

80-BUS NEWS

MARCH—APRIL 1984

VOL. 3 ISSUE 2

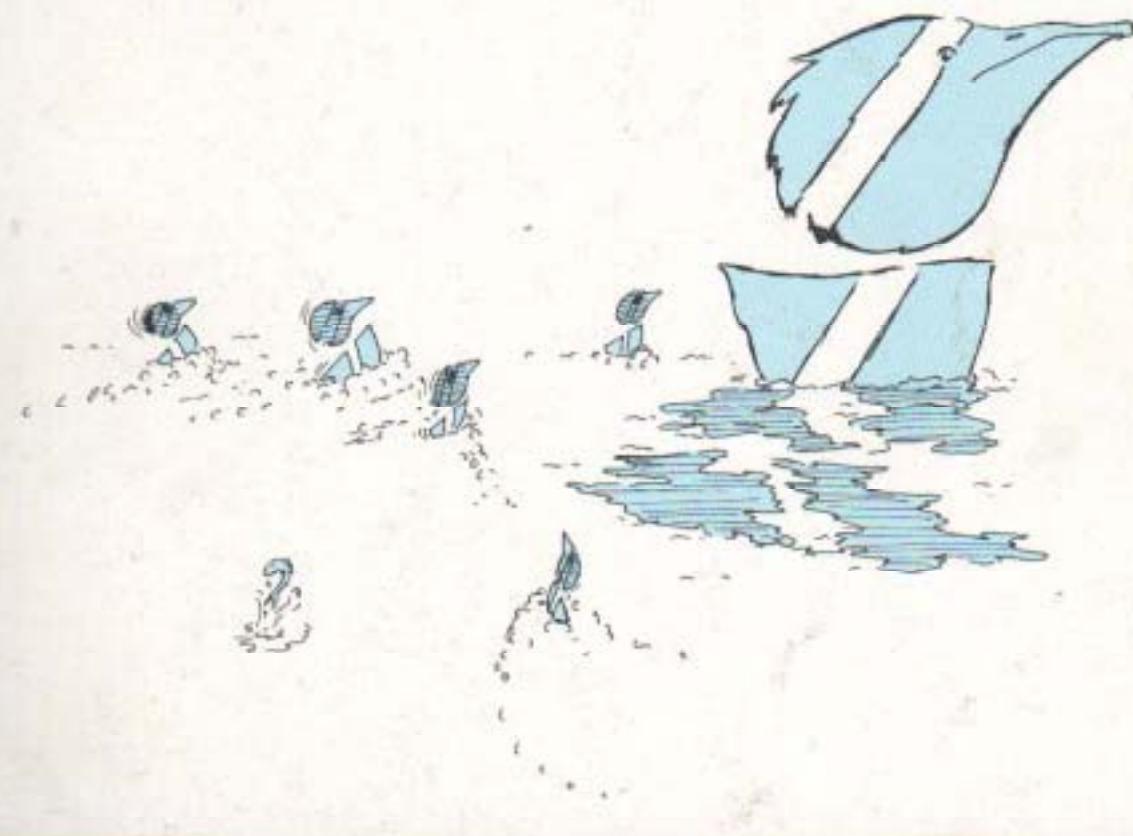
- REVIEWS

NEW GEMINI PROGRAMMER

NASCOM ENHANCED BASIC

- THE IVC AND THE SVC

- NASPEN FOR POLYDOS



The Magazine for
GEMINI & NASCOM USERS

£1.50

March-April 1984.

80-BUS NEWS

Volume 3. Issue 2.

CONTENTS

Page 3	Letters to the Editor
Page 9	MONITOR.COM and Other Stuff
Page 17	Doctor Dark's Diary - 21
Page 18	Some Private Sales
Page 19	A POLYDOS Version Of NasPen
Page 22	Coming Next Issue
Page 23	NASCOM BASIC Disassembled - Part 6
Page 31	REVIEWS - Gemini GM860 EPROM Programmer
Page 35	- Nascom Enhanced BASIC
Page 42	Some More Private Sales
Page 43	An Insight into the Gemini IVC and SVC
Page 50	Ads

All material copyright (c) 1984 by Gemini Microcomputers Ltd. No part of this issue may be reproduced in any form without the prior consent in writing of the publisher except short excerpts quoted for the purposes of review and duly credited. The publishers do not necessarily agree with the views expressed by contributors, and assume no responsibility for errors in reproduction or interpretation in the subject matter of this magazine or from any results arising therefrom. The Editor welcomes articles and listings submitted for publication. Material is accepted on an all rights basis unless otherwise agreed. Published by Gemini Microcomputers Ltd. Printed by The Print Centre, Chesham.

SUBSCRIPTIONS

<u>Annual Rates (6 issues)</u>	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.

PRIVATE SALES

Free of charge to Subscribers by sending to:

ADDRESS: 80-BUS News,
c/o Gemini Microcomputers Ltd.,
18 Woodside Road,
Amersham, Bucks.

Letters to the Editor

Nascom Plus Typewriter

Congratulations to you and your team on 80-BUS News. Has anyone tried using an Olivetti or Sanyo dual purpose printer/electric typewriter on a Nascom 2 yet? For some user-friendly advice on this would be well received. Electric typewriters have limited graphics but good quality printing.

With all good wishes, Ambrose Lambert, Stratton on the Fosse, Bath.

POLYDOS

Hello, angry of Tonyrefail here (again). Just writing to let you know that (i) my Nascom has just committed suicide, (ii) my dandruff has entered the terminal stage, and (iii) since writing to you a few months ago on the subject of starting a program exchange group for POLYDOS users, a la Dr. Dark's 'circle of iron', I have received NO letters from you or anyone else (I counted them twice, just to make sure, and it's definitely NONE!).

I know my letters tend to be a little flippant, but I hope you took the time to read my last letter before throwing it in the bin! Or perhaps the people who use Polydos are a bit more apathetic than 'yer average NAS-NUT, and I'm the only one 'wot wrote to you'.

Anyway, if you do hear from anyone interested in starting a Polydos fan club would you please TELL ME ABOUT THEM. Or, if such a group already exists, could you put ME in touch with THEM?

Do you know how lonely it is living half-way up a Welsh mountain? It would be very nice to talk to someone. Well, it'd make a change from talking to the sheep, they all have BBC micros, and hence, can't run POLYDOS (BAA)!!!

Yours, D.G. Richards, Tonyrefail, Mid Glamorgan.

Nascom - Who Are They?

Could somebody who reads this letter who is someway involved with Nascom/Lucas Logic please tell us why we, that is the devoted keyboard bashers, never hear or see anything advertised or reviewed to run on the micro? I personally have had a Nascom/Gemini etc., since it first came out (October 1979 I think) and whilst Gemini have continued to advertise on their own and through the Microvalue group, Lucas apparently have gone into limbo, is it the lull before the storm?

I now am presently looking to upgrade my system to high resolution colour with a Pluto board, if I can find somebody that stocks it and doesn't only get to special order, so that I can receive via the computer the weather satellite pictures transmitted from such as Meteosat 2. I have at present got a system working on a BBC model B which works, but the colour resolution and definition isn't high enough (note this is linked to the Nascom via the RS423/RS232 port), so if there is anybody reading this who wants to get rid of a Pluto board (cheap!!) I would be very interested and then I could write a piece telling you how I did it.

One other point can somebody at Henry's tell me if there are any more SYS's for CP/M since number 15 and if the Utilities disc has been added to or revised in the last six months.

Yours faithfully, D.J. Roche, B. Tech (Hons) A.M.I.E.E., Workshop, Notts.

[Ed. - On the question of SYS, this was withdrawn from sale some months ago, as outlined in previous issues of this mag. As regards one of your other points "Dear Nascom - is there anyone alive there?"]

NasPen Problem Revisited

Thank you for the answers to my questions in a recent 80-BUS which have now been extracted for filing in an appropriate place (not a waste bin).

We do have our wires crossed on the bit "NasPen Problem". I am actually running a Nascom 2 with Nas-Sys 3 monitor BUT the NasPen was the same set of EPROMs that I had for Nas-Sys 1 monitor. It was during the time between writing to you for help, and the letter being published, and getting more frustrated in between, that I learned that it was all to do with the off-set on the READ command. I have since had my EPROMs modified and all is now well. What did cause some confusion was that in all the literature I could find, no mention was made that NasPen was different for Nas-Sys 3. (My original 2708 ROMs are labelled NasPen VS 1-1 and NasPen VS 1-2.) It occurs to me that it might be worth while mentioning this to anyone who might be considering obtaining NasPen from someone who no longer requires it. In short NasPen to run with Nas-Sys 3 is not the same as NasPen made to run with Nas-Sys 1.

Incidentally, when I contacted Lucas about it, they would not accept that my original copy was from a bona fide dealer at the time. I bought my NASCOM as a kit when RAM A cards were being included for free.

Thanks again for the reply and I felt that the record had to be straightened out.

Yours sincerely, H.B. Piper, Sheffield.

Some Suggestion

Here are a number of suggestions I would like to make and they maybe worth a few words in a future issue:

Submitting programs and/or articles

- Do you prefer "camera-ready" listings or tapes? (We don't all have disks.) If tapes are used, which format would be both general or convenient? (We don't all have the same word-processor if any.)
- Some feedback on submitted material would be, to say the least, encouraging! (I submitted a software version of a real-time clock and my last hope of hearing about it is that it was published in the issue I did not receive.)

To be pragmatical why not print in each issue the list of received and not yet published material? It would be enough to have one line per entry stating for example the name, status (reviewed or not/will/will not be published/kept for special issue/ etc.), and any particular comments (unreadable/simplistic/send more of the same kind/etc.)

Subscriptions Why not print the "subscription due date" on the mailing stickers? It could be a useful reminder!

Thanks for the magazine quality, Jean-Michel Dasnoy, Brussels, Belgium

[Ed. - Thanks for the suggestions, comments are always welcome and will be ignored in the usual fashion acted on where appropriate. Some form of magnetic media is always preferable to hard listings - preferably Nascom 2 or Gemini RP/M formats if tapes are to be sent, we can't cope with Nascom 1 tapes. As far as the actual data format, we can cope with NasPen or WordStar type files, and failing that just send a straight ASCII file. As far as sub. reminders are concerned, these are always included in the last mag(s) on your current sub.]

Deliberate Errors!

Many thanks for publishing my two programmes, COMPARE and INTEL-HEX DUMP, in the (rather belated!) Vol. 2 Iss. 6 80-BUS NEWS. To save you all the aggro, I'll tell you where DRH put in the deliberate mistakes when he transcribed them!

Firstly, in COMPARE, in sub-routine TADDR (D025), a CALL CONOUT should follow the LD E,0AH at D02E. In INTEL-HEX DUMP, in sub-routine LOADBUFF (8015), an LD BC,16 should follow the LD DE,BUFFER at 8018. Again in INTEL-HEX DUMP, in sub-routine TYPENDFL (80BE), the instruction at 80D1 should be LD A,01 and not LD A,L. I hope the above help to solve any problems which anyone may have had.

On the topic of GEMPEN and DISKPEN, I have GEMPEN VG:1. My question now is:- Can I purchase DISKPEN VG:3 on tape to run under RP/M, from any of the usual GEMINI dealers, or does this latest version need to access disks whilst in use, for such things as HELP overlays. One further question:- presumably the GM829 FDC card copes satisfactorily with the interfacing to a TEAC FD 55E or 55F drive, but can GEMINI'S distribution CP/M disk be simply slotted in and expected to run? TEAC drives are somewhat cheaper than Micropolis!

Did DRH make any sense out of the AMTOR spec I sent him some time ago?

Is there any advantage in swapping to the latest version of the GM812 IVC monitor, which I now believe is in a 2732 as opposed to my version in a 2716, when I don't have the latest keyboard? Exactly what are the differences between the two versions - does the later one have any extra "goodies" in?

Sorry I haven't had time to submit anything else for your perusal, but hopefully others will have done so. Keep up the good work on the magazine - rest assured that it's much appreciated. Please, no more 6-month gaps though!

Yours truly, Martin Davies, Tewkesbury, Glos.

[Ed. - As far as I am aware it is not possible to use the latest PEN on tape as, as you have observed, it needs to be able to access the Overlay files, and coping with this with a tape recorder would present quite a problem!

With regard to the Teac drives and Gemini CP/Ms, yes these will run, although the standard Gemini products assume the 10mS track-to-track time of the Micropolis drives, rather than the 3mS capability of the Teacs. They will still run perfectly alright, it's just that they will be a little slower than they should be, and for some reason the Teacs make quite a bit more noise when stepping at the slower rate. The much awaited (and awaited) Gemini CP/M with BIOS 3 will get around this feature when (if) it appears!

The IVC monitor situation is covered in David Parkinson's article in this issue, and hopefully that will answer your final question.]

POLYDOS

In the latest pricelist from Gemini I noticed that Polydos will soon be discontinued. If you should find it interesting for 80-BUS News' readers we are still supplying PolyDos and other PolyData Nascom software.

Yours truly, Piezodan Aps, Bakkedraget 55, Dk-3480 Fredensborg, Denmark

Nascom Dying?

Please find my next years subscription enclosed. What reminded me, was the receipt of the final Nascom Newsletter. This seems to indicate that Nascom is dying. I trust that Gemini is going to keep the breed gong. How about publishing a review of where Gemini are and what we might expect in terms of product development. Thanks for an excellent magazine.

Yours truly, M G Nixon, Chelmsford

Don't Forget

Please don't forget us humble Nascom 2 owners without disks. Otherwise, keep up the good work.

Yours truly, K Ward, Kent

Comments And Praise

I did not write to complain but to add my voice to the many people who have praised your efforts over the last two or three years. The magazine has been interesting in so many ways, often when the articles have been on subjects outside my own particular interests. The useful point is that as my ideas have matured and the hardware built up - so ideas and articles that were not of interest before now become relevant.

The only comment that I can make about future issues is that I would like to see more descriptions of system software capabilities written either by the originator or by experienced users. As I live some way from any dealer, knowing what one would get by buying a particular software package is somewhat vague. The articles on CP/M have been useful but could go further. Don't laugh, but I would like to run Cobol. Is CIS-COBOL at #400 the only option? Adverts in Byte indicate it is available for CP/M at \$29! I would have liked to have seen comparative reviews of PolyDos vs. NasDos before I bought my FDC. And so on. Thanks again for a great magazine that gets the words worn off by reading again and again.

Yours truly, Nigel Chetwood, Tewkesbury, Glos

Mis-quoted

My congratulations on yet another splendid issue of the 80-BUS News. Lawrence (alias Mr D G Richards) has rather quoted anything I may have said to him (whilst tired and emotional) out of context - honest! I don't really think you are the Mafia

Yours truly, C Blackmore, Taunton

Nas-Graphpac

I have been catching up on my reading and came across the free dump of Graphpac in one of last year's issues. I have typed it in, and find it works perfectly on my Nascom 3 under NasSys 3 and NasDos, with ROM Basic. The only minor difficulty is in plotting or drawing circles in the pen-flip mode - presumably because some points on the circumference are plotted twice and hence flip twice (i.e. don't change). I find a ratio of 375 is right for my screen, and 400 or 415 to get a round circle in a screen dump to my MX80FT 2. Thanks for the magazine and the free software.

Yours truly, C R Case, Rugby

Utility Disk

I recently bought a copy of "Henry's Incredible CP/M Utilities Disk" for use on my ageing Nascom/Gemini system running CP/M Version 1.4. As the documentation tells you (when you have worked out how to decipher the DQC files - hint, TYPE file UTILITY.HLP), not all the goodies will run under CP/M 1.4. Worse, some of the most useful programs which were version independent, refused to perform, or had irritating quirks.

Fortunately, some very minor patches soon cured these problems. Here they are (addresses assume programs are loaded at 100H):-

HELP.COM For the NASCOM display, change the byte at address 02A3H from 17H to 0DH. This changes the page length to the correct value. Use DISKPEN or similar to reduce line length in the HLP files to 47 or less characters.

UNERA11.COM The program needs the track number at 03FAH to be changed from 02 to 03. I also changed the skew table (all of it - hence the repetition after 18 values) commencing at 03DFH to the values on the G805 Disk System. These are:-

01, 05, 09, 0D, 11, 03, 07, 0B, 0F, 02, 06, 0A, 0E,
12, 04, 08, 0C, 10, 01, 05, 09, 0D, 11, 03, 07, 0B.

CAT.COM & CATUD.COM This cataloguing suite was one of the most useful features of this disk, and it refused to work! I found later by accident that it functioned perfectly if I first hit the RESET key to disable SYS, then ran on the skeletal CBIOS in ROM. The clue to the problem was in the documentation, the only two programs of the suite which would not function under SYS were those that DRH had 'improved'.

The problem is calls to the BDOS to find the User code, with register C set to 20H. Replacing all these calls with three NOP's (00H) was crude, but seems to work. Locations were, in CAT.COM - 016AH, 0182H and 01C1H and in CATUD.COM - 0325H, 033CH and 0A30H.

None of the above changes are guaranteed bug-free, but they work on my system. Together with the disk-repair utility, the squash-unsquash programs, the SUB replacement to SUBMIT and the 8080 to Z80 translation program, all of which seem to function quite satisfactorily, these programs are an essential feature of any disk system (no, I don't get any commission!).

Yours truly, I P Forbes, Brighton.

Reply To Waldo

People of 80-BUS, greetings.

Being, as I am, an exceptionally cool person, and a lunatic of some years standing (piles, you know), I couldn't let Waldo 'D.R.' Dobbs' letter (80-BUS, Jan/Feb 84) go without comment.

As can be plainly seen from Waldo Dobbs' letter, the man is a genius! He has an insight into the ways of non-Nascom 'computer' users which can only be described as prophetic, or at least, inspired.

I know the above to be true, for, know you, I have seen the 'worthless diseased molluscs' creeping around the magazine racks in W.H. Smiffs. More, I have seen these poor misguided non-Nascom fools salivating over the displays of Vic 20's, ZX Spectrums, Orics and, may my larynx drop out for uttering it, BBC micros, in such shrines of capitalism as Dixon's and Curry's.

At the present time, I am in considering prosecuting B&T's The Chemist for breach of the Consumer's Act. Do you know what they did? They opened a department in their Cardiff branch in which they sell the above-mentioned 'computers', and they call it their Computer Department. Now, as we enlightened ones know, the definition of a computer department (in this context) is, and I now quote from Dai Pugh's classic work 'A Boyo's Guide to Mysteries of the Universe', "...a computer department, look you, is a section of a shop reserved for the sale of Nascoms, Geminis and 80-BUS-related products, look you, mun...".

Going back to the subject of Waldo Dobbs, I knew a guy called Waldo, in, like, one of my previous lives, you know. I wonder if it's the same Waldo? Do you know where Waldo Dobbs has his being? Wouldn't it be a real downer if it turns out that Waldo is nothing more than an A.I. algorithm in a program called Dobbs, heavy!! If you hear from Waldo Dobbs again, tell him I know a guy who knows a guy who grows the greenest weed this side of 'frisco. This stuff is g'teed to be a ticket to Cloud 9.

Talking of getting high, I hear that there will definitely be 6 issues of 80-BUS this year. It's already August and I've only had one issue so far, that means there are five issues to come between now and Xmas. Wouldn't that be a trip!

The PolyDos Users Group, which was the subject of my letter printed in Jan/Feb issue of the mag., is now under way (sort of). Membership of the group now stands at six, one of which was in response to the letter you printed.

As you've probably noticed, my letters tend to be a little on the 'Crazy' side, this is in an attempt to overcome the mind-numbing apathy which seems to go hand-in-hand with owning a Nascom.

Yours, until the end of this life,
'Head-crash' Richards, Tonyrefail, Glam.

And Waldo Again

Dear Slimewort, Are you like, totally blind, man? How anything can read SCUZZBALL as SAUZZBALL is something I find totally unbelievable, man. It should be obvious that a scuzzball is a monstrous and vicious citrus fruit, whereas a sauzzball is a totally, like, plastic imitation, man.

Like, finally, man, if you an issue devoted to the totally worthless greaseballs of amateur radio, you will find it inserted in a totally awesome part of your anatomy, man. Any circuits for spark-gap type RFI generators (for, like, N-1, with original NASBUG pre-T2 compatible software, man), would be mega-cosmically total, man.

Like, totally, (and very slightly less hazardous) yours, man.
Waldo 'D.R.' Dobbs (man!)

[Ed. - we couldn't read your writing in your first letter, and in this one the second paragraph makes no sense at all. Any offers to educate this man(?)
man? No prizes for guessing the missing word. Until next time, Mr Humphries?]

More About Nascom

Having just received Vol 3 Iss 1 of 80-BUS News, I was pleased to see the letter from Dr D Plews concerning the lack of interest from Lucas Logic.

I have had some correspondence with Lucas over the last six months and have been disgusted with their attitude, also their decision to cease supporting publication of the Nascom Newsletter, which has therefore died.

I also believe that Nascom is dead.

Lucas have told me they don't like the tone of my letters but they don't seem to realise that it was the loyal customer who made Nascom what it was and still could be. Several weeks ago, I wrote asking what would be available from Nascom in the future and also how I could legally join Prestel and MicroNet with my Nascom 2 but have had no reply. Lucas have brought out very few new products and don't even have a legal modem system for use with Prestel/MicroNet.

I know of many people who have sold their Nascom due to the lack of support and interest from Lucas. Their promises when taking over Nascom seem to have been forgotten. I have looked around at other computers but the Nascom is still the best and most versatile. It would appear now that my only option is to keep my Nascom 2 and expand it with 80-BUS products, but even so, is the legal Prestel/MicroNet expansion possible?

One final plea: With the death of Nascom Newsletter could we have a regular (say every 8 to 10 weeks) edition of 80-BUS News.

Yours truly, Robin Scadden (G3TFM9), Stratford-on-Avon

NASDOS

CLIFF WERNHAM would like to hear from any other NASDOS users who also feel cut off from the outside world! Phone 01-398-1948.

Introduction

This "article" is a collection of hopefully useful bits and pieces, on both the hardware and software front. It is hopefully the first of several such articles, assuming the Editor approves and deems this blurb fit for publication. It hopefully contains something of interest for everybody, but mention of a certain operating system is made occasionally.

HS1N Revisited

Some of you may recall that a while back I wrote a review of the HS1N digital cassette system for the Nascom. Somewhere in the article I mentioned the possibility of a Cassop-HS1N interface. What has happened to it? I hear you ask. Well, it's like this... At about the time the article was published, a friend at work, Tim, bought a Cassop system. Armed with the source code for the Cassop (and a tape with some Cassop files on it) I had soon patched my HS1N software to read the Cassop tape. Well, the catalogue actually, for some reason I couldn't read in any files. The effort on that sort of fell by the wayside as I proceeded to upgrade to disks (see below). The end result is that the interface software is half-written, and my trusty HS1N is up for sale, at a very resonable (i.e. cheap) price (see the Ads. section).

I had an interesting letter from a Mr I.P Coole of Imperial College, London. He too owns an HS1N, and has reported that the tape drives increase speed with age. This had lost him a lot of software before he found out what was causing it. Well, I haven't noticed any problems, and neither has Tim reported any problems, so maybe it is a batch problem. Maybe any other HS1N or Cassop owners can comment? The cure suggested by Mr Coole (short of using a frequency meter which he describes as "messy") is to always transfer all the required files onto a new tape before adding any files if the old cassette is more than a month old. I tended to do this anyway - I prefer assembler work, and the size of files I was using precluded the saving of more than two on one side of a tape. Thus when editing a file, I would always save the new version to a new tape. Being the only file on the tape, it would thus improve the load-time for next time. This may explain why I never noticed the speed problem.

Cheap Disks (or Doing it the Hard Way)

When I bought my Nascom 2 over four years ago, I did so primarily to learn about micros. But I kept on adding this and that to it, but at no time did I consider fitting disks to it. Disks are, after all, a bit too expensive for something that's just a hobby. That's why I bought my HS1N system in the first place. Shortly before my HS1N article was published, someone (I forgot who) in the local user group (NasTug) announced the possibility of getting some cheap disk drives. Pertec were discontinuing the FD250 drive and were flogging them untested, unguaranteed at some #50-#60 each. I saw my chance to finally upgrade to disks at a price I could afford, so I put my name down on the list.

Several months later, my name finally reached the top of the list, and despite comments of "Just wait till they fall apart" from A Certain Person (a shameful thing to say about a product his company was only too happy to sell to customers a few months previously), I collected my drives (and handed over the money). The drives had been tested by someone in the club and declared "working". A bit of high-pressure salesmanship by a certain local(ish) company ensured that nearly all of us used the (then) new Vfc. It made sense to me: there "ain't a lot of room" in my Kenilworth case, and anything that puts an

80x24 video and FDC on one card seemed like good news. Naturally, I bought the FDC-only kit version and it went together without any major snags (it needed a 'scope to set up, which I was able to borrow from work).

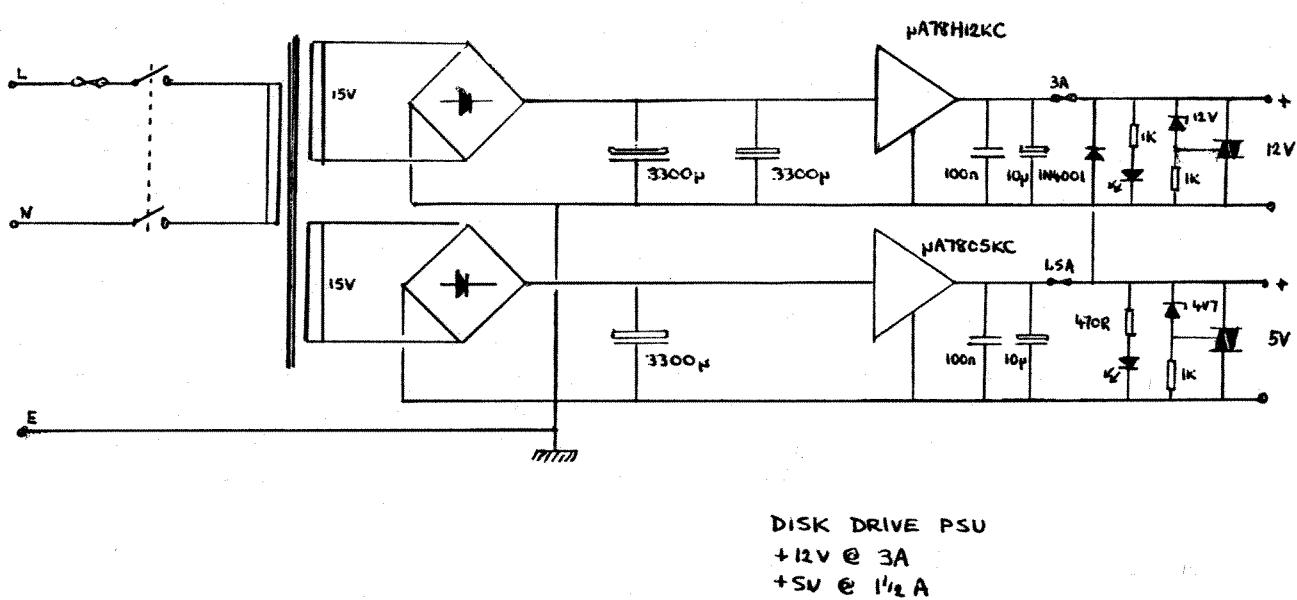
Next, I needed a PSU to power the drives. "No problem" I said, looking at the GM804K in Gemini's catalogue. "Don't do 'em any more" a smiling Dave Hunt informed me when I walked into Henry's to buy one. Well, I didn't want to buy the Gemini switch-mode job just to power a couple of disk drives, so I decided to design my own. The result is below. I hesitate to say "professionally designed", since I don't exactly earn my pennies doing hardware design. Anyway, the spec. is as follows:

12 volts at 3 amps

5 volts at 1.5 amps

both outputs have short-circuit protection, are fused, and are crowbar protected.

The design is fairly standard, and can be built for under #30, including the crowbar protection.

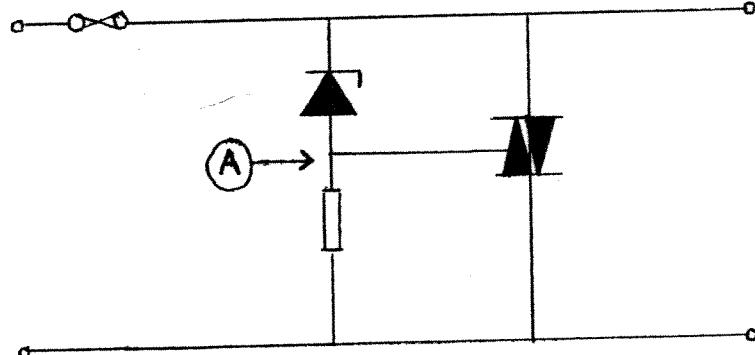


Protect Your Computer... With a Crowbar!

Many of you are probably wondering what the hell "Crowbar protection" is. Well, it isn't a way of protecting your prized computer against hoardes of plastic-box owners hell-bent on terrible revenge for all the nasty things we say about them... no, it's not that sort of crowbar. It is instead a neat little circuit you stick on the output of your power supply, and it ensures that should your 5v regulator in your computer PSU fail, and put unregulated voltage, or worse, mains, onto the 5v rail, your computer is safe. It works as follows: (see figure below).

Let us assume that all is well, and the PSU is outputting 5v. The voltage across the zener diode is 4.7v. This means that the voltage at point A is 0.3 volts. This isn't enough to trigger the triac, so nothing happens. Now let us assume that a fault has occurred, and the output voltage begins to rise. Because the voltage across the zener is constant at 4.7 volts, the voltage at point A will begin to rise. As soon as it reaches 0.7 volts, the triac turns

hard on, effectively short-circuiting the output and blowing the fuse. The triac will remain turned on until current ceases to flow through it (i.e. the fuse blows and/or the power is turned off). Note that the crowbar is fitted after the fuse: if it were fitted before the fuse, then should the fuse blow the computer would no longer be protected against peripherals putting what they like onto the 5v rail. I shall, in the best tradition of these things, leave it as "an exercise for the reader" to verify that the circuit also works should the power supply go more negative than -0.7v. It goes without saying that if you change the zener, you can use it on the other power rails.



CROWBAR PROTECTION CIRCUIT

Monitor.Com - The Bugfix(es).

Once I had got CP/M going, the first thing I did (alright - the second, first thing I did was to backup the master disk) was to install Monitor.Com, using NasSys 3 as the starting point. As many of you who are using this excellent piece of software are probably aware, there are a couple of minor "bugs" in the software, which are only significant if you (a) use overlays, and (b) like filling the disk up! Before describing how to fix these "features", I shall describe how I found them and what I did.

To help ease the transition to disk and CP/M, I modified Zeap, Nasdis, Debug to use the revised screen addresses, and I also took the opportunity to relocate them to 1000 hex. I was soon "REVASing" files, and then loading Zeap in over NasDis to edit the file. Needless to say it didn't appear to work: the start of the "REVASed" file appeared to be total garbage. The excellent debug facilities of NasSys soon pointed me in the right direction: the Fetch routine appeared to be loading the last 128-byte block twice. I verified (using DDT) that it wasn't a "feature" of CP/M itself, and so dis-assembled the disk drivers that Chris had added. I then referred to the CP/M manuals to find out what the code did.

It appeared that the Fetch routine was loading each 128-byte block, and then checking the BDOS error code. Thus when the end-of-file was reached, this wouldn't be detected until after the buffer (which contained the previously-loaded block) had been copied (again). Luckily, most of the disk drivers in Monitor.Com appear to be written in 8080 code, rather than more compact Z80 code, so the "patch" (what a horrible word "Patch" is!) is fairly simple. It consists of moving a few instructions and changing a few more instructions. (Note that you can use Monitor.Com for this one - anyone using DDT earns themselves the "Dodo of the Decade" award.) Note also that all addresses given assume that Monitor.Com was typed in at the same addresses as in the original (Micropower), and sub-sequent, articles.

- 1 - Move the bytes at #A21 to #A30 inclusive up to #A24 using the I command.
- 2 - Add the following bytes at #A21: B7 20 12.
- 3 - Add the following bytes at #A34: 18 E3.

The code should now look something like this:

```
getblock:
    ld de,fcb
    ld c,read_seq
    call jbdos
; New code follows
    or a          ; set flags
    jr nz,err     ; eof, so jump
; original code
    ld hl,buffer
    ld de,(arg1)
    ld bc,128
    ldir
    ld (arg1),de
; New code
    jr getblock
; original code
err:
    cp 1
    ...etc
```

This appears to work: I've had no more problems since patching this in. I also took the opportunity to change the bytes at #996, #9B5, #A50, #A65 and #A8D to #C9. This changes the SCAL ZMRET codes to simple RET codes, and enables the F and P routines to be called from user programs (eg the ":" command in Zeap).

Now for the second problem: the P routine appears to "hang" when the disk is full. The patch ain't so simple: it requires additional code on the end of Monitor.Com, and where this goes will depend on where your particular copy ends (I had tacked a printer driver onto the end of mine).

- 1 - Change the bytes at #97C to C3 xx xx where xxxx is some spare memory after your Monitor.Com (the code here should look like this):

```
ld de,fcb
ld c,write_seq
call jbdos
cp 0          ; Set error flags
jp z,write_block
; Here if error - disk full - New code
    jp full_error
; original code, no longer executed
    ...
```

- 2 - Add the following code at address xxxx :

```
11 5C 00 0E 10 CD 05 00 CD 24 09 EF 44 69 73 6B 20 66 75 6C 6C 2E 00 C9.
```

The code should look something like this:

Full_error:

```

ld de,fcb
ld c,close_file
call jbdos
call swap      ; Swap CP/M and NasSys page 0
rst prs
defm "Disk full."
defb cr,0
ret

```

A Fix for Sys

Many of you disk users who are running under a certain O.S. are doubtless using Richard Beal's excellent, although unfortunately no longer available, Sys overlay BIOS. I certainly am, and excellent though Sys is, I have encountered one or two minor problems. The "fix" I am suggesting here is to make the "Automatic Page Throw" code a bit more intelligent. One problem is that although my printer (an MX80 F/T) recognises form feeds, it doesn't do an automatic "skip over perforation", so I set the appropriate option to allow Sys to do this. When you do this, Sys converts form feeds to CR/LFs. I didn't like this, so I patched (there's that word again) Sys to output form feeds unaltered. A problem exists, however, in that if a form feed is issued when the printer is already at the top of a page, an extra page throw is sent which results in a page being wasted. Things are further complicated when you consider that some utilities may try to "help" by issuing extra CR/LFs to ensure the perforation is skipped. An example may be NasDis running under Monitor.Com. With Sys and Revas both supplying extra CR/LFs, the listing can get "out of step", so to speak.

The solution I have come up with is to make the "Skip perforation" code in Sys more intelligent: if the printer is already at the top of the page, Sys will "throw away" any extra form feeds, CRs and LFs that may be sent before any printable characters are sent for that page. This code hasn't had the fullest of tests, but I have checked it against a few trial runs of outputting files which, because I have also got the "line length" check built into Sys, had caused problems before. The listing is the part of module Sysb4.Mac, which contains the printer driver. Both "new" and "old" code is shown, with switches to select the one you want. This should help you if you want to type in my "additions".

[Ed. - Mr Perkins article now goes into a discussion as to the rights and wrongs of the current legal situation re. BIOS copyright, SYS, etc. As the comments he makes are wholly mis-informed they have been removed, as to comment on them would be sub-judice.]

NASIO

It's good to see that the Editor is trying to draw up a map of all the I/O ports used by the various boards available. However, the bit about Nasio support perhaps doesn't go far enough, since some manufacturers don't do Nasio properly. The 80-Bus specification says that Nasio is taken low to indicate a Nascom I/O address, which to me means whenever you try to access ports 0-7 inclusive, which are the ports used on the N2 card. Therefore it should be high at all other times. It is also an open-collector line. Now some manufacturers simply invert address line A7 and call it "Nasio". I know that Map do this on both the ram and vfc. The clue comes in their documentation, which says that the Nasio link is to be made "...only if no other card is generating Nasio". This implies that the line is NOT an OC line, and that bus

```

; Part of Sysb4.Mac. Original code by Richard Beal.
; Modified by Adrian Perkins 1-Jan-1984 to support the following:
; aip001 - Perform Page throw by form feeds, not CR,LF
; aip002 - When at top of page, "throw away" spurious CR, LF, CS
;
; aip001 equ true      ; true for form feed, false for CR,LF
; aip002 equ true      ; true for "intelligence", false original
;
; if tycs or lpcxs or ulcs
; Routine to support automatic form feed
; and perforation skip
; if aip002
top_flag:    defb 0      ; 0 = top-of-page, #0 = not at top
lhs_flag:    defb 0      ; 0 = at lhs of page, #0 = chars on this line
; Ignore CR, LF, CS if at top lhs
ckcs:        ld a,c
            cp lf
            jr z,ck_top
            cp cr
            jr z,ck_top
            cp cs
            jr z,ck_top
            cp cs
            jr z,ck_top
            ld hl, lhs_flag
            r
            ld (hl), 1
            jr ck_top
            ld a,(top_flag)
            r
            or a
            jr nz,ckcs1 ; If NZ not at top
            ld a,(lhs_flag)
            r
            or a
            jr nz,ckcs1 ; If NZ not at lhs
            pop hl
            ret
ckcs1:       ld a,c
            cks:      ld a,c
            endif
            cp lf
            jr z,lfl
            cp cs
            ret nz
            ; Handle CS
            pop hl
            ld (lsc1),hl
            ; Get return address
            r
            if aip001
            jr skips
            else
            ld a,(lcnt)
            ; Get line count
            r
            or a
            jr z,skip
            dec a
            ld (lcnt),a
            ; Perforation skip
            ; Next line
            r
            call feed
            ; Do CR/LF
            r
            jr cs1
            ; Loop until end of page
            ; Handle line feed
            if aip002
            lfl:    ld hl,top_flag
            r
            ld hl, lhs_flag
            r
            ld (hl),1
            ; Save return address
            r
            call feed
            ; Do LF
            r
            ld hl,lent
            ; Decrement
            r
            dec (hl)
            ; Get return address
            ret nz
            ; Get line counter
            if aip001
            ld c,cs
            endif
            ; Perforation skip
            if aip002
            skips:   ld hl,top_flag
            r
            ld (hl),0
            ld hl, lhs_flag
            r
            ld (hl),0
            else
            skips:   ld a,(lskip)
            r
            or a
            ; Could be none
            ld b,a
            skips:   ld a,(lskip)
            r
            or a
            ; Lines to skip
            call feed
            r
            pop bc
            dinz skips1
            endif
            cksx:    ld a,(plpag)
            r
            ; Reset line counter
            pop bc
            r
            ld (lent),a
            r
            if aip002
            jr feed2
            else
            ret
            endif
            ; Send CR/LF through the calling routine
            feed:   ld c,cr
            jr feed
            r
            ld hl, (lsc1)
            ; Caller address
            r
            push hl
            ; Return address
            r
            jp (hl)
            ; Caller address
            if aip002
            feed2:  ld hl, (lsc1)
            r
            endif
            ld hl, (lsc1)
            ; Caller address
            r
            jp (hl)
            endif

```

conflicts would ensue if more than one card tried to generate Nasio. This is borne out by the circuit diagram for the ram. (Incidentally, Map say that the circuit diagram for the VFC is "now available".) Now some of you may look up the NasBus issue 3 spec (in the N2 manual) and cry "Nasio is not an OC line!", so who is right? Well, my Nascom I/O card generates Nasio correctly, and treats it as an OC line, so I guess it was either a misprint or Nascom changed their minds.

I know this may seem a trivial point to go on about, it's just that I like all I/O peripheral cards to generate Nasio, so that I can unplug, say, the I/O card and still have a working Nascom. I initially connected the Nasio link on my Map VFC, as well as having the I/O card generating Nasio, and consequently took a VERY long time getting the RS232 on the I/O card to work properly, all because the I/O card and the VFC were arguing over Nasio.

RS232

This brings me neatly onto my next topic. I was surprised to read that Rory O'Farrell has been having problems PIPing COM files over the RS232 interface. A short time ago I had a similar problem, in that I needed to get some files transferred from an ICL PC to my Nascom. I tried using PIP, and like Rory didn't have much success. It would open the file, but didn't seem to "know" when the ICL PC had finished transmitting. The file was probably in memory, but I didn't know where. I also encountered problems of different buffer sizes (no RS232 handshaking!). I solved this by writing a very simple program to run under Monitor.Com which read characters from the serial interface into memory. Then when the ICL PC had finished transmitting, I pressed RESET, re-loaded Monitor.Com, and, because the cold boot loader leaves the TPA alone, there was my file in memory, waiting to be saved.

The ICL PC had preceeded the file by sending about 40 NULLs, and terminated the file with about 40 more nulls (and a control-z character!), but by using DDT on the ICL PC I was able to determine the exact size of the file, and how it began, so it was easy to identify where the required program was. Very simple, and so much easier and quicker than using a disassembler!

Re-Inventing The Wheel

It has occurred to me while upgrading to CP/M that I would need to translate all my Zeap files to Macro-80 format, and all my Nascom Basic files to MBasic format. This will doubtless require the writing of some simple (or not so simple?) program. Now how many other people have written such utilities? Quite a few, I would imagine. What a total waste of time and effort if such routines already exist! I wholeheartedly agree with Dr. Darks comments about "idle bodies" in this context. Such routines would be so massively useful to Nascom (and Gemini owners who have switched from Nascom?) that I am sure Mr Editor would publish such an article were it to be submitted. This is why Dr. Dark's "Ring of Rust" software circle has got to be a good idea, in that everyone can swap such useful little utilities. As I write, I am not on the "ring" yet, but then I only wrote to Chris a week or so ago asking to be included.

Why have I not written such utilities yet? Well, I have just written a Zeap to Macro-80 preprocessor, which will be put on the "ring" in due course, and, of course, written up into an article for this magazine just as soon as I have written a Nascom Basic to MBasic preprocessor.

Another Hardware Tip

Here is a hardware tip that is so massively useful, and yet has never been put into print... The scenario: you've just built (from a kit) the latest goodie for your computer. The time comes to check the board for bad joints, etc. Problem: the board is covered in flux splashes, making it difficult to check the board properly. I have discovered that proprietary Dry-Cleaning fluid is excellent at removing (fresh) flux from boards.

The technique is simple. Hold the board to be cleaned upside down (solder side uppermost), and at an angle (with the edge connector at the top). Apply a smallish amount of the fluid to the board, enough to wet the board. Using an old toothbrush (or similar), scrub the solder joints to ensure that the flux is loosened and dissolves in the fluid. Allow the fluid to run to the bottom of the board, carrying the flux with it (now you see why the edge connector has to be at the top - flux is a good insulator!), and allow to drip off the board and evapourate. Repeat with another small amount of fluid to "rinse" the board. If, after this lot has evapourated the board is sticky to the touch, there is still some flux left on the board, and another scrub/rinse is required.

There are several points to note: this stuff evapourates very fast, so unless you can afford to "slosh it around" work on small areas of the board at a time. Second, and more important, this stuff is POISONOUS, so make sure the room is WELL VENTILATED (don't want to lose any readers!). Third, for some reason this only works on fresh flux: it won't work on that board you made six months ago. This is probably due to the flux oxidising, or something.

The end result will be a VERY professional finish, and it will be simplicity itself to find bad soldered joints. I tried this on my Ram after it didn't work first time, and was surprised how many bad joints this revealed. Once these had been fixed, I plugged my Ram back in, and it burst into life (so to speak).

CP/M 3

There has been a little discussion on CP/M 3 in this magazine, together with Mr Editor's thoughts on the matter. I know next to nothing about CP/M 3 except what has already been said in these hallowed pages. Now some of you may be wondering why Map80 are supplying CP/M 3 when Gemini aren't. Well the reason is simple (so I am told). CP/M 3 works with the MapRam, but Gemini can't get it to work on their GM833 Ram Disk! Mind you, my source was rather biassed... [Ed. - Yes, it was! Gemini could have supplied CP/M Plus (that is its official name) some 18 months ago, when they had it running using multiple GM802 RAM boards. However they decided that there would be insufficient demand given the market at that time, and the costs (to the customer) involved. With the arrival of the Gemini GM862 256K board I suppose there is always an outside chance that Gemini may change their minds, but I wouldn't hold my breath waiting]

Open Letter to Richard Beal & Dave Parkinson

In one of his book reviews, Rory said "oh, for a book on CP/M using Z80 mnemonics!". Doubtless the entire disk-based readership of this magazine agreed. So some on you two, you know more about CP/M than the rest of us put together, so how about a Z80 CP/M book? (It could even be published by the 80-BUS News Publishing Company Ltd Plc Inc...).

DOCTOR DARK'S DIARY - EPISODE 21.

This looks as if it will be a short article, judging by the notes I have made, which are also short. The plain fact of the matter is that I have been wickedly neglecting the trusty Marvin for another computer! Actually, several computers, some of which are quite large, while one of them is a Spectrum. The non-Spectrum computers belong to British Telecom. Well guessed, it's a not very obscure reference to Prestel.

For the benefit of anyone from another planet who may be reading this, Prestel is a mighty collection of databases you can access by phone, if you have the right gear to do it. At present, Marvin is not the right gear at all. I hope that one day he will be, however, even though the technical people at Micronet thought I was joking when I rang them up to ask about the chance of connecting a Nascom. All it will need is a suitable modem and the easy bit, some software (a joke). The modems available at the moment are quite good, but I think I can wait long enough to be able to get a very good one for less money. The Miracle one, with autodial and auto-answer, costs about £200 by the time you have added VAT and the like. Once it has a bit more in the way of competition, perhaps the prices will slide down. Then I will be able to flog the Spectrum and VTX 5000 modem I am using, and put Marvin to work as a Prestel terminal, as well as all his other uses.

The ability to store frequently read Prestel pages on disc, so that they can be read without piling up telephone call charge units would be nice, as would the clarity of the Pluto display, compared with the blur of a Spectrum! Now that you know I am on Pretzel, you know how to send messages to me without all the expense of buying postage stamps. Incidentally, for those of you wondering if you should subscribe to Micronet 800, I can honestly say that they are hopelessly slow to update their pages, so much so that they make Micropower magazine look regular (it is, it just happens to be an annual!) by comparison. The letters to the editor, which Micronet say is updated "at least twice a week" have now stayed the same for in excess of three weeks. The main thing the "micro-news" reporters write about is what it is like to be a micro-news reporter, what they ate at the Sinclair QL's launch, what they drank at the Microfair, and so on. They are nearly as bad as yours truly...

Mention of the possibility of a price slide for modems has reminded me of another thing that should have slid, but has not. The price of 4164's seems to be very stable, which is not good news, as I have a nice new 256K RAM board with 32 empty sockets. The terrible world shortage of silicon must be to blame! Joking aside, the shortage of RAMs and TTL is getting to be a real pain. My new board has 33 sockets empty, actually, as Maplin have let me down. I tried to buy a 74LS32 from them, a simple enough request, you might think. But no! The message from their computer was "code D", which on translation seems to mean "we haven't got them, and we don't know when we will get them, and we bet you can't get them anywhere else before we get them". I also got the impression that they didn't much care, either. Maybe one of the other firms will come up with the goodies... (Generous readers with spare 74LS32's, please don't send them! I have just spotted one I can pinch from a board I am not using at present!)

Spectrum Tape Format Conversions.

I have been sent a couple of interesting programs to do with this. Dr M D Hendry sent a machine code program to run on the Spectrum, which makes it dump to tape in Nas-Sys tape format at 300 Baud. Very crafty, and I think I suggested he should send it in for possible publication. Dennis McLaren sent in a Nascom machine code program that reads Spectrum tapes, dumping the result

in memory. It is quite short, and I would have put it here, but for two things. One, I would have deprived the author of the chance of getting paid for his work. Two, the program doesn't distinguish between the header blocks and the program data, it just stores everything it reads. But the amount of tape format compatibility between the two machines is a surprise. If I ever find the time, I really must write a program to transfer things to and fro properly. Or perhaps one or both of the above named programmers will get it together first, and save me the bother!

More Bit Power Please!

I thought I saw an advertisement for a sixteen bit Galaxy computer recently! If there is such a thing in existence, it would be nice to know more about it, such as what processor it uses, whether the board is to 80-BUS standard and whether it can be bolted onto Marvin. If the manufacturers would like it reviewed by me, they have only to pop one in the post! Meanwhile, I have sent a request via Prestel to the ACC for information about their 68000 board, which could well make a nice project for me. I have not built anything for a long time, and feel the need to solder something! I was not able to get the 64 transputers I wanted...

General Stirring.

There must be more than one school using Nascoms, judging by the different photographs one sees in various computer programmers' magazines. It would be nice to hear from the people who are using these machines, and to know what they are using them for, whether they read this magazine, and whether they want any kind of help from us!

And if I was in charge of the Nascom part of Lucas, I know which magazine I would send my newsletters to, so that they would be published in the year they were written!

Sorry this is so short, but I have to get back to M203, the Open University cure for sanity, otherwise I will never know enough maths for the unbelievable graphics program I will one day get round to writing!

END.

SMALL ADS

Wanted.

Pertec FD 250 double-sided, single-density disk drive, suitable for GM805 system. J M Harding (051-608) 1796.

For Sale

Nascom 2 cased, with 64K RAM, 'A' board (4 MHz, no waits). #220 ono. Cliff Wernham, 01-398 1948.

Nascom 2 with 32K RAM, BASIC toolkit, graphics ROM, assembled in case, with manuals, #100. Crofton 9" green screen monitor, #40. J Curtis, 01-669 7061, (Wallington, Surrey).

Nascom 2, 32K RAM, together with IVC card, GM815 Dual Floppy Disk System and GM809 Controller card, Grange micro-cassette system, Imprint ROM, and PolyText, PolyDos and full documentation/manuals on above. Offers please on whole system or parts thereof - any reasonable offer considered. Call Peter Wood, Potters Bar 54896.

A POLYDOS Version Of NasPenBy Alan Melia

Because many Nascom users, who, like myself, started with cassette systems, would have bought and installed NASPEN, it was felt that a POLYDOS compatible version might be more acceptable than a change to say POLYTEXT. Personally, being mean, and having spent my pocket money for this month, I wanted to get my cassette text files onto this excellent disk operating system. The advantage of POLYDOS is a good manual, with some very useful examples of how to use the PolyDos subroutines, and a relatively simple guide to producing purpose-built file-handling overlays. Armed with these tools and a trusty Nasdis listing of my Naspen, I set to work to try and produce a usable system.

First Naspen was copied to disk, and the ROM was removed to leave RAM at B000 to BFFF. A careful investigation of the Nasdis listing showed that the NASPEN command decoding table started at B999H. The table consisted of a three byte entry for each command, the table being terminated by a catch-all jump to a 'command error' subroutine. The first byte is the ASCII value of the command symbol, the second and third give the address of the routine to execute the command (bytes reversed, of course). I determined to reallocate the "R" and "W" commands to the new disk routines to retain some compatibility with the original usage. I further decided that I required to keep the cassette routines to retain compatibility with several other local 'Nascomaniacs'. A skim through the handbook for Naspen will confuse by offering duplicate commands for cursor-up and cursor-down, "+" & "-" as well as the normal cursor keys. Thus I determined to replace the "+" & "-" with commands "r & w" for cassette read and write, and altered the "R & W" entries in the table to point to the new disk routines in the overlay. This means that only a simple patch is required, rather than a full re-assembly job. A further modification that was found necessary was to re-program the "return to Nas-sys". The original Naspen return reinitialises Naspen by jumping to 0000H rather than that well-loved line, DF 5B. Unfortunately it was not sufficient to add the "DF 5B" code, as the reentry to PolyDos leaves the keyboard in "typewriter" mode. So a few more bytes of code were required to reset "K0". First, a separate file was written and debugged. This file contained the code necessary to access the PolyDos routines, and had to be loaded before Naspen was called. The ploy worked quite well, but flushed with success, I decided that it would be far tidier if the extra file was loaded automatically when Naspen was called. The procedure used was to rename Naspen with an extension "NP" instead of the more usual "GO" machine-code extension. An overlay routine was then written and called "NPfh.OV". The overlay program is loaded and executed automatically when you type "NASPEN". The overlay loads NASPEN at B800 and the extension file "NASPEX" at B000. It then initialises the printer and cold-starts NASPEN. The sequence of operations is then as follows:-

- 1) Call Naspen by typing "NASPEN" after the "\$"prompt
- 2) PolyDos sees that NASPEN has the extension "NP" so it loads the file "NPfh.OV" into the overlay area (C800 onwards), and executes it.
- 3) NPfh.OV loads the NASPEN file and the extensions "NASPEX.GO" and jumps to the start of NASPEN at B800H. If this confuses, read the POLYDOS manual on overlay file handlers.

```

;***** NASPEN extension file "NASPEN.G0"*****
;      SYSQU      REFS
;      REF          SYSQU      REFS
;      ; This Program is an add-on for NASPEN to allow
;      ; text files to be loaded and dumped to disk via
;      ; POLYDOS using the NASPEN commands W & R .The
;      ; tape commands are retained by replacing the
;      ; + - commands by r & w. This program and the
;      ; main Naspen code are loaded by the overlay
;      ; 'NPFH.RV' which also performs a cold start.
;      ; The printer routine intercepts the codes and
;      ; converts a "CTRL D" into an "escape" code
;      ; which should allow printer control strings to
;      ; be embedded in the text.
;
;      WARMST EQU 0B806H
;
;      ; ORG 0B000H
;      ; IDNT $,$
;      ; JP WARMST      ; Warm start NASPEN from B000H
;      ; JR DREAD       ; Read file routine is at B003H
;      ; DWRITE        ; Write file entered from B006H
;      ; PRINT         ; Exit routine (resets printer)
;      ; QUIT          ; SET UP FOR
;      ; LD HL,0         ; NASSYS KO (UPPER CASE DEFAULT)
;      ; SCAL          ; BACK TO POLYDOS
;      ; ZMRET
;
;      ;***** Text entered from disk *****
;      ; Comments as POLYDOS manual
;      DREAD RST PRS
;      ; Load which file? ,0
;      DB ZINLIN
;      SCAL HL,DE
;      ADD DE,HL
;      EX DE,HL
;      D1 LD HL,S1FCB
;      LD B,110B
;      SCAL ZCFS
;      SCAL ZCKER
;      SCAL ZRDIR
;      SCAL ZCKER
;      SCAL ZCKER
;      SET 4,B
;      SET 5,B
;      SCAL ZLOOK
;      SCAL ZCKER
;      LD HL,(S1FCB+FLDA)
;      LD DE,(S1FCB+FSEC)
;      LD A,(S1FCB+FNSC)
;      LD B,A
;      SCAL ZRDIR
;      SCAL ZCKER
;
;      DWRITE RST DB
;      ; File name? ,0
;      PRS
;      ; Comments as POLYDOS manual
;      DB ZINLIN
;      SCAL HL,14
;      ADD HL,DE
;      EX DE,HL
;      LD HL,S1FCB
;      LD B,100B
;      SCAL ZCFS
;      SCAL ZCKER
;      SCAL ZDIR
;      SCAL ZCKER
;      LD HL,0
;      LD (S1FCB+FSFL),HL
;      LD HL,(NFTSEC)
;      LD (S1FCB+FSEC),HL
;      EX DE,HL
;      LD A,101BH
;      SUB 1OH
;      LD B,A
;      LD A,(101AH)
;      OR A
;      JR Z,P1
;      INC B
;      PUSH BC
;      LD HL,0
;      LD L,B
;      LD (S1FCB+FNSC),HL
;      LD HL,1000H
;      LD (S1FCB+FLDA),HL
;      LD (S1FCB+FEXX),HL
;      LD BC
;      POP BC
;      SCAL ZDIR
;      SCAL HL,S1FCB
;      CALL ENTR
;      RET
;      SCAL ZENTER
;      RET Z
;      CP 31H
;      RET NZ
;      PUSH HL
;      LD HL,FSFL
;      ADD HL,DE
;      BIT 0,(HL)
;      LD A,33H
;      JR NZ,SKIP
;      SET 1,(HL)
;      POP HL
;      JR ENTR
;      POP HL
;
;      SKIP RET
;      ; THIS IS A ROUTINE FOR THE EPSON RX80 PRINTER
;      ;with CENTRONICS INTERFACE

```

```

;***** POLYTOS Overlay "NPfh.OV" *****
REFS    SEQQU
REF    0B007H
PRINT   EQU
;This Program is a POLYTOS file handler for
;NASPEN. This overlay allows text files to be
;loaded and dumped to disk via an extension
;program NASPEX. The extension program
;adds the disk facilities using the NASPEN
;commands "W" & "R". The cassette commands
;are retained by replacing the "+" & "-"
;commands by "r" & "w". The extension file is
;loaded at 8000H, and contains a printer driver
;for the Epson printer allowing some of the
;control codes to be output to the printer. This
;overlay initialises the printer on a cold start
ORG    0C800H
IDNT   $,0
DB     'NPFH'
DE     ,FNAME1
LD     DE,FNAME1
CALL   D1
LD     DE,FNAME2
CALL   D1
LD     DE,FNAME3
CALL   D1
JP     0B800H
;Cold start NASPEN
;***** DISK FILE LOADING ROUTINE *****
D1    HL,SIFCB
LD     B,110B
LD     B,110B
SCAL   ZCFBS
SCAL   ZCKER
ZDIR   ZDIR
SCAL   ZCKER
SCAL   ZCKER
SET    4,B
SET    5,B
SCAL   ZLOOK
SCAL   ZCKER
HL,(SIFCB+FUDA)
LD     DE,(SIFCB+FSEC)
LD     A,(SIFCB+FNSC)
LD     B,A
ZDRD   ZDRD
SCAL   ZCKER
RET
;***** PRINTER INITIALISATION *****
PRINIT HL,INIT$G
LD     B,(HL)
INC   HL
A,(HL)
CALL   PRINT
INC   HL
DNNZ   P2
CALL   PRINT
INC   HL
P2
;FILES TO BE LOADED BY OVERLAY ROUTINE
FNAME1 DB  'NASPEN.NP' ;MAIN NASPEN CODE
FNAME2 DB  'NASPEX.G0' ;FILE WITH EXTENSIONS
INIT$G DB  0BH,1BH,40H,43H,42H,1BH,4EH,6,1BH
6CH,8
END

```

;

BASE EQU 4H

CENPAD EQU BASE

CENPBD EQU BASE+1

CENPAC EQU BASE+2

CENPPC EQU BASE+3

;

;INITIALISE THE PORTS, PORT B TO 0/P, PORT A B+D

PRINT PUSH BC

PUSH DE

PUSH HL ; SAVE CHARACTER

START PUSH AF

CALL INIT

;EXAMINE BUSY LINE TIL FREE

LOOP IN A,(CENPAD)

RRA

JR C,LOOP

;PRINTER FREE SO SEND A CHAR

POP AF

PUSH AF ; Is it "CTRL D"

CP 4 ; no

JR NZ,NOTESC ; yes, output an "escape" code

LD A,1BH

NOTESC OUT (CENPBD),A

NOP

LD A,0FH

OUT (CENPAD),A

NOP

LD A,0FH

OUT (CENPAD),A

POP ODH

CP NZ,DONE ; is end of line

JR LD A,0AH ; no

START JR QAH ; yes, then output a lf char.

DONE CP JR NZ,NOTLF ; did we op a lf?

LD A,0DH ; no

NOTLF POP HL ; yes, put back cr for RET to main

POP DE ; routines

POP BC ; get back registers

RET

INIT LD A,0FH ; set up PIO for centronics H/S

OUT (CENPAD),A

OUT (CENPBD),A

OUT (CENPAC),A

LD A,0FH

OUT (CENPAC),A

LD A,0FH

OUT (CENPPC),A

A XOR (CENPPC),A

OUT (CENPPC),A

RET

END

The extension file is a chance to add some extra facilities to NASPEN. I have included a printer driver for a centronics driven RX80. The main reason for this was that I found I had three program modules fighting over the print format if I used the built-in POLYDOS routines. It also gave the opportunity to invent a way of putting control sequences to printer. The main problem is that Naspen will not store an "escape" character, because this code is used to terminate a Naspen command. I have used a "Control D" which can be stored in with the text, to replace the "escape". This means that most if not all of the print-modes of the EPSON RX80 can be switched on and off, at will, by embedded "escape" strings. A further addition of the extension file is that the system can be restarted by a NAS-SYS "EB000" command which warm-starts NASPEN. The overlay file will have been lost, because the "Exec" overlay will have been loaded by the return to POLYDOS but it is not required once the main Naspen programs have been loaded and cold-started the first time. The warm start proved most necessary with the disk system, because it is easy to call a wrong file-name, or one that already exists. Either of these errors will blast you back into POLYDOS command level. (Rather like a touch of the snakes and ladders.) Obviously a rather more sophisticated file handling routine would allow for default file extensions and would automatically overwrite an existing file, renaming the existing version as a ".BK" backup version. The fun of getting that going is all yours!! An even more useful addition would be to toggle the screen with some unused memory and allow the operator to view the disk directories whilst in Naspen.

Some further minor changes were made to Naspen to indicate differences from the original prom version, the title was changed to read "NASPEN PDOS4". The initialisation routine was changed to give 32000 bytes free instead of the original 11000. Although this is supposed to slow down the operation of some commands, I have not found it as big a disadvantage as the space restriction.

There are a few points to take note of before committing your epic novel, or autobiography to "Polypen". DON'T use the "NP" extension for any other files. I use the extension "nt" for my Naspen text files. Remember to patch the printer reflection on your files before you submit them to "Polypen". The disk save routines save all memory from 1000H.

To avoid compromising copyright on the PolyDos listing I have not included the comments. If you have bought PolyDos you will have a manual which explains all. I was amazed by the ease with which I managed to effect these changes, and it speaks volumes for the quality of the manual that the first successful assembly of the file-writing routine worked first time. (For a comparison note how many non-Digital Research textbooks are recommended for CP/M tinkerers).

Next Issue

Subject to probable change the following items should appear in the next issue of 80-BUS News, a special 'back-from-holiday' issue !

- a review of the new PEN
- Dave Hunt back from holiday
- the FINAL part of the BASIC disassembly
- Part 2 of the IVC/SVC Insight
- Lawrence back from holiday
- an improved(?) NAS-SYS 3
- Aunt Agatha back from holiday
- 128K on a RAM A !!

Is there any truth in the rumour that Dave Hunt, Aunt Agatha and Lawrence all went away together?

Dis-assembly of NASCOM ROM BASIC Ver 4.7

		PAGE	77
F75B CD44F8	DIV10: CALL STAEEP	; Save FPREG on stack	
F75E 012084	LD BC, 8420H	; BCDE = 10.	
F761 110000	LD DE, 0000H		
F764 CD54F8	CALL FPBCDE	; Move 10 to FPREG	
F767 C1	DIV: POP BC	; Get number from stack	
F768 D1	POP DE		
F769 CD13F8	DVBCDE: CALL TSTSZN	; Test sign of FPREG	
F76C CAB0E3	JP Z,DZERR	; Error if division by zero	
F76F 2EFF	LD L,-1	; Flag subtract exponents	
F771 CDD1F7	CALL ADDEXP	; Subtract exponents	
F774 34	TNC (HL)	; Add 2 to exponent to adjust	
F775 34	INC (HL)		
F776 2B	DEC HL	; Point to MSB	
F777 7E	LD A,(HL)	; Get MSB of dividend	
F778 321210	LD (DIV3),A	; Save for subtraction	
F77B 2B	DEC HL		
F77C 7E	LD A,(HL)	; Get NMSB of dividend	
F77D 324010	LD (DIV2),A	; Save for subtraction	
F780 2B	DEC HL		
F781 7E	LD A,(HL)	; Get MSB of dividend	
F782 320A10	LD (DIV1),A	; Save for subtraction	
F785 41	LD B,C	; Get MSB	
F786 EB	EX DE, HL	; NMSP, LSB to HL	
F787 AF	XOR A		
F788 4F	LD C,A	; Clear MSB of quotient	
F789 57	LD D,A	; Clear NMSB of quotient	
F78A 5F	LD E,A	; Clear LSB of quotient	
F78B 321510	LD (DIV4),A	; Clear overflow count	
F78E E5	DIVLP: PUSH HL	; Save divisor	
F78F C5	PUSH BC		
F790 7D	LD A,L	; Get LSB of number	
F791 CD0910	CALL DIVSUP	; Subt' divisor from dividend	
F794 DE00	SBC A,0	; Count for overflows	
F796 3F	CCF		
F797 D2A1F7	JP NC,RESDIV	; Restore divisor if borrow	
F79A 321510	LD (DIV4),A	; Re-save overflow count	
F79D F1	POP AF	; Scrp divisor	
F79E F1	POP AF		
F79F 37	SCF	; Set carry to	
F7A0 D2	DEFB (JP NC,mn)	; Skip "POP BC" and "POP HL"	

NASCOM**ROM****BASIC****DIS—ASSEMBLED****PART 6**

BY CARL LLOYD—PARKER

Dis-assembly of NASCOM ROM BASIC Ver 4.7		PAGE	78		PAGE	79
F7A1 C1	RESDIV: POP BC	; Restore divisor	F7EF CD13F8	OVTST1: CALL CPL	TSTSZN	; Test sign of FPREG
F7A2 E1	POP HL	; Get MSB of quotient	F7F2 2F	POP	HL	; Invert sign
F7A3 79	LD A,C	; Put it back	F7F3 E1	POP	HL	; Clean up stack
F7A4 3C	INC A	; Get NMSB of quotient	F7F4 B7	OVTST2: OR	A	; Test if new exponent zero
F7A5 3D	DEC A	; Double it	F7F5 E1	OVTST3: POP	HL	; Clear off return address
F7A6 1F	RRA	; Bit 0 to bit 7	F7F6 F233F6	JP P,RESZR,	OVERR	; Result zero
F7A7 FA54F6	JP M,ROUND	; Done - Normalise result	F7F9 C3BCF3	JP		; Overflow error
F7AA 17	RLA	; Restore carry	F7FC CD5F78	MLSP10: CALL BCDEFP	BCDEFP	; Move FPREG to BCDE
F7AB 7B	LD A,E	; Get LSB of quotient	F7FF 78	LD A,B	A,B	; Get exponent
F7AC 17	RLA	; Double it	F800 B7	OR A	A	; Is it zero?
F7AD 5F	LD E,A	; Put it back	F801 C8	RET Z	Z	; Yes - Result is zero
F7AE 7A	LD A,D	; Get NMSB of quotient	F802 C602	ADD A,2	A,2	; Multiply by 4
F7AF 17	RLA	; Double it	F804 DABCE3	JP C,OVERR	C,OVERR	; Overflow - ?OV Error
F7B0 57	LD D,A	; Put it back	F807 47	LD B,A	B,A	; Re-save exponent
F7B1 79	LD A,C	; Get MSB of quotient	F808 CDCDF5	FPADD CALL	CALL	; Add BCDE to FPREG (Times 5)
F7B2 17	RLA	; Double it	F80B 21E710	LD HL,FPEXP	HL,FPEXP	; Point to exponent
F7B3 4F	LD C,A	; Put it back	F80E 34	INC (HL)	INC (HL)	; Double number (Times 10)
F7B4 29	ADD HL,HL	; Double NMSB,LSB of divisor	F80F CO	RET NZ	NZ	; Ok - Return
F7B5 78	LD A,B	; Get MSB of divisor	F810 C3BCE3	JP OVERR		; Overflow error
F7B6 17	RLA	; Double it				
F7B7 47	LD B,A	; Put it back				
F7B8 3A1510	LD A,(DIV4)	; Get VLSB of quotient	F813 3AE710	TSTSZN: LD A,(FPEXP)	A,(FPEXP)	; Get sign of FPREG
F7B9 17	RLA	; Double it	F816 B7	OR A	A	; RETurn if number is zero
F7BC 321510	LD (DIV4),A	; Put it back:	F817 C8	RET Z	Z	; Get MSB of FPREG
F7BF 79	LD A,C	; Get MSB of quotient	F818 3AE610	LD A,(FPREG+2)	A,(FPREG+2)	; Test sign
F7C0 B2	OR D	; Merge NMSB	F81B FE	DEFB (CP 2FH)	(CP 2FH)	; Invert sign
F7C1 B3	OR E	Merge LSB	F81C 2F	RETREL: CPL		; Sign bit to carry
F7C2 C2BEF7	JP NZ,DIVLP	; Not done - Keep dividing	F81D 17	RLA		; Carry to all bits of A
F7C5 E5	PUSH HL	; Save divisor	F81E 9F	FLGDL: SBC A,A	A,A	; Return -1 if negative
F7C6 21E710	LD HL,FPEXP	; Point to exponent	F81F CO	RET NZ	NZ	; Bump to +1
F7C9 35	DEC HL	; Divide by 2	F820 3C	INC INC A	INC A	; Positive - Return +1
F7CA E1	POP HL	Restore divisor	F821 C9	RET RET		
F7CB C2AEF7	JP NZ,DIVLP	Ok - Keep going	F822 CD13F8	SZN: CALL TSTSZN	TSTSZN	; Test sign of FPREG
F7CE C3BCE3	JP	Overflow error	F825 0638	FLGREL: LD B,80H+8	LD B,80H+8	; 8 bit integer in exponent
F7D1 78	ADDEXP: LD A,B	Get exponent of dividend	F827 110000	RETINT: LD DE,0	DE,0	; Zero NMSB and LSB
F7D2 B7	OR A	Test it	F82A 21E710	RETINT: LD HL,FPEXP	HL,FPEXP	; Point to exponent
F7D3 C4FF7	Z,OVTST3	Zero - Result zero	F82D 4F	LD C,A	C,A	; Save exponent
F7D6 7D	LD A,L	Get add/subtract flag	F82E 70	LD (HL),B	(HL),B	; CDE = MSB,NMSB and LSB
F7D7 21E710	LD HL,FPEXP	Point to exponent	F82F 0600	LD B,0	B,0	; Point to sign of result
F7DA AE	XOR (HL)	Add or subtract it	F831 23	INC HL	INC HL	; Set sign of result
F7DB 80	ADD A,B	Add the other exponent	F832 3630	LD (HL),80H	LD (HL),80H	; Carry = sign of integer
F7DC 47	LD B,A	Save new exponent	F834 17	RLA CONPOS	CONPOS	; Set sign of result
F7DD 1F	RRA B	Test exponent for overflow	F835 C3ABF6	JP		
F7DE A8	XOR B	Get exponent				
F7DF 78	LD A,B	Positive - Test for overflow				
F7E0 F2F4F7	JP P,OVTST2	Add excess 128				
F7E3 C680	ADD A,80H	Save new exponent				
F7E5 77	LD (HL),A	Zero - Result zero				
F7E6 CA54F7	Z,POPHRT	Set MSBs and sign of result				
F7E9 CD79F8	CALL SIGNS	Save new exponent				
F7EC 77	LD (HL),A	Point to MSB				
F7ED 2B	DEC HL	RET				
F7EE C9						

Dis-assembly of NASCOM ROM BASIC Ver 4.7

PAGE 80

PAGE 81

Dis-assembly of NASCOM ROM BASIC Ver 4.7

F838 CD13F8 ABS: CALL TSTSGN ; Test sign of FPREG
 F83B F0 RET ; Return if positive
 F83C 21E610 INWSGN: LD HL,FPREG+2 ; Point to MSB
 LD A,(HL) ; Get sign of mantissa
 80H ; Invert sign of mantissa
 LD (HL),A ; Re-save sign of mantissa
 RET ; Old sign to carry

F844 EB STAKFP: EX DE,HL ; Save code string address
 LD HL,(FPREG) ; LSB, NLSB of FPREG
 EX (SP),HL ; Stack them, get return
 PUSH HL ; Re-save return
 LD HL,(FPREG+2) ; MSB and exponent of FPREG
 LD (SP),HL ; Stack them, get return
 EX HL ; Re-save return
 PUSH DE,HL ; Restore code string address
 EX DE,HL ; Restore code string address
 RET ; New sign of result

F851 CD62F8 PHLTFFP: CALL FPBCDE: EX LOADFP ; Number at HL to BCDE
 LD DE,HL ; Save code string address
 (HPREG),HL ; Save LSB, NLSB of number
 H,B ; Exponent of number
 LD L,C ; MSB of number
 LD (FPREG+2),HL ; Save MSB and exponent
 DE,HL ; Restore code string address
 EX RET ; New sign of result

F85F 21E410 BCDEFP: LD HL,FPREG ; Point to FPREG
 LOADFP: LD E,(HL) ; Get LSB of number
 INC HL ; Get NMSB of number
 LD D,(HL) ; Get MSB of number
 INC HL ; Get MSB of number
 LD C,(HL) ; Get MSB of number
 INC HL ; Get exponent of number
 LD B,(HL) ; Used for conditional "INC HL"
 INC HL ; Used for conditional "INC HL"
 RET ; New sign of result

F86B 11E410 FPTHL: LD DE,FPREG ; Point to FPREG
 DETHL4: LD B,4 ; 4 bytes to move
 DETHL8: LD A,(DE) ; Get source
 LD (HL),A ; Save destination
 INC DE ; Next source
 INC HL ; Next destination
 INC B ; Count bytes
 DEC B ; Loop if more
 JP NZ,DETHLB ; Drop RET

F879 21E610 SIGNS: LD HL,FPREG+2 ; Point to MSB of FPREG
 LD A,(HL) ; Get MSB
 RLCA ; Old sign to carry
 SCF ; Set MSB
 RRA ; Set MSBit of MSB
 LD (HL),A ; Save new MSB
 CCF ; Complement sign
 RRA ; Old sign to carry

F884 23 INC HL ; Set sign of result
 F885 77 LD (HL),A ; Get MSB
 F886 79 LD A,C ; Old sign to carry
 F887 07 LD RLCA ; Set MSBit
 F888 37 LD SCF ; Set MSBit of MSB
 F889 1F LD RRA ; Save MSB
 F88A 4F LD XOR (HL) ; New sign of result

F88D C9 CMPNUM: LD A,B ; Get exponent of number
 OR B7 ; Zero - Test sign of FPREG
 JP Z,TSTSZN ; Return relation routine
 LD Z,TSTSZN ; Save for return
 LD HL,RETREL ; Test sign of FPREG
 CALL TSTSZN ; Get MSB of number
 RET Z ; FPREG zero - Number's MSB
 LD HL,FPREG+2 ; MSB of FPREG
 XOR (HL) ; Combine signs
 LD A,C ; Get MSB of number
 RET M ; Exit if signs different
 CALL CMPFP ; Compare FP numbers
 LD A,C ; Get carry to sign
 RRA C ; Combine with MSB of number

F896 E5 CALL F896 CD13F8 ; New sign of result

F899 79 LD A,C ; Test sign of FPREG
 CALL TSTSZN ; Get MSB of number
 RET Z ; FPREG zero - Number's MSB
 LD HL,FPREG+2 ; MSB of FPREG
 XOR (HL) ; Combine signs
 LD A,C ; Get MSB of number
 RET M ; Exit if signs different
 CALL CMPFP ; Compare FP numbers
 LD A,C ; Get carry to sign
 RRA C ; Combine with MSB of number

F8A8 23 CMPFP: INC HL ; Point to exponent
 LD A,B ; Get exponent
 CP (HL) ; Compare exponents
 RET NZ ; Different
 LD HL ; Point to MBS
 DEC HL ; Get MBS
 LD A,C ; Compare MSBs
 CP (HL) ; Different
 RET NZ ; Point to NMSB
 LD A,D ; Get NMSB
 CP (HL) ; Compare NMSBs
 RET NZ ; Different
 LD HL ; Point to LSB
 A,E ; Get LSB
 SUB (HL) ; Compare LSBs
 RET NZ ; Different
 POP HL ; Drop RETurn
 POP HL ; Drop another RETurn

F8B1 7A LD A,D ; Get LSB
 F8B2 BE SUB (HL) ; Compare LSBs
 F8B3 CO CP (HL) ; Different
 F8B4 2B DEC HL ; Point to LSB
 F8B5 7B LD A,E ; Get LSB
 F8B6 96 SUB (HL) ; Compare LSBs
 F8B7 C0 RET NZ ; Different
 F8B8 E1 POP HL ; Drop RETurn
 F8B9 E1 POP HL ; Drop another RETurn

F8BA C9

Dis-assembly of NASCOM ROM BASIC Ver 4.7		PAGE	82	Dis-assembly of NASCOM ROM BASIC Ver 4.7	
F8BB 47	FPINT:	LD	B,A	<- Move	
F8BC 4F		LD	C,A	<- exponent	A,B
F8BD 57		LD	D,A	<- to all	C
F8BE 5F		LD	E,A	<- bits	Z
F8BF B7		OR	A'	Test exponent	A,16
F8C0 C8		RET	Z	Zero - Return zero	HL,HL
F8C1 E5		PUSH	HL	Save pointer to number	C,BSERR
F8C2 CD5PF8		CALL	BCDEF0	Move FPREG to BCDE	DE,HL
F8C5 CD79F8		CALL	F90B EB	Set MSBs & sign of result	HL,HL
F8C8 AE		CALL	F90C 29	Combine with sign of FPREG	DE,HL
F8C9 67		XOR	(HL)	Save combined signs	NC,MLDBL
F8CA FCDFFF		LD	H,A	Negative - Decrement BCDE	HL,BC
F8CF 90		CALL	M,DCBCDE	24 bits	C,BSERR
F8CD 3D		LD	A,80H+24	Bits to shift	A
F8D0 CD92F6		SUB	B	Shift BCDE	NZ,MLDBL
F8D3 7C		CALL	F915 3D	Get combined sign	JP
F8D4 17		RLA	A,H	Sign to carry	RET
F8D5 DC65F6		CALL	C,FPRND	Negative - Round number up	"-"
F8D8 0600		LD	B,0	Zero exponent	AF
F8DA DC7EF6		CALL	C,COMPL	If negative make positive	Z,CVNVM
F8DD E1		POP	HL	Restore pointer to number	"+"
F8DE C9		RET			Z,CVNVM
F8DF 1B		DCBCDE:	DEC	Decrement BCDE	DEC
F8E0 7A		LD	A,D	Test LSBs	RESZR
F8E1 A3		AND	E		B,A
F8E2 3C		INC	A		D,A
F8E3 C0		RET	NZ	Exit if LSBs not FFFF	E,A
F8E4 0B		DEC	BC	Decrement MSBs	C,A
F8E5 C9		RET			GETCHR
F8E6 21E710	INT:	LD	HL,FPEXP	MANLP:	C,ADDIG
F8E9 7E		LD	A,(HL)	F931 DA77F9	"."
F8EA FE98		CP	80H+24	F934 FE2E	Z,DPOINT
F8EC 3AE410		LD	A,(FPREG)	F936 CA52F9	"E"
F8EF DO		RET	NC	F939 FE45	NZ,CONEX
F8F0 7E		LD	A,(HL)	F93B C256F9	GETCHR
F8F1 CDBBF8		CALL	RINT	F93E CD36F8	SGNEXP
F8F4 3698		LD	(HL),80H+24	F944 CD36F8	GETCHR
F8F6 7B		LD	A,E	F947 DA99F9	C,EDIGIT
F8F7 F5		PUSH	AF	F94A 14	D
F8F8 79		LD	A,C	F94B C256F9	NZ,CONEX
F8F9 17		RLA		F94E AF	A
F8FA CD1BF6		CALL		F94F 93	SUB
F8FD F1		POP		F950 5F	LD
F8FE C9		PBT		F951 0C	INC

Dis-assembly of NASCOM ROM BASIC Ver 4.7		PAGE	83
F8FF 210000	MLDEBC: LD	HL,0	; Clear partial product
F902 78	LD A,B	C	; Test multiplier
F903 B1	OR C	Z	; Return zero if zero
F904 C8	RET	A,16	; 16 bits
F905 3E10	LD ADD	HL,HL	; Shift P.P left
F907 29	MLDBLP: ADD	C,BSERR	; ?BS Error if overflow
F908 DA45F0	JP DE,HL	DE,HL	; Shift multiplier left
F90B EB	EX ADD	HL,HL	
F90C 29	EX DE,HL	DE,HL	
F90D EB	NC NOMLAD	NC,NOMLAD	; Bit was zero - No add
F90E D215F9	ADD HL,BC	ADD	; Add multiplicand
F911 09	JP C,BSERR	C,BSERR	; ?BS Error if overflow
F912 DA45F0	DEC A	A	; Count bits
F915 3D	DEC NZ	NZ,MLDBLP	; More
F916 C207F9	JP RET		
F919 C9			
F91A FE2D	ASCTFP: CP	"."	; Negative?
F91C F5	PUSH AF	Z,CNVNUM	; Save it and flags
F91D CA26F9	CP "+"	Z,CNVNUM	; Yes - Convert number
F920 FE2B	DEC HL	RESER	; Positive?
F922 CA26F9	CNVNUM: CALL LD	B,A	; Yes - Convert number
F925 2B	CD33F6 DEC	LD	; DEC ^cos GETCHR INCS
F926 CD33F6	CALL JP	D,A	; Set result to zero
F929 47	MANLP: CALL CP	E,A	; Digits after point count
F92A 57	EXPLP: CALL CP	C,EDIGIT	; Sign of exponent
F92B 5F	CD36F9 INC D	D	; Exponent of ten
F92C 2F	CPL INC A	A	
F92D 4F	CD36F9 LD C,A	C,A	; Before or after point
F92E CD36F8	MANLP: CALL GETCHR	GETCHR	; Get next character
F931 DA77F9	JP C,ADDIG	C,ADDIG	; Digit - Add to number
F934 FE2E	CP "	"	; "." - Flag point
F936 CA5F9	Z,DPOINT	"E"	
F939 FE45	CP NZ,CONEXP	JP NZ,CONEXP	; Not "E" - Scale number
F93B C256F9	CALL GETCHR	GETCHR	; Get next character
F93E CD36F8	CALL SGNEXP	SGNEXP	; Get sign of exponent
F941 CD70E9	CALL GETCHR	GETCHR	; Get next character
F944 CD36F8	CALL C,EDIGIT	C,EDIGIT	; Digit - Add to exponent
F947 DA99F9	JP INC D	D	
F94A 14	XOR A	A	
F94B C256F9	SUB E	E	
F94E AF	LD E,A	C	
F94F 93	INC C	C	
F950 5F	DPOINT: INC C	C	
F951 0C	CONEXP: PUSH Z,MANLP	JP Z,MANLP	
F952 0C	PUSH HL	HL	
F953 CA2EE9	CALL DIV10		
F956 E5	LD A,E	A,E	
F957 7B	SCALM1: CALL P,SCALPL	P,SCALPL	
F958 90	POP AF	AF	
F959 F46FF9	JP ENDCON	ENDCON	
F95C F265F9	PUSH AF	AF	
F95F F5	CALL DIV10	DIV10	
F960 CD52F7	POP AF	AF	
F963 F1	INC A	A	
F964 3C			

Dis-assembly of NASCOM ROM BASIC Ver 4.7

PAGE 84

PAGE 85

Dis-assembly of NASCOM ROM BASIC Ver 4.7

F965 C259F9 ENDCON: JP NZ, SCALMI ; More to do
 F968 D1 POP DE ; Restore code string address
 F969 F1 POP AF ; Restore sign of number
 F96A C03CF8 CALL Z, INVSZN ; Negative - Negate number
 F96D EB EX DE, HL ; Code string address to HL

F96E C9 RET ; Code string address to DE

F96F C8 SCALPL: RET Z ; Exit if no scaling needed
 F970 F5 MULTEN: PUSH AF ; Save count
 F971 CDRCF7 CALL MLSP10 ; Multiply number by 10
 F974 F1 POP AF ; Restore count
 F975 3D DEC A ; Count multiples

F976 C9 RET ;

F977 D5 ADDIG: PUSH DE ; Save sign of exponent
 F978 57 LD A, B ; Save digit
 F979 78 ADC A, C ; Get digits after point
 F97A 89 LD B, A ; Add one if after point
 F97B 47 RE-SAVE COUNTER ; Re-save counter

F97C C5 PUSH BC ; Save point flags
 F97D E5 PUSH HL ; Save code string address
 F97E D5 PUSH DE ; Multiply number by 10
 F97F CDRCF7 CALL MLSP10 ; Restore digit

F982 F1 POP AF ; Make it absolute
 F983 D630 SUB "0" ; Re-scale number
 F985 CD8EF9 CALL RSCALE ; Restore code string address

F988 E1 POP HL ; Restore point flags

F989 C1 POP EC ; Restore sign of exponent

F98A D1 POP DE ; Get another digit

F98B C32EF9 JP MANLP ; Put number on stack

F98E CD44F8 RSCALE: CALL STAKFP ; Digit to add to FPREG
 F991 CD25F8 PADD: CALL FLGREL ; Restore number

F994 C1 JP IPADD ; Add BCDE to FPREG and return

F995 D1 JP ADD ; Times 5

F996 C3CDF5 EDIGIT: LD A, E ; Times 2

F999 7B ADD: RLCA ; Times 4

F99A 07 ADD: RLCA ; Get digit

F99B 07 ADD: A, E ; Times 5

F99C 83 ADD: RLCA ; Times 10

F99D 07 ADD: A, (HL) ; Add next digit

F99E 86 SUB "0" ; Make it absolute

F99F D630 LD E, A ; Save new digit

F9A1 5F EXPLP ; Look for another digit

F9A2 C344F9

F9A5 E5 LINEIN: PUSH HL ; Save code string address
 F9A6 2146E3 LD HL, TNMSG ; Output " in "
 F9A9 CD10F2 CALL PRS ; Output string at HL
 F9AC E1 POP HL ; Restore code string address

F9AD EB PRNTHL: EX DE, HL ; Code string address to DE

F9AE AF XOR A ; 24 bits

F9AF 0698 LD B, 80H+24 ; Return the integer
 F9B1 CD2AF8 CALL RETINT ; Print number string

F9B4 210FF2 LD HL, PRNUMS ; Save for return

F9B7 E5 PUSH HL ; Convert number to ASCII

F9B8 21E910 NUMASC: LD HL, PBUFF ; Save for return

F9B9 E5 PUSH HL ; Test sign of FPREG

F9BC CD13F8 CALL TSTSN ; Space at start

F9BF 3620 LD (HL), " Positive - Space to start
 F9C1 F226F9 JP "-" ; "-" sign at start

F9C4 362D LD (HL), "-" ; First byte of number

F9C6 23 INC HL ; "0" if zero

F9C7 3630 LD (HL), "0" ; Return "0" if zero

F9C9 CA7CFA XOR Z, JSTZER ; Save buffer address

F9CC E5 PUSH HL ; Negate FPREG if negative

F9CD FC3CF8 CALL M, INVSZN ; Zero A

F9D0 AF PUSH HL ; Save it

F9D1 F5 PUSH AF ; Test number is in range

F9D2 CDB2FA CALL RNGTST ; BCDE = 99999.9

F9D5 01A391 SIXDIG: LD BC, 9143H ; Compare numbers

F9D8 11FB4F LD DE, 4FF8H ; CMPNUM

F9DB CD8EF8 CALL OR A ; > 9999.9 - Sort it out

F9DE B7 JP PO, INRNG ; Restore count

F9DF E2F3F9 POP AF ; Multiply by ten

F9E2 F1 CALL MULTEN ; Re-save count

F9E3 CD70F9 PUSH AF ; Test it again

F9E6 F5 CALL SIXDIG ; Divide by 10

F9E7 C3DF59 POP AF ; Get count

F9EA CD5BF7 GTSIXD: CALL DIV10 ; Count divides

F9ED F1 POP AF ; Re-save count

F9EE F5 PUSH AF ; Test number is in range

F9F0 CDB2FA CALL RNGTST ; Add 0.5 to FPREG

F9F3 CDBBF5 INRNG: CALL ROUND ; Add one

F9F6 3C INC A ; F.P to integer

F9F7 CDBBF8 CALL FPINT ; Move BCDE to FPREG

F9FA CD54F8 CALL FPBCDE ; 1E+06 to 1E-03 range

F9FD 010603 LD BC, 0306H ; Restore count

F9AO F1 POP AF ; 6 digits before point

F9A1 81 ADD A, C ; Do it in "E" form if < 1E-02

F9A2 3C INC A ; More than 99999 ?

F9A3 FA0FFA JP M, MAKNUM ; Yes - Do it in "E" form

F9A6 FE08 CP NC, MAKNUM ; Adjust for exponent

F9A8 D20FFA INC A ; Exponent of number

F9A9 3C LD B, A ; Make it zero after

F9A0 47 LD A, 2

```

Dis-assembly of NASCOM ROM BASIC Ver 4.7

FA0F 3D      MAKNUM: DEC    A
FA10 3D      DEC    A
FA11 E1      POP   AF
FA12 F5      PUSH  AF
FA13 1195FA  LD    DE, POWERS
FA16 05      DEC    B
FA17 C220FA  DEC    B
FA1A 367E  DIGITXT: DEC    B
FA1C 23      INC    HL
FA1D 3630  INC    HL
FA1F 23      INC    HL
FA20 05      INC    HL
FA21 362E  DIGITXT: DEC    B
FA22 362F  CALL  "(HL), \""
FA23 C569FB  CALL  Z, INCHL
FA26 C5      PUSH  BC
FA27 E5      PUSH  HL
FA28 D5      PUSH  DE
FA29 C5DFFF8 CALL  BCDEFPP
FA2C E1      POP   HL
FA2D 062F  TRYAGN: INC    B
FA2F 04      INC    B
FA30 7B      LD    A,E
FA31 96      SUB   (HL)
FA32 5F      LD    E,A
FA33 23      INC    HL
FA34 7A      LD    A,D
FA35 9E      SBC   A,(HL)
FA36 57      LD    D,A
FA37 23      INC    HL
FA38 79      LD    A,C
FA39 9E      SBC   A,(HL)
FA3A 4F      LD    C,A
FA3B 2B      DEC    HL
FA3C 2B      DEC    HL
FA3D D2FFFA  CALL  NC, TRYAGN
FA40 CD72F6  CALL  PLUCDE
FA43 23      INC    HL
FA44 CD54FB  CALL  FPBCDE
FA47 EB      EX    DE, HL
FA48 E1      POP   HL
FA49 70      LD    (HL), B
FA4A 23      INC    HL
FA4B C1      POP   BC
FA4C 0D      DEC    C
FA4D C220FA  DEC    B
FA50 05      DEC    B
FA51 CA60FFA SUPTLZ: DEC    B
FA54 2B      JP    Z, DOEBIT
FA55 7E      LD    A,(HL)
FA56 FE30  SUPTLZ: DEC    B
FA58 CA54FFA CP    "O"
FA59 FE2E  CP    "Z"
FA5D CA469FB CALL  NZ, INCHL

```

Dis-assembly of NASCOM ROM BASIC Ver 4.7

Dis-assembly of NASCOM ROM BASIC Ver 4.7

PAGE 89

```

C1          ; Get base
POWER:    POP BC
          POP DE
          CALL TSTSGN
          LD A,7F
          JP Z,EXP
          JP P,POWER1
          OR A,DZERR
          JP Z,0/0 Error
          JP A,BASE zero?
          JP Z,RETURN zero?
          JP Z,SAVEXP
          PUSH DE
          PUSH BC
          LD A,C
          OR 0111111B
          CALL BCDEFP
          JP P,POWER2
          PUSH BC
          CALL INT
          POP BC
          DE AF
          PUSH AF
          CALL CMPNUM
          POP HL
          LD A,H
          RR A,F
          POP HL
          LD (FPREG+2),HL
          POP HL
          LD (FPREG),HL
          CALL C,NEGACT
          CALL Z,TINVSGN
          PUSH DE
          PUSH BC
          CALL LOG
          POP BC
          CALL FPMULT
          POP DE
          CALL CD088F7
          ; Test sign of power
          ; Get exponent of power
          ; Make result 1 if positive base - or
          ; Positive base - or
          ; Zero to negative
          ; Yes - ?/0 Error
          ; Base zero?
          ; Yes - Return zero?
          ; Save base
          ; Get MSB of base
          ; Get sign status
          ; Move power to BC
          ; Positive base -
          ; Save Power
          ; Get integer of power
          ; Restore power
          ; MSB of base
          ; Power an integer
          ; Restore MSB of base
          ; but don't affect
          ; Exponent odd or
          ; Restore MSB and
          ; Save base in FPR
          ; LSRs of base
          ; Save in FPREG
          ; Odd power - Negate
          ; Negative base -
          ; Save power
          ; Get LOG of base
          ; Restore power
          ; Multiply LOG by

```

THE JOURNAL OF CLIMATE

EXPTAB:		8	Table used by EXP
74	DEFB	040H, 02EH, 094H, 074H	-1/71 (-1/540)
75	DEFB	070H, 04FH, 02EH, 077H	-1/6 (-1/720)
76	DEFB	06EH, 002H, 088H, 07AH	-1/5 (-1/120)
77	DEFB	0E6H, CA0H, 02AH, 07CH	-1/4 (-1/24)
78	DEFB	050H, DA0H, 0AAH, 07EH	-1/3 (-1/6)
79	DEFB	0FFFH, 0FFH, 07FH, 07FH	-1/2 (-1/2)
80	DEFB	000H, 000H, 080H, 081H	-1/1 (-1/1)
81	DEFB	000H, 000H, 000H, 081H	1/0 (-1/1)

Disassembly of NASCOM ROM BASIC Ver 4.7

PAGE 90

FB5B CD44F8	SUMSER: CALL	STAKFP	; Put FPREG on stack
FB5E 1106F7	LD DE,MULT	DE,MULT	; Multiply by "X"
FB61 D5	PUSH DE	DE	; To be done after
FB62 E5	PUSH HL	HL	; Save address of table
FB63 CD5FF8	CALL BCDEFP	BCDEFP	; Move FPREG to BCDE
FB66 CD08F7	CALL FPMULT	FPMULT	; Square the value
FB69 E1	POP HL	HL	; Restore address of table
FB6A CD44F8	CALL STAKFP	STAKFP	; Put value on stack
FB6D 7E	LD A,(HL)	A,(HL)	; Get number of coefficients
FB6E 23	INC HL	HL	; Point to start of table
FB6F CD51F8	CALL PHLTFP	PHLTFP	; Move coefficient to FPREG
FB72 06	DEFB (LD B,n)	(LD B,n)	; Skip "POP AF"
FB73 F1	POP AF	AF	; Restore count
FB74 C1	POP BC	BC	; Restore number
FB75 D1	POP DE	DE	
FB76 3D	DEC A	A	; Cont coefficients
FB77 C8	RET Z	Z	; All done
FB78 D5	PUSH DE	DE	; Save number
FB79 C5	PUSH BC	BC	
FB7A F5	PUSH AF	AF	
FB7B E5	PUSH HL	HL	; Save count
FB7C CD08F7	CALL FPMULT	FPMULT	; Save address in table
FB7F E1	POP HL	HL	; Multiply FPREG by BCDE
FB80 CD62F8	CALL LOADFP	LOADFP	; Restore address in table
FB83 E5	PUSH HL	HL	; Number at HL to BCDE
FB84 CDCDF5	CALL FPADD	FPADD	; Save address in table
FB87 E1	POP HL	HL	; Add coefficient to FPREG
FB88 C373FB	JP SUMLP	JMP SUMLP	; Restore address in table
			; More coefficients

LAST**PART****NEXT****ISSUE!**

GEMINI GM860 EPROM PROGRAMMER REVIEW**By D. W. Parkinson**

At long last the Gemini EPROM programmer is with us. At least I assume it is with us and not a holographic projection, as our Editor claimed it didn't exist an issue or two ago! (I quote - "What EPROM programmer?" - Vol 2 Iss 5.)

What you get

- i) A box 8.25" x 4.25" x 2" - the programmer.
- ii) A disk.
- iii) A manual.
- iv) A connecting cable.

What you need.

- i) A disk-based MultiBoard system. (e.g. Gemini Galaxy)
- ii) A mains plug.
- iii) Money. (#150+VAT)
- or alternatively
- i) A Z80 based system with a PIO.
- ii) A mains plug.
- iii) The ability to program & an understanding of Z80 assembly language.
- iv) Money. (#150+VAT)

What does it do?

It will program all Intel compatible single supply EPROMs from a 2716 (2kx8) to a 27256 (32kx8).

Now taking each item in turn:

The Box and cable: This is a mains powered unit containing the programmer. It has a 28-pin ZIF (Zero Insertion Force socket) sticking out of the top along with two LEDs, (one to indicate 'power on', the other for 'ZIF socket powered up'). On the front is a mains on/off switch and fuse, and on the rear a mains lead emerges and there is a 34-way IDC connector. The programmer has been designed to interface to the main computer via a PIO. The interface comprises an 8-bit bi-directional bus, together with a clock line and an address line. Gemini have selected and ordered the connections such that the programmer can be plugged straight into the Centronics socket on the back of a standard Galaxy. The supplied connecting cable is just under 3' long, with a 34-way IDC connector on one end, and a Centronics plug on the other.

Note: This does not mean that the programmer can be plugged into any Centronics type interface. On the Galaxy the Centronics socket is connected internally directly to a PIO without any intervening buffers. Thus the full bidirectional capability of the PIO's data port is available to the control software. Similarly the Centronics 'Busy' printer status line can be used as an output line in this application.

If you don't have a Galaxy, or would rather connect the programmer direct to another PIO within the Galaxy you have two options open:

- a) Make up your own cable to connect the 34-way IDC of the programmer to the usual 26-way PIO connection of the Nasom or Gemini boards. (Details are given in the documentation.)
- b) Buy the internal 'PIO-to-Centronics' cable that is used within the Galaxy and plug the supplied cable into that. (The 'PIO-to-Centronics' cable is in the current Gemini catalogue at #15).

The internal power supply of the programmer provides all the necessary voltages, and no power is drawn from the main system. As well as the various programming voltages (25V, 21V, 12.5V) the EPROM supply voltage can be raised to 6V, the level necessary if the "intelligent programming algorithm" is to be implemented for the larger EPROMs (see below).

The programmer's box is made of metal and consists of an interlocking U-shaped base (coloured black), and a U-shaped lid (coloured grey). With the metal construction of the box and the internal mains transformer the weight of the programmer is such that it sits nicely on a flat surface without any tendency to scoot about while EPROMs are changed. Taking the lid off the box reveals a very neat and compact arrangement. The main programmer logic is contained on a single PCB that runs the length of the box. Underneath this PCB is a toroidal mains transformer and a smaller pcb that carries the power supply components.

The Disk: This holds three programs:-

PROG.COM The programmer software.

CPROG.COM A configuration program for PROG.COM

PROGINIT.MAC The low level assembly language drivers used within PROG.COM.

The disk is in Gemini QDSS format (i.e. single sided Galaxy format - 96tpi) which can also be read by double-sided systems (QDDS format). The manual indicates that you should be able (via your dealer) to exchange it for DDDS or SDDS formats (the early Gemini 48tpi formats). Failing that you can ask for a paper listing of PROGINIT.MAC instead. Some dealers may be able to provide other formats.

The Manual: The eight page manual covers the installation of the programmer, and the use of CPROG and PROG. Also a brief section is included on how Microsoft's M80/L80 combination can be used to create ROMable programs. There is no description of the internal circuitry of the programmer nor is there any information on how it should be driven by any program other than that supplied. (But see PROGINIT.MAC below.)

Using the Software

The configuration program CPROG allows the user to change two parameters within PROG. The first is the base address of the PIO that is driving the programmer, and the second is the default EPROM type that PROG assumes when it is invoked. The latter option saves the user the irritation of having to use the 'Change EPROM type' option every time that PROG is used.

The main program PROG drives the programmer, and transfers data in both directions between disk files and a EPROMs via a buffer that it maintains in memory. When invoked it puts up a heading together with a menu from which the following options can be selected:

- 0 - Return to CP/M.
- 1 - Change EPROM type. This brings up a sub-menu offering the choice of 2716, 2732, 2732A, 2764, 27128, and 27256.
- 2 - Read an EPROM into the buffer. This prompts for an EPROM to be inserted into the socket, and then reads it into the buffer.
- 3 - Read a disk file into the buffer. (A prompt for a file name will appear.)

- 4 - Edit the contents of the buffer. Allows simple editing of the contents of the buffer.
- 5 - Display the buffer. Displays the buffer contents on the screen using the conventional Hex/ASCII display format (as in Gemdebug, Nas-Sys etc).
- 6 - Write the buffer to a disk file. (A prompt for a file name will appear. If the file name you give already exists you will be asked for confirmation that it is to be overwritten.)
- 7 - Write the buffer to an EPROM. (i.e. Program the EPROM.)
- 8 - Verify the EPROM is erased. Prints out any locations within the EPROM that are not OFFh.
- 9 - Verify an EPROM against the buffer. The two are compared and any differences are listed to the screen.

The heading also displays three things of interest. i) Whether the memory buffer contains any data. ii) The type of EPROM currently selected. iii) The name of the last file read/written to/from the memory buffer. Items (ii) and (iii) I find quite useful, especially if you're trying to program multiple files into multiple EPROMs.

The Intelligent Programming Algorithm

The software implements the 'Intelligent programming algorithm' (IPA) for the larger EPROMs (2764 and upwards). For those of you that haven't met this, this is a technique for vastly speeding up the programming of the latest EPROMs.

In general EPROMs are programmed by presenting the appropriate address and data to the EPROM along with a suitable write pulse. (Certain 'programming' voltage levels have to be present on other pins.) The write pulse duration is specified at 50ms. This figure of 50ms has to cover the worst case, and in fact most data cells within an EPROM program in a fraction of this time (Intel claim 16% of the time i.e. 8ms), and the difficulty lies in determining when a cell is adequately programmed. With devices such as the 2716 the resultant programming time of about 100 seconds was not a burden, but 14 minutes for a 27128 is another matter entirely, (let alone 28 minutes for a 27256!). The latest EPROMs have been designed and tested so that a (faster) modified programming technique can be used, which still results in the same guaranteed long term data retention characteristics. A simplified version of this algorithm is as follows:

- 1) Set up the programming conditions and set Vcc=6v (rather than the usual 5v)
- 2) Apply address & data and set a software counter N=0.
- 3) Apply a 1ms write pulse. Set N=N+1.
- 4) Read back the data (with Vcc still at 6v) and compare with the original.
If it has not programmed go to 3 (unless N=15).
- 5) Apply a pulse of 4NmS to complete the programming.
- 6) Verify & move on to next address.

i.e. You try in stages until the byte is programmed, then you give it a final big 'bash' (4 x what you've given it so far) to ensure that the cells are well and truly past the threshold. The 6v Vcc level also ensures that the internal threshold level that determines whether it is a '1' or a '0' that you are reading is back is higher than normal. This gives an increased programming margin that leads to increased reliability.

In practice I found that 2764s I was using programmed in about 90 seconds using this algorithm.

Observations

PROG (Version 0.7) is a reasonable program that provides all that is necessary (and a little more) for easy straight-forward programming of EPROMs. The ability to load and save data to/from various files is a useful feature, as is the limited editing. Those requiring more extensive editing features (such as block moves or block fills) will have to resort to loading the data file under a debugger such as Gemdebug, ZSID, or DDT and using their facilities. The display option is useful as it lets you visually check the buffer contents before programming. (This ensures that you have loaded the correct file and that it does contain what you think it does.)

The only obvious omission is the ability to program a limited range of addresses within the EPROM. At the moment it is a case of 'all or nothing'. (Mind you the software does not bother to program any byte that is OFFh, so this can be got around in a roundabout way although the 'verify' code may complain rather!)

In general the ESCape key acts as an abort key to return you to the main menu. However this feature does not work during programming, and there is no visual echo of the progress of the programming operation. I assume this is to reduce the display/polling overhead that would otherwise occur and cause an increase in the overall programming time, but it is a pity that it has been omitted. Perhaps this will be changed in a later release of the software.

General comments

It is nice to see the programmer as a separate unit rather than an 80-BUS card. If anyone has ever used a system where the programmer was on a plug-in card they will know the frustrations of that approach. (You have to work with the lid off the computer and have to juggle the cards around to get the EPROM card into a position where you can actually get EPROMs into and out of the socket - The socket is more often than not vertical, and when you flick the lever to release the EPROM it often falls out into the guts of the computer!)

If you do not have a disk-based system, or want some unusual feature in your control software, the file PROGINIT.MAC contains the low level drivers that form part of PROG. If you are writing software to be 'PROM'ed then it is likely that you should be able to add your own control routine to PROGINIT although you will have to do a certain amount of detective work to work out exactly what to do. PROGINIT.MAC - I think it should actually be PROGINIT.C judging by the comments at the start - starts off with a section of comments that tells you what bits where control what. (e.g. Bit X of the control latch selects a Vcc of 5v or 6v, set bit Y for 21v programming voltage, etc.) What it leaves you to work out for yourself are the bit combinations required for the various EPROM types.

Conclusions

The Gemini EPROM programmer is fairly obviously targeted on Gemini Galaxy and Quantum users, but any Multiboard/Gemini/Nascom owner running CP/M should be able to use the package without any difficulty. (N.B. The screen display of PROG.COM is obviously orientated towards the Gemini IVC/SVC.) Those without disk systems will have to write some of the support software themselves.

The price of #150 (+VAT) seems a little on the high side to me, but it is neatly packaged (unlike the earlier Bits & PC's programmer which came as a bare board) and it definitely does work.

REVIEW OF NASCOM ENHANCED BASIC (CRYSTAL BASIC 3)**By Roger Dowling****GENERAL**

The following review of Enhanced BASIC for the Nascom computer is based upon the version to run under the Nas-Dos disk operating system, although presumably the CP/M version is similar and the tape version is lacking the disk commands, most of which will be obvious.

The Nas-Dos version supplied on disk contains two Crystal BASIC interpreters. XBASD is the standard disk version and one other which has additional commands for use with the Nascom AVC (GXBASD). The former is approximately 14k in length and the graphics version is some 22k long. XBASD returns 34044 bytes free and GXBASD returns 26304 bytes free when run under a 48k RAM system. It will run under either Nas-Sys 1 or 3.

Also on the disk are some demonstration XBASIC programs, some of which are also examples of disk (or tape!) file handling given in the manual. The manual contains around 90 pages including appendices and is well documented although a bit hap-hazard in layout. Nothing like the early Nascom manuals though for anyone who goes that far back! The only faults with the manual that I could fairly say is that some of the default values in the workspace area are incorrectly shown and the examples given for adding extra commands seem to assume tape XBASIC and thus the locations for the pointers to the extra tables and commands are different for the disk version. No great hardship after half an hours work though.

Also present on the disk are various pictures and demo programs to be run with the AVC under GXBASD. These are virtually the same as those which come with the AVC board although I did manage to find one or two extra things in there.

One other nice program on the disk is a ROM BASIC to XBASIC conversion utility which, when run, sets up the command table to accept two extra commands namely DLOAD and TLOAD to load and convert a ROM BASIC program from disk or tape and then to convert to XBASIC format.

XBASIC supports up to four disk drives and one tape drive and occupies RAM from 1000H to approximately 4660H. Its workspace is from 1006H to 1180H and a complete useful listing of the workspace area is given in the manual.

Additional commands may be incorporated into the interpreter in two ways -permanent extensions to the command set extend the interpreters length by adding the routines and extra word tables onto the end of the XBASIC code. The complete interpreter is then re-written to disk to include the extra commands. Temporary additions such as toolkit type commands which have no use during the running of a program are added at the top of RAM and can be removed by a cold start. An interesting point is that the Auxiliary Reserved word table (the table that contains any extra commands) is scanned before the Standard Reserved word table and this means that it is possible to use complete words from the existing table as part or whole Auxiliary words.

There are a total of 115 commands (including operators), 44 (I think!) that are in addition to those available with ROM BASIC. There are 14 commands that are similar to those of ROM BASIC but which have a changed format or are enhanced. XBASIC is an interpreter in its own right but where commands are the same as in ROM BASIC then their action is similar too. In XBASIC, there is no unscrolled top line. Thus all 16 lines scroll. It took a long time to get used to the fact that it is the cursor down key rather than the Enter key that scrolls the screen. The Enter key merely places the cursor (if it is not already there) at the first free line on the VDU. The Editor uses an internal "VDU" high up in RAM and a line length of up to 127 characters is achieved using this technique coupled with an input buffer located at 0C80H. By moving

this location, with the use of the pointer command, longer line lengths may be achieved.

Variables may be named to any length but only the first five letters are distinguished. Integer variables are defined by the use of a % suffix i.e. A%, B% etc. Any numerical variable followed by % will be truncated to its integer form. Hex numbers are catered for by preceding any hex number with an ampersand sign (&).

EDITOR

The XBASIC editor takes two forms, namely screen editor and line editor. Normally under command mode, the Editor is in screen edit mode and during a program run is in line edit mode, although this can be altered for special purposes (see IOM command).

SCREEN EDITOR

As under ROM BASIC with the additional facilities of:

CTRL A - Home cursor to top left corner of screen. CTRL W - Erase whole line that cursor is sitting upon irrespective of where it is in that line. CTRL X -Erase to end of line from the current cursor position. CTRL O - Erase to end of screen from current cursor position. CTRL P - Print screen contents to printer. ESC - Abandon a line. This does not delete the line as it does in ROM BASIC. The Ok prompt is printed. This took a while to get used to.

LINE EDITOR

As for ROM BASIC in input mode within a program except that CTRL P will dump screen contents to printer. Cursor left wil delete previous input during input mode.

OUTPUT DEVICES

As supplied, three devices are assigned under XBASIC. Device 0 is the default value and assigns the Nascom output to the VDU and input from the keyboard. Device 0 is the only one that uses the screen editor. Device 1 is output to a printer (may be serial or parallel) and is the device utilised when CTRL P is used. Input is not assigned and is as for device 0. Device 2 is output to serial port and input from the serial port.

The output device is changed by use of the PRINT# command followed by the device number. e.g. PRINT#1:LIST would list the current XBASIC program to the printer. A number of devices may easily be added to cater for a modem or other such periphiral attached to a UART on the Nascom I/O board for example. A total of 255 (0 to 254) output and 255 input devices may be assigned! All disk and tape file handling is passed via device 255 and is therefore not assignable.

COMMANDS

First the commands that have the same name as those within ROM BASIC that we all know and love but which are enhanced or act differently.

CLEAR - As for ROM BASIC except when followed by up to two values sets up the top most location available to BASIC and the size of the stack in addition to clearing the variables.

e.g. CLEAR&C000,300 increases the size of the stack from its default and minimum value of 256 to 300 bytes and the highest location available to XBASIC as BFFF hex. Any OBJ file loaded after this command will be loaded from C000H irrespective of its resident address when it was saved.

POKE - Easiest shown as an example: POKE&C000,252,253,254,255 etc. This places the hex code for 252d into location C000H,253d into C001H,254d into C002 etc.

DOKE - As for POKE above only, of course, in double byte form.

LIST - Followed by up to three values. The first value is the line to list from, the second is the number of lines to list at a time before pausing and the third is the line number to list up to. The default value of lines to list at a time, as supplied is 65535 but this can be altered to suit your taste. e.g. LIST 300,10,1000 will list lines 300 to 1000 pausing every 10 lines until a key is hit to display the next ten. The value of ten lines at a time will remain until set otherwise by a command such as LIST,5 when the whole program will start to be listed pausing every 5 lines displayed. A nice feature of XBASIC is that it is not necessary to press ESC to abort a listing. During a pause, any of the cursor keys will abandon the listing while at the same time moving the cursor in the desired direction in order to make an edit. Listings are made obvious that they are XBASIC format by the fact that the start of each line is indented by two spaces on the VDU. Another interesting point is that graphic characters are listed as their true representation rather than their reserved word counterpart.

MON - Abbreviated form of the MONITOR command available in ROM BASIC.

POS - POS(J) where J is an expression which must evaluate to an integer in the range 0 to 255 is used to obtain the current output column or row position according to the value of J.

POS(0) returns the print column count up to a maximum of 255 when it is zeroed. It is independent of screen size. It is a command mainly used in conjunction with printers.

POS(1) returns the current column position of the cursor on the VDU.

POS(2) returns the current row position of the cursor on the VDU. The latter two are designed to be used alongside the PRINT@ facility.

PRINT@ - Allows printing of expressions at specified points on the screen. Similar in use to the ROM BASIC SCREEN command except that the coordinates 0,0 are from the top left of screen. The coordinates may be up to 255 and in this case wrap-around will occur.

PRINT# - When data files are open PRINT# followed by strings or numerical variables will allow writing to the file.

INPUT# - As above but in the reverse direction! Superbly easy to use.

SET - As for ROM BASIC except as with the PRINT@ command above the coordinates 0,0 are at top left. Incidentally, the SET command is not followed by the coordinates within brackets as it must be within ROM BASIC. Similar rules apply to the RESET command.

RND - RND(1), as for ROM BASIC, returns a random number in the range 0 to 1 although it is a true random number, the routine making several uses of access to the refresh register of the Z80 CPU.

RND(0) returns the last random number produced.

RND(I) where I is in the range 2 to 65535 will return an integer random number ranging from 0 to I-1. e.g. RND(9) produces an integer random number in the range 0 to 8.

RUN - Programs may be run direct from disk by just entering RUN followed by the name of the program as stored in the disk directory. e.g. RUN"TEST" will load the program TEST from disk and proceed to run it.

TAB - As for ROM BASIC except that when additionally followed by two values will print characters specified by the second value until the print head reaches the column specified by the first value. e.g. PRINT TAB(20,46) assuming the cursor is placed on the first column of a line, will print twenty full stops.

WIDTH - Sets the width of the current output device when an automatic CRLF is generated. Set by default to 0 when no automatic CRLF is generated.

LOG - Followed by a suitable number returns the base 10 logarithm. The ROM BASIC LOG command returns the natural (base e) logarithm.

LN - Followed by a suitable number returns the natural logarithm!

And now the exclusive command set:

AUTO - Followed by up to two values. An automatic line numbering whilst entering programs facility. e.g. AUTO 100,5 will start auto numbering from line 100 in steps of 5. Default values are from line 10 in steps of 10. Unfortunately you can fool this command by editing another line whilst in the auto mode and then entering this new line. Although this line will be entered correctly the next line number automatically printed will have increased by the increment even though there has not been a line entered with the previously printed auto line number. The only way to revert to normality is to abort the auto mode by ESC and then issuing a new AUTO command. Absolutely essential to program writing though!

CALL - Followed by a suitable expression will call a machine code routine at the address specified. The routine should normally end with a RET instruction. e.g. CALL &C800 will call the machine code routine located at C800H.

DEL - Followed by two values will delete the XBASIC program lines between the lines specified. e.g. DEL100,200 will delete all lines from 100 to 200 inclusive.

DRIVE - The default drive in the disk version is the disk drive A (logical drive 0) and this may be altered with the DRIVE command. Any read or write commands after this will default to the drive specified unless specified otherwise in the command (see later disk commands). e.g. DRIVEB will now default to the second disk drive. DRIVET will now read and write data to and from tape when required.

ELSE IF THEN : ELSE - need anymore be said?

EVAL - I'm afraid the usefulness of this command escapes me. The command will evaluate text in a string expression. e.g. A\$="1+X-EXP(X/3)":Y=EVAL(A\$).

FMT - Requires a lot of words to explain its detailed use but suffice to say it is a PRINTUSING type of command which can specify the number of figures to

be printed in front of and behind a decimal point.

HOLD - A real beaut. this one - especially when used with the CHAIN command. HOLD followed by up to two line numbers will hold a range of lines for view in a program so that another program may be appended to it or so that this range only may be renumbered and thus moved to another part of the program. The effect is that the rest of the program seems to have disappeared. In fact it is still resident in memory but cannot be found with a LIST command nor executed with a RUN command. The listable area can be modified however and even RUN without affecting the hidden areas. LOADING another program removes the listable area and also the upper hidden area if there is one but does not destroy the lower hidden area. e.g. HOLD100,199 leaves lines 100 to 199 in view and a command such as a renumbering command will only renumber that which is in view.

MGE - Restores sanity to a held program.

HEX\$ - Followed by up to two values returns the hexadecimal string corresponding to the first value as a string of characters whose length is dependant upon the second value (4 assumed if not specified). e.g. HEX\$(1234) returns the string 04D2. HEX\$(100,2) returns the string 64.

INCH - Returns the ASCII value of the next input character which it waits for. e.g A=INCH places the ASCII code of the next key pressed into the variable A.

INCH\$ - Returns the next key pressed as for INCH above but as a one character string. e.g A\$=INCH\$. Also useful in conjunction with tape or disk files when certain characters may cause undesired effects under an input condition. See later commands on file handling.

KBD - Similar to INCH but scans keyboard and does not wait for a character.

KBD\$ - As for INCH\$ but does not wait for input.

RENUM - Renumbers a held program or the whole of it if a HOLD has not been previously issued. Used in conjunction with HOLD and MGE it is a very versatile command.

IOM - Followed by two values. This command alters the bits in the IOMOD word in the workspace which consists of sixteen one bit flags of which seven are used at present in XBASIC. By setting the bits to 0 or 1 the operation of certain areas can be modified. These require more explanation than is possible here but it is with this command that one prevents the switch from Screen Edit mode to Line Edit mode during a program run or disable the use of the break key etc.

MOD - An operator which is the remainder from a division. It can be defined as follows: A MOD B = A - B * INT(A/B). e.g 5 MOD 3 returns 2.

MUL\$ - Followed by a string and a number returns the string repeated a number of times dependant upon the number. e.g. MUL\$("*",5) returns the string *****.

PI - Returns the value 3.14159. Very useful with the AVC and is faster than using a variable to hold the number for pi. This function actually uses 3.141593 but is only printed to five decimal places.

POP - Removes one address off the stack of GOSUB addresses so the next RETURN will branch one statement beyond the second most recently executed GOSUB.

PTR - Followed by two values allows the setting of selected scratch pad locations without using POKE or DOKE. The advantage of this is that the XBASIC workspace area is different for various implementations. For example the worksp-ace locations in the Nas-Dos version starts at 1006H but in the CP/M implementation, it is at 0100H. Thus some compatibility is achieved. There are 24 selected locations. e.g. A=PTR(14) places the address (in decimal!) of the end of the XBASIC program currently in memory into variable A.

SCRN\$ - Followed by a number which must be less than the number of rows on the screen (16 on a standard Nascom), returns the string of characters from this row number, the length of which will always be equal to the number of columns on the screen (48 on a standard Nascom).

SEP - Followed by an ASCII value alters the separator found in input statements - normally a comma. e.g. SEP 47 makes / the separator. SEP 0 allows any characters to be input in INPUT statements including commas - although in this case, only one input at a time would be allowed.

SIZE - Returns the current space available for program use. As for FRE(0) in ROM BASIC.

SPEED - Followed by a number from 0 to 255 sets a delay in the character output to the current output device. 0 is very slow whilst 255 is normal speed.

SWAP - Followed by two variables swaps the contents of the variables which may be numeric or string types. Saves a lot of space in sort routines.

ZONE - Followed by two numbers sets the print zone (tab) width and the largest column for which printing to the next zone will stay on the same line. Default values for a standard Nascom are 14 and 36 respectively.

ON ERR - Errors can be trapped and dealt with as you desire. It is possible to deal with certain errors only and then continue in the program whilst unexpected errors will still break out of the program. The function ERL returns the line number where the error occurred, ERR returns the number of the type of error -there is a list in the manual, ERR\$ returns the error string message without the word "error".

OFF ERR - Turns off the ON ERR command. An ON ERR command will turn off anyway after an error has occurred in case the error routine has an error in it(!) but the error dealing routine can always turn it back on again at the end of the routine.

ON EOF - As for ON ERR except that this deals specifically with the encounter of an end of file when reading disk or tape data files and which would normally cause an End of Text Error. If ON ERR and ON EOF are both in force then the latter takes precedence if an end of file condition is sensed. The main difference between the two error handling statements is that the ON EOF routine does not turn off after an EOF error.

OFF EOF - As for OFF ERR, turns off above command. The OFF commands are not required if the program ends normally as both modes are automatically turned off.

DIR - Displays the directory of the currently logged disk drive. This is shown in a different format to when displayed under Nas-Dos. Each file name is shown with the type of file (XBS for XBASIC, OBJ for object code, ASC for ASCII type files or how you wish to suffix them!). Nas-Dos object code files and ROM BASIC files are shown with strange file types. Locked files are indicated with an asterisk in front of the filename and the size of each file is given after the name to the nearest lk above its actual size.

LOAD - Followed by a filename loads a file from tape or disk. e.g. LOAD"TEST" will load the XBASIC program called TEST from the current logged drive, disk or tape. Any program in memory is lost but variables are not destroyed. LOAD"B:TEST.OBJ" will load the object code file called TEST from disk drive B (irrespective of the current logged drive) into memory starting at the location reserved for it by the CLEAR command.

SAVE - Similar in operation to LOAD command above only, of course, saves the file to disk or tape. Three types of file may be saved with this direct command -XBASIC programs, object code and a program as an ASCII file. If the filename is already present on disk then the file with that name will be erased before saving the current program!

ERA - Followed by a filename will erase the disk file of that name from the directory.

LOCK - Followed by a filename will lock the file so any attempt to erase or save a file of the same name will produce an error message. UNLOCK removes the facility.

REN - Followed by two filenames will rename one file for the other.

VERIFY - Followed by a filename will verify the file, reporting any checksum error as a Bad Data Error.

CHAIN - Followed by a filename will load and execute a program whilst preserving variables. Particularly useful when used in conjunction with the HOLD command such that common subroutines at the beginning of a program may be "held" and then another program CHAINED from disk, an automatic MGE being performed, the resulting program then being able to make use of the previously "held" subroutines. Thus any number of programs may be "chained" which make use of a common set of routines. I find this command most useful.

CREATE - Followed by a valid filename will create a file for subsequent data storage using the PRINT# command. Random or sequential access are both catered for.

OPEN - Followed by a filename will open a previously created file for subsequent writing (PRINT#) or reading (INPUT#).

APPEND - Followed by a filename will open the file for subsequent writing (PRINT#) or reading (INPUT#).

APPEND - Followed by a filename will open the file for subsequent writing onto the end of the current data in the file.

CLOSE - Followed by the appropriate string handler for a file will close the file. Unlike the ROM BASIC routine handler within Nas-Dos, XBASIC allows any number of files to be open at one time subject to memory limitations. Close by itself will close all files that are open at that time.

The file handling within XBASIC is very easy to use. File handling in ROM BASIC is, in my view, cumbersome, although it was reasonably straight forward to store strings but numeric data was more of a pain. In XBASIC, file storage and retrieval of strings and numeric data is made as simple as I think it can be whilst still retaining versatility.

Apart from the previously mentioned bug in the AUTO command, there are at the time of writing (July 1983) several known bugs in the Nas-Dos version of XBASIC which I am told is being looked into by Nascom. The two most drastic ones are:-1) ASC files cannot be saved to disk although they can be saved to tape and 2) attempts to retrieve programs or data from tape proves fruitless. Hopefully these will be corrected shortly.

CONCLUSIONS

I find that I now no longer use ROM BASIC because XBASIC does all that ROM BASIC does and more. Certain commands are lacking which need to be added as debugging aids;namely TRACE and FIND routines etc. No doubt these will appear as XBASIC becomes more widely used. Failing that I will have to write the routines myself! My first attempt at writing an OLD command to retrieve NEWed programs resulted in disaster as I could only recover the first line!

All in all I find XBASIC to be a useful interpreter and one that any Nas-Dos user should not be without if only for the added disk commands that make file handling so much easier than in ROM BASIC.

SMALLADS (continued)

For Sale

RAM-A board, 32k RAM, 2716 conversion (no EPROMs). With INMC mods, but still won't work at 4MHz without waits (dunno why...) #45 ono. HS1N twin-drive digital cassette system. As reviewed 80-BUS News Vol 1 iss. 4. With original ROMs (2708s, assembled for #D000 and ports #F8 to #FF) or my enhanced OS (2716 - assembled for #D000, and ports #78 to #7F); 4118s, connecting leads and some tapes. #170 ono Adrian Perkins Tel. Bracknell (0344) 485816.

Gemini Galaxy 2 Computer System, as new. Complete with WordStar and MailMerge. #1300 ono. Tel. Peter Forsyth, Chesham (0494) 782449.

Colour/Sound/RTC, etc, card (Holmes - R&EW) as advertised in Vol 3, Iss 1. Also HENELEC GM805 Disk Controller card. Any reasonable offers considered. Chris Bowden (0209) 860480 for details.

An Insight into the Gemini IVC and SVC

By D. W. Parkinson

Over the years I have waited for a review of the Gemini IVC to appear in the pages of 80-BUS News. As nothing has appeared to date, I have decided to put pen to paper to rectify the situation. What follows is not a review, but an insight into the IVC (GM812) and the SVC (GM832). Why an insight? Well I had some influence on the specifications for the boards, and I am the author of the IVC and SVC monitors. As such I am unlikely to be an un-biased reviewer, but hopefully this article will give you a deeper understanding of the whys and wherefores of the software, and the capabilities of the boards.

Background

Back in the dark ages, when there were only Nascoms, and the Nascom 2 was stuck in that black hole of Receivership from which only a trickle emerged, the Gemini system was born. (The demand was there, and the dealers needed something to sell.) After much discussion Gemini decided that in their system the video section should be separate from the CPU card, and that it should have its own dedicated processor. Although this was a more costly approach than that of the Nascom 2 it had various advantages:

- (1) Those requiring a CPU card for dedicated control applications need only buy a CPU card (GM811).
- (2) Flexibility.
- (3) Increased performance.

Item (3) needs some further explanation as, on first examination, one would think that nothing could be faster than a memory-mapped display. After all, with a memory-mapped approach, the CPU is putting the characters directly into the display, so why take any other approach? The reasons for doing this are many, and I'll try and give the pros and cons.

Loss of CPU memory space I. A memory mapped display eats up CPU address space, (c.f. the 32k BBC. Put it into hi-res graphics and you are only left with 12k), although this could be got round by switching the display memory into the normal address space only when it is required (as done by the Nascom AVC).

Loss of CPU memory space II. A memory mapped display requires software to drive it, and the more sophisticated the software the more space it takes up. Once again the software could be contained in ROM on the video card, being paged in along with the display memory (Not done by the Nascom AVC).

Speed. Unless the video driver is incorporated within the applications programs (which would ensure they were non-portable), there is the overhead of the driving software to consider, and the case of 'transparent access'. The screen memory is usually shared between the CPU and the display controller. If the CPU updates the screen memory whenever it wants to without regard to what the display controller is doing, the result is visible interference on the display. This can be very irritating and most systems aim to provide an interference free display by synchronising the CPU and display controller. In general this is most easily achieved by only allowing the CPU access to the shared memory during the intervals when the display is blanked, and the resultant 'waiting' wastes a certain amount of the CPUs time. Similarly scrolling a full screen takes an appreciable amount of time, although this could be reduced with specialised hardware. Anyway, by the time you have a paged display card including on-board software in EPROM, it is only a small incremental cost to give it its own processor, and to talk to it via a few I/O

ports. This now gives you parallel processing. The main CPU can pass a character over to the slave CPU saying `deal with this please', and can then get on immediately with its job of running your program.

I have recently heard of a problem somebody has been having with the Nascom AVC that nicely illustrates the point. He is (was?) writing a communications package to run on his Nascom, and had run into trouble because he couldn't get incoming characters into the display fast enough. The IVC can suffer from a similar problem, (scrolling takes an appreciable time), but this can be circumvented very easily by implementing a small circular buffer and alternately polling the UART and the IVC [1]. Here one processor is handling the display and keyboard while the other is acting as a postman between it and the UART, and the postman can temporarily hold onto things if the display is busy. However with the AVC the problem is more fundamental as the postman and display driver programs can only run sequentially, not concurrently. Having given the output routine a character to display the AVC driver was sometimes taking so long to return that several incoming characters were lost. This could be circumvented in two ways:

- i) duplicate the screen driving code, and add `poll UART & buffer' calls to it in the middle of all routines that take an appreciable length of time to execute. (i.e. Combine the `postman' and `display driver' in a single program, interleaving the two like a `riffle shuffle' with a pack of cards.)
- ii) Run the UART under interrupts.

The latter approach needs careful thought if the screen driver is busy paging various parts of the memory in and out! Also it may interfere with any software approach to transparent screen access.

The IVC hardware

The IVC is basically a small SBC (single board computer) containing the following: A Z80A, 6845 display controller, 2K of shared screen memory, 4k of character generator that may be a mixture of RAM and EPROM (the standard is 2k of EPROM, 2k of RAM), 2k of workspace memory, EPROM program memory (2716 or 2732), and a few I/O ports. To the host 80-BUS system it appears as three I/O ports. One port is a read/write port through which data and commands are passed to/from the IVC, one port is a read-only status byte that holds the full/empty flags for the data port. The final port is a write-only port that resets the IVC's processor.

Within that hardware context the features of the IVC are provided in software by the on-board monitor IVC-MON. There have been several versions of the monitor, and in the following sections I will cover various aspects of the monitor, finishing off with a description of the differences between each version.

IVC-MON

I don't intend to regurgitate the software manual here - those that are interested should be able to obtain a copy through their friendly Gemini dealer - but table 1 is a copy of the command summary from the manual for those that are not familiar with it.

Command Summary

NOTE:

1. These are the control codes that the IVC responds to. They are normally sent to the IVC by a User program or the Host's operating system. (See section 4 - PUTVID).
2. Programs that request information from the IVC will have to read the information back from the IVC. (See section 4 - GETVID).
3. It may not be possible to issue commands to the IVC direct from the keyboard attached to a system, as the operating system may modify the characters typed. (eg the standard CP/M line input routine would echo ``[`` to the IVC if the ESCAPE key were pressed, rather than the ESCAPE code. - However note that this problem can be solved in CP/M systems by selecting "EDIT mode", where all characters are echoed to the IVC exactly as typed).
(Also see section 6.2 - Nested Escape sequences).

Command Summary	<u>Screen format</u>	1B 31.. <ESC> 1B 32.. <ESC> 1B 33.. <ESC> 1B 46.. <ESC> 1B 42.. <ESC> 1B 56.. <ESC> 1B 49.. <ESC> 1B 4A.. <ESC> 1B 41.. <ESC> 1B 4E.. <ESC> 1B 4D.. <ESC> 1B 4F.. <ESC>
<u>NOTE:</u>		
1. These are the control codes that the IVC responds to. They are normally sent to the IVC by a User program or the Host's operating system. (See section 4 - PUTVID).		
2. Programs that request information from the IVC will have to read the information back from the IVC. (See section 4 - GETVID).		
3. It may not be possible to issue commands to the IVC direct from the keyboard attached to a system, as the operating system may modify the characters typed. (eg the standard CP/M line input routine would echo "```" to the IVC if the ESCAPE key were pressed, rather than the ESCAPE code. - However note that this problem can be solved in Gemini CP/M systems by selecting "EDIT mode", where all characters are echoed to the IVC exactly as typed). (Also see section 6.2 - Nested Escape sequences).		
General	<u>Character set</u>	1B 43.. <ESC> 1B 63.. <ESC> 1B 47.. <ESC> 1B 52.. <ESC> 1B 53.. <ESC> 1B 48.. <ESC> 1B 68.. <ESC>
Cursor movement	<u>Block graphics</u>	1B 47.. <ESC> 1B 52.. <ESC> 1B 53.. <ESC> 1B 54.. <ESC>
1C ^] Cursor left 1D ^[Cursor right 1E ^A Cursor up 1F ^B Cursor down 1B 3D..- <ESC> "=" RR CC Cursor addressing	<u>Keyboard</u>	1B 66.. <ESC> 1B 6B.. <ESC> 1B 4B.. <ESC> 1B 58.. <ESC>
Additional cursor operations	<u>Miscellaneous</u>	1B 57.. <ESC> 1B 4C.. <ESC> 1B 55.. <ESC> 1B 50.. <ESC> 1B 76.. <ESC>
Screen editing		
OB ^K Delete line and scroll up OE ^N Insert line 16 ^V Delete character in line 17 ^W Insert character in line 1A ^Z Clear screen		
1B 16 <ESC> ^V Delete character in screen 1B 17 <ESC> ^W Insert character in screen 1B 25 <ESC> "%" Delete to end-of-screen 1B 2A <ESC> "*" Delete to end-of-line 1B 5A <ESC> "Z" Return contents of current line		

Odd features of the IVC stem from the Nascom background. e.g. the alternative screen format being 48 characters per line. The block graphics are those of the Nascom 2, rather than the teletext standard. (Same symbols, just controlled by different bits within the byte.)

The IVC accepts 8-bit characters from the host system. All the standard printable ASCII characters (whose codes are in the range 20-7F), and all characters for the alternate character generator (whose codes are in the range 80H-FFH), are placed directly into the display at the current cursor position. Characters in the range 00-1FH are interpreted as control characters, and are used to control the extensive features of the IVC.

One reason why I have been looking for a review of the IVC is to see if anybody feels that important features are lacking from the escape sequences. (Do you all want protected fields? Do you want a "page" mode as well as a "Scroll" mode?) My feeling, echoed recently in these pages by David Hunt, is that 97% of users do not use anything other than the few basic functions like `cursor addressing' that are used by programs such as Wordstar and dBase II.

Attributes

The IVC supports a single flexible attribute. This is the programmable character generator (pcg), sometimes referred to in the IVC documentation as the alternate or upper character generator. On power-up the contents of the standard character generator (an EPROM) are copied to the alternate character generator and are inverted on the way. This means that by using the alternate character generator, (either by setting the msb of a character before it is sent to the IVC, or using the sequence <ESC> A to make the IVC do it for you), video-inverse characters can be displayed. (e.g. to highlight headings and menus within Wordstar and other such programs.) Note that inverse video is the default setting, there is nothing to stop you programming other bit combinations into the character boxes. For example you could set up a Greek alphabet or a series of specialist mathematical symbols for use by a custom wordprocessing package. The Henry's package HI-RES uses the pcg to offer a pseudo high resolution graphics mode on the IVC.

Transparent access

As I mentioned earlier, the absence of visual interference on the screen when the on-board processor accesses the screen memory is essential. There are various ways that this can be achieved, but this to some extent depends upon the choice of controlling microprocessor. If a Motorola 6800 had been used for the IVC processor, then a very neat arrangement could have been implemented where the processor and display accesses could have been interleaved, leaving each controller in total ignorance of the other. However history dictated that a Z80 was used. The Z80 has asynchronous memory cycles, (asynchronous to the extent that its memory cycles are not all the same length), and so it can not easily be arranged to access the display memory between each displayed character. With the IVC design the easy way out was taken, and the Z80 was given priority access to the display memory. (i.e. it can barge in whenever it likes.) However the horizontal and vertical sync pulses and the display blanking signal from the display controller are routed to a port so that the Z80 can monitor their state. Thus, by monitoring this port, it knows when it can safely `barge in' and update the screen memory without causing any interference on the display.

In general, there is a surprisingly small impact on performance in taking this approach. This can be seen by considering the usual practical overheads in displaying a line of text on the screen. To do this an application program loads a byte from memory into a register, updates and saves the pointer into the character string, and then calls the operating system to output that character. The operating system then determines where to send the character, (e.g. checking the IOBYTE in CP/M), and finally sends it on its way. A simple assumption that can be made in this scenario is that the IVC can put characters into the screen display at a maximum rate of one per displayed line, i.e. one every 64uS during the line blanking interval. (A remote terminal would have to work at a line rate of 156k baud to match this!) This display rate of one byte per 64uS is likely to be of the order of the Host system's overhead, if not a bit faster, and so the display rate is actually limited by something other than the internal display technique of the IVC. (Note that once the IVC has accepted the character for display the Host system is free to go off and collect the next character. Thus the IVC's 'average' waiting time of 32uS to get that character into the display is not wasted time, as the Host is not held up by it.)

The one time that the display technique does have an impact on the performance of the IVC is in the scrolling rate. Don't get me wrong - the IVC doesn't scroll exceptionally slowly - it's just that it could go faster! I did briefly experiment with using a hardware feature of the display controller that resulted in a scroll rate of over 1000 lines/sec but dropped it for various reasons:

- i) It was far too fast and would require another escape sequence to let users define the scroll rate they wanted.
- ii) It complicated the monitor software in various places.
- iii) It made the implementation of the 'non-scrolling' lines more complicated, and if a significant number of lines were 'locked' on the screen the situation returned to the original problem. (Either 'locked' lines are left alone and the rest of the screen memory contents moved, or the main screen stays in place in the display memory and the 'locked' lines have to be moved to the new display start address!)

Keyboards

The IVC includes a port to which a keyboard can be attached. This makes sense as it effectively turns the IVC into a terminal which is then connected to the Host 80-BUS system. (The connection is via the 80-BUS backplane rather than a conventional RS232 link.) The IVC continually monitors the keyboard port and passes all characters from the keyboard to the Host via an internal 64-character buffer. Thus, whenever a key on the keyboard is pressed, the character is read and stored in the buffer from whence it is extracted when and if the Host system asks for it. Thus a fast typist can hammer away secure in the knowledge that nothing will be missed. In fact it is quite useful to be able to 'type ahead' with some programs, although this should be done with caution until you are sure that the program will accept it. (Some programs poll the keyboard often, and 'swallow' characters they don't recognise. Others stop polling as soon as they receive a single character that is not immediately applicable, but they hold onto it for later.)

The point I'm trying to make is that once you are familiar with certain programs you can improve your throughput by typing in responses before the next prompt appears. For example you can speed up the time taken to load a Wordstar file and to get to a certain position within it. Having typed WS to

invoke Wordstar wait until Wordstar has loaded and has started on its first screen display. (It seems to `swallow` the odd character if you start too soon.) You can than type (without any pauses) a command like **DARTICLE.DOC<RETURN>^QFKeyboards<RETURN>2<RETURN>**. This magic invocation (for those unfamiliar with Wordstar) would open a document file ARTICLE.DOC, start to load it into the memory buffer, search through the file and put the cursor at the second occurrence of the word "Keyboards". WORDSTAR would do this without all the normal in-built delays, and also without bothering to display the first page of the document.

One very useful feature came in with release 2.0 of IVC-MON. At this time Gemini introduced a new keyboard, GM827, which included a row of extra function keys along the top, together with a numeric pad on the right-hand side. IVC-MON 2.0 catered for this keyboard, and while providing default settings for these extra keys, it allowed users to define alternative character strings for them (including the numeric pad). Note the use of the word "strings". If required, pressing one of these keys can result in an entire sentence (or paragraph!) being returned by the IVC. I find them useful in a variety of contexts: During program development they can be used as an instantaneous SUBMIT feature (e.g. define a key as **<M80 =PROG^M L80 PROG,PROGLIB,PROG/N/E^M>** - which runs the assembler followed by the linking loader). For standard programs they can be set up to suit the program's command codes. (e.g. Keys for INSERT, DELETE etc in editors). Oft used words or phrases can be put a single key (e.g. Please send the money).

The keyboard port on the IVC is an 8-bit port that is sub-divided into 7 data bits and a strobe pulse. This meant that some way had to be found for the IVC to distinguish the special function keys from the 128 standard ASCII characters that are generated by the other keys on the keyboard. This was done by having the GM827 keyboard return double-byte codes for the special keys, and, as the ESCAPE code (0Bh) was chosen as the lead-in code, the ESCAPE key is in fact one of the special keys. (The IVC-MON software however does not allow it to be reprogrammed to an alternative code or character string.)

Every time one of these special double-byte codes is received, the IVC converts it to a single byte code in the range 80H-BDH and stores the byte in the keyboard buffer. (Note that the normal keys can only produce codes in the range 00-7FH.) When the IVC passes bytes from the keyboard buffer to the host system it checks the msb of each byte. If the msb is found to be set it then looks for that keycode in an internal table it has its workspace RAM. The code in the table marks the start of the string that is to be sent in place of the function key. IVC-MON then sets a flag to say "get characters from this string rather than the keyboard buffer", which remains set until the string is exhausted. One point to notice from this is that a function key only uses up one byte of the `type ahead` buffer. So, if you're somebody who often goes charging ahead of PIP or your assembler, and occasionally `bust` the buffer, the answer is to put some of your typing under function keys (if this is possible). The programmable function keys I find a powerful feature, and this the IVC provides at zero overhead to the Host system thanks to it having its own processor on-board.

IVC-MON Release history.

Vers.

1.0 The original release.

- 1.1 Various small items were changed. (The unlikely sequence ESC ESC % % used to crash the system). 'Delete line and scroll up' now cleared the bottom line of the display if the cursor was there. (Previously nothing happened.) Cursor addressing, Set, Reset, and Test now accepted nested escape sequences during the coordinate string.
- 1.2 A few small internal changes made. The input buffer (used to hold incoming characters while the screen is being scrolled) was reduced to 64 bytes from 128 bytes, (128 bytes was a bit of a waste), and the internal stack area was increased.
- 2.0 Support for the Rotec keyboard and programmable function keys added, hence the jump to 2.0. The monitor size now exceeded 2k and required a 2732 rather than the 2716 of the earlier releases.
- 2.1 The current release of IVC-MON. Sections of the monitor code were restructured to give a faster response to keyboard poll requests. This has a noticeable effect on the screen update rate for operating systems such as CP/M where the BDOS polls the keyboard everytime it outputs a character. Another Escape sequence was added to allow lines to be 'locked' at the bottom of the screen.
- (3.0?) Although the IVC has now been discontinued one further update of the software is envisaged. (When time permits - don't expect it tomorrow!) This will include SOME of the features that have appeared in the SVC monitor (see below).

The SVC

With a new design on the drawing board Gemini was faced with the usual decision - Do they use a 'state-of-the-art' device and design a system around that and accept the attendant constraints? (e.g. the in-built character set and screen formats offered by some graphics controllers.) Or should they remain as far as possible compatible with the IVC? They - wisely in my view - opted for the latter case.

So following the IVC came the SVC. Why SVC? well they had to call it something and it does bear some relationship to the IVC. In fact it is upwards compatible with the IVC. (i.e. You can replace an IVC by an SVC and 99.9% of programs will continue to run as before.) So what do you get by moving from an "I" to an "S"? Well the Z80A changes to a Z80B (6MHz operation), the screen memory goes up in size from 2k to 8k (although some of it doubles as the character generator), a 256x256 pixel graphics mode appears, a 40 x 25 display option is provided (the 0.01% incompatibility), a UART can be added on a piggy-back board to make the SVC a stand-alone terminal, a range of attributes appears, a buzzer is added, a Gemini serial keyboard can be used and finally you get ESP.

I think that is about enough for now, and so I'm afraid you will have to wait until the next issue for part II to find out about the ESP.

REFERENCES.

1. PARKINSON D.W., "Remote Terminals and the Galaxy", 80-BUS NEWS 1-4 Nov-Dec 1982, pp45-46.
-

HIGH SPEED ARITHMETIC PROCESSOR

SPEED-UP YOUR PASCAL PROGRAMS

The HSA-88B floating-point arithmetic processor is a 80-BUS/Nasbus compatible board which uses a microprogrammed 16/32 bit microcomputer IC which performs arithmetic and trigonometric calculations 10 to 100 times faster than the best Z80 software routines. For example, a 32 bit floating-point division takes just 90 microseconds and a 32 bit arctangent executes in only 2500 microseconds. A large number of 16/32 bit integer and floating-point functions from $x+y$ to x^y is accessible with simple single-byte commands. All accesses to the HSA-88B are via two I/O ports (selectable from 80H to FOH). The HSA-88B is a true simultaneous co-processor capable of performing one operation while your Z80 CPU is doing something else. This is ideally suited to animated graphics where the CPU, the HSA-88B and the graphics card can perform their functions at the highest possible speed.

The HSA-88B is easily used from within assembly language programs. High level language programs require a compiler with modified run-time

TRANSFORM YOUR GAMES AND BIT-MAPPED GRAPHICS

routines. We are offering with every HSA-88B a FREE latest Hisoft HP5 Pascal compiler which has been specially adapted to compile HSA-88B-oriented code. This compiler is already extremely fast and with the HSA-88B it outperforms all other Z80 Pascal compilers, in many cases by an order of magnitude. The standard Pascal variable types plus 32 bit integers (ideal for financial applications) are supported together with a full range of maths functions rarely seen in Pascals or Basics. The size of the run-time routines is greatly reduced over other compilers because the HSA-88B performs the arithmetic functions in hardware.

The complete package consists of the HSA-88B processor card, HP5 compiler on Gemini 5½" DSDD disc (other formats available including Nascom 5½" and IBM 8" SSSD) and HSA-88B and HP5 documentation and programming examples. Package price £268 plus VAT, UK postage free. Not suitable for Nascom 1.

BELECTRA LTD. 11 Decoy Road, Worthing, West Sussex BN14 8ND Telephone 0903-213131

NEW SUPER DISKPEN (PENVG:3)

DISKPEN has been rewritten and revised. This popular text editor/formatter now includes a 'HELP' facility, and new features for the print control of the most popular printers, underline, bold, etc (also user patchable for the less popular types). New features include block delete, better move commands, new cursor control, optional hyphenation, visible indentation setting and lots more. A major enhancement is the ability to handle overlay files so that PEN can use auxiliary packages such as the MAXIFILE free field file searching utility or the print spooling utility.

The new DISKPEN is useable on all Gemini multiboard computers (Galaxy, Kenilworth, Quantum) and Nascom/Gemini hybrids, (MAPPEN is available for

BDOSZ

Yes, you guessed it. Some enterprising person has now 'disconboobered' the BDOS in CP/M and rewritten it as a Z80 program. Its fully compatible with the original with no bugs found to date. Because it's written in Z80 code it's smaller, this has allowed room for tidying up all the annoying stupids in the original BDOS so that errors like:

BDOS ERROR ON X: R/O

which usually causes you to lose everything you've just done, becomes the far more helpful:

Disk x: is set R/O

Do it anyway? (Y/N/^C)

Which, of course, means you don't lose anything. It even allows you to change disks when they are full without loss of data. In all, a lovely piece of software. Available in most popular 5.25" formats (please state when ordering) at 11.50 inc. VAT.

Carriage & packing 50p

users of the MAP video card). DISKPEN is available as an upgrade to earlier DISKPENs and GEMPENs at 17.25 inc. VAT., or to new purchasers at 57.50 inc. VAT. (Please state disk format when ordering.)

MAXIFILE overlay 23.00 inc. VAT (20.00 + VAT)

SPoolER overlay 17.25 inc. VAT (15.00 + VAT)

Carriage & packing 50p

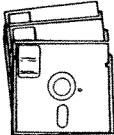
ALLDISC VARIABLE DISC FORMAT UTILITY

ALLDISC is a new approach to the problem of lots of different machines all with different disk formats. Designed for use with the Gemini and Nascom CP/M computers, ALLDISC replaces the existing disk drivers and the CP/M disk parameter headers from an archive of different formats. Up to 6 drives and two controller cards are supported, the drives may be a mixture of 8", 5.25 or 3.5" types in either single or double density and may be single or double sided. 96 tpi 5.25" drives can be made to 'double step' to read and write 48 tpi disks. The Gemini GM833 512K virtual disk RAM is also supported. All this adds up to a very powerful system where the drives fitted to the system may be reconfigured to read and write disks of different formats. The limitations are that the disks must be to IBM3740 CRC standards and must be soft sectored (or in other words, it will cope with most disks).

The archive is supplied with nine useful formats when supplied, with others being made available to registered users free of charge for the first six months from registration. The archive can be edited and new formats created. Disks can be formatted from the archive so there is no need for preformatted disks when transferring software.

ALLDISC is supplied with full documentation and hints and pointers to discovering unknown disk formats. ALLDISC costs 172.50 inc VAT (150.00 + VAT).

Carriage & packing 50p



ORDER BY POST OR TELEPHONE OR CALL IN AND SEE FOR YOURSELF

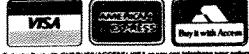
COMPUTER SHOP

404/406 Edgware Road, London, W2 1ED

Telephone: 01-402 6822

OPEN 6 DAYS A WEEK

Credit Sales available ask for details.
Official orders welcome.



DRH 840613

E. & O.E.

TRANSDATA MODEM CARDS.

Brand new, tested and aligned 300 BAUD answer and originate acoustic telephone modem cards manufactured by TRANSDATA for their model 317 acoustic modem. Full CCITT specs. Requires only a few components to complete (no case available). Computer interface is 300 BAUD RS232 I/O. The power supplies of +12 volts at 180mA and -12 volts at 170mA are required. Card size 350mm x 105mm. May be directly connected using the GEC LTU 11 coupler (see below). Full circuit description, drawings and connection data supplied. Suitable for use with computers fitted with 300 BAUD RS232 I/O. Software for Nascom & Gemini published in the last issue.

Card only at 29.95 inc. VAT. (26.04 + VAT)

Supplied with LEDs connector, switches and transducers at 34.95 inc. VAT. (30.39 + VAT)

Supplied with LEDs connector, switches and GEC LTU 11 directcoupler at 45.95 inc. VAT. (39.96 + VAT)

Carriage & packing 1.00

BIG BIG POWER SUPPLIES

Ex-equipment GOULD power supplies for those who need a lot of power.

Type 1 +5 volts at 20 amps. 7" x 4" x 3.5" o.a. Switch mode. 220-240 V AC in.

39.95 inc. VAT. (34.73 + VAT)

Carriage & Packing 2.00

Type 2 +12 volts at 4 amps, +5 volts at 40 amps

-5 volts at 1 amp, -12 volts at 1 amp

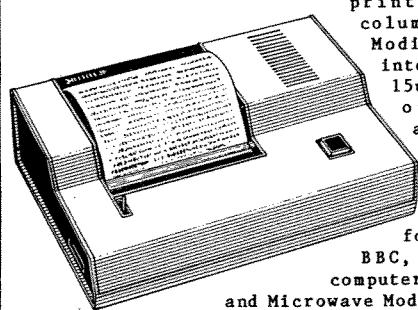
14.5 x 6" x 3.5" o.a. (including fan)

Thyristor switch mode. 220-240 V AC in

79.95 inc. VAT. (69.52 + VAT)

Carriage & packing 4.00

MULTITECH MPF-II PRINTER



The Multitech MPF-II thermal matrix line printer features 40 column print width. Modified Centronics interface (requires 15uS strobe or use our interface adaptor). 150 - 180 lines per minute. Bidirectional 7 x 10 matrix. 4.4" paper. Suitable for Nascom, Gemini, BBC, Dragon, Tandy computers, etc., and Tona and Microwave Modules RTTY readers.

Unrepeatable value at 49.95 inc VAT. (40.43 + VAT.)

Interface adaptor with leads 14.95 inc. VAT (13.00 + VAT). Paper 2.95 per roll inc. VAT. UK Carriage & packing 1.00

GEC LTU 11 Mk II PRETEL MODEM CARD

Manufactured for the GEC stand alone PRETEL unit, computer interface is TTL logic level at 75 BAUD transmit and 1200 BAUD receive. (Suitable for use with most computers fitted with speed selectable UARTs or twin independant serial I/O at TTL levels.) Output level is at -13dBm 600 ohms, input at -36dBm, for connection to the GEC LTU 11 direct coupler. Power requirement +5 volts at approx. 250 mA. Designed for direct connection to the GEC LTU 11 direct coupler, to provide line connect and autodial facilities. Card size 266mm x 140mm. Supplied with full circuit description and connection data.

Card only at 14.95 inc. VAT. (13.00 + VAT)

Supplied with GEC LTU 11 direct coupler at 27.95 inc. VAT. (24.30 + VAT)

Carriage & packing 1.00

THE 69SD5 HALL EFFECT KEYBOARD

Compact, 64 key + 5 function keys, Hall effect keyboard with reprogrammable (2716) ASCII output decoder EPROM. Steel key frame for good rigidity. Negative going strobe. Requires +5 volt and -12 volt supplies. 29.95 inc. VAT (26.04 + VAT). Carriage & Packing 1.00.

STC NOVATEL PRETEL TERMINAL

**50 ONLY
AT THIS PRICE**



Features

7" diag. green screen, 240V AC mains operated, detachable key pad, robust case 14" x 12" x 7½", cassette recorder data store facility, video output for other monitors, security lock, etc.

Ideal for travel agents, offices, stock market, educational, home and all Prestel information!

Fully tested 'as new' and guaranteed

£149.95 inc. VAT
[£130.39+VAT] [UK C/P + ins £3.55]

ORDER BY POST OR PHONE OR CALL IN AND SEE FOR YOURSELF

GEC LTU 11 DIRECT TELEPHONE COUPLER

Telephone direct coupler, with isolation transformer to the rigorous 1978 (5KV) spec. (not the more recent, 2KV, spec.). Relays are provided for line select and autodial. Power supply (for line control relays) +5 volts at 200 mA. TTL logic inputs for the line control, tone input to the unit at -13dBm, output at -35dBm. Supplied in a plastic case 195mm x 108mm x 65mm complete with PO type 96 5-pole plug on lead. Suitable for use with the GEC LTU 11 Mk II and the TRANSDATA modem cards. Full connection details and circuits are supplied.

Supplied complete at 14.95 inc. VAT. (13.00 + VAT)
Carriage & packing 1.00

ITT COMPUTER CASES



Professional computer case with keyboard cutout. 18" x 15.5" x 4.5" (front slopes). Ideal for single board computers like the Nascom or Gemini Multiboard (3 cards max.) Very heavy gauge (.25") plastic with metal base. Attractive silver gray finish. 27.50 inc. VAT. (23.91 + VAT). UK Carriage & Packing 2.10

HENRY'S INCREDIBLE CP/M UTILITIES DISK

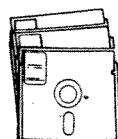
All the things you ever wanted: The disk cataloguing and file dating suites. File compare, string and byte search utilities, ASCII file compression suite, system independant disk repair utilities and all sorts of other goodies. Most are true CP/M system independant utilities and will work with any CP/M system. Supplied on most popular 5.25" formats (please state when ordering) at 17.25 inc. VAT.
Carriage & packing 50p

CCPZ

Replaces the CCP in CP/M and provides hierarchical file searches through user areas (invaluable when using more than one user area on high capacity drives). New commands for getting and executing files, and lots more. Now available rewritten as an M80 .MAC file using Z80 mnemonics as well as the earlier .ASM file, with two recently discovered bugs fixed. Supplied in most popular 5.25" formats (please state when ordering) at 11.50 inc. VAT.
Carriage & packing 50p

MDIS THE INTELLIGENT DISASSEMBLER

MDIS is a CP/M disassembler useable on most standard CP/M machines, switchable for either 8080 or Z80 mnemonics. Command syntax is designed to be similar to the Microsoft M80 assembler, and redirected input from .DAT files is allowed. The output files to printer or disk may include the disassembled code or in M80 format and output label types are differentiated by different letter prefixes. A powerful package at a price which is a lot less than its competitors. Supplied in most popular 5.25" disk formats (state type when ordering) at 57.50 inc. VAT.
Carriage & packing 50p



ORDER BY POST OR TELEPHONE OR CALL IN AND SEE FOR YOURSELF

HENRY'S

COMPUTER SHOP
404/406 Edgware Road, London, W2 1ED
Telephone: 01-402 6822

OPEN 6 DAYS A WEEK

Credit Sales available ask for details.
Official orders welcome.

