

80-BUS NEWS

NOVEMBER—DECEMBER 1982

VOL. 1 ISSUE 4

- POLYDOS BACKUP
- WHAT IS CP/M ?
- REVIEWS

Battery Backed RAM

Digital Cassette

Sound Board

Real Time Clock



The Magazine for
NASCOM & GEMINI USERS

£1.50

November—December 1982.

2.

80-BUS NEWS

Volume 1. Issue 4.

CONTENTS

Page 3	Editorial
Page 4	Review - Microcode battery backed RAM
Page 5	Review - Sound Board
Page 6	Doctor Dark's Diary - Episode 13
Page 11	Interfacing the Epson MX80
Page 17	What is CP/M?
Page 18	Classified Ads.
Page 19	Review of HS1N Digital Cassette
Page 24	Polydos Tape Backup
Page 29	Review - Gemini GM822 Real Time Clock
Page 42	Nascom Program Library
Page 44	RAM A and 2732s
Page 45	Remote Terminals and the Galaxy
Page 47	Random Rumours (& Truths)
Page 48	Lawrence Meets the Commercial Software Syndrome
Pages 49-51	Advertisements

All material copyright (c) 1982 by Interface Data 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 Interface Data Ltd and printed by Excelsior Photoprinting Ltd., High Wycombe.

SUBSCRIPTIONS

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

Subscriptions to 'Subscriptions' at the address below.

EDITORIAL

Editor : Paul Greenhalgh Associate Editor : David Hunt

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

ADVERTISING

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

ADDRESS: 80-BUS News,
Interface Data Ltd.,
Oakfield Corner, Sycamore Road,
Amersham, Bucks, HP6 5EQ.

EDITORIAL

Life seems to be a constant round of exhibitions at the moment! Having only just got PCW out of the way, a couple of weeks ago it was Compec's turn. But Compec must be THE show of the year. It goes under the title of 'Trade Only', but a quick walk around reveals that this is not really the case. It is the show, however, that every manufacturer HAS to go to, and one at which many products are often seen for the first time.

When I made my first visit to Compec (in 1978) it was mainly a show for minis and mainframes, with hardly a micro to be seen. Over the last few years the balance of the show has shifted, and this year it reached the stage of being probably about 85-90% micros - a real reflection of the way in which the market has moved. With the advent of cheaper memory, cheaper mass storage, and more intelligent peripherals, the range of 64K systems with, say, 10 Megabytes of hard disk storage for less than £3000 is growing rapidly. And below the £200 mark things are moving even more rapidly.

In 1978 the Nascom 1 was not only one of the very few micros available in the UK, but at less than £200 was an absolute bargain. Now everybody and his brother is producing, or announcing, a sub-£200 and even sub-£100 micro. And yet these are totally different. Expansion is limited, and consequently the life of the product is too. Many Nascom owners have had their systems for 4 years now, and yet they have available to them, if they wish, all of the facilities of today's £200 or £3000 systems. Yes, it has cost them more to get there, but that is the cost of buying ANY electronic goods sooner rather than later.

It is the bus that makes Nascom and Gemini products so flexible. Purchase a single board computer, watch technology change, and watch your computer become out-moded by its inherent inflexibility. Buy a multiboard (or MultiBoard!) micro, and add colour, speech, floppy-disks, hard-disks, A to D or IEEE488 capability etc. as need, technology, or even finance permits. And that, in a round about sort of way, brings me back to Compec.

Nascom and Gemini were both at Compec. Nascom were showing various Nascom 3s sporting floppy-disks, AVCs and running Nas-Net. There was also a Kenilworth Computer on show, more of which in a moment. Gemini were sharing their stand with Quantum. There were two Galaxy 2s on show, one standard and one with Winchester, and three Quantums, one standard, one with Pluto, and one with Winchester. Plus another Kenilworth Computer (see on!). But a fair proportion of the stand was dedicated to showing Gemini's MultiBoard range. Gemini are now finding that, because of the flexibility of the bus, and because of the growing range of bus compatible product from themselves, and a number of other British manufacturers, that OEM sales (i.e. sales of boards to other manufacturers to use in their own 'brand named' equipment) are growing rapidly. And that is good news for everybody, as many OEMs want guarantees that the equipment will still be available in two or three years time, and that obviously means that 80-BUS must continue to grow.

And so, what is the Kenilworth Computer? Well, it is rather difficult to describe, but in the absence of a photo I will have a go. It is available in two versions, one Nascom based and one Gemini based. It is portable, weighing about 28 lbs. It contains two slimline 400K drives and a 9" screen. Starting at the bottom, the cards are mounted vertically on the left, and the drives, side by side, on the right. Above this is the screen, and above that the PSU. Total dimensions approx. (guessed) 9" wide, 13" deep and 17" high. The Nascom based version contains N2, AVC, RAM, FDC with N2 keyboard. The cheaper Gemini based version contains the GM813 CPU-RAM, GM812 IVC and GM809 FDC, with the GM827 extended keyboard. A novel looking unit and I will be watching with great interest to see how it goes. Osborne watch out!

Well, that's my bit over once again. A Merry Christmas to you all, and may Santa stumble down your chimney with a Winchester unit!

A report on the Microcode Control battery backed RAM board.B. M. Farrelly

Whilst looking for an EPROM board which I could afford, an ad. offering a 32K battery backed RAM board which would also accomodate single rail EPROMs caught my eye. It was too good a chance to miss, especially as the manufacturer was willing to supply a bare board at a reasonable cost. I wasn't sure what the battery backed RAM would be used for, but no doubt I would think of something.

A cheque was despatched, and before long the board arrived, complete with manual. Parts were ordered from a certain mail order firm, which had better remain anonymous, and after several weeks, and not a few phone calls, TWO parcels arrived. The bad news was that two deductions had also been made to my Access account, perhaps I should have bought a made up version.

For my money I got an 80-BUS compatible double sided board. It is silk screened and solder masked, and is made to a high standard. The only reservation I would have is that the supply rail tracks are a bit light, but to be fair it does not seem to be causing any problems. There were a couple of minor errors in the silk screening, but nothing serious.

The documentation was a different story though. It seems I have heard that cry before!! One useful feature is a "How it works" section, which other producers would do well to follow. Not so good were the examples of wiring the link options. They were WRONG!!!

Construction was simple enough, anyone who has built a Nascom will have no trouble here. One nice feature was that even though DIL resistor arrays were specified, extra rows of holes had been left on the board so that ordinary resistors would fit. A small saving but worth having. I also liked the idea of having all the 80-BUS lines available - even those that are not used go to holes, making mods. easier.

The board will hold 16 memory chips, which may be 6116, 2516 or a mixture of both, depending on whether RAM or EPROM is required. Battery backup is provided by an on-board PCB battery, and each memory IC may be battery backed or not using the link options, but for obvious reasons EPROMS should not use this facility. The board may be configured as one 32K, or two 16K pages, in 4K blocks. The blocks need not be consecutive, and if two pages are selected the addresses may overlap, but if so be sure not to enable them both together. Options are set by wire links. The "off the peg" board uses special, no solder, push-in connectors for this. Since I did not have any, wire wrap pins were used instead. Working out what to link to where is not too difficult so long as no notice is taken of the examples given.

Paging is different from the normal Nascom/Gemini method in that the board may be enabled on reset regardless of which page it has been set to. If this is done however the page control bit works in reverse i.e. to turn the board off the appropriate bit must be set.

One important point to note is that, although the board responds to RAMDIS it does not generate it, a shortcoming in a board which claims to support EPROMs, but I can see the problem on a board with interchangeable RAM and EPROM, of deciding whether to originate RAMDIS or respond to it. Since my system has 64K of RAM, well it doesn't yet but it will have, a RAMDIS output was essential, so a slight mod was needed. The only other problem was that the board would not run at 4MHz without waits. The makers claim 6MHz but maybe I have a slow chip somewhere. No matter, a little thought and some more surgery, and I now have an EPROM board which will also support battery backed RAM.

In conclusion if you want a low cost EPROM board you could do worse, and if you need battery backed RAM this is the board for you.

I know what that RAM will be used for. A permanent record of which programs are on what tapes, preferably automatically updated every time a program is saved. All it needs is someone to write the software!

SOUND BOARD REVIEW

by E. Cameron

Product : WT910 Sound Board
 Manufacturer : Winchester Technology
 Supplier : Amersham Computer Centre
 Documentation : Yes, one manual

The WT910 sound board is 80-BUS/Nasbus compatible (Ed.- see note below), marketed in 'bare bones' style, 'bare bones' meaning that there are enough chips to enable the programmer to produce sounds. There is an on-board amplifier and speaker as well.

Setting up

The most difficult part of getting the board to work was unscrewing the top of Sidney's case. (Yes, my computer has a name - so does every appliance in the house!) So while the top was off, I took the time to straighten the GGIDW panel (Good Grief It Doesn't Work) anyhow.

Operational/Programming

The board was constructed as a 'write only' board and occupies addresses 0000 to 000F. You have to insert a wait state in order to produce sounds - this is explained but it caused me ten minutes of (Ab) use to the GGIDW panel. Programming can be done in either BASIC, by use of Poke and Doke, or by assembler. Producing a sound involves the following sequence:

Step	Functional Block	Operation
1	Tone generator	Program tune frequencies for each channel
2	Noise generator	Program noise frequency
3	Mixers	Enable tone and/or noise on selected channel
4	Amplitude control	Program 'fixed' or envelope control on selected channels
5	Envelope generator	Program period and select envelope shape

Five easy steps, and there are three channels to play with. This is the second product from Winchester Technology that I have seen, and if the first is anything like the sound board, then they are on the right track - good manuals and excellent hardware. Also included in the basic board are spaces for the addition of the AY-3-1350 chip, along with its support circuitry. This chip extends the capabilities of the sound board and is a pre-programmed micro that plays the first 25 notes of 25 different tunes.

One last item - you can also add the Astec 1286 video/sound modulator. This will allow you to combine the computer video and WT910 sound output into one signal path. However, this is not recommended if also using their colour board, due to harmonics in the 6MHz region.

Impressions

I have never bought a 'bad' Gemini or Nascom board - yet. This board is built to the same quality as the other boards that I have. Being a confirmed DIY type, it is a pleasure to solder components onto boards that have firmly anchored PCB runs. This board does, and space to spare. Aimed at anyone needing sound effects for programming, or desiring to be a Beethoven, this is the board for you.

(Ed.'s Note - In a Nascom/Nas-Sys environment this board lives 'overlaid' on the Nas-Sys ROM, which is OK, but it is not strictly 80-BUS compatible as a complete Gemini system is purely RAM based, and therefore the fact that this board is memory mapped could lead to all sorts of complications. I also understand (regrettably) that WT are no longer trading, although your local friendly dealer may still have some boards in stock.)

Doctor Dark's Diary – Episode 13. Pascal Episode.

Well, the new batch of home brew is just about drinkable, so I had better get on with this, before the rot sets in...

Let's kick off with the "Rory is right" section. He said that P.J.Brown's new book, "Pascal from BASIC" should be Computer Book of the year, and I agree wholeheartedly. If you have been putting off learning Pascal because you didn't like the look of the text books that were available, you now have no excuse. I have found a mistake as well, however, on page 100. The equation for the addition of two complex numbers should read:-

$$(a,b) + (c,d) = (a+c, b+d)$$

but this is a minor quibble, and anyway the Pascal procedure to do the job is correct, so perhaps we were being tested to see if we were paying attention...

Free procedure number 1.

Yet another version of my favourite way of clearing the screen, which I have now implemented in four different languages, or was it five? Anyway, here is that old chestnut, the spiral screen wipe, yet again...

```
PROCEDURE spiral(col : colour);
VAR
  lox, hix, loy, hiy, i, j : integer;
BEGIN
  {Set the size of the first box to be drawn.}
  lox := 0; hix := 95; loy := 0; hiy := 44;
  REPEAT
    {Draw a box one pixel wide round the screen.}
    BEGIN
      {Line along the top of the screen.}
      FOR i := lox TO hix DO
        BEGIN
          GRAPH(col,i,loy);
          {This line gives a short delay each time it appears.
          There is no reason why it should not itself be a procedure.}
          FOR j := 1 TO 10 DO BEGIN END
        END;
      {Line down the right hand side of the screen.}
      FOR i := loy+1 TO hiy DO
        BEGIN
          GRAPH(col,hix,i);
          FOR j := 1 TO 10 DO BEGIN END
        END;
      {Line backwards across the bottom of the screen.}
      FOR i := hix-1 DOWNTO lox DO
        BEGIN
          GRAPH(col,i,hiy);
          FOR j := 1 TO 10 DO BEGIN END
        END;
      {Upwards line on the left of the screen.}
      FOR i := hiy-1 DOWNTO loy+1 DO
        BEGIN
          GRAPH(col,lox,i);
          FOR j:= 1 TO 10 DO BEGIN END
        END;
    
```

```

{Reduce the size of the box being drawn.}
lox := lox+1;  loy := loy+1;
hix := hix-1;  hiy := hiy-1
END
UNTIL loy = hiy;
{Which means we have reached the middle, almost!
The next bit fills in the remaining blank line.}
FOR i := lox TO hix DO
BEGIN
  GRAPH(col,i,loy);
  FOR j := 1 TO 10 DO BEGIN END
END
END;
PROCEDURE spiralwipe;
BEGIN
  spiral(on);
  spiral(off)
END;

```

The procedure is called by including the word "spiralwipe" in your program, this spirals round twice. The first time round, all the pixels making up the spiral are switched on, and on the second spiral, they are switched off. Alternatively, you can use the call
 spiral(invert);
 which will spiral round the screen just once, inverting all the graphics characters as it goes. Once again, impress your friends...

Medium sized review of a large book.

I don't think Rory has done this one yet, although it's a mystery how he gets through so many! This is about "Microcomputer Architecture and Programming" by John F. Wakerly, which is published by Wiley. It is the last book I bought from the Computer Book Club, before I told them I had had enough of their methods. (See last vitriolic episode for the boring details...)

Anyway, as I said, this is a large book, with 692 pages. The author describes the book as being suitable for an introductory course on (micro) computer organisation and assembly language programming. Since it is so large, I have read only a few chapters so far. I do like the style, which is somewhat like a transatlantic version of P.J.Brown. Reviews are allowed to contain reasonably sized chunks from the book in question, so here is an example, from chapter 3, which is headed "Data Structures in Pascal Programs."

"Niklaus Wirth, the designer of Pascal, has written a programming textbook called "Algorithms + Data Structures = Programs". From the previous chapter, you should already have a good idea of what programs and algorithms are. To find out what data structures are, just subtract, and you don't have to read this chapter..."

The two chapters before that give a review of "fundamental concepts" and Pascal respectively. They are good, although the experienced reader will be able to skip through them pretty fast. The greater part of the book, as yet unread in detail, I have to describe by glancing at it, or reading the description at the start of the book. There are four chapters in the first, "introductory and remedial", part of the book. Part 2 describes basic principles that apply to all computers, using as examples two hypothetical processors, which just happen to have instruction sets and features that are subsets of those of the Motorola 6809 and the Zilog Z8000. I wanted to learn more about the Z8000, so I have looked ahead to the section concerned, and it seems very comprehensive, not to mention comprehensible.

Part 3 describes in detail the workings of several processors, including the 68000, the Texas 9900, the PDP-11(!), Z8000, and the Intel 8086. The last is of particular interest, as it is very similar to the 8088 used in the Pluto graphic board. (Any reader who doesn't wish he or she had a Pluto has already sold his/her soul and bought one!) This is not a cheap book, and although I like it so far, I can see that it would not suit every user's needs. If you don't want to know about Pascal, which is used throughout for examples, and think that 16 bit processors are never going to be cheap enough for you, or are happy running other people's programs, the book is definitely not for you. (And why are you reading this?) If what I have said made it sound interesting, then "Look before you buy", is my advice. I think you'll probably buy...

Tuesday's fabulous excitement.

This consisted, once again, of taking Marvin to the Taunton Computer Club, much to the astonishment of the rest of the serious members. (I am beginning to see that that club is much like this one, in that it has a great proportion of members who prefer to do nothing at all. As an example of this, at our annual general meeting, I promised to arrange a coach to the Personal Computer World Show, if enough members wanted to go. Just about everyone who was there put their hand up as a person who would go to the show. But when it came to sending in the application form on our club newsletter, it turned out that the entire membership had lost their pens! This saved me the bother of arranging a coach trip, and by the time you read this, I and the rest of the "Serious Members" will have probably been and seen the show anyway.) In the meantime, I had a great time, showing the younger members of the club that I was still one step ahead of even the most brilliant BASIC programmers amongst them. If you rush through a Pascal program or two, even the most remarkable youthful genius types can be impressed, so long as they only know BASIC. The trouble is, they will probably all go out and buy "Pascal from BASIC", and then I will have to learn something else in a hurry. In emergencies, I use little snippets of number theory from Hofstadter's book to subdue these youthful upstarts. It is for their own good, of course. If they realised how clever they actually are, they would become so unbearable that they would have to be beaten up...

Mispnirt of the Year (so far).

I nominate Personal Computer World's September issue, page 274. The I.B.S. Ltd advertisement offers readers the opportunity to purchase "6/12 Slut Mother Boards". Entries on a pound note, please, and note that any entry involving "Band rates" will be disqualified instantly, as will anyone who mentions any of my errors.

How to Win £5000.

See the same issue of Personal Computer World for details of a competition with a prize of the size mentioned above. The first part of the competition requires you to find a mystery number, subject to the following conditions. The number required is the lowest palindromic number (in other words, it reads the same forwards and backwards) which, if you square it and subtract a million, gives a result which contains at least one of each of the digits 0 to 9. Well, that is at least a ten digit result, you are saying, and that means it is not easy to write a program to solve it. The simple fact is that few machines can handle numbers of that sort of size easily. Scores of you may write and tell me if I'm wrong, but I don't think Nascom ROM BASIC can operate on numbers of that order, unless one or more of the recently available extension BASICs can do the job.

So, how have I solved part one of the puzzle? (I have, honestly!) I wrote a program in Hisoft Pascal 4, which generated each palindromic number in ascending order, and then tested them to see if they met the conditions. The program used the following declarations to set up a data structure capable of holding the big numbers.

```

TYPE
  DIGIT = 0..9;
  NUMBER = ARRAY [1..6] OF DIGIT;
  BIGNUMBER = ARRAY [1..12] OF DIGIT;
VAR
  STARTAT, VALUE : NUMBER;
  RESULT : BIGNUMBER;

```

The rest of the program consists mainly of routines which manipulate the arrays of digits, which could be of any length, if you wanted enormous accuracy. One procedure puts the square of VALUE into RESULT, another knocks a million off, and so on. Of course, there is no room to print the whole program here. When run, it took seven minutes to come up with the answer, which I am not going to tell you, as that would not help you to improve your programming skills. A more recent revision of the same program, which generates its palindromic numbers a faster way, gives the six suitable numbers less than a million in just under ten minutes, and could be made to go even faster, if I could be bothered to do it, simply by changing the routine that squares the number to take advantage of the symmetry of the number. The most important thing, I think, to come out of this, unless I get the £5000 at the end of it all, is the realisation that the people who used to annoy me by going on about "Problem Solving" with Pascal were right. All except one...

Great Stupid Remarks of the Twentieth Century.

In a recent edition (16/8/82) of Datalink, Margaret Park writes:- "Edsger Djikstra the programming guru credited with the invention of structured programming, damned the language completely by saying that programmers who had started on Basic (sic) were unteachable. "As potential programmers," he said, "they are mentally mutilated beyond hope of regeneration."

It is important to realise that these people who are "credited" with inventing this, that, or indeed the other, have hardly ever done so. If you think Arthur C. Clarke really invented the communication satellite, you need your head looked at. And if Marshall McLuhan was right, I can send in a blank tape, instead of going to all the bother of writing this article at all, on the grounds that "the medium is the message". All these "gurus" can do is make a nice after dinner speech. They each select a theme, and choose a set of suitably far-out, trendy things to say, and make a fortune for themselves on what Douglas Adams has rightly identified as the "Chat Show Circuit". Structured programming is a method of programming that is independant of the programming language in use. You can write structured programs in zarking machine code. You can write them in BASIC, and it may even be possible to write them in Pilot. You may have had a Nascom 1 when there was no BASIC available for it, and have written structured programs for that, too. And along comes this guy Djockstrap, and tells you that your brain is too curdled for you to learn any new language, ever, no matter how you try. Well, I wonder what language Djikstra actually learned to program in, 'cos he looks very old in the photograph in Datalink. It was almost certainly Fortran. Nascom guru Doctor Dark, inventor of the Hyperspace Beer-Drive writes, "Wow, he was lucky he didn't learn Cobol first. That makes your w***y drop off..."

What is wrong with Pascal? (Shock Horror!)

Honestly, there really is one thing missing from Pascal, and it just happens that this is the one thing BASIC is better at than almost anything else (that takes care of the ones I don't know! PIC X(12), indeed!) String handling is easy in Pascal, as long as all the strings are the same length! You can not use things like

A\$ = LEFT\$(B\$,4) + MID\$(A\$,3,3) + " Wowee!"
 even if you want to. But of course, you can define a set of procedures and functions that would make this sort of thing possible. I have started work on just such a set of procedures, and have made a little progress, some of it in the right direction...

It is necessary to learn how to use the "dynamic variables" of Pascal, as the other way of approaching the problem, using a huge array of characters, while probably faster, takes up memory all the time. That defeats one of the objects, so the dynamic approach it is. The things BASIC can do to its string variables include setting them to "empty", concatenation, and the ever useful LEFT\$, MID\$ and RIGHT\$. Also useful are the ability to type into a string from the keyboard, and to print strings out. Some sort of clue as to how difficult such things are to program in Pascal can be gained from P. J. Brown's remarks on the matter, when his exponent of structured programming, Professor Primple remarks, "My program to analyse English sentences works well, but there is a small restriction that all sentences must contain ten words and each word must be of five letters."

The stuff I have written so far starts out with these declarations:-

```
CONST
  longeststring = 256; {Can be more, or less.}
  longestname = 8; {Length of stringnames.}

TYPE
  stringname = PACKED ARRAY [1..longestname] OF CHAR;
  stringlength = 0..longeststring;
  stringcontents = PACKED ARRAY [1..longeststring] OF CHAR;
  stringitem =
    RECORD
      name      : stringname;
      length    : stringlength;
      contents  : stringcontents;
      next      : ^stringitem
    END;
  stringpointer = ^stringitem
```

In most versions of Pascal, "stringpointer" would have been declared before "stringitem", and "next" would be of type stringpointer. The Hisoft version won't let you use the slight forward declaration involved, but this is not a handicap, as it is easily avoided, as shown. Anyway, the declarations above show the sort of data structure I am trying to use in my string handling routines. Whenever a new string is needed, the program includes a NEW(soandso), where the "soandso" is the name of a pointer. Space is then allocated by the run time routines for another record of the type shown above. It may prove to be unnecessary to write the system after all, since an Apple owner I know has suggested that the UCSD string handling routines might work. I'll let you know if they do... And now a rest, while I try to beat my score of 232 at Adventure. How DO you get past the Plover Room, without falling down a pit?

Interfacing the EPSON MX80.

by Big H

The following article(s) if this prose is worthy of print, will attempt to describe the ins and outs (ups and downs) of getting that very expensive printer PRINTING. Although specific references apply to the MX80 type III, much is applicable to the whole MX-range. Since the interface is a standard 'Centronics type', the general principle applies to any number of printers. I refuse to class that race of tanks dressed up as printers (you know... 5 bit code etc.) as printers. By the way, does any one want a slightly footmarked Creed 444? Advertising over - Included will be notes on how to patch into NAS-PEN, NAS-DIS and ZEAP (if it isn't already obvious) and a listing of a fairly comprehensive relocatable routine to set up and drive the printer through the on board PIO. Later on for those with the MX80 types II or III (and a larger overdraft than everyone else), use of the HI-RES graphic facilities is described. These can be used to define your own characters and with suitable software.... produce a dot for dot dump of the Nascom screen. Do I hear cries of 'Cor - What a cop out'? Well you can take it all back because a listing of such a program will be given. With luck, it will not only provide a simple screen dump but also true/inverted and half/normal/double width images of any defined screen window. (This guy is just unbelievable! - Ed.)

Epson Interface.

But first the basic printer interface and its driving software. The rightmost columns of Table 1 show the pin connections of the Epson interface. All signals are (naturally) TTL compatible. Synchronisation is achieved by the STROBE signal and handshaking provided by either ACKNLG or BUSY. I have chosen to use the BUSY line, but for no particular reason. Data transfer is achieved by waiting for the printer to be 'NOT BUSY', applying the data, waiting for it to stabilise and then asserting STROBE for a minimum of 0.5 microsec. Apart from ACKNLG, the only other signals (on the printer I/F) not used are SLCT IN and SLCT. The function of these (to select the printer/tell you it is selected respectively) was disabled using the Epson's internal DIL switches. This is OK as I don't see me operating a bank of MX's from my Nascom - not this year at least!

DIL Switches.

The internal DIL switches definitely deserve a mention as they allow you to set up many of the programmable functions immediately on power up. Things like character size, form length, line spacing and even whether or not your zero has a slash across it can all be set.

The Electric String.

The assignment of the PIO ports is shown by diagram 1 and the actual connections made by the cable is shown by the middle columns of table 1. The Epson manual tells you to use twisted pair cable and the return earth signals. Twisted cable is not easy to get hold of and shouldn't be necessary so long as the

cable doesn't trail up stairs to the back room. Admittedly, my cable is only a couple of feet long but it was all I could find when the printer arrived.

EPRINT Listing.

The commented listing of EPRINT should be relatively simple to follow but I shall explain some areas such as setting up the PIO and printer. EPRINT by the way stands for Epson Printout Ready In No Time or alternatively Extra Playing Round In Nascom Trauma.

NAS-PEN, ZEAP etc.

When executed, the first questions asked involve what software you want to run the printer with. Dependent on the replies, the vectors held in RAM for the ZEAP and NAS-PEN output routines may be set to OUTCHA, the character output routine. Unfortunately NAS-DIS has no RAM vector and so can only use the printer (as with BASIC) by setting OUTCHA as the user output routine. Needless to say, NAS-PEN or ZEAP must be initialised before executing EPRINT or they will just overwrite the vectors when 'cold started'. If ZEAP is selected, then the printer AUTO FEED EXT line will have to be disabled. This is because while the ZEAP sends a CR and LF, the printer will automatically do a LF on reception of the CR. Hence two LF's and double line spacing, very nice but paper is expensive. NAS-PEN, and software using the user output routine will only send a CR and so AUTO FEED EXT is left low unless ZEAP is selected. A value stored in D is used to set this line later on.

Setting Printer Status.

Several questions are asked relating to the way you require the printer to be configured. Many more variations could be catered for (column width, print size etc.) but it becomes tedious setting too many whenever you switch on or start using some different software. You will usually configure the printer the same way and anyway codes can be embedded in the text to set anything else.

As mentioned above AUTO FEED EXT is controlled by the level of that line, all the other features are set by sending control codes to the printer. Before any codes are sent to the printer, it is reset by holding INIT low for longer than 60 microseconds. There is no need to compensate for CPU clock speeds (2/4 MHz. etc.) as all timing is done assuming a 4MHz. clock and all delays may be exceeded without effect.

The length of page is set by ESC C nn (nn is a hex. value 0<nn<127.).

The paper end detector is set on by ESC 9 or off by ESC 8. This stops the buzzer going off while using single sheets which is annoying because it is LOUD and buzzes for ages.

The printer is told to skip over the perforation at the end of each page by ESC E 6. The 6 tells it to skip 6 lines which with normal spacing is one inch. Note that the DIL switches have been left to cause the printer to initialise in the NO SKIP condition.

Having set all these conditions, the printer status is tested and suitable messages output to the screen.

SETUP has been written as a subroutine so that it may be used by other programs such as the screen dump to be described later. There is one other section of code that probably warrants explanation, setting up the PIO which is described below.

Configuring the PIO.

In keeping with explaining setting up the printer before explaining setting the PIO, I have configured PORT B first and will explain it last. Confused?

PORT A:

This is set up in bit (control) mode with all the bits as outputs. In order, the bytes sent to PACONT to do this are: fFF;0;7;3. fFF selects mode 3 (control mode) and the following 0 makes all the bits outputs. The 7 is the interrupt control word which is set to prevent any interrupts. The final 3 disables all interrupts on PORT A just to be sure.

PORT B:

This is identical to the configuring of PORT A except for the second byte. This is fF8 (1111 1000) because we require bits 0,1 and 2 to be outputs and bits 3,4 and 5 to be inputs. We don't care about 6 and 7 and so could have sent f38 etc.

The next installment.

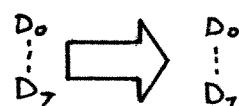
The screen dump program will have to wait until next time, while I think of some suitable abbreviated names for it. I hope all this is of interest to someone and maybe even useful. In this extremely unlikely event I would be glad to supply a cassette of the ZEAP source to those too busy (lazy) to type it all in. A cheque for £1-50 (to cover costs) to COMPWARE at 57 Repton Dr. Haslington Crewe Cheshire CW1 1SA will secure this.

DIAGRAM 1 : SCHEMATIC SUMMARY OF CONNECTIONS.

NASCOM PIO

EPSON I/F

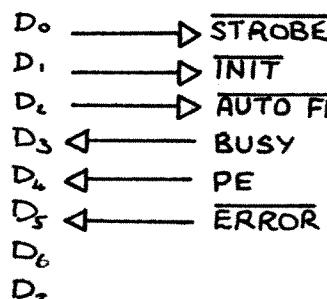
PORT A



PORT B

NOT USED:

+5v, ASTB, ARDY,
BSTB, BRDY



<u>NOT USED:</u>
10 <u>ACKNLG</u>
13 <u>SLCT</u>
36 <u>SLCT IN</u>
35 +5v PULL UP

Ov (16,18)

— Ov (19-30,16,33)

Table 1: Nascom 2 PIO / Epson MX80 connections..	
PIO	CABLE
	EPSON
1 B5	B5 1 32 ERROR
2 B4	B4 2 12 PE
3 B6	- 3 -
4 B3	B3 4 11 BUSY
5 B7	- 5 -
6 B2	B2 6 14 AUTO FEED EXT
7 ARDY	- 7 -
8 B1	B1 8 31 TRIT
9 ESTB	- 9 -
10 BO	BO 10 1 STROBE
11 ASTB	- 11 -
12 BRDY	- 12 -
13 AO	AO 13 2 D _o
14 NC	- 14 -
15 AI	A1 15 3 D ₁
16 OV	OV 16 A11 OV pins
17 A2	A2 17 4 D ₂
18 OV	OV 18 A11 OV pins
19 A3	A3 19 5 D _s
20 +5v	- 20 -
21 A4	A4 21 6 D ₄
22 +5v	- 22 -
23 A5	A5 23 7 D ₅
24 A7	A7 24 9 D ₇
25 A6	A6 25 8 D ₈
26 NC	- 26 -
	(Twisted pair) earth returns)
30 INIT	30 -
31 ERROR	31 -
32 OV	32 -
33 NC	33 -
34 NC	34 -
35 +5v Pull up	35 -
36 SLCT IN	36 -
	EPSON
1 B5	1 STROBE
2 D _o	2 D ₁
3 D ₁	3 D ₂
4 D ₂	4 D ₃
5 D ₃	5 D ₄
6 D ₄	7 D ₅
7 D ₅	8 D ₆
8 D ₆	9 D ₇
9 D ₇	10 ACKNG
11 BUSY	11 PE
12 PE	12 D _o
13 SLCT	13 SELCT
14 ATTF FEED EXT	14 ATTF FEED EXT
15 AI	15 NC
16 OV	16 OV
17 CHASSIS-GND	17 CHASSIS-GND
18 NC	18 NC
19 \	19 \
20 to	20 to
21 A4	21 A4
22 +5v	22 +5v
23 A5	23 A5
24 A7	24 A7
25 A6	25 A6
26 NC	26 NC
	(ZEARAD), HL
30 RST PRS	30 RST PRS
31 DEFN /Nass-pen? /	31 DEFN /Nass-pen? /
32 YESNO	32 YESNO
33 JR C, APENUQ	33 JR C, APENUQ
34 POP HL	34 POP HL
35 LD D, EF	35 LD D, EF
36 PUSH DE	36 PUSH DE
	RST PRS
37 DEFN /Nass-pen? /	37 DEFN /Nass-pen? /
38 LD A, EC3	38 LD A, EC3
39 LD (PENADD), HL	39 LD (PENADD), HL
40 LD (PENAD-1), A	40 LD (PENAD-1), A
41 PUSH HL	41 PUSH HL
42 RST PRS	42 RST PRS
43 DEFN /Skp over defn?	43 DEFN /Skp over defn?

Table 1: Nascom 2 PIO / Epson MXBO connections.


```

B11C 3E07 1640 LD A,7 OUT (PACONT),A 2150 ; 60 mic. sec. DELAY
B11E D306 1650 LD A,3 OUT (PACONT),A 2160 ;
B120 3E03 1660 LD A,C OUT (PACONT),A 2170 ;
B122 D306 1670 LD A,C OUT (PACONT),A 2180 DEL60 LD B,17
B124 3E1B 1680 LD A,ESC OUT (PACONT),A 2190 LOOP DJNZ LOOP
B126 D777 1700 LD A,ESC OUT (PACONT),A 2200 RET
B128 3E43 1710 RCAL OUTCHA 2210 ; PRINTER DRIVER ROUTINE
B12A D773 1720 LD A,C 2220 ;
B12C C1 1730 RCAL OUTCHA 2230 ;
B12D 79 1740 POP BC 2240 OUTCHA PUSH AF
B12E D76F 1750 LD A,C 2250 BUSY IN A, (PBDATA)
B130 3E1B 1770 LD A,ESC 2260 BIT 3,A
B132 D768 1780 RCAL OUTCHA 2270 JR NZ, BUSY
B134 7B 1790 LD A,B 2280 POP AF
B135 D768 1800 RCAL OUTCHA 2290 OUT (PADATA), A
B137 7B 1820 LD A,E 2300 NOP
B138 FE4E 1830 LD A,E 2310 IN A, (PDATA)
B13A 2B0C 1840 CP "N" 2320 DEC A
B13C 3E1B 1850 JR Z, NOSKIP 2330 OUT (PADATA), A
B13E D75F 1860 LD A,ESC 2340 INC A
B140 3E4E 1870 RCAL OUTCHA 2350 OUT (PADATA), A
B142 D75B 1880 LD A,"N" 2360 IN A, (PDATA)
B144 3E06 1890 RCAL OUTCHA 2370 RET
B146 D757 1900 LD A,6 2380 ;
B148 DB05 1920 RCAL OUTCHA 2390 ;
B14A CB67 1930 NOSKIP IN A, (PBDATA) 2400 ;
B14C 2B18 1940 BIT 4,A 2410 ; Labels etc. :
B14E EF 1950 JR Z, PAPOK 2420 NUM EQU £C21 ; Store for val from NUM
B14F 5072696E 1960 RST PRS 2430 USROUT EQU £C78 ; vectored jump to USROUT
B16A EF 1970 DEFN /Printer out of paper./ 2440 PRS EQU £2B ; NAS-SYS print string, RST
B16B 201E 2010 RST PRS 2450 ROUT EQU £30 ; " O/P A to screen, "
B16C 5072696E 2020 DEFN /Printer in error status./ 2460 BLINK EQU £7B ; " blink for I/P, SCAL
B170 6F57420 2050 RST PRS 2470 INLIN EQU £63 ; " input a line,
B171 6F62070 2060 RST PRS 2480 NUM EQU £64 ; " get line value,
B172 6F62070 2070 RST PRS 2490 MRET EQU £5B ; " monitor return,
B173 6F62070 2080 RST PRS 2500
B174 6F62070 2090 RST PRS 2510 ;
B175 6F62070 2100 RST PRS 2520 NUM EQU £C21 ; Store for val from NUM
B176 6F62070 2110 RST PRS 2530 USROUT EQU £C78 ; vectored jump to USROUT
B177 6F62070 2120 RST PRS 2540 ZEPAD EQU £F05 ; ZEAP vector to printer
B178 6F62070 2130 RST PRS 2550 PENADD EQU £101E ; NAS-PEN vector to printer
B179 6F62070 2140 RST PRS 2560
B180 6F62070 2150 RST PRS 2570 ; PIO PORT ADDRESSES:
B181 6F62070 2160 RST PRS 2580
B182 6F62070 2170 RST PRS 2590 ;
B183 6F62070 2180 RST PRS 2600 PACONT EQU 6 ; Port A control
B184 6F62070 2190 RST PRS 2610 PADATA EQU 4 ; " A data
B185 6F62070 2200 RST PRS 2620 PBCONT EQU 7 ; " B control
B186 6F62070 2210 RST PRS 2630 PBDATA EQU 5 ; " B data
B187 6F62070 2220 RST PRS 2640
B188 6F62070 2230 RST PRS 2650 ; ASCII CODES:
B189 6F62070 2240 RST PRS 2660
B190 6F62070 2250 RST PRS 2670 ;
B191 6F62070 2260 RST PRS 2680 CR EQU ED ; Carriage return
B192 6F62070 2270 RST PRS 2690 ESC EQU £1B ; Escape
B193 6F62070 2280 RST PRS 26A0
B194 6F62070 2290 RST PRS 26B0
B195 6F62070 2300 RST PRS 26C0
B196 6F62070 2310 RST PRS 26D0
B197 6F62070 2320 RST PRS 26E0
B198 6F62070 2330 RST PRS 26F0
B199 6F62070 2340 RST PRS 2700
B200 6F62070 2350 RST PRS 2710
B201 6F62070 2360 RST PRS 2720
B202 6F62070 2370 RST PRS 2730
B203 6F62070 2380 RST PRS 2740
B204 6F62070 2390 RST PRS 2750
B205 6F62070 2400 RST PRS 2760
B206 6F62070 2410 RST PRS 2770
B207 6F62070 2420 RST PRS 2780
B208 6F62070 2430 RST PRS 2790
B209 6F62070 2440 RST PRS 2800
B210 6F62070 2450 RST PRS 2810
B211 6F62070 2460 RST PRS 2820
B212 6F62070 2470 RST PRS 2830
B213 6F62070 2480 RST PRS 2840
B214 6F62070 2490 RST PRS 2850
B215 6F62070 2500 RST PRS 2860
B216 6F62070 2510 RST PRS 2870
B217 6F62070 2520 RST PRS 2880
B218 6F62070 2530 RST PRS 2890
B219 6F62070 2540 RST PRS 2900
B220 6F62070 2550 RST PRS 2910
B221 6F62070 2560 RST PRS 2920
B222 6F62070 2570 RST PRS 2930
B223 6F62070 2580 RST PRS 2940
B224 6F62070 2590 RST PRS 2950
B225 6F62070 2600 RST PRS 2960
B226 6F62070 2610 RST PRS 2970
B227 6F62070 2620 RST PRS 2980
B228 6F62070 2630 RST PRS 2990
B229 6F62070 2640 RST PRS 3000
B230 6F62070 2650 RST PRS 3010
B231 6F62070 2660 RST PRS 3020
B232 6F62070 2670 RST PRS 3030
B233 6F62070 2680 RST PRS 3040
B234 6F62070 2690 RST PRS 3050
B235 6F62070 2700 RST PRS 3060
B236 6F62070 2710 RST PRS 3070
B237 6F62070 2720 RST PRS 3080
B238 6F62070 2730 RST PRS 3090
B239 6F62070 2740 RST PRS 3100
B240 6F62070 2750 RST PRS 3110
B241 6F62070 2760 RST PRS 3120
B242 6F62070 2770 RST PRS 3130
B243 6F62070 2780 RST PRS 3140
B244 6F62070 2790 RST PRS 3150
B245 6F62070 2800 RST PRS 3160
B246 6F62070 2810 RST PRS 3170
B247 6F62070 2820 RST PRS 3180
B248 6F62070 2830 RST PRS 3190
B249 6F62070 2840 RST PRS 3200
B250 6F62070 2850 RST PRS 3210
B251 6F62070 2860 RST PRS 3220
B252 6F62070 2870 RST PRS 3230
B253 6F62070 2880 RST PRS 3240
B254 6F62070 2890 RST PRS 3250
B255 6F62070 2900 RST PRS 3260
B256 6F62070 2910 RST PRS 3270
B257 6F62070 2920 RST PRS 3280
B258 6F62070 2930 RST PRS 3290
B259 6F62070 2940 RST PRS 3300
B260 6F62070 2950 RST PRS 3310
B261 6F62070 2960 RST PRS 3320
B262 6F62070 2970 RST PRS 3330
B263 6F62070 2980 RST PRS 3340
B264 6F62070 2990 RST PRS 3350
B265 6F62070 3000 RST PRS 3360
B266 6F62070 3010 RST PRS 3370
B267 6F62070 3020 RST PRS 3380
B268 6F62070 3030 RST PRS 3390
B269 6F62070 3040 RST PRS 3400
B270 6F62070 3050 RST PRS 3410
B271 6F62070 3060 RST PRS 3420
B272 6F62070 3070 RST PRS 3430
B273 6F62070 3080 RST PRS 3440
B274 6F62070 3090 RST PRS 3450
B275 6F62070 3100 RST PRS 3460
B276 6F62070 3110 RST PRS 3470
B277 6F62070 3120 RST PRS 3480
B278 6F62070 3130 RST PRS 3490
B279 6F62070 3140 RST PRS 3500
B280 6F62070 3150 RST PRS 3510
B281 6F62070 3160 RST PRS 3520
B282 6F62070 3170 RST PRS 3530
B283 6F62070 3180 RST PRS 3540
B284 6F62070 3190 RST PRS 3550
B285 6F62070 3200 RST PRS 3560
B286 6F62070 3210 RST PRS 3570
B287 6F62070 3220 RST PRS 3580
B288 6F62070 3230 RST PRS 3590
B289 6F62070 3240 RST PRS 3600
B290 6F62070 3250 RST PRS 3610
B291 6F62070 3260 RST PRS 3620
B292 6F62070 3270 RST PRS 3630
B293 6F62070 3280 RST PRS 3640
B294 6F62070 3290 RST PRS 3650
B295 6F62070 3300 RST PRS 3660
B296 6F62070 3310 RST PRS 3670
B297 6F62070 3320 RST PRS 3680
B298 6F62070 3330 RST PRS 3690
B299 6F62070 3340 RST PRS 3700
B300 6F62070 3350 RST PRS 3710
B301 6F62070 3360 RST PRS 3720
B302 6F62070 3370 RST PRS 3730
B303 6F62070 3380 RST PRS 3740
B304 6F62070 3390 RST PRS 3750
B305 6F62070 3400 RST PRS 3760
B306 6F62070 3410 RST PRS 3770
B307 6F62070 3420 RST PRS 3780
B308 6F62070 3430 RST PRS 3790
B309 6F62070 3440 RST PRS 3800
B310 6F62070 3450 RST PRS 3810
B311 6F62070 3460 RST PRS 3820
B312 6F62070 3470 RST PRS 3830
B313 6F62070 3480 RST PRS 3840
B314 6F62070 3490 RST PRS 3850
B315 6F62070 3500 RST PRS 3860
B316 6F62070 3510 RST PRS 3870
B317 6F62070 3520 RST PRS 3880
B318 6F62070 3530 RST PRS 3890
B319 6F62070 3540 RST PRS 3900
B320 6F62070 3550 RST PRS 3910
B321 6F62070 3560 RST PRS 3920
B322 6F62070 3570 RST PRS 3930
B323 6F62070 3580 RST PRS 3940
B324 6F62070 3590 RST PRS 3950
B325 6F62070 3600 RST PRS 3960
B326 6F62070 3610 RST PRS 3970
B327 6F62070 3620 RST PRS 3980
B328 6F62070 3630 RST PRS 3990
B329 6F62070 3640 RST PRS 4000
B330 6F62070 3650 RST PRS 4010
B331 6F62070 3660 RST PRS 4020
B332 6F62070 3670 RST PRS 4030
B333 6F62070 3680 RST PRS 4040
B334 6F62070 3690 RST PRS 4050
B335 6F62070 3700 RST PRS 4060
B336 6F62070 3710 RST PRS 4070
B337 6F62070 3720 RST PRS 4080
B338 6F62070 3730 RST PRS 4090
B339 6F62070 3740 RST PRS 4100
B340 6F62070 3750 RST PRS 4110
B341 6F62070 3760 RST PRS 4120
B342 6F62070 3770 RST PRS 4130
B343 6F62070 3780 RST PRS 4140
B344 6F62070 3790 RST PRS 4150
B345 6F62070 3800 RST PRS 4160
B346 6F62070 3810 RST PRS 4170
B347 6F62070 3820 RST PRS 4180
B348 6F62070 3830 RST PRS 4190
B349 6F62070 3840 RST PRS 4200
B350 6F62070 3850 RST PRS 4210
B351 6F62070 3860 RST PRS 4220
B352 6F62070 3870 RST PRS 4230
B353 6F62070 3880 RST PRS 4240
B354 6F62070 3890 RST PRS 4250
B355 6F62070 3900 RST PRS 4260
B356 6F62070 3910 RST PRS 4270
B357 6F62070 3920 RST PRS 4280
B358 6F62070 3930 RST PRS 4290
B359 6F62070 3940 RST PRS 4300
B360 6F62070 3950 RST PRS 4310
B361 6F62070 3960 RST PRS 4320
B362 6F62070 3970 RST PRS 4330
B363 6F62070 3980 RST PRS 4340
B364 6F62070 3990 RST PRS 4350
B365 6F62070 4000 RST PRS 4360
B366 6F62070 4010 RST PRS 4370
B367 6F62070 4020 RST PRS 4380
B368 6F62070 4030 RST PRS 4390
B369 6F62070 4040 RST PRS 4400
B370 6F62070 4050 RST PRS 4410
B371 6F62070 4060 RST PRS 4420
B372 6F62070 4070 RST PRS 4430
B373 6F62070 4080 RST PRS 4440
B374 6F62070 4090 RST PRS 4450
B375 6F62070 4100 RST PRS 4460
B376 6F62070 4110 RST PRS 4470
B377 6F62070 4120 RST PRS 4480
B378 6F62070 4130 RST PRS 4490
B379 6F62070 4140 RST PRS 4500
B380 6F62070 4150 RST PRS 4510
B381 6F62070 4160 RST PRS 4520
B382 6F62070 4170 RST PRS 4530
B383 6F62070 4180 RST PRS 4540
B384 6F62070 4190 RST PRS 4550
B385 6F62070 4200 RST PRS 4560
B386 6F62070 4210 RST PRS 4570
B387 6F62070 4220 RST PRS 4580
B388 6F62070 4230 RST PRS 4590
B389 6F62070 4240 RST PRS 4600
B390 6F62070 4250 RST PRS 4610
B391 6F62070 4260 RST PRS 4620
B392 6F62070 4270 RST PRS 4630
B393 6F62070 4280 RST PRS 4640
B394 6F62070 4290 RST PRS 4650
B395 6F62070 4300 RST PRS 4660
B396 6F62070 4310 RST PRS 4670
B397 6F62070 4320 RST PRS 4680
B398 6F62070 4330 RST PRS 4690
B399 6F62070 4340 RST PRS 4700
B400 6F62070 4350 RST PRS 4710
B401 6F62070 4360 RST PRS 4720
B402 6F62070 4370 RST PRS 4730
B403 6F62070 4380 RST PRS 4740
B404 6F62070 4390 RST PRS 4750
B405 6F62070 4400 RST PRS 4760
B406 6F62070 4410 RST PRS 4770
B407 6F62070 4420 RST PRS 4780
B408 6F62070 4430 RST PRS 4790
B409 6F62070 4440 RST PRS 4800
B410 6F62070 4450 RST PRS 4810
B411 6F62070 4460 RST PRS 4820
B412 6F62070 4470 RST PRS 4830
B413 6F62070 4480 RST PRS 4840
B414 6F62070 4490 RST PRS 4850
B415 6F62070 4500 RST PRS 4860
B416 6F62070 4510 RST PRS 4870
B417 6F62070 4520 RST PRS 4880
B418 6F62070 4530 RST PRS 4890
B419 6F62070 4540 RST PRS 4900
B420 6F62070 4550 RST PRS 4910
B421 6F62070 4560 RST PRS 4920
B422 6F62070 4570 RST PRS 4930
B423 6F62070 4580 RST PRS 4940
B424 6F62070 4590 RST PRS 4950
B425 6F62070 4600 RST PRS 4960
B426 6F62070 4610 RST PRS 4970
B427 6F62070 4620 RST PRS 4980
B428 6F62070 4630 RST PRS 4990
B429 6F62070 4640 RST PRS 5000
B430 6F62070 4650 RST PRS 5010
B431 6F62070 4660 RST PRS 5020
B432 6F62070 4670 RST PRS 5030
B433 6F62070 4680 RST PRS 5040
B434 6F62070 4690 RST PRS 5050
B435 6F62070 4700 RST PRS 5060
B436 6F62070 4710 RST PRS 5070
B437 6F62070 4720 RST PRS 5080
B438 6F62070 4730 RST PRS 5090
B439 6F62070 4740 RST PRS 5100
B440 6F62070 4750 RST PRS 5110
B441 6F62070 4760 RST PRS 5120
B442 6F62070 4770 RST PRS 5130
B443 6F62070 4780 RST PRS 5140
B444 6F62070 4790 RST PRS 5150
B445 6F62070 4800 RST PRS 5160
B446 6F62070 4810 RST PRS 5170
B447 6F62070 4820 RST PRS 5180
B448 6F62070 4830 RST PRS 5190
B449 6F62070 4840 RST PRS 5200
B450 6F62070 4850 RST PRS 5210
B451 6F62070 4860 RST PRS 5220
B452 6F62070 4870 RST PRS 5230
B453 6F62070 4880 RST PRS 5240
B454 6F62070 4890 RST PRS 5250
B455 6F62070 4900 RST PRS 5260
B456 6F62070 4910 RST PRS 5270
B457 6F62070 4920 RST PRS 5280
B458 6F62070 4930 RST PRS 5290
B459 6F62070 4940 RST PRS 5300
B460 6F62070 4950 RST PRS 5310
B461 6F62070 4960 RST PRS 5320
B462 6F62070 4970 RST PRS 5330
B463 6F62070 4980 RST PRS 5340
B464 6F62070 4990 RST PRS 5350
B465 6F62070 5000 RST PRS 5360
B466 6F62070 5010 RST PRS 5370
B467 6F62070 5020 RST PRS 5380
B468 6F62070 5030 RST PRS 5390
B469 6F62070 5040 RST PRS 5400
B470 6F62070 5050 RST PRS 5410
B471 6F62070 5060 RST PRS 5420
B472 6F62070 5070 RST PRS 5430
B473 6F62070 5080 RST PRS 5440
B474 6F62070 5090 RST PRS 5450
B475 6F62070 5100 RST PRS 5460
B476 6F62070 5110 RST PRS 5470
B477 6F62070 5120 RST PRS 5480
B478 6F62070 5130 RST PRS 5490
B479 6F62070 5140 RST PRS 5500
B480 6F62070 5150 RST PRS 5510
B481 6F62070 5160 RST PRS 5520
B482 6F62070 5170 RST PRS 5530
B483 6F62070 5180 RST PRS 5540
B484 6F62070 5190 RST PRS 5550
B485 6F62070 5200 RST PRS 5560
B486 6F62070 5210 RST PRS 5570
B487 6F62070 5220 RST PRS 5580
B488 6F62070 5230 RST PRS 5590
B489 6F62070 5240 RST PRS 5600
B490 6F62070 5250 RST PRS 5610
B491 6F62070 5260 RST PRS 5620
B492 6F62070 5270 RST PRS 5630
B493 6F62070 5280 RST PRS 5640
B494 6F62070 5290 RST PRS 5650
B495 6F62070 5300 RST PRS 5660
B496 6F62070 5310 RST PRS 5670
B497 6F62070 5320 RST PRS 5680
B498 6F62070 5330 RST PRS 5690
B499 6F62070 5340 RST PRS 5700
B500 6F62070 5350 RST PRS 5710
B501 6F62070 5360 RST PRS 5720
B502 6F62070 5370 RST PRS 5730
B503 6F62070 5380 RST PRS 5740
B504 6F62070 5390 RST PRS 5750
B505 6F62070 5400 RST PRS 5760
B506 6F62070 5410 RST PRS 5770
B507 6F62070 5420 RST PRS 5780
B508 6F62070 5430 RST PRS 5790
B509 6F62070 5440 RST PRS 5800
B510 6F62070 5450 RST PRS 5810
B511 6F62070 5460 RST PRS 5820
B512 6F62070 5470 RST PRS 5830
B513 6F62070 5480 RST PRS 5840
B514 6F62070 5490 RST PRS 5850
B515 6F62070 5500 RST PRS 5860
B516 6F62070 5510 RST PRS 5870
B517 6F62070 5520 RST PRS 5880
B518 6F62070 5530 RST PRS 5890
B519 6F62070 5540 RST PRS 5900
B520 6F62070 5550 RST PRS 5910
B521 6F62070 5560 RST PRS 5920
B522 6F62070 5570 RST PRS 5930
B523 6F62070 5580 RST PRS 5940
B524 6F62070 5590 RST PRS 5950
B525 6F62070 5600 RST PRS 5960
B526 6F62070 5610 RST PRS 5970
B527 6F62070 5620 RST PRS 5980
B528 6F62070 5630 RST PRS 5990
B529 6F62070 5640 RST PRS 6000
B530 6F62070 5650 RST PRS 6010
B531 6F62070 5660 RST PRS 6020
B532 6F62070 5670 RST PRS 6030
B533 6F62070 5680 RST PRS 6040
B534 6F62070 5690 RST PRS 6050
B535 6F62070 5700 RST PRS 6060
B536 6F62070 5710 RST PRS 6070
B537 6F62070 5720 RST PRS 6080
B538 6F62070 5730 RST PRS 6090
B539 6F62070 5740 RST PRS 6100
B540 6F62070 5750 RST PRS 6110
B541 6F62070 576
```

WHAT IS CP/M? An independent look.**D. R. Hunt**

This question is often asked when the purchase of new computer systems is considered. What advantages has CP/M got to offer, is it very complicated, even what does CP/M stand for? In many ways all these questions may be answered glibly by saying that these things are not seen by the user, and therefore the user need not concern himself with them. However, this answer is not entirely true, as the choice of computer operating system is of great importance when it comes to system support and choice of available software.

CP/M stands for Control Program for Microcomputers, and the original version was designed in the late 1970's by Digital Research Inc., to allow a standard form of interchange between one computer system and another. CP/M as such has been a story of growing success, and has been progressively improved with time. What it is, is a standard control program which takes over the computer disk system and the way in which the computer talks to the outside world, through the keyboard, display monitor, printer, etc. This offers one great advantage, in that a program written obeying the CP/M rules on one type of machine will be entirely compatible when run on another type of machine. This of course opens up the availability of software to a far wider market leading to greater choice and lower prices.

In use CP/M is a fairly simple concept, and only has five straightforward commands. These are at the system level and are rarely used by computer operators concerned with business systems, who are primarily concerned with the operation of a program, stock control, financial, etc. The program is working under CP/M, where CP/M is the controlling executive handling the transfer of data and programs to and from the disks, and handling the communication of the program through the keyboard, printer and display monitor. In this respect, CP/M is transparent to the user, and it is the program in use which appears to be in control. At the system development level, where the user is concerned with the design and development of programs, CP/M offers a number of useful utilities giving access to the computer at the machine program level. Most of the CP/M utilities are useful to the machine programmer, except the context editor utility (ED.COM) which is awkward and inconvenient to use. This utility is easily replaced with others of considerably greater usefulness.

CP/M has had much adverse comment unfairly levelled at it. Unfair because the criticism has been concerned with the CP/M input and output facilities, which are not the concern of CP/M, but of the manufacturer of the machine fitted with CP/M. This has come about because, whilst CP/M itself is standardized, the machines to which it is fitted vary widely. To allow CP/M to be adapted to any machine, the machine manufacturer has to write an interface program between CP/M and the machine, this is known as the Basic Input/Output System, or BIOS. It is in this area where criticism is called for, many machine BIOSes are simplistic indeed, and lead to what can only be described as unfriendly operation, giving only simple error messages and the minimum of help. It is rare to find a CP/M machine with useful and friendly performance through the BIOS, although such machines exist, the Gemini range being a shining example.

So what has CP/M got to offer, this may be summed up in one word, compatibility! As mentioned above a program written using the CP/M rules can be run on almost any CP/M based machine. The word 'almost' was used advisedly as the Tandy range of computers have adapted CP/M for their own purposes and in the process made their implementation of CP/M incompatible with all other CP/M machines, a self defeating sales gimmick to ensure that software for their machines can only be purchased through themselves, thereby limiting choice and competition. Having said that CP/M programs are compatible, which is true, there are of course some incompatibilities, usually mechanical. The most obvious is the

mechanical incompatibility between 8" and 5.25" disks. Further, unfortunately, there is lack of standardization between the various 5.25" disk systems themselves. Usually these varying standards have been chosen with care for very valid technical reasons concerned with available disk space. However, it does mean that whilst the programs are compatible, the disks on which programs and data are stored may not be directly transferable from one machine to another.

On the brighter side, these various disk standards do not present much of a problem to software suppliers who are capable of supplying software in almost any disk standard. However, an obscure standard, in an unpopular format is less likely to be attractive to a software supplier, meaning investment in copying software is likely to show a lesser return for the effort involved compared with the popular formats, suggesting that the prices for software in unpopular formats are likely to be a little higher than those in the more popular formats. But usually software houses offset the higher cost of unpopular formats against the more popular formats, and charge a standard price for a given piece of software.

Likewise differing formats do not usually present a problem to end users except where several different CP/M based machines are in use in one situation. In this instance, the machine supplier is often in a position to supply the necessary interchange software and hardware to allow the various machines to be interconnected. In this way only one set of software need be purchased for several machines, assuming the software licences allow multiple copies to be made.

To sum up then, CP/M itself remains transparent to the user, being solely concerned with the efficient handling of the transfer of data and programs to and from the disks. The BIOS handles all the communications with the outside world, which it does to a greater or lesser degree of effectiveness depending upon the ability of the manufacturer of the machine in question. (The effectiveness of the BIOS is a good guide to the quality of support that may be expected from a manufacturer.) The main benefit is the vast choice of standard software available at reasonable prices, brought about by the compatibility that CP/M has introduced throughout the small business system and system development markets.

CLASSIFIED ADS.

Nascom 1 cased, PSU, Cottis Blandford cassette, Stuart colour board, Smart 1 buffer/32K RAM (with memory plague - no instruments/knowledge to fix). £170 ONO. Phone 0382-67896

Nascom 2 cased with 48K, Nas-Sys 3 and graphics. Imp printer, Hobbit digital cassette system. Zeap, Nas-Dis, Debug, Sargon chess. Many games, spare cassettes, books, magazines, manuals. Family circumstances force sale at £590.
Tel. Keith Brown, Colchester (0206) 841295

Teletype KSR 33. RS232 i/f fitted. Can be seen working on Nascom 2. Excellent condition. £50. Keyboard with 80 good quality keys and case. £6.
Tel. Crowthorne 6894.

REVIEW OF HS1N**A. J. Perkins****INTRODUCTION**

A number of storage systems have appeared recently based around the Philips Mini Digital Cassette Recorder (DCR). Priced at less than half the cost of a floppy disk system, they are intended as a cheap(er) alternative to floppies if you want high-speed mass storage.

At the time of writing, there are three such systems available for the Nascom: the 'Hobbit' system at £99+VAT, the Grange Electronics 'CFS' at £170+, and the MicroSpares 'HS1N' system, at £199+ for a single drive system, and £279+ for a double drive system. This is a review of the HS1N double-drive system.

WHAT YOU GET

I ordered this system last summer (1981) after reading an article on it by the designers in PCW (it was also advertised in PCW), and I sat back and waited for it to come. I waited, and waited, and waited... finally, FOUR MONTHS later, it arrived (now where have I heard that before??) courtesy of the GPO. When I unpacked it, this is what I found:

- Two mini DCR's
- One 'NASBUS Compatible' controller card
- Two leads to connect the drives to the controller card.
- Firmware in two 2708's
- Two 4118's for workspace
- One mini-cassette
- One manual

One of the connecting leads was twice the length of the other, and was wired up wrong! Luckily no damage resulted from this. The lack of a 77 way edge connector is to be deplored, as some would consider this essential to connecting the controller card to the bus. Rather than wait weeks for them to send me one (or for them to tell me where to go) I got one from my local Nascom dealer.

HARDWARE

The controller card is an 8"x8" double-sided through-plated fibreglass board, which is just roller-tinned. It would have made a more professional finish if it had been solder masked and silk-screened as well. The board is supposedly 'NASBUS Compatible' and seems to conform to the spec. in all but one aspect: the DMA daisy-chain link had been omitted. At first I thought that this was a design/assembly oversight, but I have since realised that this is to enable entrepid users to fit the DMA chip on the prototyping area of the board, enabling files to be loaded under DMA control... with suitable software, of course. This, of course, wasn't mentioned in the manual (more of which later). The lack of a link won't worry those 99.9% of users who'll never use DMA, and anyway it is the simplest of tasks to add a link should you need it.

All chips on the board are socketed, but none (yes, NONE!!!) are marked on the board. This must give headaches to the poor guy who has to assemble the thing, not to mention anyone who encounters problems, as the circuit diagram supplied is appalling! The controller circuitry occupies about two-thirds of the board, the rest being a prototyping area.

The board I received was very much an 'Issue 1' board, with quite a few wire links and broken tracks on it. Some of these links had fallen off when I unpacked the board (due, no doubt, to the solder joints being cut too close to the board) and a bit of detective work was needed to put things right. Just to

confuse the unwary, the documentation gives the clear impression that the /NASIO circuitry must be added by the user, and the manual goes on to suggest a method of implementing this (using extra chips, of course). To really confuse matters, the actual circuitry used (it IS supplied) lives on the prototyping area and uses a completely different circuit to that suggested!! The DBDR circuitry (for Nascom-1 owners) also lives on the prototyping area.

The circuit itself uses the Z80-SIO to provide the necessary parallel to serial conversion. TTL and CMOS logic provide the phase-encoding and the interface to the drives. The system uses phase-encoding to store bits of information on the tape. Briefly, this entails a change of flux on the tape representing a 1 or a 0, depending on whether this happens on a positive or negative-going clock edge. This ensures that the tape magnetisation changes at least once every clock cycle. Having seen the source listing of the CFS system (which uses the PIO) I must say I think that using the SIO was the best approach, since it avoids the machinations required to generate the serial data stream, generate checksums, sync. characters, etc., and this is reflected in the command set available, which is certainly much more extensive than that on CFS (more about the software later). It also allows an easy upgrade to DMA.

The data rate used is 6000 bits per second, which, because no start/stop bits are used, is the equivalent of 7500 bps (e.g. using CUTS, if you could get it to work that fast!). Data is recorded in blocks of 2K bytes, and the catalogue, which is at the start of the tape, also uses a 2K block. This gives a capacity of 56K bytes per side, excluding the directory. Any data less than 2K in length uses a 'padded out' block. Each data block consists of sync. characters (which serve not only to indicate the start of a block but also to synchronise the read electronics), the load address, a length word, the data stream itself, followed by the CRC characters. Thus, say, a 10 byte 'file' can be loaded without the other 2038 bytes 'above' it in RAM being corrupted.

The hardware falls down in one very important point: whereas the drive buffers use ports fF8-fFB, the SIO uses ports fFC-fFF. This is bad news indeed to those using page mode on their RAM/ROM cards, since the page mode circuitry also uses port fFF (and port fFE on the MAP 256K RAM and Gemini GM813 CPU-I/O-64K RAM boards). Use TOS, and you'll inevitably 'load' a file into ROM. Switch a RAM card back into the system & you'll be reprogramming the SIO!!! The solution is to reconfigure the SIO at a different I/O address (and change the software to suit). The port decoding is hard-wired, but luckily there are a couple of unused inverters on the board, so it is just a matter of breaking a track or two and inverting an address line to reconfigure the port decoding. This is something I have yet to do (spot the RAM-A user!). Neither are the address/data lines buffered on the card: some of the lines have several LS-TTL loads attached to them. Buffering would then make it easier (neater) to mount the 2708s/4118s on the card if your Nascom (& RAM-A) card are already chock full of ZEAP, NASDIS, DEBUG, NASPEN, etc., or has been reconfigured for 2716/6116's.

The drives themselves (of which no information is supplied) are very compact, being about 4" cubed in size. They require a 12 volt power supply, which in this case is drawn from the NASBUS. The manual says that if more than 800mA is already being drawn from the 12V line, then an extra power supply may be needed. The cassettes are very small, being the same type as those used in 'Dictaphone' machines, only the ones used are certified free from drop-outs, which they do indeed seem to be (unlike 'computer quality' C10 audio cassettes). This means that Read/Write errors are definitely a thing of the past! Uncertified cassttes may also be used, but on your own head be it!

SOFTWARE

The operating software (called 'TOS' - Tape Operating System) occupies 2K bytes and is supplied in two 2708 EPROMS. It resides at location fD000, and uses 2K bytes of workspace, which 'sits' on top of TOS at fD800. Two 4118s are

supplied for this. This prevents BASIC from overwriting TOS's workspace when cold-started. TOS overlays ROM ZEAP, but it is a simple matter to install a switch in the 4K decode line (on the N2) to switch between the two. I of course encountered difficulty in this task owing to Blocks A & B on the N2 card being in different places to that documented (i.e. for block A read block B, and vice versa).

The TOS command structure is fairly extensive, with commands being entered in the true NASSYS fashion, i.e. a single command letter followed by a number of arguments. In TOS these arguments are obligatory and are summarised below.

B d	Save BASIC file on drive d
C d	Display catalogue of drive d
D d nn	Delete file number nn from drive d
I	Initialise a blank tape in drive A
J	BASIC cold start
N	NASSYS cold start
P d	Save NASPEN file on drive d
Q	NASPEN warm start
R	Rewind both tapes, read in catalogues
R d nn	Read file nn from drive d
T d nn	Transfer file nn from drive d to the other drive
W d ssss eeee	Write file from fssss to feeee-1 to drive d
X	Rewind both tapes prior to removal
Z	BASIC warm start

I understand that the latest version of TOS has a multiple-delete command, which presumably is of the form D d nn xx yy .. which would delete file numbers nn, xx, yy, etc. from the tape in drive d. I have made alterations to TOS to include the following (extra) commands:

E xxxx	- Execute program at address xxxx. Additional arguments may be supplied, as with the NASSYS 'E' command.
R d	- Rewind the tape & read the catalogue of the specified drive (TOS only does this for both drives at once).
X d	- Rewind the specified tape to the start prior to removal.
Y	- BASIC warm start.
Z d	- Save ZEAP file on the specified drive.

When saving a file, TOS prompts for a filename, which may be upto 17 characters long. This gives ample room for, say, the version number and, in the case of machine code files, the execution address. With the Write command TOS also prompts for a single-character filetype (BASIC & NASPEN default to B and P respectively). No check is made to see if there is already a file of the name you have specified. Each file is assigned a file number, and this number is used to access the file. This reduces the chance of typing errors, which is quite possible if one had to type in 17-character filenames each time. It does mean, however, that when loading a file under program control (i.e. from another program, e.g. loading a data file into RAM) that the file number be specified. While (short) routines are given in the manual for saving a (named) file & loading a (numbered) file under program control, no routine is given for loading a named file. When you consider that the Delete command can cause the assigned filenumbers to change, this is a serious drawback. No source listing is supplied, but having disassembled the software, writing such a routine shouldn't be too difficult.

There were a couple of bugs in TOS, which interfere with the operation of the 'I' command, in which the blank catalogue written to the tape isn't really blank. If any other reader has found this problem, there are a couple of bytes to

change. Since this bug may also be present in other versions (with different addresses) the code which should be changed is just after the prompt for the tape number and the three SCALs to process the answer and get it into HL. The code should read:

E5	PUSH HL
CDFADO	CALL £DOFA ; REWIND TAPE
21ADD7	LD HL,£D7AD
012700	LD BC,£0027 ; ** WAS £23 **
1152DB	LD DE,£DB52
EDBO	LDIR ; WRITE NULL CATALOGUE
2158DB	LD HL,£DB58
2231DB	LD (£DB31),HL ; SAVE CAT. START
212100	LD HL,£21 ; LENGTH ** WAS £1D **
2233DB	LD (£DB33),HL ; SAVE LENGTH

Another undesirable feature (alright then, a bug) in TOS occurs if any errors occur when using TOS under program control. Since errors can occur at several 'depths' in TOS (e.g. an error in the number of arguments, no tape present, a read/write error, etc.) there could be problems in knowing how many return addresses should be 'thrown away' so that TOS can do a quick exit. TOS gets round this the easy way by warm-starting itself whenever errors occur. No problem when using TOS normally (i.e. from the keyboard) but when using TOS under program control, it means that control remains in TOS. No way to get back to your program. You can't just type "W" or "R" (Write or Read) because the arguments are obligatory. You can't return to NASSYS to see what they are because 'N' cold-starts NASSYS, and this resets the workspace. Unless you have designed your program carefully, you have lost all the data your program just spent a long time building. It would be far better to return, say, to the outer level of the Read/Write routines or whatever, and then return to the caller if an error is detected. The latest versions of TOS have a 'Vectored Error handling function' which I presume means that control is passed through a location in RAM to a (user) error routine.

The Write command (and B & P) automatically verifies each block after it has been written by rewinding to the start of the block and then 'reading' the block (but not loading it into RAM) and checking the CRC character at the end. If an error occurs, the block is rewritten, and if the error persists, the command is aborted & an error message is displayed. A drawback is that the catalogue will still contain an 'entry' for the erroneous file, which will have to be deleted.

The I command is used to initialise a tape, and all it does is write a blank directory to the start of the tape. TOS automatically checks to see whether the tape has already been initialised, and displays an error message if it has. No command has been provided to delete the catalogue (e.g. so that it may be reinitialised with another tape number). This is deliberate, as it avoids the accidental 'erasure' of a whole side. It can be done, however, by outputting the relevant commands to the ports, and this method is given in the manual. TOS also remembers where it is on a tape, so when reading more than one file off a tape, there is no need to read in the catalogue both times (or even the once). TOS cannot, however detect if a tape has been removed & replaced with another one. Thus the 'R' command (no arguments) is best used immediately after inserting a cassette.

TOS is written for Nascom systems using NASSYS monitors (sorry, all those NASBUG fans) and automatically detects whether NASSYS 1 or 3 is in use. Much use is made of NASSYS routines but not every call is made in the Approved Manner. The offenders are the call to Parse and to Errm. I don't see why Errm is called (when an invalid command is entered) instead of the address of TOS's own error routine (which calls ERRM anyway in the Approved Manner). These routines are called by reflections in RAM, so should your NASSYS be different (e.g. NASSYS 4 if & when,

or a 'custom' version, maybe) the solution is to single-step through TOS until the addresses have been copied to RAM, then modify these addresses to suit, then proceed.

TOS cannot be used with NASSYS running in RAM. This is because the catalogue load address depends on the drive in use (i.e. Drive A catalogue must be loaded into the Drive A catalogue space in RAM) and not on the drive used to create the catalogue. TOS gets round this by 'throwing away' the header bytes into 'ROM' at location 0. It should be a simple matter (I haven't tried it) to change the address to, say, £800 which is in the video RAM margin. TOS also uses absolute addresses to copy over the NASSYS subroutine table. Again, this is avoidable. TOS calls STMON, which initialises NASSYS, so just a simple look into \$STAB should suffice (unless I've missed something). In fact (I know 'cos I've done it) rewriting TOS to have its own PARSE routine and to look in \$STAB as above releases a bit more space, enough for FOUR whole new commands! It also allows it to run on any version of NASSYS. The bit that initialises the SIO can also be rewritten to take up a fraction of its original length. The lack of a Save ZEAP command in the original version of TOS is to be deplored, especially, as I've said, there is enough room to slot one in.

It may also be of interest to note that a software package is available from HS Design in Scotland (from the same guys who designed the HS1N) for HS1N systems, which greatly extends the commands under Crystal BASIC to include many disk-like commands. The command set looks very comprehensive, and would certainly be worth looking into if you use Crystal BASIC a lot. It will only run on the latest version of TOS (not earlier ones, though they will supply the latest version free with the BASIC Extension if you return the original ROMs), but I doubt very much whether it will run on my 'custom' version. I wouldn't buy it, but only because a) I don't have Crystal BASIC, b) I'd probably lose the extra facilities of 'my' TOS, and c) I don't use BASIC all that much anyway.

DOCUMENTATION

The documentation supplied comes in the form of a small manual which is A5 size. I've mentioned the misleading bit about the /NASIO circuitry, and also the appalling circuit diagram (which is so bad it looks more like a wiring diagram). No source listing of the software is supplied either, so what does the manual contain? Well, basically it explains the operation of the various commands fairly well (enough to enable you to save files without getting it wrong) but I found the format a little cramped. An A4 size manual with more help on using TOS under program control (I keep on about this, but since it is advertised as 'looking like' a floppy disk system & all floppy disk users use operating system commands to load/save data from their programs, I don't see why TOS users shouldn't either) would also be useful (or perhaps a commented sources listing - I've done it myself but probably didn't get it all right, and there's still one or two bits I still don't understand). Some information on the drives themselves would also be useful.

COMPATABILITY WITH OTHER SYSTEMS

I know very little about the 'Hobbit' system except that it uses the PIO, so all comments here relate to comparisons with the 'CFS' system. CFS also writes in blocks of 2K, also at 6000 bps. CFS commands are 'menu-driven' but are less extensive than those of TOS (they do include a 'save ZEAP' command, though). CFS data format is a sync. byte, followed by a load address word, followed by a length word, followed by the checksum, followed by the data bytes. This is therefore incompatible with the HS1N format, and CFS systems will be unable to read HS1N tapes, and vice-versa. One day I might 'con' a CFS user to help with some routines to overcome this. If and when they are finished I might write it up into an article for the mag. (By the way, can 80-Bus News accept NASPEN files dumped onto these min-cassettes? No, I thought not).

CONCLUSION

This system has been in use for several months now, and I must say that I am very pleased with it. I use my Nascom a damn sight more often than I did when I was relying on audio cassettes. I now use my Nascom for actual serious work (i.e. with huge ZEAP/NASPEN, etc. files)... when was the last time you dumped several 28K files to (audio) tape in an evening, eh? And how often does it refuse to verify, meaning you have to do the whole thing again? With the HS1N, no such problems occur, as verification is automatic (as I've mentioned). If a block fails to verify (which usually means you are using uncertified tapes) it will try again... in the meantime you can go away & make a cuppa or whatever. Above all, this system is so fast compared with the old CUTS system that you will soon wonder how you ever managed without it!

The few points I criticised don't detract from the system's usefulness & versatility, they just make the system look a trifle untidy. They really need attention though, for a more 'professional' finish... an 'issue 2' release, perhaps? I could make a long list of extra commands it would be 'nice' to have, in addition to the ones I have already added, e.g. Loading a named file, Append ZEAP/NASPEN/BASIC file, auto-relocation of ZEAP file when loading (useful if you need to relocate the ZEAP buffer)... but then TOS wouldn't occupy 2K (it would be more like 4K). There is no reason why the intrepid user shouldn't add his own commands, if he/she wishes...

Since I bought the system from Microspares in Edinburgh it has achieved 'Nascom Approved' status, so perhaps some of the points I have mentioned have received attention. I should also think that the HS1N will now be available through your friendly local Nascom dealer.

One final thought...this article was prepared using NASPEN. The file will take about 30 seconds to save & verify. The Editor likes NASPEN (CUTS) tapes [Ed. - or disks!]. It will probably take about half an hour to dump it & verify it. Now where did I put that old cassette recorder...

POLYDOS TAPE BACKUP

M. J. R. GIBBS

Recently (much to the disgust of my Bank Manager) I have purchased a disk system, the Gemini G809/G815 running with Polydos, which is an excellent product and I have been able to interface all my existing software so that the I/O can be either Disk or Tape.

However, one of the problems that I had was producing backup copies, and after having one disaster (entirely my own fault) doing a backup with a single disk system, I decided to write a Tape Backup Utility. This utility reads in all the files on the disk and writes them to tape, some common files can be omitted by placing the file names in a list. The user only has to load the disk, set the tape on to record and run the program, then the Backup is taken automatically. The files are written onto tape in a similar way to the Generate command. To recover the files you simply set the tape unit to play, and each file is read into memory. Stop the tape after the file required has been read and then use the Polydos 'SAVE' command to restore the data. All files are located at £1000 upwards (and must be less than £B000 long), each file is identified and the load and execute addresses are displayed. All files are copied including ones that have been 'Deleted'. I have enclosed two listings the first a full Assembly listing and the second a dump using a slightly modified version of the DUMP utility published in vol 1 issue 2 (The addresses on the left hand side are the RAM addresses), the program loads and executes at £OC80. Using this program it is possible to backup a disk in a few minutes with the cassette running at 4800 Baud, depending on how full the disk is (I run my cassette at 6000 baud without any trouble).

```
*****  
* TAPE BACKUP OF DISK FILES *  
* - - - - - *  
* M J R GIBBS 14-07-82 *  
*****  

NASSYS EQU $18  
PRS EQU $28  
ROUT EQU $30  
RDEL EQU $38  
0018  
0028  
0030  
0038  
007B  
0068  
0065  
0071  
0077  
005B  
006F  
0066  
0081  
0083  
0088  
0088  
C055  
C069  
C001  
C400  
C414  
C416  
C418  
C418  
C055  
C055  
C05D  
C05F  
C060  
C061  
C063  
C065  
C067  
C067  
OBCA  
080A  
084A  
084A  
084A  
OCOC  
OCOE  
CURSOR EQU  
OC29  
*****  
; NASSYS COMMANDS  
BLINK EQU $7B  
B2HEX EQU $68  
CRT EQU $65  
NOM EQU $71  
NNOM EQU $77  
RETNAS EQU $5B  
SOUT EQU $6F  
TBCD3 EQU $66  
DISK COMMANDS  
ZDRD EQU $81  
ZRDIR EQU $85  
ZCOV EQU $88  
S1FCB EQU $C0055  
S2FCB EQU $C0069  
DDRV EQU $C001  
BUFFER EQU $C400  
NTSEC EQU $C414  
ENDDIR EQU $C416  
DIR EQU $C418  
C418  
ORG S1FCB  
LOAD O  
FNAM DS 8  
FEXT DS 2  
FSFL DS 1  
FUBL DS 1  
FSEC DS 2  
FNSC DS 2  
FLDA DS 2  
FEXA DS 2  
LOAD ADDRESS  
EXEC ADDRESS  
SCREEN ADDRESSES  
TOPIN EQU $0BCA  
LINE1 EQU $030A  
LINE2 EQU LINE1+$0040;  
; MONITOR LOCATIONS  
;
```

* CURSOR CONTROL CHARS *

CLEAR EQU \$0C
CRET EQU \$0D
ESC EQU \$1B
TAB EQU \$09
;

0C80 AF START XOR A ;SET FILE NUMBER
0C81 32E50E LD (\$FILENO),A ;ZERO FILE NUMBER
0C84 CDDAQD CALL HEAD ;WRITE HEADINGS
0C87 214A08 STA10 LD HL,LINE2
0C8A 22290C LD (CURSOR),HL
0CBD EF RST PRS
0CBE OD09 DB CRET,TAB
0C90 4C4F4144
0C94 20204469
0C98 736B2074
0C9C 6F206265
0CA0 2044756D
0CA4 706564
0CA7 OD0D09 DB CRET,CRET,TAB
0CA8 50726573
0CAE 7320454E
0CB2 54455220
0CB6 5768656E
0CBA 20526561
0CBE 64792E
0C91 OD0D09
0C94 20202020
0C98 20204253
0CCC 43202020
0C90 746F2073
0C94 746F702E
0C9B OD0D09
0C9B 456E7465
0C9F 72205265
0C95 71756912
0C9F 6E203D5D
0C93 3D2E00
0C76 DF RST NASSYS
0C77 7B DB BLINK ;WAIT REPLY
0C78 FE0D CP CRET ;WAS CRET
0CFA 2807 JR Z,STA20
0CF0 FE1B CP ESC ;WAS ESCAPE
0CFE C2800C JP NZ,START ;NO TRY AGAIN
0D01 DF RST NASSYS
0D02 5B DB RETMAS ;STOP
0D03 CDD0E STA20 CALL WAIT ;WAIT .5 SEC
0D06 3EFF LD A,FFF ;RESET DIRECTOR READ
0D08 3201C0 LD (DDRV),A ;ZAP IT IN
0D0B 0E00 LD C,O ;FORCE READ ON DRIVE 0

```

ODD0 DF RST NASSYS ;READ DISK DIRECTORY
ODEO 83 DB ZRDIR
ODEO 2816 JR Z,STA50
OD11 EF RST PRS
OD12 0D0945552 DB CRET,TAB,'ERROR TRY AGAIN',0
OD16 52459220 STA10 ;COPY DISK ID
OD1A 54525920 STA30 LD HL,BUFFER
OD1E 41474149 DE,NAME
OD22 4E00 LD BC,20 ;LOAD IT OUT
OD24 C5870C LD DIR,LDIR ;HL = A(DIRECTORY)
OD27 2100C4 STA40 LD HL,DIR
OD2A 111FOE LD (DIRPOS),HL ;SAVE IT
OD2D 1A1400 LD HL,(DIRPOS) ;HL = POS IN DIR
OD30 EDB0 LD DE,(ENDDIR) ;DE = A(END DIR)
OD32 211BC4 STA40 LD HL,DIR
OD35 22E30E LOOP10 LD HL,(DIRPOS) ;HL = POS IN DIR
OD38 2AEE30E LD DE,(ENDDIR) ;DE = A(END DIR)
OD3B ED5B16C4 OR A ;END OF FILES ON DISK
OD3F B7 SBC HL,DE ;RESTORE ADDRESS
OD40 ED52 ADD HL,DE
OD42 19 ADD Z,START ;YES MEANT DISK
OD43 CA800C JP DE,S1FCB ;COPY TO S1FCB
OD46 115C0 LD BC,20 ;BC = L1FCB
OD49 011400 LD HL,S1FCB ;HL = THIS FILE NAME
OD4C EDB0 PUSH IX ;KEEP IX
OD4E 22E30E LD (DIRPOS),HL ;ZAP IT ACROSS
OD4F INC HL ;KEEP ACROSS
OD4E ;--- SCAN TABLE --- ;--- SCAN TABLE ---

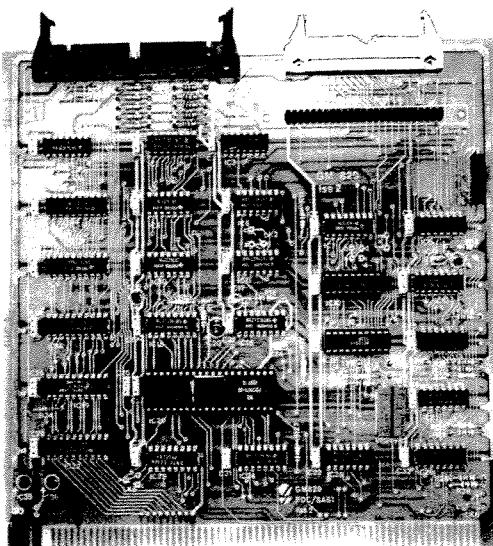
OD4E LD B,13 ;B = NO OF TABLE ENT
OD51 060D LD IX,TABLE ;IX = A(TABLE)
OD53 DD1E90E LD DE,10 ;DE = ENTRY SIZE
OD57 110AC0 LD PUSH BC ;KEEP BC
OD5A C5 LOOP20 LD B,10 ;B = SCAN LENGTH
OD5B 060A LD HL,S1FCB ;HL = THIS FILE NAME
OD5D 2155C0 PUSH IX ;KEEP IX
OD60 DDE5 LD (HL) ;GET FROM TABLE
OD62 DD7E00 LOOP30 LD A,(IX+0) ;COMPARE
OD65 BE POP IX ;FAIL
OD66 200A JR NZ,NEXT
OD68 DD23 INC IX ;INDEX TO NEXT CHAR
OD6A 23 INC HL ;REPEAT
OD6B 10F5 DJNZ LOOP20 ;GOT IT OK
OD6D DDE1 POP IX ;RECOVER STACK
OD6F C1 POP BC ;CONTINUE
OD70 18C6 JR LOOP10 ;RESET POINTER
OD72 DDE1 NEXT ADD IX,DE ;POINT TO NEXT
OD74 DD19 POP EC ;RECOVER BC
OD76 C1 DJNZ LOOP20 ;TRY AGAIN
OD77 10E1 STA50 CALL WAIT ;PRINT HEADINGS
OD79 CDDDE CDA0D CALL HEAD ;PRINT FILE INFO
OD7C CDA0D CALL HEAD ;LOAD ADDRESS
OD7F CDD01E CALL PRINT ;SECTOR ADDRESS
OD82 210010 LD HL,1000 ;A = NUMBER OF SEC
OD85 ED5B61CO LD DE,(FSEC) ;B = DIRECTORY DRT
OD89 3A63C0 LD A,(FNSC) ;C = DIRECTORY DRT
OD8C 47 LD B,A
OD8D 3A01C0 LD A,(DDRV) ;C,A
OD90 4F LD C,A

```

Additional Information on Gemini Multiboard Products

NOVEMBER 1982.

Since the Autumn Catalogue of Gemini MultiBoard products was published, a number of interesting additional items have been added to the product range. This supplementary section includes information on these new products, plus further information on other aspects of MultiBoard and 80-BUS.



GM829 – FDC/SASI BOARD

- * Single/Double Density Operation
- * Single/Double Sided Drive Support
- * Up to 4 mixed 3.5", 5.25" and 8" Drives
- * Industry Standard SASI Hard Disk Interface

The Gemini GM829 combined FDC (floppy disc controller) and SASI (Shugart Associates Systems Interface) board has been designed to allow both floppy disc drives and Winchester hard disk drives to be easily added to a MultiBoard system.

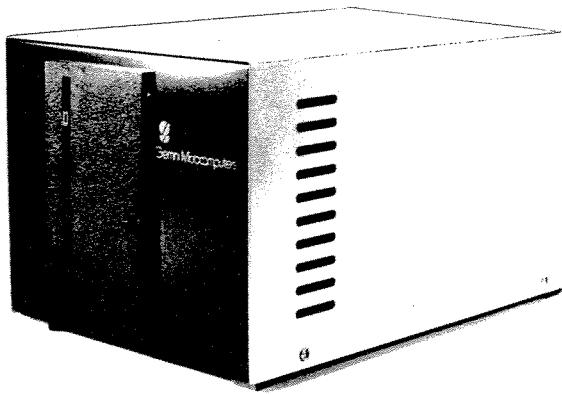
Up to four floppy disk drives may be controlled. These may be single or double sided, 48 or 96 TPI drives, in single or double density formats. The drives may also be 3.5", 5.25" or 8" types, or a combination of these. Switching between single and double density, and 3.5"/5.25" and 8" drives, are under software control.

High performance and reliability are provided by variable write precompensation and phase locked loop data recovery circuitry. The board uses the Western Digital 1397 chip set and occupies 8 Z80 I/O ports. These ports may be set to one of two positions, allowing two GM829 boards to be used in a single system.

The industry standard 50 way 'SASI' interface allows Winchester hard disk sub-systems, such as the Gemini GM835, to be simply plugged straight in.

This board is a development of the extremely popular GM809 FDC board and maintains the same elegant and reliable engineering design which has been proven on the GM809. Full 5.25" software compatibility is maintained, with the added advantage of also being able to control 3.5/5.25" 8" and hard disk drives simultaneously from the one 80-BUS board.

GM829 – FDC/SASI – £145



GM835 – WINCHESTER DRIVE SUB-SYSTEM

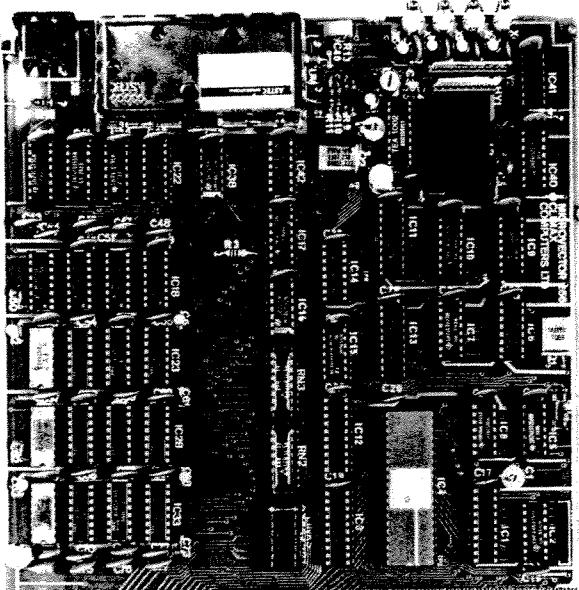
- * 5.4 Megabyte Formatted Capacity
- * Industry Standard SASI Interface
- * Integral Controller and Power Supply

The Gemini GM835 has been designed to plug directly into the new GM829 FDC/SASI board to provide the MultiBoard user with the hard-disk advantages of high performance, high speed, high reliability and high storage.

The housing, styled to match the Gemini Galaxy and GM820 floppy disk housings, contains the Winchester drive, a Z80 based Winchester controller board, and a switch mode power supply. The drive used is the British made Rodime RO201, providing 6.67 Megabytes of unformatted capacity, 5.4 Megabytes formatted. Average seek time is 90ms and the data transfer rate is 5 MBits per second. The intelligent controller provides a 512 byte sector buffer and automatic error detection and correction.

Higher capacity drives will be made available in the future.

GM835-6 – 5.4 MBYTE WINCHESTER SUB-SYSTEM – £1450



CC837 – COLOUR GRAPHICS DISPLAY INTERFACE

- * 256 x 256 16 Colour Pixel Display
- * Ultra-fast Vector and Character Generation
- * 96 ASCII Character Set
- * Audio and Light Pen Inputs
- * Available January 1983

The Climax Computers CC837 is a high performance graphics display interface board. Various graphics primitives such as vector and character generation are provided by a Thomson EF9365 Graphic Display Processor. The plotting rate is typically 1 million pixels per second, giving animation capability. Various vector and character types can be selected and characters may be scaled to give 256 different sizes.

The CC837A has a high quality PAL UHF output with an intercarrier sound facility as well as a composite 75 ohm B/W output. The CC837B has additional 75 ohm RGB outputs for connection to a video monitor.

The CC837 provides a resolution of 256 x 256 pixels in 16 colours using 32K bytes of on-board memory. The board occupies 17 Z80 I/O ports, commands being single byte control codes. A socket is provided, compatible with the AM820 light pen, to be used for interactive graphic procedures.

A comprehensive set of assembly language subroutines given in the operating manual enables the user to develop his own graphics programs quickly and efficiently.

CC837A – COLOUR GRAPHICS (PAL) – £199

CC837B – COLOUR GRAPHICS (PAL + RGB) – £220

Multinet Local Area Network

IN DESIGN

Available January 1983

GM836 – Network Interface Board

- * Low Cost
- * Up to 32 Stations
- * Simple Interface to MultiBoard Systems
- * Available January 1983

There are many occasions when it would be useful to be able to connect several MultiBoard based computer systems together, in order to communicate with each other or to share expensive resources such as disks and printers.

Gemini have developed MultiNet for this purpose, and as with all Gemini products, MultiNet is low cost, easily expandable and highly flexible.

The Gemini GM836 is a small add-on board which plugs into the 26-way connector of the PIO of a GM811 or GM813 CPU board by means of a ribbon cable. The board contains all of the necessary interfacing circuitry to allow the connection of a machine to the network. Data is transmitted serially using a differential transmission method at about 300 Kbaud along a single coaxial cable. Connection between any machine and the network is by BNC T-Junction, this allows any station to be disconnected from the network without disturbing the operation of other stations. Using this system up to 32 machines can be connected to the network and the length of cable can be up to 600m (2000 feet) end to end.

The associated software uses a CSMA/CD (Carrier Sense Multiple Access with Collision Detection) technique where station access to the network is on a contention basis, the software deferring transmission until the network is silent, transmitting, detecting a collision (or simultaneous transmission by another station) and then delaying for a random time before attempting to retransmit. Data is transmitted through the network in variable sized packets, each packet having a CRC to detect errors during transmission, retries being automatically performed if a packet is delivered incorrectly.

A MultiNet System

MultiNet allows several stations access to shared disks and printers when in a CP/M environment. Gemini are currently in the process of developing both hardware and software to allow this to be achieved extremely cost effectively.

The hardware consists of a number of MultiBoard based systems without disks but with the network interface integral to the machine, and the appropriate software to allow all disk and printer operations to be performed by another MultiBoard system acting as a combined file and printer server, which has a printer and Winchester hard disk drive and is dedicated to managing all the shared resources. Software will also be available to allow machines running under CP/M with their own disk drives (such as the Gemini Galaxy) to connect to the network and access the shared resources.

IN DESIGN

Available January 1983

GM839 – PROTOTYPING BOARD

- * Fibreglass PCB
- * 80-BUS Signal Identification
- * High Density IC Capability
- * Available January 1983

The Gemini GM839 high quality 80-BUS prototyping board provides the MultiBoard user with a convenient means of adding specialised 'one-off' boards to his system. This single sided fibreglass PCB provides extensive power supply tracking and the layout has been optimised to allow a high IC packing density. Additionally one edge of the PCB has been designed to accommodate multi-way insulation displacement type connectors.

The component side of the GM839 is silk-screened to show the positioning of the power rails and component pads, and all 80-BUS signals are identified. On the track side of the board the 80-BUS lines are identified by number. An 80-BUS specification booklet is included.

GM839 – PROTOTYPING BOARD – £TBA

MP840 – TERMINATED BACKPLANE

- * 14 Slot Backplane
- * Signal Termination
- * Extensive Ground Shielding

The Microcode MP840 is a 14 slot 80-BUS backplane which has been specially designed to overcome problems often associated with running long microcomputer backplanes. All active bus signals are terminated into a potential balanced RC filter and are interleaved with ground shield tracks, plus one side of the backplane provides a complete ground plane.

The backplane features interrupt and bus request daisy chaining and can be used in different lengths by a simple 'score and break' operation.

MP840 – 14 SLOT BACKPLANE – £47

Multiboard Based SYSTEMS



GEMINI GALAXY

The Gemini Galaxy 1, launched in January 1982, has met with considerable success in a variety of fields. Being based around the MultiBoard range it has proved particularly useful as both a hardware and software development system. In addition it has found many applications in business and education.

With the more recent availability of the GM813 board, capable of replacing both the GM811 and GM802 boards in a Galaxy type system, and with the addition of the new GM827 extended keyboard to the product range, Gemini has now launched the Galaxy 2 range of computers. The Galaxy, briefly described in the Autumn Catalogue, is available in its Mark 2 form in three alternative disk drive configurations.

GM903

The standard Galaxy 2 is the GM903, and like all of the Galaxy 2 range it is based on a three board MultiBoard set, the GM813 CPU-I/O-64KRAM, GM812 IVC and GM809 FDC, along with the GM827 extended keyboard. This configuration provides the user with two spare 80-BUS slots for future expansion.

In the GM903, twin Micropolis 1015F5 drives are fitted, providing 800K of formatted disk storage.

As in all Galaxy configurations, the CP/M 2.2 operating system is included in the package, along with a 12" video monitor.

GM904

For users who require a Galaxy 2 but either do not need twin disk drives, or wish to start with a lower cost system, the GM904 offers all of the features of the GM903 but is supplied with only one 1015F5 drive, providing 400K of formatted disk storage. All connectors and cables are included to allow the second drive to be added at a later date if required.

GM905

The third option in the Galaxy 2 range is the GM905. This system is aimed at users with particularly high storage requirements and is supplied with twin Micropolis 1015F6 drives, offering a massive 1.6 Megabytes of formatted disk storage.

GM903 – GALAXY 2 WITH TWIN 400 KBYTE DRIVES – £1495

GM904 – GALAXY 2 WITH ONE 400 KBYTE DRIVE – £1275

GM905 – GALAXY 2 WITH TWIN 800 KBYTE DRIVES – £1695

Further storage expansion capability of the Galaxy system is provided by the GM825 disk drive unit and GM835 Winchester drive sub-system, both of which have been specifically designed to complement the Galaxy range.

Purchasing Your Multiboard System

Gemini Microcomputers does not normally sell its products direct to the end user, unless that user is an OEM requiring large quantities of product. Sales are usually made via a number of dealers throughout the UK and Europe. Details of your nearest dealer can be obtained from the computer press, or by telephoning Gemini on 02403 - 28321.

NON GEMINI 80-BUS PRODUCTS

It is Gemini Microcomputer's policy to include 80-BUS compatible products in its catalogues which are manufactured and marketed by other companies. The purpose of this is to ensure that the potential customer is aware of the wide range of 80-BUS products available.

All Gemini products have product numbers which have 'GM' prefixes. Products with other prefixes are manufactured by the following companies, and are generally available from Gemini dealers.

CC -	Climax Computers Ltd., 17a Broad Street, South Molton, Devon. EX36 4JE	MP -	Microcode (Control) Ltd., 41a Moor Lane, Clitheroe, Lancs.
EV -	EV Computing Ltd., 700 Burnage Lane, Burnage, Manchester.	AM -	Arfon Microelectronics Ltd., <i>These products are currently being re-sourced. Contact your dealer for further details.</i>
IO -	IO Research Ltd., 6 Laleham Avenue, Mill Hill, London. NW7 3HL		

COMPATIBILITY WITH OTHER MICROCOMPUTERS

Gemini Microcomputers commenced production of Nascom compatible products in 1980. This range of products continued to grow during the following year, and in mid 1981 we published the 80-BUS specification. The purpose of this was to remove some of the ambiguities present in the Nasbus specification and to encourage other manufacturers to produce 80-BUS compatible products by giving the BUS structure a name unassociated with one specific company.

80-BUS, therefore, is essentially identical to Nasbus, and Gemini MultiBoard products are thus compatible with the Nascom range of computers and have been used by many people to expand their Nascoms.

80-BUS NEWS

'80-BUS NEWS' is a magazine produced six times a year for owners of 80-BUS based microcomputers. The aim of '80-BUS NEWS' is to provide owners of Gemini and Nascom systems with a comprehensive guide to the equipment that they own, and to provide information on the extremely wide range of further hardware and software products which are available for their machines.

A major objective of '80-BUS NEWS' is to act as a central discussion point for Gemini and Nascom users. Each issue of '80-BUS NEWS' contains a blend of hardware and software reviews, articles on hardware construction and on programming, a number of tips and hints, letters received from readers, and some occasional light relief - all with the aim of assisting both Gemini and Nascom owners to obtain the maximum use and/or pleasure from their chosen equipment.

ERRATUM

Please note that the price of the GM807 3A Power Supply, as shown in the Autumn Catalogue, should be £40 and NOT £140.

Gemini Microcomputers reserves the right to amend prices and specifications without notice.

Z2 ASSEMBLY LISTING PAGE 5

Z2 ASSEMBLY LISTING PAGE 6

```

OE08 21E60E LD HI,OUTTAB ;SET UART ON
OE09 DF RST NOM ;SWITCH NEW OUTPUT ON
OE0C 71 RST PRS
OE0D EF DB CRET,CRET
OE0E 0DD DB CRET,CRET
OE10 20204499 DB CRET,CRET
OE14 53EB204E DB CRET,CRET
OE18 414D4520 DB CRET,CRET
OE1C 3A2D20 DB CRET,CRET
OE1F 20202020 DB CRET,CRET
OE23 20202020 DB CRET,CRET
OE27 20202020 DB CRET,CRET
OE2B 20202020 DB CRET,CRET
OE2F 20202020 DB CRET,CRET
OE33 OD NAME DB CRET,CRET
OE34 20204649 DB CRET,CRET
OE38 4C4204E DB CRET,CRET
OE3C 4F202020 DB CRET,CRET
OE40 3A2D2000 DB CRET,CRET
OE44 3AE50E DB CRET,CRET
OE47 3C INC A,(FILENO) ;FILE NO :- ,00
OE48 27 DAA ;FILENO = FILENO+1
OE49 32E50E LD (FILENO),A ;CONV TO DEC
OE4C DF RST NASSYS
OE4D 68 DB B2HEX ;PRINT FILENO
OE4E EF RST PRS
OE4F OD202046 DB CRET,CRET
OE53 494C520 DB CRET,CRET
OE57 4E1AD45 DB CRET,CRET
OE5B 203AD2D0 DB CRET,CRET
OE5F 00 LD HL,SIFCB ;HL = A(FILENAME)
OE60 215FC0 LD B,8 ;B = L(FILENAME)
OE63 0608 LD A,(HL) ;GET CHAR
OE65 TE PRT10 LD ROUT ;PRINT IT
OE66 F7 RST OUT ;INDEX TO NEXT
OE67 25 INC HL ;LOOP ON IT
OE68 10FB INC PR10 ;INDEX ALONG
OE6A 3E2E LD A,(HL) ;GET CHARACTER
OE6C 0603 LD B,3 ;OUTPUT FILE EXTENSION
OE68 2B DEC HL
OE6F F7 PRT20 RST OUT ;PUSH IT OUT
OE70 23 INC HL ;INDEX ALONG
OE71 7E LD A,(HL) ;GET CHARACTER
OE72 10FB DJNZ PRT20
OE74 3A5FC0 LD A,(FSTL) ;A = SYSTEM FLAG
OE77 FEC1 CP 1 ;IS = LOCKED FILE
OE79 2012 JR NZ,PRT30
OE7B EF RST PRS
OE80 4C7C3C20
OE84 6542020 DB '<< Locked >>,00
OE88 3E3E00 JR PRT40 / ;IS = DELETED FILE
OE8B 1814 PRT30 CP 2 / ;IS = LOCKED FILE
OE8D FEC2 JR NZ,PRT40
OE8F 2010

```

```

OE91 EF RST PRS
OE92 203C5D20 DB CRET, LOAD AT :- ,00
OE96 44454C45 LD HL,(FILE) ;HL = LOAD ADDRESS
OE9A 54454420 DB CRET, LOAD AT :- ,00
OE9B 3E3E00 RST PRS
OE9C 4F414445 RST PRS
OE9A 44204154 RST PRS
OE9E 203A2D20 DB CRET, ;NASSYS
OE9F 00 RST PRS
OE93 2A65C0 RST PRS
OE96 DF TBCD5 ;PRINT IT
OE97 66 RST PRS
OE98 EF RST PRS
OE99 0D202045 RST PRS
OEBD 5B454320 RST PRS
OECD 41542020 RST PRS
OECE 203A2D20 DB CRET, EXEC AT :- ,00
OE99 00 LD HL,(FILEA)
OECA 2A67C0 RST NASSYS
OECD DF RST TBCD5 ;PRINT IT
OECE 66 RST PRS
OECF EF RST PRS
OEDE 0D00 DB CRET,00
OEDE 3E52 LD A,R ;FORCE A READ
OEDE DF RST NASSYS
OEDE 6F DB SOUT ;ZAP IT OUT THE UART
OEDE 5E0D LD A,CRET ;FOLLOWED BY CRET
OEDE DF RST NASSYS
OEDE 6F DB SOUT
OEDE DF RST NASSYS
OEDE 77 DB SOUT
OEDE C9 RET RST PRS
OEDE DF RST NASSYS
OEDE 6F DB SOUT
OEDE DF RST NASSYS
OEDE 77 DB SOUT
OEDE C9 RET RST PRS
OEDE DF RST NASSYS
OEDE 6F DB SOUT
OEDE AF RDELT
OEDE 47 DIRPOS DS 2
OEDE FF WAIT10 RST RDEL
OEDE 10FD OEOE 09 DIRPOS DS 1
OEDE 09 RST RDEL
OEDE C9 FILENO DS 1
OEDE 656FOO OUTTAB DB CRT,SOUT,200
OEDE 45785563 OEOF7 20202020 TABLE DB 'Exec OV'
OEDE 4F56 OEOF7 20202020
OEDE 456D7367 OEOF7 20202020
OEDE 456D7367 OEOF7 20202020
OEDE 4F56 OEOF7 20202020
OEDE 4466756E OEOF7 20202020
OE01 20202020 OEOF7 4F56
OE05 4F56 OEOF7 4563D64
OE0B 20202020 OEOF7 4F56
OEOF 4F56 DB 'Emsg OV'

```


As was mentioned in the last issue of 80BUS News, the series 'Kiddies Guide to Assembler Programming' had found its inevitable end as the topics to be covered were becoming too long and complex to be dealt with in a single issue. However, as the urge to write in a, hopefully, understandable manner is no less diminished, the search for other topics has continued. In this article the intention is to cover the Gemini GM822 real time clock kit from both the construction and the software point of view.

Of late a number of real time clocks have appeared for the Nascom and Gemini, five at the last count, ranging in price from about £25.00 to £50.00, with one at £125.00. The later is the Gemini GM816 I/O card which also contains a CTC, three PIOs and fairly comprehensive expansion capability as well, which explains the otherwise highish price. Of the cheaper clocks, these are small cards measuring up to three inches square and are intended to be piggy backed onto the host board with double sided sticky tape, or to be mounted separately as convenient. Four of the five clocks feature the National Semiconductor MM58174A chip, the fifth I have not seen, but from the price, it would be reasonable to assume the same chip. All incorporate battery backup. The clocks are provided in either built or kit form with the necessary software and documentation to build, connect and run them.

The Gemini GM822 is the only version I have had experience with, and this part of the article will be confined to a review of that item. The Gemini GM822 is supplied in kit form at £29.50 + VAT, a middle of the road price when compared with its competitors.

The kit comprises the documentation and the component parts. The documentation is extensive, covering the constructional details, a detailed description of the circuit employed, some three pages on the workings of the MM58174 itself which is considerably more understandable than the National Semiconductor data sheet for the chip (not supplied), and a simple software listing in both assembler and HEX dump for either the Nascom or Gemini machines. The routines are simple and well explained. Because they are short, however, the routines are unforgiving of input errors, but form a sound basis for more elaborate routines which may be written.

The components were all there as per the parts list, although a connecting lead to the PIO plug was not provided. The printed circuit board is a little disappointing, and not up to the usual Gemini standard. The PCB is glass fibre and is quite small, about 4cm by 6cm, the tracking is single sided only. Unfortunately the board is not coated in the usual Gemini green solder resist and it is only too easy to bridge pins of the IC sockets or the 26 way IDS connecting plug during assembly. Extra care during assembly and inspection needs to be taken. The only other problem encountered during assembly was with the tantalum bead capacitor. The documentation blithely warns about ensuring that the component is connected with the correct polarity, but as the type supplied was of the colour coded variety with no polarity markings, this could present a problem for those who do not know the correct connection. For those who do not know the correct connection, the positive lead is on the right when holding the capacitor leads down with the coloured spot facing you. No other problems were encountered during assembly.

Two clocks were ultimately constructed, the first one at home and the second at work a few days later. So, to the first clock, constructed and tested at home. The software was typed in from the HEX listings provided, and checked. The port base address, 04H for Nascom, or 0B4H for Gemini was changed throughout to 1CH to suit a spare port on my Nascom I/O card, and to remain consistent with some already existing software which had been acquired from others who already had the clock working. The clock was connected with a 6" ribbon cable folded 'Z' shaped and flattened to shorten it to about 2" and the clock stuck down on top of

an adjacent PIO chip with double sided, foamed, particularly tenacious sticky tape, making for tidy mechanical assembly. On test the clock functioned perfectly.

The clock is provided with a trimmer for setting the onboard crystal to exactly 32.768KHz, however, no advice on setting this is offered. The best method is adjustment over a long period, days or weeks, carefully taking up any errors in time noted. Connecting any external device such as a frequency counter will load the onboard clock and render inaccurate readings, also the accuracy of most cheaper frequency counters is likely to be less than the accuracy required for setting the clock to +/- 5 seconds a day, so the only choice is long term adjustment. There is also a second problem associated with clock accuracy, and that is that unless the chip is addressed completely within a specified minimum time frame, the clock will gain about 1/100th of a second each time it is addressed. More on this later. All this is coupled in my case, with the fact that my computer is now mounted in an inaccessible totally enclosed aluminium box for RF shielding, making long term adjustment an impractical proposition. The trimmer was set to mid position, and over the last two weeks the clock has gained some seven seconds. As I am only interested in hours and minutes, this is of little importance.

The second clock constructed a few days later at work, however, behaved in a most erratic fashion. This clock was connected to the onboard port of a Gemini GM811 CPU card, and the software, which had worked correctly on the home built clock, was reassembled to port OB4H. Due note was taken of the warning in the clock documentation concerning the Gemini CP/M initialisation routines setting the port for Centronics printer operation causing rubbish to be written into the clock registers, and these were duly bypassed. Regardless of how the clock was initialised, it would not return from the read routine unless a finger was placed across the crystal and then it would always return completely repeatable rubbish. After much checking of the PIO, the software and almost complete component substitution, the conclusion was reached that the crystal oscillator was either running at low level or not at all, and that the only item that could be at fault was the pcb board itself. This being patently ridiculous, we must have missed something. None the less, the pcb was rigorously cleaned to remove any last vestige of flux that could be causing the clock to run at low level as it had been noted that attaching a 'scope to the crystal would produce a visible increase in clock level, although this was probably a byproduct of the capacitance of the 'scope probe.

Then, by accident, light started to dawn. In desperation, we had by now adopted the very bad practice of not powering down the computer before making component substitution. Tim removed the clock chip with all power on and replaced it with another, and it all appeared to work. This caused some investigation of the clock power down circuitry and battery backup. However, this was not found to be at fault, and another lucky accident revealed the true cause. Having persuaded the clock to work by plugging in the chip with the power on, the failed condition could be caused by either powering up or by resetting the computer. It was apparent that it was the reset state which caused the problem, and that under this condition the clock chip was being fed with a register combination which not only jammed the clock chip, but made the chip incapable of accepting new data during any subsequent initialisation process, powering down the computer made no difference to the jammed clock as the battery backup on the clock board took over automatically. Once jammed the clock chip remained jammed. In fact the only way to unjam a jammed clock chip was to remove it from the board and then plug it back in (having initialised the port to address the chip).

On the Nascom 2 (not Nascom 1) and the Gemini, the PIO device is reset when the computer is reset and also on power up, causing the PIO to go to an input condition and to float the input/output pins which are left in a high impedance state. Normally the input/output pins float high. By attaching our in line port analyser, which sounds complicated but consists of no more than a CMOS non inverting buffer sensing each port line and lighting a LED when a 'high' appears on the port line, we noted that the port line which addressed the clock

board chip select, CS, did not float high as expected but remained resolutely at about the two volt level. This, despite the inclusion of a 100K pullup resistor on the clock board to ensure the CS line remained high. Taking in to account the loading effect of the high impedance meter used to measure this voltage, the approximate voltage on this line was guessed as being about the 2.5 volt level. Changing the PIO or the buffer made no difference, it remains a mystery where the leakage causing the line to be lower than expected is occurring.

However, accepting that this leakage does occur, and as this voltage is about the threshold of the board chip select buffer, which is a CMOS device with a threshold of about 2.5 volts, we thought that any spurious noise on this line, and there has to be some, was causing the CMOS buffer on the clock board to select the clock chip. This was confirmed with a 'scope looking at the select pin of the clock chip, after the input and power down buffers. A definite short glitch was noted on the select pin during a computer reset sequence. The cause, like the voltage leakage remains unknown. As the output lines of the PIO are directly connected to the clock chip which again is a MOS device, these are also susceptible to noise due their high impedance inputs. This appears to cause the clock chip write input to open during a computer reset sequence and so to write the rubbish on the data lines to the clock chip registers which were being addressed by the rubbish on the clock chip address lines, also directly connected to the PIO. We were relieved to cure the problem by the simple expedient of changing the 100K pullup resistor for a 22K pullup.

Unfortunately, this did not remove all the problems, the clock would now work correctly whilst the computer was left running, however, on power down and subsequent power up the clock was occasionally being corrupted. Having previously eliminated the power down and battery backup, this left the power up condition. This was confirmed as being the cause by powering down the computer with the clock connected, then unplugging the clock and powering the computer up again. Reconnecting the clock with the power on and then reading the clock, the clock worked correctly. The clock circuitry has a 100K pullup resistor in series with a 100nF capacitor to stop the CS line becoming active during power up. 100K and 100nF gives a time constant of 10mS which is the time allowed for the power supply rails to settle. This may be adequate for a switch mode power supply, as the clock at home had shown no trouble in this respect. However, the works computer was fitted with a large conventional power supply. Investigation of the power supply start up revealed that the rails did not become stable for at least 100mS, and during that time the state of the PIO was extremely indeterminate. It was obvious then that the 10mS allowed for stabilisation was inadequate for the power supply in use, and the 100nF capacitor was changed for a 2uF tantalum, giving a 200mS time constant. This cleared the remaining problem.

Both the problems noted above could do with further investigation, why were these problems not noted on the prototypes, and are the conclusions drawn above and the solutions adopted, the correct ones. Our aim at work was to make the clock work, not to conduct an investigation into the behaviour of PIOs and power supplies at reset and power up. Perhaps Gemini may like to make some comment in a future issue.

To sum up then, the Gemini GM822 represents a useful addon which is simple to construct and easily added to the system. A one evening project with a useful end result. Having cured the problems mentioned above, both clocks have behaved well. Taking the cost of the competing clock boards into account, the Gemini GM822 seems to be about par for the course in terms of value for money and facilities provided.

I didn't like the software supplied with the clock so I set to and wrote some new routines. These used large chunks of the originals and in this case I can not credit them as I do not know who wrote them in the first place, however, thank you, whoever you are. The routines are listed elsewhere and are written for use under CP/M only, but are otherwise Nascom/Gemini compatible although I have maintained port 1CH as the clock port. This could easily be changed on reassembly. The most important improvement is that they contain error checking of the input, making it impossible to program wrong data into the clock. However,

there is still room for improvement, as since writing these I have discovered that it is possible for a clock register to become programmed with OFH as a legitimate character, caused by spurious pulses on the PIO lines. Under these circumstances the program goes into a permanent loop, thinking that the clock has just changed time. Some sort of time-out is required, where say, after having had 256 goes at reading the clock and failed, the routine returns the register contents regardless.

Having fitted a real time clock (RTC) some use has to be found to justify the clock. In the circles I move in, a couple of people have already found uses for the clock. Richard has modified his M80 assembler to include the time and date on printed listings, which can be very useful. David has written an elegant little utility called INDEX which gives a CP/M directory listing and allows a one line comment after each program entry. The date being added to the index created at the time of updating. I hope INDEX will see its way round a larger circle soon, although, as it requires the GM822 RTC to be hung on a Z80 PIO at port 1CH, it does make it a dedicated program, and therefore not universally usable. Note that in each instance port 1CH has been used, and it has been mutually agreed amongst ourselves that port 1CH will be the clock port. My own interest has been to hook the clock into a compiled Basic program.

As was mentioned earlier, the MM58174A clock exhibits a peculiar characteristic, in that it gains time if access is made to it over too long a time frame. I do not have the National data sheet to hand, but if I remember correctly, the clock must be accessed within a period of roughly 15mS from reading the first register to reading the last. This means that the access software has to be very fast. On the other hand, in instances where it is not the intention to display the clock on the screen, reading it back at its fastest increment of tenths of seconds, this is of little consequence. The software provided by Gemini certainly fails in this respect in that it wastes a lot of time within the read loop determining whether the clock has changed time, and if not, to convert the incoming nibble to ASCII and to save it in the temporary workspace. The obvious method to correct for this would be to copy the registers 'as is' to the temporary workspace using the tightest possible loop, perhaps using an INI instruction or even an INIR instruction if it could be contrived, then, having copied the registers to the temporary workspace, to sort them out later. I have not given this approach much thought, but it should be possible. The later National MM58174AN chip does not suffer from this problem, but is more expensive.

Making use of the clock to display the time continually..on the screen, whilst a nice idea, is not as easy as it sounds. That is, assuming the computer is to be used for other things apart from displaying the time. It is not difficult to write a program which will continually read the clock and place the result on the screen, Gemini have already provided the clock read routines, and a simple change to the software could be made to cause the routine restart itself instead of making the clock return to the routine which called it. This is a massive waste of time as the processor is 100% wrapped up in reading the clock. There is no time left for other things. As the clock is capable of creating interrupts, one method would be to use the interrupt output connected to the strobe input of the PIO to create an interrupt every time the clock changed state. The fastest interrupt rate on the MM58174 is one every half a second, this could be used to update the clock display every second, whilst causing minimum disruption to processing of other things. As it happens, the interrupt output of the GM822 was deliberately not returned to the strobe line of the PIO, as this could be the cause of some very strange software faults by the creation of undesired interrupts when used with disk systems (which must not be interrupted), or other interrupt driven auxiliaries elsewhere in the system, where the use of the EI and DI instructions could cause other problems. The interrupt output is connected to one of the output bits which means it can only be used when mode 3, the bit masked interrupt mode of the PIO is selected. This mode, whilst more flexible is also more difficult to implement.

EV Computing in Manchester have come up with a novel solution using the Gemini GM812 IVC card, where the video card Z80 is run under a new piece of software, and interrupts are created on the video card. This overcomes all the objections to running the RTC in an interrupt mode on the main Z80, and because their modification includes its own battery backed up RAM as well, allows preset messages to be generated as required. An ingenious solution, and one which may be retrofitted to the IVC card with little difficulty.

One method of display I have yet to investigate is to incorporate the clock routines within the keyboard routine. This is a compromise solution, as the clock would only be displayed whilst the keyboard was being scanned but not actually processing an incoming character. However, as the computer spends the vast majority of its time doing just that, the system should be quite effective. Naturally the clock display would freeze when other processing was taking place, for instance, whilst fetching and putting to disk, where the keyboard is no longer scanned, but I consider these as minor inconveniences, there are not many times in an evening where the processor does not look at the keyboard at least once within a ten second period. The only time I can think of is when assembling a large program, where the processor may be occupied for three or four minutes. I intend to display the time at the top right of the screen, but not to trap the top line so that the time can always be scrolled when, say, listing a program. Again, I do not think this will matter, as the display will be restored within one second of the screen display becoming stable.

My use for the clock has been in a computer logging system written in compiled Basic. Much as I like, and need the practice at machine assembler, I am sufficiently aware of my limitations not to take on a program which requires eight overlay programs, none of which is less than 14K of Basic source. The problem remained, however, of reading the clock from Basic. The clock updates its registers every tenth of a second, and indicates the update by changing the output of the registers from a four bit BCD number to the code OFH. Trying to access the twelve registers via one port by outputting (using the Basic OUT instruction) to create the address, and cause and output strobe and then inputting the data from each register within one tenth of a second is impossible in straight Basic. I also doubt that compiled Basic would complete the task within the time allotted, as the compiler is not renown for producing efficient code. So the obvious approach was a USR type routine.

In Microsoft MBASIC the distinction between the USR and CALL user subroutines is finely drawn, although the CALL function is more appropriate when used compiled. The problem was to decide how to extract the data returned by the clock, when the Microsoft manual glosses over this rather vital point. It is also rather obscure about how to pass information to a machine code subroutine as well, although after careful reading and repeating the words slowly to myself, the technique was finally revealed. The final program was simple and straight forward, although how it works is another matter.

```

1000 TM$="*****"
1010 ADDR% = PEEK(VARPTR(TM$)+1)+256*PEEK(VARPTR(TM$)+2)
1020 CALL TIME(ADDR%) : ' Get the time using the machine code subroutine
1030 MONTH=PEEK(ADDR%): DYWK=PEEK(ADDR%+1): DAY=PEEK(ADDR%+2)
1040 HOUR=PEEK(ADDR%+3): MIN=PEEK(ADDR%+4): SEC=PEEK(ADDR%+5)

```

It all relies on making use of the way the strings are handled within the Basic. Anyone who has tried to investigate how the Microsoft Basic stores strings will have noted that the string space is allocated dynamically, that is, the string is put into a workspace and a three byte pointer is set up to point to the string and to supply the string length. This allows the strings to be shuffled about without having to reserve the maximum space for string manipulations. To save space further, any strings declared in the program, prior to use, are not copied into the string workspace, but are left exactly where they are within the Basic source, and a pointer is set to point inside the Basic source.

In the above case, TM\$ will be the variable space for the machine code clock read routine. All that is required is to pass the address of TM\$ to the machine code routine. When the Microsoft Basic CALL is used, upto three variables may be passed to the machine code routine in HL, DE and BC. If there are more variables, BC is set to the address of a table of variables and that is passed for the machine code routine to pick up. In this instance only the address of TM\$ is required. The VARPTR function returns the address of the string pointer, which holds:

- 1) The string length
- 2) The actual string address, low byte first

Line 1010 returns the actual address of TM\$, ADDR%. Note that this must be an integer variable, as only a 16 bit number can be sent to the machine code routine in HL. Line 1020 CALLs the the machine code routine and ADDR% is passed in HL. The machine code routine places the month, day of week, day, hour, minute and seconds in the first six bytes of TM\$ and lines 1030 and 1040 read these back. I would have thought that as the CALL function within Basic can pass variables to a machine code routine, it ought to be able to pass them back, however I can not find out how to do this, so if anyone knows, drop me a line.

The machine code read subroutine is very simple (refer to the program listing READ ROUTINE FOR GEMINI GM822), and is a stripped down version of my rewrite of the routines that came with the clock. It need only be a read routine as there is no facility for setting the clock within the logging program. Note, however, as I wish to return the variables as complete integers, I use a times 10 routine to convert the separate nibbles into integers. The routine has been assembled at address 0000H, and is intended to be linked into the compiled Basic program. Any line of code within the listing suffixed by an accent (such as CD 0009') means that that absolute address would need to be changed if the program were moved elsewhere. As it stands, the TIME routine can not be used with Basic without the compiler. However, as has already been discussed, strings declared within a Basic program are not moved about by Basic until required, so it could be POKEed into a string declared as the first line of the program within the Basic source. In this way, provided that no Basic lines were inserted prior to the declared string, the addresses would be absolute and the absolute code adjusted to suit. Note that this idea will only work with a specific version of the Microsoft Mbasic, as Microsoft have a tendency to move the source buffer addresses about between different versions; also that this technique makes the first line of the program unlistable, as it will contain some 128 bytes of reserved words, junk, and other nasty things.

So, having got the time from the clock routine, what to do with it? Well, one nice exercise is to display it within a Basic program, in the form say, 14.34 GMT, 21 Sep 1982. Not a difficult job, but shows some of the more useful attributes of string manipulation in Basic. Those who have read through the machine code clock setting routines will have noticed similar routines.

```

2000 MN$="JanFebMarAprMayJunJlyAugSepOctNovDec"
2010 DY$="SunMonTueWedThuFriSat"
2020 IF DAY>9 THEN D$=RIGHT$(STR$(DAY),2)+" " ELSE D$="0"+RIGHT$(STR$(DAY),1)+" "
2030 IF HOUR>9 THEN H$=RIGHT$(STR$(HOUR),2)+"." ELSE H$="0"+RIGHT$(STR$(HOUR),1)+"."
2040 IF MIN>9 THEN MIN$=RIGHT$(STR$(MIN),2) ELSE MIN$="0"+RIGHT$(STR$(MIN),1)
2050 DY$=MID$(DY$,3*DYWK-2,3)+" "
2060 MN$=MID$(MN$,3*MONTH-2,3)+" "
2070 YR$="1982"
2080 DATE$=H$+MIN$+" GMT, "+DY$+MN$+YR$

```

The points to note are the way in which the day and month are picked from the string by calculating the relevant point in the string and then using the MID\$ function. Note also the way the STR\$ function is used within a RIGHT\$ function to truncate the numeric variable and return it as a string. In line 2080 it simply becomes process of adding up the strings.

An off shoot of presenting the date is the ability to use the date in numerical sorts. The obvious method is split a date into its component parts, using the tens and units of the year, the month and the day; so that 21/09/82 would end up as 820921. This would work as each day is a unique number and days increase in ascending order. However, the number sequence is not contiguous which could cause a number of problems, not the least of which is the complicated validation to determine if a date is missing within a sequence of dates, or possibly an extra day having been entered in error. Whilst initially attractive, the method of using the date backwards has a number of disadvantages. Better then that the number sequence be contiguous. This can be achieved by starting at a base date and counting upwards. The only problem then is to decide the base date and how to cope with leap years, etc.

A numeric sequence often used for this is the Julian date. Based on the year nought. It is accurate but is confused by the normal slippage of leap years every four hundred years, also by the abnormal leap year skip which occurs every 4000 years. The next modulo 4000 leap year correction takes place in the year 2000, and the one after that in the year 6000. The one in the year 2000 might be of interest, but the one in the year 6000 will probably only be important to someone with an interest in cryogenics. However, for general purposes, the Julian number has a lot going for it.

To convert an input date to a Julian number the following simple routine may be used, it does assume that the date will be this century.

```

3000 INPUT "Date (DD/MM/YY) ";DT$
3010 DAY=VAL(LEFT$(DT$,2)) : MONTH=VAL(MID$(DT$,4,2)) : YEAR=1900+VAL(RIGHT$(DT$,2))
3020 DT=INT(30.57*MONTH)+INT(365.25*YEAR-395.25+.5)+DAY
3030 IF MONTH >2 THEN IF INT(YEAR/4)=YEAR/4 THEN DT=DT-1 ELSE DT=DT-2
3040 ' DT now equals the Julian number.

```

This is all very good, the conversion back again is also simple, with the addition in this case to convert the number to a string in the same form as the input.

```

4000 Y=INT(DT/365.26)+1
4010 D=DT+INT(395.25-365.25*Y+.5)
4020 IF INT(Y/4)*4=Y THEN D1=1 ELSE D1=2
4030 IF D>91-D1 THEN D=D+D1
4040 MONTH=INT(D/30.57)
4050 DAY=D-INT(30.57*MONTH)
4060 IF M>12 THEN M=1 : Y=Y+1
4070 YEAR=Y-1900
4080 D$=RIGHT$(STR$(DAY),LEN(STR$(DAY))-1) : IF DAY<=9 THEN D$="0"+D$
4090 J$=SPACE$(2) : LSET J$=D$ : DT$=J$+ "/"
4100 M$=RIGHT$(STR$(MONTH),LEN(STR$(MONTH))-1) : IF MONTH<=9 THEN M$="0"+M$
4110 J$=SPACE$(2) : LSET J$=M$ : DT$=DT$+M$+ "/"
4120 Y$=RIGHT$(STR$(YEAR),LEN(STR$(YEAR))-1)
4130 J$=SPACE$(2) : LSET J$=Y$ : DT$=DT$+J$
4140 ' DT$ now equals the date string

```

So having started with clock hardware, I have finished with Julian dates. Time is an interesting subject, you can do so much with it. I only wish there were a program to provide me with about forty hours in a day, then perhaps I would have time to do everything I wish to do and still find time to sleep.

Title DRIVE ROUTINES FOR THE GEMINI GM822 RTC MODULE

*Comment * Adapted from the original routines by
D. R. Hunt

*Z80

GLOBAL RTC

```

        ; Base address of control PIO
        BASE EQU 1CH

        ; CP/M equates
        CONIN EQU 1          ; Console input routine
        CONOUT EQU 2         ; Console output routine
        LININ EQU 10         ; Line input
        CR EQU 0DH           ; Carriage return
        LF EQU 0AH           ; Line feed
        BDOS EQU 0005H        ; BDOS jump vector

        ; CP/M work spaces
        ARGS EQU 0082H        ; Input arguments
        ITHOURS EQU ARG$+0    ; Input hours
        IMINS EQU ARG$+2      ; Input minutes
        IDAY EQU ARG$+4       ; Input day
        IMMONTH EQU ARG$+6     ; Input month
        IDIWK EQU ARG$+9      ; Input day of week
        PDATA EQU BASE        ; Data port for commands
        PADDR EQU BASE+1      ; Data port for commands
        PCDATA EQU BASE+2      ; Control port for data
        PCADDR EQU BASE+3      ; Control port for data

        ; Port assignments
        ALLIN EQU OFFH        ; Control mode
        ALLOW EQU 0             ; All inputs
        INTOUT EQU 10H          ; All outputs
                                ; All out except INT input

        ; Addresses for RTC
        TEST EQU 0
        TMONTH EQU 12
        START EQU 14
        INSTAT EQU 15
        NOSTRB EQU 0E0H
        RDSTRB EQU 0OH
        WRSTRB EQU 40H

        ; No strobe or select
        ; Read strobe on
        ; Write strobe on

```

***** Messages *****

```

        ; **** RTC TIME SET ***
        TIMES: DEFB CR,LF
        DEFB 'Set time in the form:- HH:MM DD:MM.D'
        OD 0A
        0000: 0D 0A 20 20
        0002: 20 20 20 20
        0006: 20 20 20 20
        000A: 20 20 20 20
        000E: 20 20 2A 2A
        0012: 20 20 52 54
        0016: 43 20 54 49
        001A: 4D 45 20 53
        001E: 45 54 20 20
        0022: 2A 2A 2A
        0025: 0D 0D 0A
        0029: 53 65 74 20
        002D: 74 69 6D 65
        0031: 20 69 6E 20
        0035: 74 68 65 20
        0039: 66 6F 72 6D
        003D: 3A 2D 20 48
        0041: 48 3A 4D 4D
        0045: 20 44 44 3A
        0049: 4D 4D 2E 44
        004D: 20 6F 72 20
        0051: 54 20 66 6F
        0055: 72 20 74 69
        0059: 6D 65 2E
        005C: 0D 0A
        005E: 45 6E 74 65
        0062: 72 20 74 69
        0066: 6D 65 20 3F
        006A: 20 00
        006C: 48 69 74 20
        0070: 61 65 79 20
        0074: 6B 65 79 20
        0078: 74 6F 72 73
        007C: 74 61 72 74
        0080: 20 65 6C 6F
        0084: 63 6B 20 00
        0088: 45 72 72 6F
        008C: 72 00
        008D: 0D 0A
        CRLF: DEFB CR,LF,0
        ; Output string '00:00:00.0
        0090: 30 30 3A 30
        0094: 30 3A 30 30
        0098: 2B 30 20 20
        009C: 58 58 58 20
        00A0: 30 30 20 20
        00A3: 58 58 58 20
        00A7: 31 39 38 32
        00AB: 0D 0A 00

```

DRIVE ROUTINES FOR THE GEMINI CM-8222 RTC MODULE M-80 14 Sep 1982 14:48 PAGE 1-2

```

DRIVE ROUTINES FOR THE GEMINI GM822 RTC MODULE M-80 14 Sep 1982 14:48 PAGE 1-5

01B8' 20 FB JR NZ,HMS ; Yes, so loop
01BA' ED A0 LDI HI,HOURS ; Copy across the tenths
01BC' 21 0090' LD PRINT ; Now display it
01BF' CD 01C3' CALL RETP ; Done

01C3' 7E ; Print the string ; Get the byte
01C4' B7 PRINT: LD A,(HL) ; Done?
01C5' C8 OR A
01C6' 23 RETN Z
01C7' 23 INC HL
01C8' 5F LD E,A
01C9' E5 PUSH HL
01CB' 02 LD C,CONOUT ; Save <HL>
01CE' E1 CALL BDOS
01CF' F2 POP HL
18 F2 PRINT

; *****
; ***** Get place in string *****
; *****

PLSTRG: LD A,(HL)
INC HL
01D1' 7E SUB 30H
01D2' 23 CALL TIME10
01D3' D6 30 ; Save the partial product
01D5' CD 01E3' LD B,A
01D8' 47 ADD A,B
01D9' 7E LD A,(HL)
01DA' D6 30 ; Multiply by 2
01DC' 80 CALL TIMES
01DD' CD 01E9' SUB 3
01EO' D6 03 ADD A,B
01E2' C9 RETP ; Add the partial product
01E3' 07 TIME10: RLCA ; Save the partial product
01E4' 47 LD B,A ; Multiply by 2 again
01E5' 07 RLCA ; and again
01E6' 07 RLCA ; Add the partial product
01E7' 80 ADD A,B
01E8' C9 RETP ; Add the partial product

; *****
; ***** Multiplication routines *****
; *****

01E9' 47 TIME3: LD B,A ; Save the partial product
01EA' 07 RLCA ; Multiply by 2
01EB' 80 ADD A,B
01EC' C9 RETP ; Add the partial product

```



```

TITLE READ ROUTINE FOR THE GM822 RTC
*280
GLOBAL TIME

BASE EQU 1CH ; Base address of control PIC

; Port assignments
BDATA EQU BASE ; Data port for commands
FADDR EQU BASE+1 ; Data port for commands
PCDATA EQU BASE+2 ; Control port for data
PCADDR EQU BASE+3 ; Control port for data

; Control instructions
CMODE EQU OFFH ; Control mode
EQU OFFH ; All inputs
EQU 10H ; All out except INT input

; Addresses for MM58174
IMONTH EQU 12
NOSTR EQU 0EOH ; No strobe or select
RDSTR EQU 80H ; Read strobe on

; **** PIO control subroutines ****
; **** Set data port to control mode, all lines input
; INDATA: LD A,CMODE ; Control mode
;          OUT (PCDATA),A ; Set mode word
;          LD A,AJIN ; Set direction
;          OUT (PCADDR),A ; Set direction
;          RET

; Set command port to control mode, all lines input
; INADDR: LD A,CMDR ; Control mode
;          OUT (PCADDR),A ; Set mode
;          LD A,AJIN ; Set direction
;          OUT (PCADDR),A ; Set direction
;          RET

; Set command port to control mode, all lines output
; EXCEPT the INT line (0BAH)
; OADDR: LD A,OFFH ; Ensure all data lines
;         OUT (PADDR),A ; ... will be 1's
;         LD A,CMDR ; Set mode
;         OUT (PCADDR),A ; Set direction
;         LD A,INTOUT ; Set direction
;         OUT (PCADDR),A ; Set direction
;         RET

```

```

READ ROUTINE FOR THE GM822 RTC M-80   11 Oct 1982   10:49   PAGE   1-1

*****
; **** Subroutine to read to the MM58174
; **** Called with:-          ****
; ****          C=PADIR - data port for command  ****
; ****          D=RTC register address   ****
; ****          E=RTC reg.+ Stroke (R or W)  ****
; ****
; **** Read a MM58174 register
; ****
; **** READ:      OUT    (C),D ; Set up address
; ****           OUT    (C),E ; Read strobe on
; ****           IN     A,(PDATA) ; Read a nibble
; ****           OUT    (C),D ; Stroke off
; ****           RETN
; ****
001F' ED 51
0021' ED 59
DB 1C
0023' ED 51
0025' C9
0027'

```

```

READ ROUTINE FOR THE GM622 RTC M-80   11 Oct 1982  10:49   PAGE    1-2
***** ; *****
***** ; ***** Multiply <A> by 10   *****
***** ; *****
TIME10: RLCA      ; Multiply by 2
        LD      B,A      ; Save the partial prod
        RLCA      ; Multiply by 2 again
        RLCA      ; Multiply by 2 again
        ADD     A,B      ; Add the partial prod
        RET
***** ; *****
0054: 07
0055: 47
0056: 07
0057: 07
0058: 80
0059: C9

```

```

; *****
; ***** Routine to read the RTC registers *****
; ***** to work space in RAM at (HL) *****
; *****

0028: CD 0012'          CALL QUADADR      ; Set command port to o/p
        CD 0000'          CALL INDATA       ; Set data port to i/p
002B: CD 007E'          LD   HL,(PTR)
        LD   B,12           LD   B,12           ; 12 registers to read
        LD   HL,TMONTH+NOSTR     ; 10's of months + no strob
        LD   B,12           LD   B,12           ; C = control port address
        LD   HL,TMONTH+RDSTRT    ; 10's of months + read str
        LD   C,PADDR         LD   C,PADDR
        LD   HL,READ          LD   HL,READ
        CALL AND             CALL AND
        LD   OFH             LD   OFH
        CP   OFH             CP   OFH
        JR   Z,RDRTC         JR   Z,RDRTC
        LD   (HL),A           LD   (HL),A
        INC  HL              INC  HL
        DEC  D               DEC  D
        DEC  E               DEC  E
        DJNZ RDRTCL         DJNZ RDRTCL
        RETF
0046: CD 001F'          CALL RDRTCL      ; Loop if more
        LD   HL,WORKSPACE    LD   HL,WORKSPACE
        LD   B,12           LD   B,12           ; Done

```

```
***** Convert nibbles into single byte *****
```

READ ROUTINE FOR THE GM822 RTC M-80 11 Oct 1982

M-80 11 Oct 1982 10:49 PAGE S

No Fatal error(s)

NASCOM PROGRAM LIBRARY

When 80-BUS News absorbed INMC80 we also acquired a quantity of programs from the program library. We now need the valuable space that they are occupying and so here are some offers that you can't refuse!! All of the programs are ones which were sent in by INMC members and are source code listings on A4 paper. In order to dispose of them quickly we are offering them at 'giveaway' prices, but must insist on a minimum program order of £1. As we run out of certain lines we will make up the value of the order with alternative programs. Postage is 30p per UK order and 60p overseas.

These offers hold until January 31, 1983 and after that date the program library will be permanently closed.

NasSys Programs

S2	Reaction Test	OC80-OFAD	0.20
	When the random digit appears, press the same digit on the keyboard. Better than 0.6 seconds is good - how good are you?		
S3	Bouncing Beastie	OC80-OCFC	0.05
	A character bounces around the screen leaving a trail behind it.		
S4	Reverse	OC80-OF99	0.25
	Arrange the randomly generated list of numbers in order in the fewest moves by reversing any number of them at a time.		
S5	Double Mastermind	OC80-OFB9	0.35
	Try and crack the Nascom's four digit code while it tries to crack yours. (It won't let you cheat!)		
S7	JJ	OC80-OEOF	0.20
	Enter the machine code of this before looking to see how it works. Written to demonstrate machine code string handling.		
S8	Unizap	OC80-OED9	0.25
	Recreate the life of the Universe. Take it from the 'big bang' to its 'ultimate destiny' by shooting stars.		
S9	Random Buzz-word Linker	OC80-OF3E	0.20
	Short of jargon? Then use the 'Functional flexibility' of this program to give you 'limited transitional communication' in your 'balanced digital engineering' !!		
S10	Quiz Program (+Italian file)	OC80-ODCB (+ODDO-OFD5)	0.25
	What is Italian for 'please' or English for 'Fratello'? Find out with this quiz. Also additional files (see below).		
S11	Cockney Rhyming Slang (for S10)	ODDO-OFD6	0.10
S12	Currency Types (for S10)	ODDO-OFCO	0.10
S13	Capital Cities (for S10)	ODDO-OFE1	0.10
S14	Ship Game	OC80-OFA4	0.30
	Blow up ships using your torpedoes. Ships come at random speed, distance & direction.		
S15	Hangman	OC80-OFA4	0.25
	Can you guess the word before you are hung? Can add own words.		
S16	Memory Test 1	OFOO-OF6F	0.10
	Tests for addressing faults		
S17	Memory Test 2	OFOO-OF61	0.10
	Tests for bit faults/pattern sensitivity.		
S24	Octal-Hexadecimal Conversion	OC80-OCE8	0.10
S28	Decimal-Hexadecimal Conversion	OC80-OD2B	0.15
S34	Random I-Ching Characters	OC80-OD25	0.10
	Great mystical significance (for those who understand these things).		
S36	Sub-Search	OC80-ODA3	0.20
	Ship traverses screen dropping random depth charges on randomly moving submarines.		

Nasbug Programs

T1	Crash	OC50-0FC9	0.35
	Land the spacecraft. Real time. Can you make NASA proud of you.		
T2	Reaction Test (as S2)	OC50-0FCC	0.25
T3	Bouncing Beastie (as S3)	OD00-0DAB	0.10
T4	Reverse (as S4)	OC60-0F88	0.25
T5	Double Mastermind (as S5)	OC50-0FD5	0.30
T6	Possum	OD00-0D84	0.10
	The hex digits are displayed and then flash one at a time. By pressing any key the character corresponding to the code will be displayed.		
T7	JJ (as S7)	OC50-0DE5	0.20
T9	Random Buzz-words (as S9)	OC50-0F35	0.20
T10	Quiz Program + Italian (as S10)	OC50-0DAA	0.25
T21	TV Test Pattern	OD00-0D19	0.05
	Displays test pattern to set up TV.		
T22	NIM	OD00-0FC4	0.45
	The classic game of NIM.		
T23	Attack	OD00-0E7F	0.20
	Shoot the descending aliens before they get you!		
T24	Octal-Hexadecimal (as S24)	OD00-0DBC	0.15
T26	24 Hour Digital Clock	OD00-0D7B	0.15
T28	Decimal-Hexadecimal (as S28)	OC90-0D07	0.10
T29	Life	OC50-0FA0	0.20
	The ubiquitous simulation of Life, using special characters to give an expanded universe. Very fast.		
T31	Compact Editor	OF70-0FDF	0.05
T32	Carre Chinois	OC50-0F3F	0.40
	We have no idea of what this game is or what it does. But it is totally written and extensively commented in French!		
T33	Lollypop Lady Trainer	OC60-0F95	0.50
	A classic! See if you can get the chickens across the road without them getting run over.		
T34	Random I-Ching (as S34)	OC50-0CF3	0.15
T35	Walled Chase.	OC50-0E27	0.50
	One player chases another, but there are invisible walls in the way which appear when you hit them. Compulsive.		
T36	Sub-Search (as S36)	OE0B-0F31	0.25
T37	Random Display	OC60-0C84	0.05
	Random patterns of asterisks.		
T38	Submarines	OC60-0E50	0.40
	Hit the randomly displayed submarines with your steerable depth charges.		
T39	Burst The Balloon	OC50-0E02	0.50
	Burst The Balloon		
	Shoot the balloons as they make their way up the screen at the mercy of random breezes. Frustrating.		

Nascom 8K BASIC Programs

B1	Hello	0.15
	Is your problem health, sex (yes please), money or your job? Let this fun program make some suggestions.	
B2	Russian Roulette	0.15
	Will you or your Nascom survive as you take it turns to fire a revolver at yourselves?	
B4	Cubist Art	0.05
	Impress the neighbours with the artistic capability of your Nascom. Sort of animated wallpaper. (Graphics ROM required.)	
B6	Calender	0.05
	Enter the year required (between 1601 and 2399) and let the Nascom calculate and display the calender for that year.	

- | | | |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| B8 | Eliza | 0.35 |
| | With this program loaded, your Nascom is especially trained in psychoanalysis. What are your particular hangups? | |
| B9 | Camel | 0.25 |
| | You have stolen the priceless idol belonging to a tribe of knock-kneed pygmies. They want it back and are in hot pursuit. Can you reach safety before the pygmies (or wild Neringi Berbers) catch you? | |
| B10 | Comrade X | 0.50 |
| | You are the premier of the communist island of Niatrib. You must decide your country's budget, harvest and economic strategy. You have 8 years in office, can you survive before the peasants revolt or you are exiled (or worse)! | |

Please indicate clearly what programs you want, e.g. S3, T3 or B3 along with some possible alternatives. Add up the program costs, plus the postage charge, and send cheques, money orders, postal orders, payable to '80-BUS News' to:

Program Offers, Interface Data Ltd., Oakfield Corner, Sycamore Rd.,
Amersham, Bucks. HP6 5EQ.

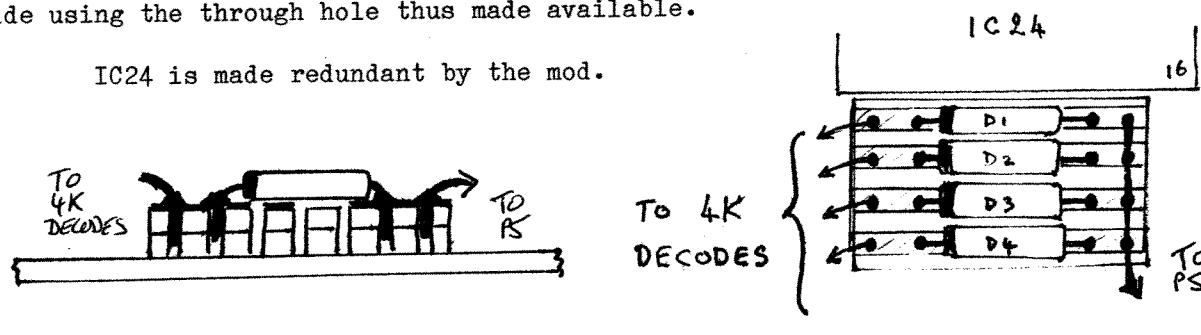
Modifying Nascom RAM A board for 2732 EPROMs

John Harrison

I decided to modify my system to take 4K EPROMs, and having read the two articles in INMC80-5 I chose the RAM board as an easier mod. than the main board. In practice, it turned out to be rather more complicated by the time I had it working.

The published scheme covered 2K and 4K chips, but the 4K version will not work. The problem with taking the 4K decodes direct to the chip selects (p20) instead of using them to enable IC24, is that P5 also drives the data highway buffer, IC26. There are two ways around this: P5 can be driven with 4 wire ORed 4K decodes and IC24 fed with A13 and A12 and used to drive the chip select lines, i.e. the 2K scheme but bigger. Alternatively the decodes can be taken direct to the chip select lines, and diode ORed to P5. The advantage of this approach is that it gives freedom to choose any set of addresses, whereas the other method constrains the 4 blocks used to have different values of A12/A13, e.g. to be 4 contiguous blocks. With 32K of RAM and a toolkit in 2 x 1K EPROMs at B000 I could not accept this constraint.

The problem then was where to lose 4 diodes, neatly. After some thought, I mounted them on a minimally sized piece of 0.1" Verostrip which just fitted between IC24 and the decode pads. I araldited this copper side up onto the board which could just be done without covering any tracks. To allow room for the tails of the diode leads, I used an extra thickness of Veroboard as a spacer. The rest of the modification is as the original one except that I cut the tracks to IC27 pins 19 and 21 on the non component side. This allowed the links to be added also on that side using the through hole thus made available.



The occasion may arise when you want to use a Gemini Galaxy (or Nascom + IVC) as a terminal on another computer system. Some communications programs exist for this, but they often include a host of non-essential features, and do not always do exactly what you require. One program I have used and can recommend is Ward Christensen's MODEM7. I have used this successfully in conjunction with a 300 Baud modem for exchanging files over the PSTN (Public Switched Telephone Network). Another great point in it's favour is that it is Public Domain software and is available on a CP/M user group disk (vol. 47) in source form and at low cost.

However if the computer to which you wish to communicate is close at hand, then the two machines are likely to be connected together directly via the RS232 serial interfaces. In this environment (with no intervening modems) high baud rates can be used, (eg 9600 baud), but problems will immediately be encountered as a result. This is due to the fact that while the IVC is scrolling the screen in response to a line-feed character, (a process which takes a few milliseconds), it can't respond to any commands, although it will accept and store any characters sent to it. As a result a conventional "echo" program would miss incoming characters while waiting for the IVC to respond to a "keyboard poll" request. (At 9600 baud the characters can come at a rate of approximately one per millisecond).

A hardware "handshake" on every character received over the RS232 interface would be a way around this problem, ensuring that no characters are sent unless the Galaxy is ready to accept them, but if you're connected into anything other than a similar CP/M micro it may be impossible to implement. But luckily on the Galaxy it is a relatively simple task to program round the problem, and this is illustrated in the listing of REMOTE below. REMOTE is a "bare bones" program that will turn your intelligent Galaxy into a dumb terminal. I have used it to turn a Galaxy into a 9600 baud terminal on a larger minicomputer system.

The program uses a first-in first-out buffer (FIFO) to get round the problem mentioned above. It polls the incoming UART register at every available opportunity, and if a character is found there, it is transferred to the FIFO. The main program loop checks the keyboard for a character, and if one is found it is sent out via the UART. It then checks to see if there is any data in the FIFO, and if there is it transfers the next character from there to the IVC. The loop then repeats.

This program can be used as the basis for a more comprehensive package. For example I have a version that performs the following:

If Control/I is typed on the keyboard then the FIFO pointers are not zeroed once the characters have been displayed. (i.e. the incoming characters are stored in the buffer and not discarded.)

If Control/D is typed the FIFO pointers are zeroed and the Control/I ("Insert") mode is switched off.

If Control/W is typed then the current contents of the FIFO buffer are appended to a previously specified disk file and the FIFO pointers are then zeroed.

If Control/E is typed then the output file is closed and the program returns to the CP/M command level.

Once again these extensions are fairly straightforward to add, and enable the user to keep a selective record of the session on the terminal. The "bare bones" are here, flesh them out to suit your requirements!

```

.*80
title Remote terminal emulation
;
; Remote terminal Emulation for GEC 4065.
;

escape equ 1bh ; Service the serial port
          ; See if IVC ready
          ; af
          ; Service the serial port
          ; See if IVC ready
          ; in
          ; pollu
          ; a,(ivcs)
          ; rrca
          ; c,pvO
          ; af
          ; Recover character
          ; out
          ; (ivcd),a
          ; Send to IVC
          ; ret

          ; Get a character from the keyboard
          ; Returns NZ if a character is read
          ; const: ld   a,escape ; Send ESC "k"
          ;         call putvid
          ;         ld   a,'k'
          ;         call putvid
          ;         call getvid ; Read reply
          ;         or    a
          ;         ret   z           ; return if none
          ;         call a,escape ; Send ESC "k" to get it
          ;         call putvid
          ;         ld   a,'K'
          ;         call putvid
          ;         call getvid
          ;         cp   'C' and 1fh ; ^C?
          ;         ret   nz           ; No, return with NZ set
          ;         jp    o             ; Yes, abort & reboot
          ; **** MAIN LOOP *****
          ; Uses a FIFO buffer to handle the data
          ; DE = Input pointer HL = Output pointer
          ; start: ld   sp,stack
          ;         ld   a,(7)      ; Get buffer limit
          ;         dec  a           ; (Make sure we don't clobber BDOS)
          ;         ld   (limit),a   ; set the buffer limit
          ;         loopO: ld   de,buffer
          ;                  hl,buffer
          ;         loop1: call const
          ;                  nz,serout
          ;                  push hl
          ;                  or    a
          ;                  abc hl,de
          ;                  pop hl
          ;                  jr   z,loopO
          ;                  ld   a,(hl)
          ;                  inc hl
          ;                  call putvid
          ;                  jr   loop1
          ; **** IVC drivers *****
          ; GETVID - gets a character from the IVC.
          ; PUTVID - sends character in <A> to the IVC.
          ; Both poll the Serial Port while waiting.

getvid: call pollu ; Check for incoming character
        in a,(ivcs) ; See if IVC ready
        rla
        jr c,getvid
        in a,(ivcd) ; Loop if busy
        ret

def: 32
stack: $

buffer equ $

end

```

RANDOM RUMOURS (& TRUTHS)**S. Monger**

I understand that a copy of the latest 'Gemini MultiBoard' leaflet went out with each of the last issues of 80-BUS, and that this has resulted in several comments along the lines of 'What has it to do with me, I'm a NASCOM owner'.....stunned silence.....well, I now wonder if we (at 80-BUS News) should pack up and go home, or if we really have omitted stating the 'obvious' (to us) in this rag. Just in case, here comes an attempt at an explanation :

Once upon a time there was a company called Nascom, producing a micro imaginatively entitled the 'Nascom 1' (N1). Now this micro took the UK market by storm, and lots and lots of them were sold. Then Nascom (NM) decided to expand the N1, and defined the Nasbus system. A memory and an I/O board were produced to fit this bus, and then the more powerful Nascom 2.

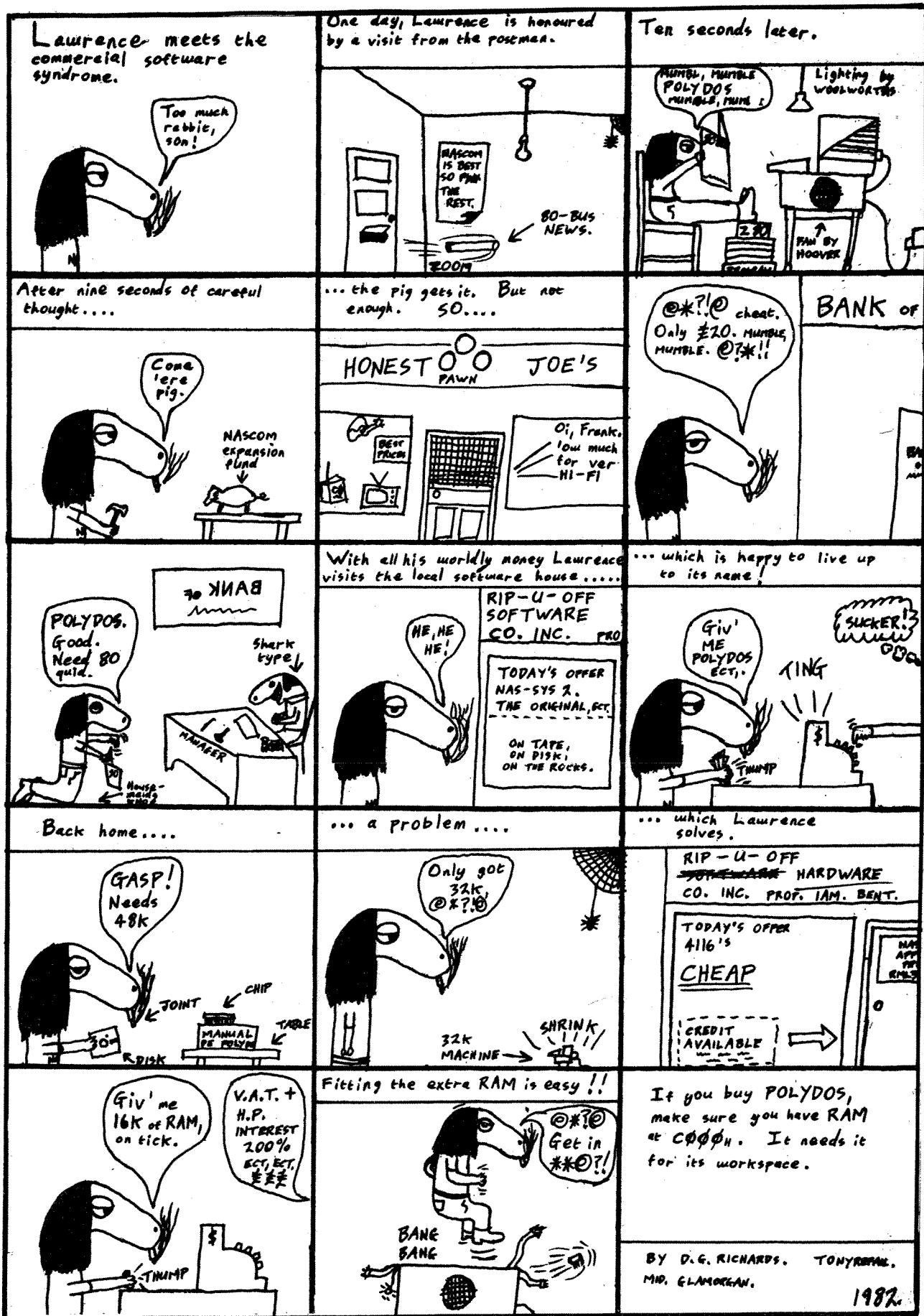
Unfortunately, for a number of reasons, NM went into receivership. Under receivership there was little advertising or production done, as a buyer was sought. Consequently the market waned considerably, and a number of dealers (including one owned by John Marshall (JM), the ex-MD of NM) got together to continue advertising NM products and to promote various 'Nasbus' add-ons of their own. Out of the success of this JM formed Gemini. Unfortunately, as this was occurring, the receiver took a dim view of the dealers using the word 'Nasbus' and started flinging writs around. So Gemini sat down, looked at the somewhat ambiguous definition of Nasbus, redefined and cleared up certain points, and renamed it as 80-BUS, a name chosen not only because of 'Z80' and the '80-way' bus, but also because it is a more generalised name, appropriate for other manufacturers to use as opposed to something like 'Gem-bus'. The 80-BUS spec. was subsequently published in INMC80, issue 4, the predecessor to this rag.

After a year in the receivers hands, NM was bought by Lucas Logic (Spring '81). Then, in my opinion, they proceeded to do 'not a lot' and have never (to date) acknowledged the existance of Gemini. Meanwhile Gemini had produced quite a number of 80-BUS (i.e. Nasbus compatible) products, which sold in large numbers to NM owners, and had also by now produced their own CPU and video cards. This made the Gemini range complete within itself, as well as all expansion cards remaining compatible with NM. With few products coming from NM, the INMC80 magazine found its subscription declining and little to write about and so it moved towards supporting 'the bus' (80-BUS and Nasbus) as a whole, instead of dedicating itself to the one manufacturer's products. In January '82, 80-BUS News was created, absorbing INMC80.

So, having wandered slightly off track, I hope that it is now obvious to all that Gemini 80-BUS expansion products shown in their catalogue, along with those from EV Computing, Arfon, IO Research and Microcode also shown in the catalogue, are in fact usable in expanding Nascom 1, 2 and 3 systems.

Well, having waffled on about that for so long I now have little room left for my usual tripe. I was going to tell you that Mike York has now implemented the UCSD system on the Nascom, and is currently doing it for the Gemini. Not that I know anything about UCSD, other than the fact that it has the usual flavour of nutty but avid followers, and that it is an alternative to the CP/M operating system. I was also going to tell you that Mike Ayres (ex. Sales Manager of Tandy) now holds the same title with Lucas Logic (Nascom) - ex. sales manager. Furthermore, I was going to mention that Gemini Winchester units (mentioned last iss.) are now a reality (only for MultiBoard/ Micropolis & Galaxy systems at the moment, although they will be producing alternative software - write to them saying what system YOU want to add one to). I was also considering mentioning the availability of a 256K RAM board from MAP Systems (and don't expect it to remain the only large RAM board), as well as the fact that Gemini are about to release a new CP/M BIOS containing virtual disk support, screen dump capability, and stand-alone terminal mode.

I might even have mentioned CP/M 3 - but there just isn't room!!



80-BUS NEWS SERVICESBACK ISSUES

INMC COMPILATION ISSUE ('Best of' INMC Issues 1 to 7)	£2.50
INMCO80 ISSUES 1 to 5	£1.50 each
80-BUS NEWS ISSUES 1 to 3	£1.50 each

P&P Please add 25p (40p overseas) for the first issue and 20p (25p overseas) for each additional one.

SPECIAL PURCHASE

BASIC PROGRAMMING FOR THE NASCOM (including P&P)	£3.50
--------------------------------------------------	-------

COMING SOON

NASCOM BASIC DISASSEMBLED — a complete disassembly of the Nascom BASIC ROM including a description of the operation and use of all major subroutines. Send for details.

Please send all orders for any of the above items to:

80-BUS NEWS, Interface Data Ltd., Oakfield Corner, Amersham, Bucks. HP6 5EQ.

PARKSTONE ELECTRICS

NASCOM
AUTHORISED
DEALER

GEMINI PRODUCTS
ALSO STOCKED

Computer Systems Built
for Special Applications

Personal PEARL
now available
on NASCOM

Create all your business
programs for less than £200

PARKSTONE
ELECTRICS
18, Station Road,
Lower Parkstone,
Poole, Dorset, BH14 8UB.
Tel: Parkstone (0202) 746555

UCSD p-system for Nascom

The renowned UCSD p-system has now been adapted for Nascom microcomputers with virtually any combination of Nascom or Gemini drives, 5" or 8".

A complete operating system, with screen editor, macro assembler, optional Pascal, BASIC and FORTRAN, Native Code Generator and TurtleGraphics, there is a comprehensive range of business and applications software available.

Basic p-system (includes screen editor and macro assembler).....	£225
Compilers:	
Pascal.....	£140
FORTRAN.....	£195
BASIC.....	£125

Add VAT @ 15%.

Gemini/Galaxy versions soon.

Large SAE to: Mike York, 9 Rosehill Road,
LONDON SW18 2NY (Tel. 01 874 6244) for
further details.

HENRY'S INCREDIBLE CP/M UTILITIES DISK.

All the things you ever wanted: The disk cataloguing and file dating suites. File compare and string and byte search utilities. Text file compression (and it's complementary expansion) programs. Universal disk repair utility (works on Nascom and the Gemini SD, DD and QD formats). Revised and correctly working SUBMIT.COM with interactive line input (no more XXX.SUB files unless you want to). The revised DRI PIP.COM including all the official fixes published a couple of months back. And many more. All are standard CP/M utilities, not system dependant. Available in Nascom and all Gemini formats. The price of this gift? A mere £15.00 + VAT (£17.50 + VAT in Gemini SD format as it's too big to fit on one disk.)

Richard Beal's SYS overlay BIOSes for Gemini disk systems. SYSN7 the latest and last for Nascom/Gemini G805 systems. SYSB11 universal version for both Nascom/Gemini G809 and Gemini Multiboard systems (not Micropolis drive versions). £10.00 + VAT. Please state version and system configuration required.

Maths pack double precision for Nascom Basic	£13.00 + VAT
Maths pack handler for Nascom Basic	£9.95 + VAT
Command extender for Nascom Basic	£9.95 + VAT

HENRY'S RADIO

404, Edgware Road,
London NW1 1ED

phone 01-402 6822

E & OE

256Kb RAM CARD FOR NASCOM/GEMINI 80 BUS SYSTEMS

8" x 8" Plug in card with memory arranged in four 64K blocks. Buy it now as a standard 64K ram card and expand it later by simply adding more ram chips. Versatile page mode options or more extensive memory mapping facilities. Operation at 4MHz without wait states. Software support available for CP/M virtual disk — Speed up those programs that access the disk excessively.

NOW AVAILABLE IN KIT FORM

Bare PCB with full instructions	£42.50
64K Kit complete	£105
64K Card fully built and tested	£150
64K Expansion kits P.O.A.	

All prices are exclusive of VAT. Add £1.50 for carriage and packing.

COMING SOON

Low cost 80 column video card for Nascom.

MAP 80 Systems. Ltd.

333 Garret Lane, London SW18
Tel: 01-874 2691



Gemini Microcomputers Ltd

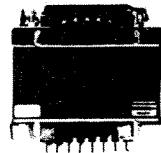
GEMINI 'SUPERMUM AND 'HYPERMUM'

GM810K – COMBINED PSU/MOTHERBOARD

Suitable for Nascom 2 and Gemini MultiBoard systems, the Gemini GM810K 'HYPERMUM' is a 12" x 8" board, providing a powerful 5A PSU in conjunction with an 8 slot motherboard. The motherboard incorporates full bus daisy chaining and an extensive ground plane to reduce electrical noise in the system to an absolute minimum. The power supply outputs are:

+12V at 1A	+5V at 5A
-5V at 0.5A	-12V at 0.5A

The GM810K 'Hypermum' kit is £69.50 + VAT



GM806AK – COMBINED BUFFER BOARD/MOTHERBOARD

Suitable for use in expanding Nascom 1 systems, the Gemini GM806AK is a 12" x 8" board, providing a buffer board for the Nascom 1 in conjunction with a 5 slot Nasbus motherboard. The board mounts parallel with the Nascom and includes synchronised reset circuitry and selectable 'Reset jump' addresses. The motherboard incorporates extensive ground planing.

The GM806AK 'Supermum' kit is £49.50 + VAT

These products are available from the MicroValue Group, and other Gemini dealers.

AMERSHAM COMPUTER CENTRE

80-BUS HARDWARE

GM811	Z80A CPU Board	£125.00
GM813	Z80A CPU +64K RAM Board	£225.00
GM812	Z80A Video Controller Board	£125.00
GM802K	16KRAM Kit (64k Board)	£ 80.00
GM802	64K Dynamic RAM Board	£125.00
GM809	Floppy Disk Controller	£125.00
GM803K	Eeprom/Rom Board (Kit)	£ 55.00
GM816	I/O Board (RTC, CTC, 3*PIO)	£125.00
RB32KC	32K Cmos Batt/Backed RAM	£170.00
IO824	8ch - 8bit A/D Board	£120.00
PLUTO	High-Res Colour Graphics Board, 2 Screen Memories (640h*288v) 3 bit Pixels	£399.00
BABY PLUTO	(320h*288v) 3 bit Pixels	£299.00
NAS AVC	Colour Graphics Board 80col (320h*256v) Pixels	£185.00
WT625	Teletext Colour Graphics Board, 40 col* 24 rows	£136.00
WT910	Nasbus Sound Board	£ 49.50
NAS I/O	Kit Not populated	£ 45.00
GM810	8 Slot Motherboard 5A PSU	£ 69.50

MONITORS

9" AVT	High Res,Green or Amber	£ 99.00
12"GEM	High Res,Green or Amber	£130.00
14"BMC-1401	Colour Monitor,R G B (ideal for baby Pluto)	£299.00
14"HANTAREX	Hi-Res Colour,R G B This unit is one of the best monitors on the market today.Ideal for PLUTO.	£600.00

PRINTERS

EPSON MX80T/3	Dot/Mat/Tractor	£349.00
EPSON MX80FT/3	Dot/Mat/Friction/Trac	£389.00
EPSON MX100T/3	132col (as above)	£499.00
NEC PC8023BC	Dot/Mat/Friction/Trac	£389.00
SEIKOSHA GP100A	Dot/Mat/Tractor	£215.00
SMITH-CORONA TPI	Daisywheel/12cps	£485.00

NASCOM SOFTWARE TAPES

LEVEL 9	Extension Basic Tape	£ 15.00
CCsoft	Nas-Graphpac Tape	£ 20.00
NASCOM	Pascal Compiler Tape	£ 45.00
SIGMA	Zeap 2.0 Assembler Tape	£ 20.00
GEMINI	Nas-Dis/Debug Tape	£ 20.00
L-SOFT	Logic Soft Relocater Tape	£ 13.00

LEVEL 9 NASCOM GAMES TAPES

5 Games	(Gunner/Wumpus etc)	£ 6.00
Missile Defence		£ 8.00
Asteroids		£ 8.00
Space Invasion		£ 7.00
Bomber		£ 5.00
Fantasy		£ 6.00
Nightmare Pork		£ 5.00
Colossal Adventure (32K)		£ 10.00

NASCOM FIRMWARE

LEVEL 9	Extension Basic (4*2708)	£ 25.00
NASCOM	Pascal Compiler	£ 75.00
POLYDATA	Polytext Text Editor	£ 35.00
GEMINI	Naspen Text Editor	£ 20.00
GEMINI	Nas-Sys 3 (2708/Nascom 1)	£ 20.00
GEMINI	Nas-Sys 3 (2716/Nascom 2)	£ 20.00
GEMINI	Nas Dis/Debug (4*2708)	£ 30.00
GEMINI	Imprint (For Imp Printer)	£ 20.00
SIGMA	Zeap 2.0 (2716)	£ 37.50
BITS & PC's	Programmers Aid (N/Sys1)	£ 20.00
BITS & PC's	Programmers Aid (N/Sys3)	£ 20.00

DISK OPERATING SYSTEMS FOR NASCOM

POLYDOS 1	for use with Gemini GM805	£ 90.00
POLYDOS 2	for use with GM815 + GM809	£ 90.00
CP/M 2.2	for use with GM815 + GM809	£100.00

GEMINI RPM/CPM SOFTWARE

COMAL 80	Stuctured Basic	£100.00
* GEM PEN	Text Editor / Formatter	£ 45.00
* GEM ZAP	Assembler(screen editing)	£ 45.00
* GEM DEBUG	Debug/Disassembler	£ 30.00
COPY SB	Superbrain to Gemini (DDDS)	£ 30.00
LIST/REPAIR	Recovers Lost Data Etc	£ 25.00
DATAFLOW	Information Processor	£125.00
* G BASIC	Graphics Basic (IVC)	£ 25.00
GEM GRAPHPAC	Links to Mbasic (IVC)	£ 25.00
COM-PAS	Pascal.Generates M/Code	£120.00
QUIBS	Integrated Business Package	£400.00
CP/M 2.2	Disk Operating System	£ 90.00

* Tape also available.

When ordering disks please specify the format.

MEDIA

	EACH	BOX(10)
SCOTCH C10	Cassettes	£ .60 £ 6.00
DYSAN 1042D	D/S disk (M or P)	£ 5.00 £43.00
GEMINI 2D	D/S disk (M or P)	£ 3.25 £30.00
DYSAN 1041D	S/S disk (P)	£ 4.75 £40.00
WABASH 1D	D/S disk (P)	£ 3.20 £30.00
(M)	Suitable for Micropolis Drives	
(P)	Suitable for Pertec Drives	

80-BUS SYSTEMS

QUANTUM	QM 2000 System (2.4M Bytes)	£2250.00
GEMINI	Galaxy 1 System (800K Bytes)	£1450.00
NASCOM 3	48K with Nas Sys 3 & Graphics	£ 499.00
*GEMINI	GM815 1 disk system (single)	£ 325.00
*GEMINI	GM815 2 disk system (dual)	£ 550.00
GEMINI	GM805 1 disk system (single)	£ 375.00
GEMINI	GM805 2 disk system (dual)	£ 580.00

* requires GM809 controller board

ORDERS ACCEPTED BY MAIL AND PHONE.
ACCESS AND VISA CARD HOLDERS WELCOME.

Prices subject to the addition of relevant VAT charge + P&P



WE ARE OPEN MONDAY TO SATURDAY 9-5.30. PERSONAL CALLERS WELCOME.

Amersham Computer Centre Ltd., Oakfield Corner, Sycamore Road, Amersham,
Buckinghamshire, HP6 6SU. Tel: 02403 22307. Telex 837788

