal, hence everything stays in step.

With the 'Packet Radio' concept, the data rate is comparatively slow, 1200 BAUD or thereabouts, so clock recovery could be in software, and all functions could be controlled by software and not expensive beasts like the Zilog Z8530 SCC device. Of course, the data recovery is not the whole story. What about the routing and address information? Well this forms another rather nice feature of 'Packet Radio'. The International Standards Organisation (ISO) have developed a model network structure in a truly generalised form. It's a hierarchical thing with seven layers from the low level primitives which form the sending and receiving part through to the final high level controllers and routing protocols. All clever stuff, and I don't understand half of it - yet!

But whereas, in the last issue, I was thinking of playing with AMTOR, perhaps I'll change my mind and have a go at this instead. With the amateur radio regulations in the state they are, and the very woolly UK definitions regarding data transmission, it should be alright on 2 metres or 70 cms.

Nascom News

Since Lucas took over Nascom, I sometimes wonder what's been happening. In the last two years, Nascom have got the disk system out of the door, rewritten all the manuals, brought out a Nascom 2 in a box, called the Nascom 3, have produced a colour card, made their Nasdos based network work, and bought a lot of software that no-one seems to want. Although this represents a lot of consolidation of the Nascom range, in real terms, it's not a lot. Well Lucas seem to be getting concerned about the lot of nothing which seems to be happening in Warwick. A few heads have rolled and a general shake up seems to be taking place. Work seems to have been happening, and they have a very nice (no, excellent is a better word) piece of software running under CP/M called LOTTI. LOTTI is a CAD package (CAD stands for Computer Aided Design) using the Nascom AVC to full advantage. Coupled to a multicolour chart plotter it is capable of A2 and A3 drawing of considerable complexity. The name Nascom seems to be in the decline however, it's now Lucas Microcomputers, with Nascom being a model name within the range. There's a new range of machines called the LX (looking suspiciously like the Quantum 2000 although with different cards fitted), two printers carrying Lucas badges, and a couple of monitors also carrying Lucas badges. The future of kit Nascoms seems a little doubtful, as the emphasis is now on built, ready to go machines.

Letters and things

A couple more letters have come my way. So lets' have a look at them. The first from A. M. Davies of Tewkesbury encloses two programs written for RP/M or CP/M which appear elsewhere in this issue, he also raises a couple of points. Firstly the topic of DISKPEN VG:1 and GEMPEN VG:1. As most readers are aware, these are almost identical products. The confusion arises from DISKPEN VG:3, this is so much of a revision of DISKPEN VG:1 as to be considered as an entirely new product. With version VG:1, distribution was through Gemini, hence the greater number of GEMPEN's around. With DISKPEN VG:3, distribution is through the majority of Gemini dealers, but the product originates from the authors rather than Gemini. Mr. Davies asks, why, having returned the registration form, he was not informed of the enhancements to his DISKPEN/GEMPEN VG:1 (he does not say which). Well this is difficult to answer, if he owns a DISKPEN, then he may have been, as about 50% of DISKPEN owners have been circularized with the remainder to follow in December. If he owns

```
                        .Z80
                        Title Memory Block Comparison Program

                        .Comment "

                        COMPARE V1.1 15/06/83,  A. M. Davies
                        Written source supplied, transcribed DRH 19/11/83.

                        This program compares all the bytes in two blocks
                        of memory and types out on the console the HEX
                        values of the addresses in the lower block where
                        the bytes are different from the corresponding
                        bytes in the higher block. The start addresses of
                        the two blocks are defined as block1 and block2,
                        and the top address of block1 to be checked is
                        loaded into <BC>, at blok1top. The program runs
                        from 0D000H.

                        The program was originally used to find a bad byte
                        in an EPROM version of MBASIC-80, which did not
                        run correctly although the tape version of the
                        same did. It took about 500mS to find the one byte
                        in 24K that was incorrect.

                        The program is not claimed to optimised in any way
                        whatever."

E100                    stcksave equ    0e100h
E000                    newstack equ    0e000h

0000'                           aseg
                                org     0d000h

D000    ED 73 E100              ld      (stcksave),sp   ; Save the RP/M stack
D004    31 E000                 ld      sp,newstack     ; Set up new stack

D007    21 0100         block1: ld      hl,100h         ; Start addr. of lo. block
D00A    11 6100         block2: ld      de,6100h        ; Start addr. of hi. block
D00D    1A              getbyte: ld     a,(de)          ; Get byte from hi. block
D00E    BE                      cp      (hl)            ; Compare with lo. block
D00F    C4 D025                 call    nz,taddr        ; If not equal report error
D012    23                      inc     hl              ; Increment both pointers
D013    13                      inc     de
D014    E5                      push    hl              ; Save <HL>
D015    B7                      or      a               ; Clear any carry flag
D016    01 6100         blok1top: ld    bc,6100h        ; Check against top addr..
D019    ED 42                   sbc     hl,bc           ; ..for end
D01B    28 03                   jr      z,exit          ; If top then quit
D01D    E1                      pop     hl              ; Restore <HL> and..
D01E    18 ED                   jr      getbyte         ; ..loop for next byte

D020    ED 7B E100      exit:   ld      sp,(stcksave)   ; Get the stack back..
D024    C9                      ret                     ; ..and return
```



```
D025    F5              taddr:  push    af              ; Save the registers
D026    C5                      push    bc
D027    D5                      push    de
D028    E5                      push    hl
D029    1E OD                   ld      e,0dh           ; Type a CR and...
D02B    CD D05E                 call    conout
D02E    1E OA                   ld      e,0ah           ; ... an LF

                        ; Now put out the current address
D030    E1                      pop     hl              ; Get <HL> back
D031    E5                      push    hl              ; Resave <HL>
D032    7C                      ld      a,h             ; Type the hi. order byte
D033    CD D03F                 call    tbyte
D036    7D                      ld      a,l             ; Type the lo. order byte
D037    CD D03F                 call    tbyte
D03A    E1                      pop     hl              ; Restore the registers..
D03B    D1                      pop     de
D03C    C1                      pop     bc
D03D    F1                      pop     af
D03E    C9                      ret                     ; ..and return

D03F    F5              tbyte:  push    af              ; Save <AF>
D040    07                      rlca                    ; Rotate high nibble ...
D041    07                      rlca                    ; ... into low nibble
D042    07                      rlca
D043    07                      rlca
D044    CD D04C                 call    thex            ; Print the nibble
D047    F1                      pop     af              ; Get <AF> back
D048    CD D04C                 call    thex            ; Print the nibble
D04B    C9                      ret

D04C    E6 OF           thex:   and     0FH             ; Mask the high nibble
D04E    CD D056                 call    hexas
D051    5F                      ld      e,a
D052    CD D05E                 call    conout          ; Print the nibble
D055    C9                      ret

D056    C6 30           hexas:  add     a,'0'           ; Convert to ASCII
D058    FE 3A                   cp      '9'+1           ; Check if > 9
D05A    D8                      ret     c               ; Return if not
D05B    C6 07                   add     a,7             ; Convert to character
D05D    C9                      ret

D05E    F5              conout: push    af              ; Save the registers
D05F    C5                      push    bc
D060    D5                      push    de
D061    E5                      push    hl
D062    0E 02                   ld      c,2
D064    CD 0005                 call    0005h
D067    E1                      pop     hl
D068    D1                      pop     de
D069    C1                      pop     bc
D06A    F1                      pop     af
D06B    C9                      ret

                                end
```

```
                        .Z80
                        Title INTEL-HEX dump program

                        .Comment "

                        INTEL-HEX DUMP V1.3,    A. M. Davies
                        Written source supplied, transcribed DRH 19/11/83.

                        This program dumps out memory contents to the
                        screen and to the serial I/O (if ^P is active) in
                        INTEL-HEX format. The program runs from 8000H and
                        requires two parameters to be entered before
                        running - the start address of the memory area to
                        be dumped must be in locations 8800H and 8801H
                        (LSB first), and the number of 16 byte blocks to
                        be dumped must be in locations 9000H and 9001H
                        (LSB first). For example, 8800H and 8801H might
                        contain 00H and 01H, and 9000H and 9001H might
                        contain 80H and 00H. This gives the starting
                        address as 0100H with a total of 80H blocks, ie,
                        128 * 16 = 2048 bytes to be dumped. Note that the
                        start address is updated during the running of the
                        program, but the number of blocks value is not.
                        This facilitates further dumps from the next area
                        of memory without requiring further entry of
                        parameters.

                        The program was originally written to enable
                        memory contents to be dumped to an EPROM
                        programmer, and could no doubt be improved and
                        made much more user-friendly. "

8400                    buffer  equ     8400h           ; 16 byte data buffer
8800                    startadd equ    8800h           ; Start address of data to be
                                                        ; dumped. N.B. This value
                                                        ; incs as data is dumped.
8880                    reccnt1 equ     8880h           ; 16 byte block counter
9000                    recnum  equ     9000h           ; Number of 16 byte blocks to
                                                        ; dump. Not incremented by
                                                        ; program.
9010                    reccnt2 equ     9010h           ; Same as reccnt1
9200                    stcksv  equ     9200h
E000                    newstack equ    0e000h
000D                    cr      equ     0dh
000A                    lf      equ     0ah

0000'                           aseg
                                org     8000h
8000    ED 73 9200              ld      (stcksv),sp     ; Save the old stack
8004    31 E000                 ld      sp,newstack     ; Set new stack pointer

8007    3E 00           start:  ld      a,0             ; Clear reccnt1 ...
8009    32 8880                 ld      (reccnt1),a
800C    32 8881                 ld      (reccnt1+1),a
800F    32 9010                 ld      (reccnt2),a     ; ... and reccnt2
8012    32 8881                 ld      (reccnt1+1),a
```



```
8015    2A 8800         loadbuff: ld    hl,(startadd)   ; Get the start pointer
8018    11 8400                 ld      de,buffer       ; Copy data to buffer
801B    ED B0                   ldir
801D    CD 804A                 call    typerec         ; Type out the data

8020    11 0010                 ld      de,16           ; Update the pointer
8023    2A 8880                 ld      hl,(reccnt1)    ; in reccnt1
8026    19                      add     hl,de
8027    22 8880                 ld      (reccnt1),hl
802A    2A 8800                 ld      hl,(startadd)   ; Update the value
802D    19                      add     hl,de           ; in startadd
802E    22 8800                 ld      (startadd),hl

8031    2A 9000                 ld      hl,(recnum)     ; Get no. of blocks to send
8034    ED 5B 9010              ld      de,(reccnt2)    ; Get block count value ..
8038    13                      inc     de              ; .. bump the count ..
8039    ED 53 9010              ld      (reccnt2),de    ; .. and resave
803D    B7                      or      a               ; Clear any carry flag
803E    ED 52                   sbc     hl,de           ; Test for end
8040    20 D3                   jr      nz,loadbuff     ; If count <> size, do next ..

8042    CD 80BE                 call    typendfl        ; .. else end
8045    ED 7B 9200              ld      sp,(stcksv)
8049    C9                      ret

804A    CD 8083         typerec: call   tcrlf           ; Send a CR LF
804D    1E 3A                   ld      e,':'           ; Send a block marker
804F    CD 8075                 call    conout

8052    16 00                   ld      d,0             ; Clear checksum to zero
8054    06 10                   ld      b,16            ; Set block length in <B>
8056    3E 10                   ld      a,16            ; Set block length in <A>
8058    CD 808E                 call    tcbyte          ; Send the block
805B    2A 8880                 ld      hl,(reccnt1)    ; Send <HL> as the address
805E    CD 80B5                 call    tcaddr

8061    AF                      xor     a               ; Clear <A>, then send <A> ..
8062    CD 808E                 call    tcbyte          ; .. as the block type

8065    21 8400                 ld      hl,buffer       ; Get buffer in <HL>
8068    7E              typebyte: ld    a,(hl)          ; Get a byte ..
8069    CD 808E                 call    tcbyte          ; .. and send it
806C    23                      inc     hl              ; Point to next
806D    10 F9                   djnz    typebyte        ; Do for <B> times

806F    AF                      xor     a               ; Clear <A>
8070    92                      sub     d               ; Get checksum ..
8071    CD 8096                 call    tbyte           ; .. and send it
8074    C9                      ret

8075    E5              conout: push    hl              ; Save the registers
8076    D5                      push    de
8077    C5                      push    bc
8078    F5                      push    af
8079    0E 02                   ld      c,2             ; Load <C> with conout
```

```
807B    CD 0005             call    0005h           ; Call FDOS
807E    F1                  pop     af
807F    C1                  pop     bc
8080    D1                  pop     de
8081    E1                  pop     hl
8082    C9                  ret

8083    1E 0D       tcrlf:  ld      e,cr            ; Send a CR
8085    CD 8075             call    conout
8088    1E 0A               ld      e,lf            ; Send a LF
808A    CD 8075             call    conout
808D    C9                  ret

808E    4F          tcbyte: ld      c,a             ; Save the byte in <C>
808F    82                  add     a,d             ; Add to checksum in <D>
8090    57                  ld      d,a
8091    79                  ld      a,c             ; Swop byte back to <A>
8092    CD 8096             call    tbyte
8095    C9                  ret

8096    F5          tbyte:  push    af              ; Save the byte
8097    07                  rlca                    ; Rotate high nibble ..
8098    07                  rlca                    ; .. into low nibble ..
8099    07                  rlca
809A    07                  rlca
809B    CD 80A3             call    thex            ; .. and send it
809E    F1                  pop     af              ; Get back byte ..
809F    CD 80A3             call    thex            ; .. and send low nibble
80A2    C9                  ret

80A3    E6 0F       thex:   and     0fh             ; Mask high nibble
80A5    CD 80AD             call    hexas           ; Convert to ASCII
80A8    5F                  ld      e,a
80A9    CD 8075             call    conout          ; Send it
80AC    C9                  ret

80AD    C6 30       hexas:  add     a,'0'           ; Convert to ASCII
80AF    FE 3A               cp      '9'+1           ; See if > 9
80B1    D8                  ret     c               ; Return if not ..
80B2    C6 07               add     a,7             ; ..else convert to letter
80B4    C9                  ret

80B5    7C          tcaddr: ld      a,h             ; Get hi. byte of address
80B6    CD 808E             call    tcbyte          ; Convert to ASCII & send
80B9    7D                  ld      a,l             ; Get low byte of address
80BA    CD 808E             call    tcbyte          ; Convert to ASCII & send
80BD    C9                  ret

80BE    CD 8083     typendfl: call  tcrlf           ; Send a CR LF
80C1    1E 3A               ld      e,':'           ; Send a block marker
80C3    CD 8075             call    conout
80C6    AF                  xor     a               ; Clear <A> and <D>
80C7    57                  ld      d,a
80C8    CD 808E             call    tcbyte          ; Send a zero record length
80CB    21 0000             ld      hl,0            ; Clear HL,
80CE    CD 80B5             call    tcaddr          ; thus current address
```



```
80D1    7D                  ld      a,l             ; Load EOF rec. type in <A>
80D2    CD 808E             call    tcbyte          ; Send it

80D5    AF                  xor     a               ; Clear <A>
80D6    92                  sub     d               ; Get checksum ..
80D7    CD 8096             call    tbyte           ; .. and send it

80DA    06 3C       null:   ld      b,60            ; Send 60 nulls
80DC    1E 00       null1:  ld      e,0
80DE    CD 8075             call    conout
80E1    10 F9               djnz    null1
80E3    C9                  ret

                    end
```


Macros:

Symbols:

8400	BUFFER	8075	CONOUT	000D	CR
80AD	HEXAS	000A	LF	8015	LOADBUFF
E000	NEWSTACK	80DA	NULL	80DC	NULL1
8880	RECCNT1	9010	RECCNT2	9000	RECNUM
8007	START	8800	STARTADD	9200	STCKSV
8096	TBYTE	80B5	TCADDR	808E	TCBYTE
8083	TCRLF	80A3	THEX	8068	TYPEBYTE
80BE	TYPENDFL	804A	TYPEREC		

No Fatal error(s)

GEMPEN, then his registration document will have been returned to Gemini, who as far as I know have not circularized the owners of GEMPEN. This is probably because they are more interested in supporting WORDSTAR than later versions of DISKPEN, see the WORDSTAR commercial on vol 2 iss 5 p 24 in the middle of the unsolicited appraisal of DISKPEN. [Ed. - Unsolicited?? Well, I believe you! But the reason that the comment (not commercial) was added was to make sure that anyone reading the article was aware that the 'equal space justification' that the article was printed in is a feature of WordStar, and that PEN output would be visually different. You wouldn't like anyone to be mis-led, would you?] What the Editor omitted to say was they he has a conversion program that turns PEN text into WORDSTAR text, so that either is equally acceptable.

COMAL-80

Mr Davies also asks about COMAL-80 and the disappearance of the tape based version. The simple answer is that although it existed, it didn't work properly in tape based form. The trouble stems from the way in which it tries to save a file to tape and the simplfied workings of RP/M compared with CP/M. I don't remember exactly what was wrong, but it was to do with COMAL checking to see if a file existed and RP/M faking an answer which implied it did, at this point COMAL tried to delete the file and found it couldn't. That doesn't look right, but it was something on those lines, resulting in COMAL getting in a knot. As far as I know no-one has attempted to cure the problem.

Disk Content

Mr. Piper of Sheffield writes complaining of the disk based content of the mag, understanding that we publish what we're sent. He goes on to write that he'd like disks, but lack of pennies prohibits this, "Could someone provide bare pcbs, etc, to allow these facilities to be provided as pocket money allows." The simple answer is yes, a few of the early Henelec/Gemini GM805 single density controllers are still available in both 'pcb plus circuits' form, and in complete kit form at much reduced prices. Disk drives are still a problem, but drives are becoming cheaper and can be bought new for about £170 each by reading the small ads. Secondly, there are few second hand Gemini GM809's knocking around which have been taken as 'trade in's' for the GM829 controller at about £70.00, likewise some dealers have second hand drives, traded in for the same reasons, at prices between £100 and £150 depending upon condition. So it's worth phoneing around. The last pricey thing on the list is an operating system. This I'm afraid is going to hurt the pocket whatever you opt for. One dealer has a few old CP/M 1.4's for the GM805 available, but CP/M 2.2 for most other permutations will cost about £120. Polydos is a viable alternative for the Nascom owner, and cheaper, but will cost about £103. The answer is that you won't get much change out of £300 to put a disk system together, but I'll bet that's a lot cheaper than you thought. The above prices include VAT.

NASPEN Problem

Mr Piper also comes up with a permutation problem with NASPEN. He says he can't do a Read or a Join with Nascom 1 running NAS-SYS 3. Now this is one I haven't heard of. As far as I know, it works, which suggests the EPROMs are corrupt. Get them reprogrammed by Henry's, but check which version, there was a NASPEN VN.1 and a NASPEN VN:1 (subtle difference), the VN:1 version was later. As I said, as far as I know, they should both work as neither Read nor Join was monitor dependant.

MORE HAPPY TALK

By Rory O'Farrell

My note on computer to computer communications in 80-BUS News Vol 2 No 5 has rapidly been outdated. I wrote then of the method of turning the file into HEX. Recently, I found an article in Microsystems (July 83) on the same subject, but much more elegant! Giving credit where it's due, the author is Steven Fisher.

His method of getting the file into HEX is to use a Public Domain utility called UNLOAD. This utility takes a .COM file, and converts it into a file in .HEXformat. He gives the entire object code for this in the form of a HEX listing. This is reprinted below. It can be typed up using a suitable editor, and saved to disc as UNLOAD.HEX. Then you use the command:

```
A>LOAD UNLOAD
```

to convert it back to an object file. If you have any mistypings, these will show up as checksum errors, causing LOAD to protest and abort. Correct such errors by retyping or otherwise editing the lines in question, and go through the LOADing process again. When you have UNLOAD on your disc as a COM file, then you can try it out.

It is used by typing:

```
A>UNLOAD FILE 0100
```

where FILE must have the extension .COM, which you need not specify, and 0100 is the address at which it is to live. This will create, on the same drive as UNLOAD, a new file called FILE.HEX. So far, so good!

Listing 2 is a short HEX listing of a file called PIPIO.HEX. This should be typed up in the same way as UNLOAD.HEX. Do not try to LOAD this. It contains the patch for PIP for communications.

With ZSID or DDT, patch as follows:

```
ZSID PIP.COM
*** ***          ;ZSID answers with a size
£IPIPIO.HEX      ;Insert the name of the file to be used
£RO              ;Read it, Zero offset
£GO              ;Exit
SAVE 30 PIP.COM
```

A copy of PIP with this modification will need to be on both receiving and transmitting machines. The easiest way is to PIP the HEX files from one to the other, as the HEX files are ASCII, and short enough not to cause any problems with data loss through disc accesses.

To use this method of transfer, set up the PUN:to be a PTP: and the RDR: to be a PTR: on both machines. UNLOAD the file you require to transmit, then on the receiving machine type:

```
PIP FILENAME.HEX=INP:
```

and on the transmitter:

```
PIP PUN:=FILENAME.HEX,EOF:
```

During the course of the transfer, the computers will each respect the other's need for disc accesses, and the cursor will be restored when the file is finally completely transmitted. Then you need only LOAD it on the receiving machine, and voila!

If you get nulls in front of the .HEX file on the receiving machine, use [H] as a final parameter on the receiver command line. LOAD and UNLOAD only work on the same DRIVE as the .COM or .HEX, so this sets an effective limit to the size of the file you can handle - roughly 25% of your drive capacity, as the .HEX file is 2.81 times the size of the original .COM file. UNLOAD only works on .COM files, so if your file is of another type, then REN it to a .COM type, remembering to change from a .COM type at the receiver.

What happens during transmission is that the transmitter sends a character, then waits for the receiver to transmit the character back again. The receiver gets a character and retransmits it immediately. If either machine is busy, then the acknowledgement does not occur, but the unread transmitted character is held in the UART buffer until read.

I hope that this very simple method of transmission is as much use to readers as it has already been to me - 400k transmitted, with no errors!

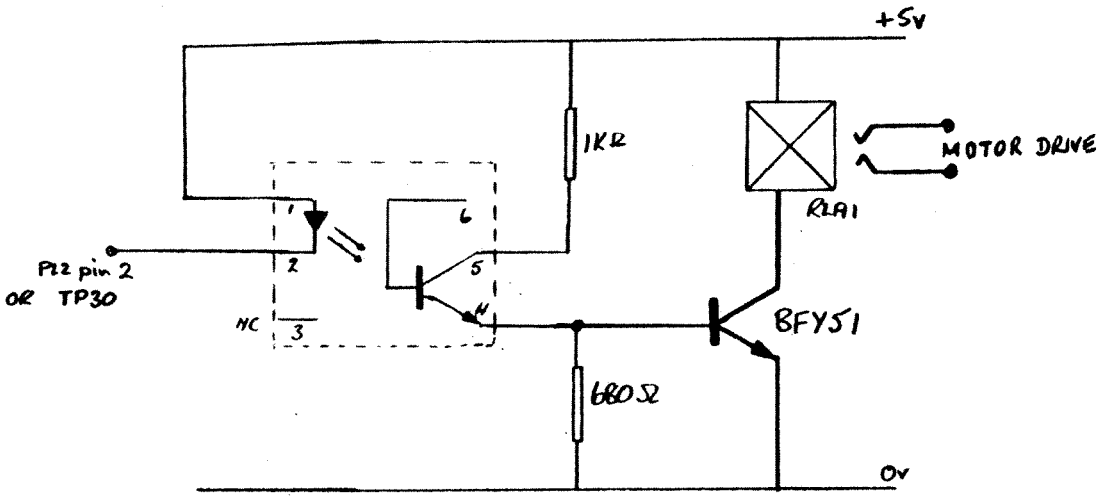
LISTING 1 ---- UNLOAD.HEX

```
:100100002A06002BF9C311017E1223130DC2080128
:10011000C9215C00111F010E09CD0801C34601 0071
:1001200000000000000000000434F4D00000000000F0
:10013000000000000000000000000000000000000BF
:10014000ED03000400000C3B6012A4201EB2A44017A
:100150007D937C9ADA9F01210000224401EB2A4220
:1001600001 7B957A9CD291012A400119EB0E1ACDA0
:10017000050011F010E14CD0500B7C28B011180BF
:10018000002A44011922440 1C35D012A4401 22428C
:1001900001 1180000E1ACD0500210000224401EB60
:1001A0002A400119EB2A42017DB43E1AC81A2A449A
:1001B000012322440 1C9AF322B01323F0121000447
:1001C000224201224401 0E0F111F01CD05003CC245
:1001D000ED010E0911DD01CD0500C300000D0A4E31
:1001E0004F204946494C45204649 4C4524215C0056
:1001F00011FB010E09CD0801C3220200000000001E
:10020000000000048455800000000000000000009
:10021000000000000000000000000000ED070004E6
:100220000000C3A502F52A1E02EB2A20027D937C62
:100230009ADA950221 0000222002EB2A1E027B9509
:100240007A9CD287022A1C0219EB0E1ACD050011E6
:10025000FB010E15CD0500B7C268021180002A20EF
:1002600021 9222002C33A020E091 17402CD0500C0
:10027000F1C300000D0A4449534B2046554C4C3AFB
:10028000204F46494C45241180000E1ACD050021 0F
:1002900000000222002EB2A1C0219EBF1122A200294
:1002A00023222002C9AF320702321B02210004229E
:1002B0001E0221 00002220020E1311FB01CD0500B9
:1002C0000E1611FB01CD05003CC2ED020E0911D73F
:1002D00002CD0500C300000D0A4E4F2044495220B4
:1002E000535041434 53A204F46494C452421000193
:1002F000060011 6D001A13D630DA1403FE0ADA0B69
:1003000003D607DA1403FE10D21403292929294F32
:1003100009C3F50222EA03CD4901CA8A03F53E3A30
:10032000CD2502AF32EC033E10CD4E033AEB03CDA8
```

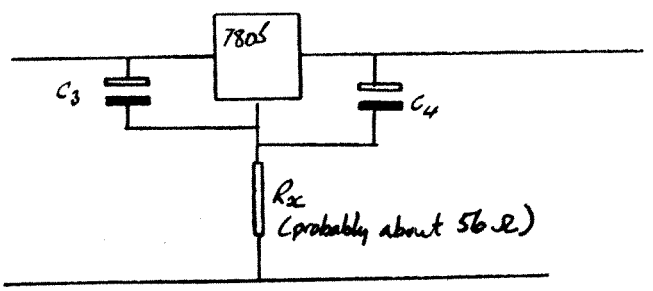
```
:100330004E033AEA03CD4E03AFCD4E03F10610C58E
:10034000CD4E03C105CA7003CD4901C33F034F3AE7
:10035000EC039132EC03790F0F0F0FCD5F0379E6B9
:100360000FC630FE3ADA6A03C607C5CD2502C1C9F9
:100370003AEC03CD4E033E0DCD25023E0ACD2502BB
:100380002AEA03111000019C314033E3ACD250206D0
:1003900005AFC5CD4E03C105C291033E0DCD25026B
:1003A0003E0ACD25022A20027DE67FC2B103221E2D
:1003B000023E1AF5CD2502F1C2A5030E1011FB0174
:1003C000CD05003CC2E7030E0911D203CD0500C3E1
:1003D000E7030D0A43414E4E4F5420434C4F5345C3
:1003E000204F46494C4524C3000000000000000097
:1003F00000000000000000000000000000000000FD
:0000000000
```

LISTING 2 ---- PIPIO.HEX

```
:10010300C30A01C31901000E03CD05003209015FC3
:100113000E04CD0500C9590E04CD05000E03CD050F
:02012300000C911
:0000000000
```



CIRCUIT No 1



CIRCUIT No 2.

THREE SIMPLE NASCOM 2 MODS.

By K. Hamlyn

Tape Motor switching

For some while there has been a lack of urgency in the switching on and off of the tape recorder under firmware control. Having tried various commercial and homemade interfaces, it was realised that under the R and W command (also CLOAD and CSAVE), LED2 was illuminated (quick, this fellow - eh?). A stirring of the grey matter brought to mind the opto-isolator, which after all requires no more driving than a normal LED. See circuit 1. The internal transistor is effectively switched on by the illumination of the internal LED. It was decided that transistor output was not on as large currents occur in the motor drive, particularly at switch on and off; proven by the number of fatalities of BFY51's having meant bulk purchases in the past. Thus the final motor control was chosen to be electromechanical, under the drive of the BFY51. The whole exercise is heavily over engineered, but this is to ensure a long and healthy life.

PSU Droop

One of the problems of the Nascom 3A power supply, or at least mine, is that the no load output of the +5 volt rail is a nominal +5 volts, but under load drops to +4.8 volts. The first steps in attempting to raise the output voltage, and also providing a 5 amp output, was to replace the LM323 (IC1 on the PSU circuit diagram) with a 78H05 5 volt 5 amp regulator. This was attached to a very large heat sink remote from the board, and connected with very thick cables. To further improve the current handling capacity of the supply, the 9 volt winding on the mains transformer was dispensed with and a new 12 volt 50 VA transformer used for the 5 volt rail instead. The current capacity was now there, but the supply resolutely remained at 4.8 volts.

A resistor, in this case 56R was introduced into the earth lead of the regulator, and had the desired affect of raising the voltage to + 5 volts under load. See circuit 2. (Note: that inserting a resistor this way is quite legit, but as the case of the regulator is the earthy connection it means that the regulator must be isolated from the heat sink, or the heat sink must be isolated from system ground (0 volts). -DRH.) A word of caution though. The exact effect on the voltage of inserting a resistor is difficult to predict, even if you are good at sums, as the final value is device dependant. It is best to disconnect the computer and replace it with a dummy load whilst selecting the appropriate resistor, as life can become expensive if the voltage is allowed to rise above 5.25 volts (7 volts is absolute max. according to the book, but I agree with the precaution. -DRH). Certainly there has been an improvement in the video quality, probably caused by the rumoured increase in speed of TTL at the correct operating voltage (not of course the computer itself, as this is crystal controlled). I thoroughly recommend the use of silicon based heat sink compound between the heatsink and the regulator.

Backplane mod.

Finally, there appears to be some mileage in 'flow-soldering' the power tracks of the Vero backplane. Not a new trick, but it does make the whole thing more solid. Mind you it does take some time to avoid indvertently bridging tracks.

No claims for originality are made for these modifications, particularly the last two, but they might help someone out of a hole!!

RANDOM RUMOURS (& TRUTHS?)

By S. Monger

This is where they are.

In my last bit, in Vol. 2 Iss. 3, I asked the whereabouts of several products that had been announced, but at that time had not actually materialised. We are now six months further on, and so what is the current position? Well, the Sinclair Microdrive has appeared and is NOT a disk, but a continuous loop tape - consequently the person that I'd mentioned who intended to try to interface one to his 80-BUS machine has long since abandoned the idea. I believe that Lucas now have Winchesters available, and understand that if you open them up (the units, not the Winchesters !) and look inside you'll find a Gemini GM829 FDC/SASI board! The Gemini GM818 serial board has now materialised, but beware! The GM818 is now a daughter board for the GM816 I/O board, and contains two 8250 type UARTs - the originally announced synchronous board has been redesignated the GM848 and is still some weeks away from availability. IO Research's Pallette is now available, for those that can afford it, and for those that can't the Mini Pallette is a meagre £300ish 'cheap' lower spec alternative. Still no sign of IO-Net.

Other New Products.

One item that was sneaked out without my prior knowledge was the Belectra Arithmetic Co-Processor board, but this was covered in some depth in the last issue by David Parkinson. Another co-processor board has also been announced, but no delivery date has yet been given. The Gemini GM886 will contain an iAPX186 (otherwise known as 80186 - i.e. a super GT 8088/8086) and a 'large' amount of RAM (256K?) as well as a socket for the 8087 (ultra high speed, ultra high price number cruncher). As already mentioned, the board will be added to an 80-BUS system as a co-processor board. i.e. the existing Z80 system will continue to handle all keyboard, screen, printer and disk requirements, whilst the 186 will be capable of running CP/M86, Concurrent CP/M and MS-DOS (Gemini are vague as to which they will be making available).

Much more imminently available is the Gemini GM832 SVC (Super Video Controller) board. This is a considerably enhanced GM812 IVC. For a start the 4MHz Z80A has been replaced with a 6MHz Z80B, and consequently, along with the effects of a number of other hardware and software changes, the board is much faster. The programmable character set has been extended to all 256 displayable characters and so the board becomes much more suitable for supporting foreign languages (as well as being more flexible for 'Space Invader' type applications!). Character attributes are also available - half-intensity, blinking and inverse characters, and half-tone backgrounds, plus combinations of these (although different attributes cannot be used on different character cells at the same time). A buzzer is also fitted. But to many the most interesting addition will be the graphics mode, providing a 256x256 bit-mapped display, with graphics primitives of line and circle drawing, polygon fill etc being included in the on-board monitor program. It is here that the Z80B comes into its own, and the plotting speed is very impressive. Estimated price is £195 (+VAT), with availability from 2Q84.

And Finally.

Mention has been made in the last couple of these rags of alleged infringement of copyright of Gemini's CP/M BIOS. Partially because of this, and partially because of Gemini's policy of constantly adding further facilities to their BIOS anyway, I understand that BIOS Version 3.0 is almost upon us, and that this will provide some 'interesting' facilities. More importantly (for existing customers), it will be made available as an upgrade at a 'very reasonable' price. Watch this page for further details.

CMOS RAM BOARD RB32KC

RB32KC C.M.O.S. BATTERY BACKED RAM BOARD

POWER FAILURE? Memory contents are preserved during power failure by a Ni-cad battery that is automatically recharged when the board is powered up. Contents are secure for up to 40 days.

PAGE MODE SCHEME SUPPORTED. The board may be configured as one 32K page or two 16K pages.

FLEXIBLE ADDRESS DECODING. Any 4K memory block may be located on any 4K boundary in the processor address space.

NO SOLDERING. All options are selected by wire links pushed into gold plated connectors.

EPROM OPTION. On this board, RAM and EPROMs may be mixed.

EPROM PROGRAMMER/ERASERS ARE NO LONGER REQUIRED because by loading a RAM block and then using the hardware write protect link, the block becomes equivalent to EPROM.

FEATURES HARDWARE AND/OR SOFTWARE READ/WRITE PROTECTION.

FULLY 80-BUS AND NASBUS COMPATIBLE.

BACKPLANE BP14C

BP14C 14 SLOT 80-BUS & NASBUS TERMINATED BACKPLANE

(Bare Board Weighs 15oz!)

* Ground plane on one side of double sided 2.4mm thick printed circuit board.
* All active BUS signals interlaced with ground 'shield' tracks.
* All active BUS signals terminated into a potential balanced R.C. filter.
* Proper implementation of both the interrupt and BUS request daisy chains.
* Large tracks for power lines.
* Size 15" x 8". Fits neatly into 19" rack.
* Easily cut for smaller systems.

Supplied built and tested without connectors.

PRICES RB32KC 32K Bytes - £225.00 BP14C - £57.00
16K Bytes - £193.00 Memory
2K Bytes - £165.00 HM6116LP-3 - £5.00
Plus £1.50 per board post and packing, plus VAT.

SPECIAL OFFER. While stocks last

£30.00 off any RB32KC board ordered with this advertisement. Applies to mail order sales from this address only. Offer closes July 31st, 1984. Cash sales only.

Cheques/PO's made payable to:-

MICROCODE (CONTROL) LTD.
40 Well Terrace, Clitheroe,
Lancs., U.K. Tel. 0200 27890

SPARTACODE LTD

Your South Coast 80-BUS Dealer

Mr John Stuckey

69 London Road

BOGNOR REGIS

PO21 1DE

0243 826161

TRIED AND TESTED CP/M SOFTWARE SPECIALLY SELECTED FOR YOUR GALAXY COMPUTER

Phone or write today for latest details and price list

WORD PROCESSING

		£
Wordstar (3.3)	MicroPro	295
Mailmerge (3.3)	MicroPro	145
Spellstar (3.3)	MicroPro	145
Starindex (3.3)	MicroPro	116
Wordstar Professional	MicroPro	414*

DATABASE & FILE MANAGEMENT

		£
Infostar	MicroPro	295
dBase II	Ashton-Tate	437
Friday!	Ashton-Tate	195
Quickcode	Fox & Geller	200
Personal Pearl	Pearl Software	190

FINANCIAL PLANNING

		£
Calcstar	MicroPro	116
Supercalc	Sorcim	126
Supercalc 2	Sorcim	195

CRITICAL PATH ANALYSIS ETC.

		£
Milestone	Organic Software	233
Datebook	Organic Software	233
Pertmaster	Abtex	650

LANGUAGES

		£
BASIC interpreter	Microsoft	259
BASIC compiler	Microsoft	295
ECO-C	Ecosoft	233
Pro Fortran	Prospero	220
CIS Cobol	Microfocus	425
Macro-80	Microsoft	149
Pro Pascal	Prospero	220

GAMES

		£
Deadline	Infocom	40
Enchanter	Infocom	40
Infidel	Infocom	40
Planetfall	Infocom	40
Starcross	Infocom	33
Suspended	Infocom	40
Witness	Infocom	40
Zork I	Infocom	33
Zork II	Infocom	33
Zork III	Infocom	33

*Special introductory offer

Add 15% VAT to all prices

Access and Visa card holders welcome

Either enclose a cheque made payable to 'Business Computer Developments' or quote your credit card number. Please state which disk format you require. Products referred to are trademarks or registered trademarks of the companies of origin. CP/M is a registered trademark of Digital Research Inc.

We specialise in many business applications in addition to the popular products listed above. Don't waste a second, phone or write today for further details.

B·C·D

Business Computer Developments
The Saddlery
113 Station Road
Chingford
E4 7BU Phone 01-524 2537