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

£1.50

The Magazine for NASCOM & GEMINI USERS

November—December 1982. **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

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 flexibilty 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 annonimous, 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 $\overline{STROBE}$ signal and handshaking provided by either $\overline{ACKNLG}$ 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 stablise and then asserting $\overline{STROBE}$ for a minimum of 0.5 microsec. Apart from $\overline{ACKNLG}$, the only other signals (on the printer I/F) not used are $\overline{SLCT\ IN}$ 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:£FF;0;7;3. £FF 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 exept for the second byte. This is £F8 (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 £38 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.

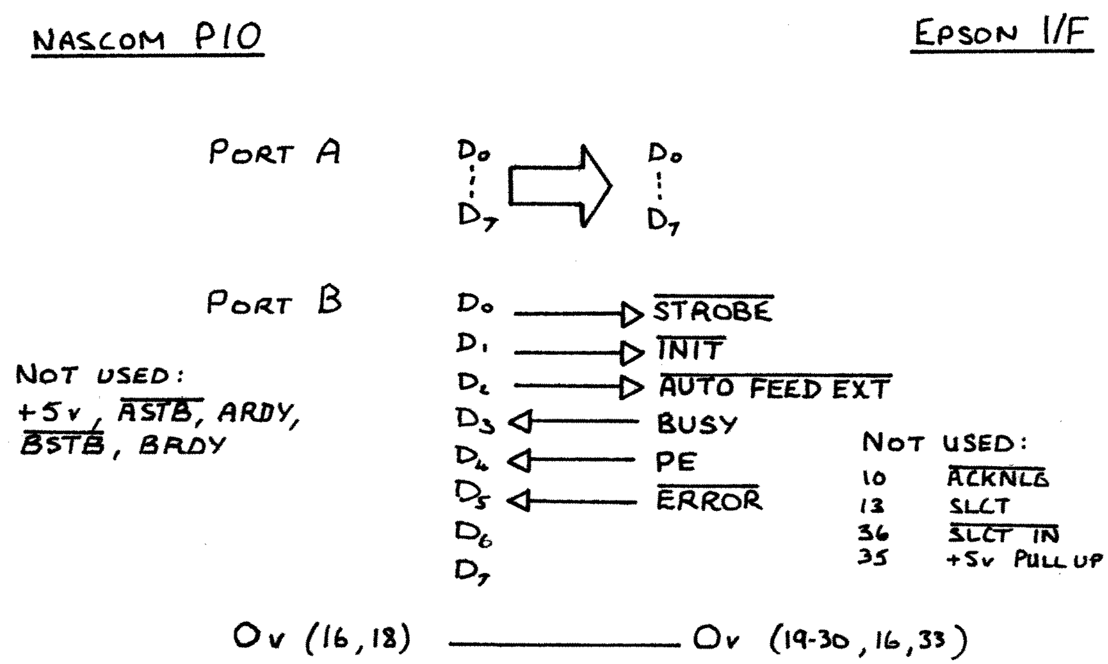


Table 1: Nascom 2 PIO / Epson MX80 connections.

PIO

1	B5
2	B4
3	B6
4	B3
5	B7
6	B2
7	ARDY
8	B1
9	BSTB
10	B0
11	ASTB
12	BRDY
13	A0
14	NC
15	A1
16	0v
17	A2
18	0v
19	A3
20	+5v
21	A4
22	+5v
23	A5
24	A7
25	A6
26	NC

CABLE

PIO		EPSON	
B5	1	32	ERROR
B4	2	12	PE
-	3	-	-
B3	4	11	BUSY
-	5	-	-
B2	6	14	AUTO FEED EXT
-	7	-	-
B1	8	31	INIT
-	9	-	-
B0	10	1	STROBE
-	11	-	-
-	12	-	-
A0	13	2	D_0
-	14	-	-
A1	15	3	D_1
0v	16	All 0v pins	
A2	17	4	D_2
0v	18	All 0v pins	
A3	19	5	D_3
-	20	-	-
A4	21	6	D_4
-	22	-	-
A5	23	7	D_5
A7	24	9	D_7
A6	25	8	D_6
-	26	-	-

EPSON

1	STROBE
2	D_0
3	D_1
4	D_2
5	D_3
7	D_4
7	D_5
8	D_6
9	D_7
10	ACKNLG
11	BUSY
12	PE
13	SLCT
14	AUTO FEED EXT
15	NC
16	0v
17	CHASSIS-GND
18	NC
19	\
to	! (Twisted
30	/ earth returns)
31	INIT
32	ERROR
33	0v
34	NC
35	+5v Pull up
36	SLCT IN

ZEAP Z80 Assembler - Source Listing

```
                ;-----------------------------------------
                ;
                ;          EPRINT V1.0 19/7/82
                ;
                ; Epson (Centronics I/F) initialisation
                ; and driver routines for Nascom 2. These
                ; are fully relocatable use NAS-SYS and
                ; may be run in ROM.
                ;
                ;       Copyright (c) COMPWARE 1982 (MRH)
                ;-----------------------------------------
                ;
                ;
8000                   ORG  £8000              ;\Nominal assembly ad.
                ;
                ;+++++++++++++++++++++
                ;Simple driver routine
8000 D702       DRIVER RCAL SETUP              ;\Execute here
8002 DF5B              SCAL MRET               ;\Return to monitor
                ;
                ;+++++++++++++++++++++
                ;
                ; INITIALISATION SUBROUTINE:
                ;
                ;
                ;                              ;\Find address of
8004 D700       SETUP  RCAL CALADD             ;'SETUP' and use it
8006 E1         CALADD POP  HL                 ;to calculate the
8007 119901            LD   DE,OUTCHA-CALADD   ;actual address of
800A 19                ADD  HL,DE              ;'OUTCHA'
800B 1603              LD   D,3                ;\Set D for AUTO FEED
800D D5                PUSH DE                 ;\Save
800E E5                PUSH HL                 ;registers
                ;
                ;
800F EF                RST  PRS                ;\Ask if using ZEAP
8010 5A656170          DEFM /Zeap? /
     3F20
8016 00                DEFB 0
8017 D748              RCAL YESNO              ;\Get answer (C = no)
8019 3809              JR   C,NEXQ
801B E1                POP  HL
801C D1                POP  DE                 ;\If answer yes set
801D 160F              LD   D,£F               ;D for non AUTO FEED
801F D5                PUSH DE
8020 E5                PUSH HL
8021 22050F            LD   (ZEAPAD),HL        ;and set ZEAP O/P addr
8024 EF         NEXQ   RST  PRS                ;to 'OUTCHA'
8025 4E61732D          DEFM /Nas-pen? /        ;\Ask if using NAS-PEN
     70656E3F
     20
802E 00                DEFB 0
802F D730              RCAL YESNO
8031 380A              JR   C,APENUQ
8033 E1                POP  HL
8034 221E10            LD   (PENADD),HL        ;\If yes, then change
8037 3EC3              LD   A,£C3              ;the seriel O/P call
8039 321D10            LD   (PENADD-1),A       ;to 'JP OUTCHA'
803C E5                PUSH HL
803D EF         APENUQ RST  PRS
803E 536B6970          DEFM /Skip over perforation? /
```

```
     206F7665
     72207065
     72666F72
     6174696F
     6E3F20
8055 00              DEFB O
8056 D709            RCAL YESNO        ;\Ask if printer to
8058 E1              POP  HL           ;skip last inch of
8059 D1              POP  DE           ;page
805A C659            ADD  A,"Y         ;\Restore ASCII value
805C 5F              LD   E,A          ;\Set E to "Y or "N
805D D5              PUSH DE
805E E5              PUSH HL
805F 1819            JR   PENULQ
             ;
             ; GET Y/N ANSWER
             ;
8061 DF7B    YESNO   SCAL BLINK        ;\Blink cursor for I/P
8063 30FC            JR   NC,YESNO     ;\Loop 'til keypress
8065 FE59            CP   "Y
8067 2806            JR   Z,ANSOK
8069 FE4E            CP   "N
806B 2802            JR   Z,ANSOK
806D 18F2            JR   YESNO        ;\Loop if not "Y or "N
             ;
806F F7      ANSOK   RST  ROUT         ;\Output answer and CR
8070 F5              PUSH AF
8071 3E0D            LD   A,CR
8073 F7              RST  ROUT
8074 F1              POP  AF
8075 D659            SUB  "Y           ;\Generate carry if "N
8077 C9              RET
             ;
             ;
8078 18E7    DUMCAL  JR   YESNO        ;\Dummy call point
             ;                         ;since RCAL can't
             ;                         ;reach
             ;
807A EF      PENULQ  RST  PRS
807B 57686174        DEFM /What page length (HEX.)? /
     20706167
     65206C65
     6E677468
     20284845
     582E293F
     20
8094 0D00            DEFB CR,0         ;\Ask for page length
8096 DF63            SCAL INLIN        ;\Input line via NAS.
8098 DF64            SCAL NUM          ;\Convert value (NAS.)
809A 38DE            JR   C,PENULQ     ;\Repeat if error
809C 2A210C          LD   HL,(NUMV)    ;\Retrieve number in L
809F 4D              LD   C,L          ;and save it in C
80A0 0638            LD   B,"8         ;\Set B for paper end
80A2 E1              POP  HL           ;detector off
80A3 D1              POP  DE
80A4 C5              PUSH BC
80A5 D5              PUSH DE
80A6 E5              PUSH HL
             ;
80A7 EF              RST  PRS
80A8 53657420        DEFM /Set paper end detector on? /
     70617065
     7220656E
     64206465
     74656374
     6F72206F
     6E3F20
80C3 00              DEFB O
80C4 D79B            RCAL YESNO        ;\Ask if to set det'r
80C6 3808            JR   C,LASQ       ;on
80C8 E1              POP  HL
80C9 D1              POP  DE
80CA C1              POP  BC
80CB 0639            LD   B,"9         ;\If "Y change B to
80CD C5              PUSH BC           ;detector on
80CE D5              PUSH DE
80CF E5              PUSH HL
             ;
80D0 EF      LASQ    RST  PRS
80D1 53657420        DEFM /Set as user output routine? /
     61732075
     73657220
     6F757470
     75742072
     6F757469
     6E653F20
80ED 00              DEFB O
80EE D788            RCAL DUMCAL
80F0 E1              POP  HL
80F1 3803            JR   C,NOSET      ;\Set as USR O/P if Y
80F3 22780C          LD   (USROUT),HL  ;\(Basic/NAS-DIS)
             ;
             ;  (N.B. User routine must be turned on
             ;   by the U command before it is active)
             ;
             ;
             ; SET UP PIO FOR PRINTER INTERFACE
             ;
             ; SET UP PORT B
             ;
80F6 F3      NOSET   DI                ;\Disable CPU interrupts
             ;
80F7 3EFF            LD   A,-1         ;\Select 'CONTROL' (BIT)
80F9 D307            OUT  (PBCONT),A   ;mode
80FB 3EF8            LD   A,£F8        ;\1111 1000 (1=I/P 0=O/P)
80FD D307            OUT  (PBCONT),A   ;Allocate bits IN or OUT
80FF 3E07            LD   A,7          ;\Interrupt control word
8101 D307            OUT  (PBCONT),A   ;(disable interrupts)
8103 3E03            LD   A,3          ;\Disable interrupts on
8105 D307            OUT  (PBCONT),A   ;PORT A
             ;
8107 3EFF            LD   A,£FF        ;\Set all O/P bits
8109 D305            OUT  (PBDATA),A   ;of PORT B
810B 3EFD            LD   A,£FD        ;\Reset (assert) INIT
810D D305            OUT  (PBDATA),A   ;line of printer
810F D775            RCAL DDEL60       ;\Hold low for 60 mic. s
8111 D1              POP  DE           ;\Restore DE
8112 7A              LD   A,D          ;\Use D to pick state of
8113 D305            OUT  (PBDATA),A   ;AUTO FEED EXT line
             ;
             ;
             ; SET UP PORT A
             ;
             ;
8115 3EFF            LD   A,-1         ;\Select 'CONTROL' (BIT)
8117 D306            OUT  (PACONT),A   ;mode
8119 3C              INC  A            ;\Set A to zero
811A D306            OUT  (PACONT),A   ;\Select all bits as O/Ps
```

```
811C 3E07     1640          LD   A,7            ;\Interrupt control word
811E D306     1650          OUT  (PACONT),A     ;with disabled interrupts
8120 3E03     1660          LD   A,3            ;\Disable PORT B int's
8122 D306     1670          OUT  (PACONT),A
              1680 ;
              1690 ;
8124 3E1B     1700          LD   A,ESC          ;\Set form length using
8126 D777     1710          RCAL OUTCHA         ;value stored in C
8128 3E43     1720          LD   A,"C
812A D773     1730          RCAL OUTCHA
812C C1       1740          POP  BC
812D 79       1750          LD   A,C
812E D76F     1760          RCAL OUTCHA
              1770 ;
8130 3E1B     1780          LD   A,ESC          ;\Set paper end detector
8132 D76B     1790          RCAL OUTCHA         ;on/off according to the
8134 78       1800          LD   A,B            ;value stored in B
8135 D768     1810          RCAL OUTCHA
              1820 ;
8137 7B       1830          LD   A,E            ;\Set skip/no skip
8138 FE4E     1840          CP   "N             ;according to the value
813A 280C     1850          JR   Z,NOSKIP       ;stored in E
813C 3E1B     1860          LD   A,ESC
813E D75F     1870          RCAL OUTCHA
8140 3E4E     1880          LD   A,"N
8142 D75B     1890          RCAL OUTCHA
8144 3E06     1900          LD   A,6            ;\Skip 6 lines
8146 D757     1910          RCAL OUTCHA
              1920 ;
8148 DB05     1930 NOSKIP   IN   A,(PBDATA)     ;\Get printer status
814A CB67     1940          BIT  4,A
814C 2818     1950          JR   Z,PAPOK        ;\Paper OK if B4 low
814E EF       1960          RST  PRS
814F 5072696E 1970          DEFM /Printer out of paper./
     74657220
     6F757420
     6F662070
     61706572
     2E
8164 0D00     1980          DEFB CR,0
8166 CB6F     1990 PAPOK    BIT  5,A
8168 201E     2000          JR   NZ,NOERR       ;\No error if B5 low
816A EF       2010          RST  PRS
816B 5072696E 2020          DEFM /Printer in error status./
     74657220
     696E2065
     72726F72
     20737461
     7475732E
8183 0D00     2030          DEFB CR,0
8185 C9       2040          RET
              2050 ;
              2060 ;
8186 1812     2070 DDEL60   JR   DEL60          ;\Dummy call point
              2080 ;                            ;for delay routine
              2090 ;
8188 EF       2100 NOERR    RST  PRS
8189 5072696E 2110          DEFM /Printer ready./
     74657220
     72656164
     792E
8197 0D00     2120          DEFB CR,0
8199 C9       2130          RET
              2140 ;
              2150 ;
              2160 ; 60 mic. sec. DELAY
              2170 ;
819A 0611     2180 DEL60    LD   B,17
819C 10FE     2190 LOOP     DJNZ LOOP
819E C9       2200          RET
              2210 ;
              2220 ; PRINTER DRIVER ROUTINE
              2230 ;
819F F5       2240 OUTCHA   PUSH AF             ;\Save character
81A0 DB05     2250 BUSY     IN   A,(PBDATA)     ;\Wait 'til printer
81A2 CB5F     2260          BIT  3,A            ;BUSY line is low
81A4 20FA     2270          JR   NZ,BUSY
81A6 F1       2280          POP  AF             ;\Restore character
81A7 D304     2290          OUT  (PADATA),A     ;\Apply to data lines
81A9 00       2300          NOP                 ;\Pause to stabilise
81AA DB05     2310          IN   A,(PBDATA)     ;\Keep port B status
81AC 3D       2320          DEC  A              ;\Reset STROBE (B0)
81AD D305     2330          OUT  (PBDATA),A
81AF 3C       2340          INC  A              ;\Set STROBE
81B0 D305     2350          OUT  (PBDATA),A
81B2 DB04     2360          IN   A,(PADATA)     ;\Return with character
81B4 C9       2370          RET
              2380 ;
              2390 ;
              2400 ;
              2410 ;-----------------------------------------
              2420 ;
              2430 ; LABELS ETC. :
              2440 ;
81B5 0028     2450 PRS      EQU  £28  ;NAS-SYS  print  string, RST
81B5 0030     2460 ROUT     EQU  £30  ;   "     O/P A to screen, "
81B5 007B     2470 BLINK    EQU  £7B  ;   "     blink for I/P, SCAL
81B5 0063     2480 INLIN    EQU  £63  ;   "     input a line,     "
81B5 0064     2490 NUM      EQU  £64  ;   "     get line value,   "
81B5 005B     2500 MRET     EQU  £5B  ;   "     monitor return,   "
              2510 ;
81B5 0C21     2520 NUMV     EQU  £C21   ;Store for val from NUM
81B5 0C78     2530 USROUT   EQU  £C78   ;Vectored jump to USROUT
81B5 0F05     2540 ZEAPAD   EQU  £F05   ;ZEAP vector to printer
81B5 101E     2550 PENADD   EQU  £101E  ;NAS-PEN vector to prin'r
              2560 ;
              2570 ;
              2580 ; PIO PORT ADDRESSES:
              2590 ;
81B5 0006     2600 PACONT   EQU  6 ;Port A control
81B5 0004     2610 PADATA   EQU  4 ;  "  A data
81B5 0007     2620 PBCONT   EQU  7 ;  "  B control
81B5 0005     2630 PBDATA   EQU  5 ;  "  B data
              2640 ;
              2650 ;
              2660 ; ASCII CODES:
              2670 ;
81B5 000D     2680 CR       EQU  £D     ;Carriage return
81B5 001B     2690 ESC      EQU  £1B    ;Escape
```

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 availabilty 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 straight forward 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 know 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 incompatiblities, usually mechanical. The most obvious is the mechanical incompatiblity 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.

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 £F8-£FB, the SIO uses ports £FC-£FF. This is bad news indeed to those using page mode on their RAM/ROM cards, since the page mode circuitry also uses port £FF (and port £FE 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 £D000, and uses 2K bytes of workspace, which 'sits' on top of TOS at £D800. 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 £ssss to £eeee-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
CDFAD0      CALL £D0FA     ; REWIND TAPE
21ADD7      LD HL,£D7AD
012700      LD BC,£0027    ; ** WAS £23 **
1152DB      LD DE,£DB52
EDB0        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 that 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 calalogue 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 automaticaly. 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 £0C80. 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).

Z2 ASSEMBLY LISTING .... PAGE 1

```
0F61                          ;****************************
0F61                          ;* TAPE BACKUP OF DISK FILES *
0F61                          ;* ------------------------- *
0F61                          ;*                           *
0F61                          ;* M.J.R GIBBS      14-07-82 *
0F61                          ;****************************
0018        NASSYS EQU   £18
0028        PRS    EQU   £28
0030        ROUT   EQU   £30
0038        RDEL   EQU   £38
0038                                  ;NASSYS COMMANDS
007B        BLINK  EQU   £7B
0068        B2HEX  EQU   £68
0065        CRT    EQU   £65
0071        NOM    EQU   £71
0077        NNOM   EQU   £77
005B        RETNAS EQU   £5B
006F        SOUT   EQU   £6F
0066        TBCD3  EQU   £66
0066                                  ;DISK COMMANDS
0081        ZDRD   EQU   £81          ;READ
0083        ZRDIR  EQU   £83          ;READ DIRECTORY
0088        ZCOV   EQU   £88          ;CALL OVERLAY
0088                                  ;DISK LOCATIONS
C055        S1FCB  EQU   £C055
C069        S2FCB  EQU   £C069
C001        DDRV   EQU   £C001        ;DIRECTORY DRIVE
C400        BUFFER EQU   £C400
C414        NXTSEC EQU   £C414        ;NEXT FREE SECTOR
C416        ENDDIR EQU   £C416        ;END  OF DIRECTORY
C418        DIR    EQU   £C418        ;DIR  ADDRESS
C418                                  ;DUMMY FCB
C055               ORG   S1FCB
C055               LOAD  0
C055        FNAM   DS    8            ;FILE   NAME
C05D        FEXT   DS    2            ;FILE   EXTENSION
C05F        FSFL   DS    1            ;SYSTEM FLAG
C060        FUFL   DS    1            ;USER   FLAG
C061        FSEC   DS    2            ;SECTOR ADDRESS
C063        FNSC   DS    2            ;NUMBER SECTORS
C065        FLDA   DS    2            ;LOAD   ADDRESS
C067        FEXA   DS    2            ;EXEC   ADDRESS
C067                                  ;
C067                                  ;SCREEN ADDRESSES
C067                                  ;
0BCA        TOPLN  EQU   £0BCA
080A        LINE1  EQU   £080A
084A        LINE2  EQU   LINE1+£0040
084A                                  ;
084A                                  ;MONITOR LOCATIONS
084A                                  ;
0C0C        ARG1   EQU   £0C0C
0C0E        ARG2   EQU   £0C0E
0C29        CURSOR EQU   £0C29
```

Z2 ASSEMBLY LISTING .... PAGE 2

```
0C29                                      ;
0C29                                      ;CURSOR CONTROL CHARS
0C29                                      ;
000C              CLEAR  EQU  £0C
000D              CRET   EQU  £0D
001B              ESC    EQU  £1B
0009              TAB    EQU  £09
0009                                      ;
0C80                     ORG  £0C80
0C80                     LOAD £8000
0C80
0C80 AF           START  XOR  A           ;SET  FILE NUMBER
0C81 32E50E              LD   (FILENO),A  ;ZERO FILE NUMBER
0C84 CDDA0D              CALL HEAD        ;WRITE HEADINGS
0C87 214A08       STA10  LD   HL,LINE2
0C8A 22290C              LD   (CURSOR),HL
0C8D EF                  RST  PRS
0C8E 0D09                DB   CRET,TAB
0C90 4C4F4144
0C94 20204469
0C98 736B2074
0C9C 6F206265
0CA0 2044756D
0CA4 706564              DB   'LOAD  Disk to be Dumped'
0CA7 0D0D09              DB   CRET,CRET,TAB
0CAA 50726573
0CAE 7320454E
0CB2 54455220
0CB6 5768656E
0CBA 20526561
0CBE 64792E              DB   'Press ENTER When Ready."
0CC1 0D0D09              DB   CRET,CRET,TAB
0CC4 20202020
0CC8 20204553
0CCC 43202020
0CD0 746F2053
0CD4 746F702E            DB   '      ESC   to Stop.'
0CD8 0D0D09              DB   CRET,CRET,TAB
0CDB 456E7465
0CDF 72205265
0CE3 71756972
0CE7 6564204F
0CEB 7074696F
0CEF 6E203D3D
0CF3 3D3E00              DB   'Enter Required Option ===>',00
0CF6 DF                  RST  NASSYS
0CF7 7B                  DB   BLINK       ;WAIT REPLY
0CF8 FE0D                CP   CRET        ;WAS  CRET
0CFA 2807                JR   Z,STA20
0CFC FE1B                CP   ESC         ;WAS  ESCAPE
0CFE C2800C              JP   NZ,START    ;NO   TRY AGAIN
0D01 DF                  RST  NASSYS
0D02 5B                  DB   RETNAS      ;STOP
0D03 CDDD0E       STA20  CALL WAIT        ;WAIT .5 SEC
0D06 3EFF                LD   A,£FF       ;RESET DIRECTOR READ
0D08 3201C0              LD   (DDRV),A    ;ZAP  IT IN
0D0B 0E00                LD   C,0         ;FORCE READ ON DRIVE 0
```

```
------------------------------------
 0D0D DF                RST   NASSYS
 0D0E 83                DB    ZRDIR        ;READ DISK DIRECTORY
 0D0F 2816              JR    Z,STA30
 0D11 EF                RST   PRS
 0D12 0D094552
 0D16 524F5220
 0D1A 54525920
 0D1E 41474149
 0D22 4E00              DB    CRET,TAB,'ERROR TRY AGAIN',0
 0D24 C3870C            JP    STA10
 0D27 2100C4    STA30   LD    HL,BUFFER    ;COPY DISK ID
 0D2A 111F0E            LD    DE,NAME
 0D2D 011400            LD    BC,20
 0D30 EDB0              LDIR               ;LOAD IT OUT
 0D32 2118C4    STA40   LD    HL,DIR       ;HL = A(DIRECTORY)
 0D35 22E30E            LD    (DIRPOS),HL  ;SAVE IT
 0D38 2AE30E    LOOP10  LD    HL,(DIRPOS)  ;HL = POS IN DIR
 0D3B ED5B16C4          LD    DE,(ENDDIR)  ;DE = A(END DIR)
 0D3F B7                OR    A
 0D40 ED52              SBC   HL,DE        ;END  OF FILES ON DISK
 0D42 19                ADD   HL,DE        ;RESTORE ADDRESS
 0D43 CA800C            JP    Z,START      ;YES  NEXT DISK
 0D46 1155C0            LD    DE,S1FCB     ;COPY TO S1FCB
 0D49 011400            LD    BC,20        ;BC = L(FCB)
 0D4C EDB0              LDIR               ;ZAP  IT ACROSS
 0D4E 22E30E            LD    (DIRPOS),HL  ;KEEP DIR POSN
 0D4E                                      ;
 0D4E                                      ;--- SCAN TABLE ---
 0D4E                                      ;
 0D51 060D              LD    B,13         ;B = NO OF TABLE ENTRIES
 0D53 DD21E90E          LD    IX,TABLE     ;IX = A(TABLE)
 0D57 110A00            LD    DE,10        ;DE = ENTRY SIZE
 0D5A C5        LOOP20  PUSH  BC           ;KEEP BC
 0D5B 060A              LD    B,10         ;B  = SCAN LENGTH
 0D5D 2155C0            LD    HL,S1FCB     ;HL = THIS FILE NAME
 0D60 DDE5              PUSH  IX           ;KEEP IX
 0D62 DD7E00    LOOP30  LD    A,(IX+0)     ;GET  FROM TABLE
 0D65 BE                CP    (HL)         ;COMPARE
 0D66 200A              JR    NZ,NEXT      ;FAIL
 0D68 DD23              INC   IX           ;INDEX TO NEXT CHAR
 0D6A 23                INC   HL
 0D6B 10F5              DJNZ  LOOP30       ;REPEAT
 0D6D DDE1              POP   IX           ;GOT  IT OK
 0D6F C1                POP   BC           ;RECOVER STACK
 0D70 18C6              JR    LOOP10       ;CONTINUE
 0D72 DDE1      NEXT    POP   IX           ;RESET POINTER
 0D74 DD19              ADD   IX,DE        ;POINT TO NEXT
 0D76 C1                POP   BC           ;RECOVER BC
 0D77 10E1              DJNZ  LOOP20       ;TRY AGAIN
 0D79 CDDD0E    STA50   CALL  WAIT         ;WAIT .5 SEC
 0D7C CDDA0D            CALL  HEAD         ;PRINT HEADINGS
 0D7F CD010E            CALL  PRINT        ;PRINT FILE INFO
 0D82 210010            LD    HL,£1000     ;LOAD ADDRESS
 0D85 ED5B61C0          LD    DE,(FSEC)    ;SECTOR ADDRESS
 0D89 3A63C0            LD    A,(FNSC)     ;A  = NUMBER OF SECTORS
 0D8C 47                LD    B,A          ;B  = NUMBER OF SECTORS
 0D8D 3A01C0            LD    A,(DDRV)     ;A  = DIRECTORY DRIVE
 0D90 4F                LD    C,A          ;C  = DIRECTORY DRIVE
```

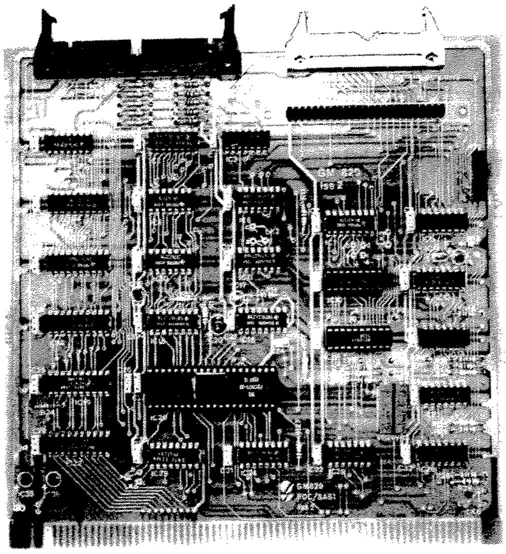


```
------------------------------------
 0D91 DF                RST   NASSYS
 0D92 81                DB    ZDRD         ;READ IT IN
 0D93 C2A70D            JP    NZ,ERRORD    ;ERROR
 0D96 210010            LD    HL,£1000
 0D99 220C0C            LD    (ARG1),HL    ;SET  UP FOR WRITE
 0D9C 0E00              LD    C,0
 0D9E 09                ADD   HL,BC        ;CALC END ADD
 0D9F 220E0C            LD    (ARG2),HL
 0DA2 DF                RST   NASSYS
 0DA3 57                DB    'W'
 0DA4 C3380D            JP    LOOP10       ;RETURN
 0DA4                                      ;
 0DA4                                      ;--- DISK ERROR ---
 0DA4                                      ;
 0DA7 F5        ERRORD  PUSH  AF           ;KEEP ERROR NUMBER
 0DA8 EF                RST   PRS
 0DA9 0D094572
 0DAD 726F7220
 0DB1 3D3D3D3E
 0DB5 2000              DB    CRET,TAB,"Error ===> ",00
 0DB7 F1                POP   AF
 0DB8 DF                RST   NASSYS
 0DB9 88                DB    ZCOV         ;LOAD ERROR OVERLAY
 0DBA 456D7367          DB    "Emsg"       ;PRINT MESSAGE
 0DBE EF        DSKEND  RST   PRS
 0DBF 0D444953
 0DC3 4B20492D
 0DC7 4F204552
 0DCB 524F5220
 0DCF 53544F50
 0DD3 50454420
 0DD7 00                DB    CRET,'DISK I-O ERROR STOPPED ',00
 0DD8 DF                RST   NASSYS
 0DD9 5B                DB    RETNAS
 0DD9                                      ;
 0DD9                                      ;--- SUBROUTINE HEAD ---
 0DD9                                      ;
 0DDA EF        HEAD    RST   PRS
 0DDB 0C00              DB    CLEAR,00
 0DDD 21D40B            LD    HL,TOPLN+10  ;SETUP CURSOR POSN
 0DE0 22290C            LD    (CURSOR),HL  ;SET IT UP
 0DE3 EF                RST   PRS
 0DE4 54415045
 0DE8 20424143
 0DEC 4B555020
 0DF0 464F5220
 0DF4 4449534B
 0DF8 5300              DB    'TAPE BACKUP FOR DISKS',00
 0DFA 210A08            LD    HL,LINE1     ;RESET FOR NORMAL PRINT
 0DFD 22290C            LD    (CURSOR),HL
 0E00 C9                RET
 0E00                                      ;
 0E00                                      ;--- SUBROUTINE PRINT ---
 0E00                                      ;
 0E01 3E0C      PRINT   LD    A,CLEAR
 0E03 DF                RST   NASSYS
 0E04 6F                DB    SOUT
 0E05 CDDD0E            CALL  WAIT         ;WAIT .5 SEC
```

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.

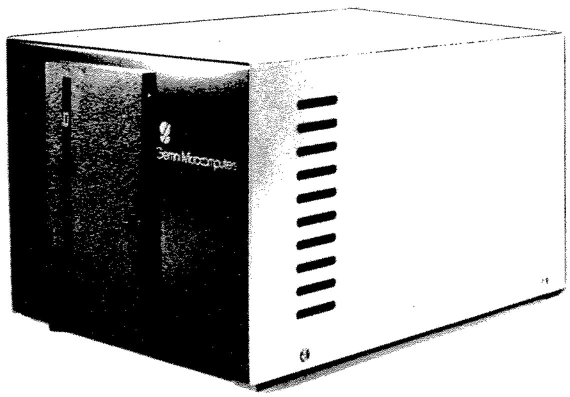
Up to four floppy disk drives may be controlled. These may be single or double sided, 48 or 96 TPI drives, in single or double density formats. The drives may also be 3.5", 5.25" or 8" types, or a combination of these. Switching between single and double density, and 3.5"/5.25" and 8" drives, are under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

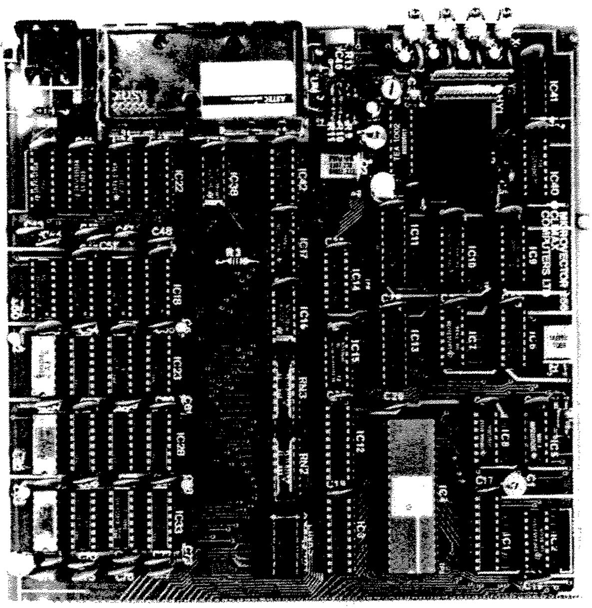
- 5.4 Megabyte Formatted Capacity
- Industry Standard SASI Interface
- Integral Controller and Power Supply

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 GM825 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.

Multiboard Based SYSTEMS

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 interlaced 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

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/0-64KRAM, 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

EV – **EV Computing Ltd.,**
700 Burnage Lane,
Burnage,
Manchester.

IO – **IO Research Ltd.,**
6 Laleham Avenue,
Mill Hill,
London. NW7 3HL

MP – **Microcode (Control) Ltd.,**
41a Moor Lane,
Clitheroe,
Lancs.

AM – **Arfon Microelectronics Ltd.,**
These products are currently being re-sourced. Contact your dealer for further details.

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
-------------------------------------
 0E08 21E60E            LD    HL,OUTTAB   ;SET  UART ON
 0E0B DF                RST   NASSYS
 0E0C 71                DB    NOM         ;SWITCH NEW OUTPUT ON
 0E0D EF                RST   PRS
 0E0E 0D0D              DB    CRET,CRET
 0E10 20204449
 0E14 534B204E
 0E18 414D4520
 0E1C 3A2D20            DB    '  DISK NAME :- '
 0E1F 20202020
 0E23 20202020
 0E27 20202020
 0E2B 20202020
 0E2F 20202020
 0E33 0D        NAME    DB    '                          ',CRET
 0E34 20204649
 0E38 4C45204E
 0E3C 4F202020
 0E40 3A2D2000          DB    '  FILE NO   :- ',00
 0E44 3AE50E            LD    A,(FILENO)  ;A  = FILE NO
 0E47 3C                INC   A           ;FILENO = FILENO+1
 0E48 27                DAA               ;CONV TO DEC
 0E49 32E50E            LD    (FILENO),A  ;SAVE IT
 0E4C DF                RST   NASSYS
 0E4D 68                DB    B2HEX       ;PRINT FILENO
 0E4E EF                RST   PRS
 0E4F 0D202046
 0E53 494C4520
 0E57 4E414D45
 0E5B 203A2D20
 0E5F 00                DB    CRET,'  FILE NAME :- ',00
 0E60 2155C0            LD    HL,S1FCB    ;HL = A(FCB)
 0E63 0608              LD    B,8         ;B  = L(FILE NAME)
 0E65 7E        PRT10   LD    A,(HL)      ;GET  CHAR
 0E66 F7                RST   ROUT        ;PRINT IT
 0E67 23                INC   HL          ;INDEX TO NEXT
 0E68 10FB              DJNZ  PRT10       ;LOOP ON IT
 0E6A 3E2E              LD    A,'.'       ;A  = '.'
 0E6C 0603              LD    B,3         ;OUTPUT FILE EXTENSION
 0E6E 2B                DEC   HL
 0E6F F7        PRT20   RST   ROUT        ;PUSH IT OUT
 0E70 23                INC   HL          ;INDEX ALONG
 0E71 7E                LD    A,(HL)      ;GET  CHARACTER
 0E72 10FB              DJNZ  PRT20
 0E74 3A5FC0            LD    A,(FSFL)    ;A  = SYSTEM FLAG
 0E77 FE01              CP    1           ;IS = LOCKED FILE
 0E79 2012              JR    NZ,PRT30
 0E7B EF                RST   PRS
 0E7C 203C3C20
 0E80 4C6F636B
 0E84 65642020
 0E88 3E3E00            DB    ' << Locked  >>',00
 0E8B 1814              JR    PRT40
 0E8D FE02      PRT30   CP    2           ;IS = DELETED FILE
 0E8F 2010              JR    NZ,PRT40
```

```
Z2 ASSEMBLY LISTING ........ PAGE  6
-------------------------------------
 0E91 EF                RST   PRS
 0E92 203C3C20
 0E96 44454C45
 0E9A 54454420
 0E9E 3E3E00            DB    ' << DELETED >>',00
 0EA1 EF        PRT40   RST   PRS
 0EA2 0D20204C
 0EA6 4F414445
 0EAA 44204154
 0EAE 203A2D20
 0EB2 00                DB    CRET,'  LOADED AT :- ',00
 0EB3 2A65C0            LD    HL,(FLDA)   ;HL = LOAD ADDRESS
 0EB6 DF                RST   NASSYS
 0EB7 66                DB    TBCD3       ;PRINT IT
 0EB8 EF                RST   PRS
 0EB9 0D202045
 0EBD 58454320
 0EC1 41542020
 0EC5 203A2D20
 0EC9 00                DB    CRET,'  EXEC AT   :- ',00
 0ECA 2A67C0            LD    HL,(FEXA)
 0ECD DF                RST   NASSYS
 0ECE 66                DB    TBCD3       ;PRINT IT
 0ECF EF                RST   PRS
 0ED0 0D00              DB    CRET,00
 0ED2 3E52              LD    A,'R'       ;FORCE A READ
 0ED4 DF                RST   NASSYS
 0ED5 6F                DB    SOUT        ;ZAP  IT OUT THE UART
 0ED6 3E0D              LD    A,CRET      ;FOLLOWED BY CRET
 0ED8 DF                RST   NASSYS
 0ED9 6F                DB    SOUT
 0EDA DF                RST   NASSYS
 0EDB 77                DB    NNOM        ;RESET OUTPUT TABLE
 0EDC C9                RET
 0EDC                                     ;
 0EDC                                     ;--- SUBROUTINE WAIT ---
 0EDC                                     ;
 0EDD AF        WAIT    XOR   A           ;A  = 0
 0EDE 47                LD    B,A         ;B  = 0
 0EDF FF        WAIT10  RST   RDEL        ;NASSYS WAIT
 0EE0 10FD              DJNZ  WAIT10      ;LOOP IT
 0EE2 C9                RET
 0EE3           DIRPOS  DS    2
 0EE5           FILENO  DS    1
 0EE6 656F00    OUTTAB  DB    CRT,SOUT,£00
 0EE9 45786563
 0EED 20202020
 0EF1 4F56      TABLE   DB    'Exec    OV'
 0EF3 456D7367
 0EF7 20202020
 0EFB 4F56              DB    'Emsg    OV'
 0EFD 4466756E
 0E01 20202020
 0E05 4F56              DB    'Dfun    OV'
 0F07 45636D64
 0F0B 20202020
 0F0F 4F56              DB    'Ecmd    OV'
```

```
Z2 ASSEMBLY LISTING ........ PAGE  7
-------------------------------------
  OF11 45646974
  OF15 20202020
  OF19 4F56              DB    'Edit    OV'
  OF1B 496E666F
  OF1F 20202020
  OF23 494E              DB    'Info    IN'
  OF25 42536668
  OF29 20202020
  OF2D 4F56              DB    'BSfh    OV'
  OF2F 42537574
  OF33 20202020
  OF37 4F56              DB    'BSut    OV'
  OF39 42536472
  OF3D 20202020
  OF41 4252              DB    'BSdr    BR'
  OF43 464F524D
  OF47 41542020
  OF4B 474F              DB    'FORMAT  GO'
  OF4D 4241434B
  OF51 55502020
  OF55 474F              DB    'BACKUP  GO'
  OF57 535A4150
  OF5B 20202020
  OF5F 474F              DB    'SZAP    GO'
  OF61 505A4150
  OF65 20202020
  OF69 474F              DB    'PZAP    GO'

  ARG1   OCOC ARG2   OCOE B2HEX  0068 BLINK  007B
  BUFFER C400 CLEAR  000C CRET   000D CRT    0065
  CURSOR 0C29 DDRV   C001 DIR    C418 DIRPOS OEE3
  DSKEND ODBE ENDDIR C416 ERRORD ODA7 ESC    001B
  FEXA   C067 FEXT   C05D FILENO OEE5 FLDA   C065
  FNAM   C055 FNSC   C063 FSEC   C061 FSFL   C05F
  FUFL   C060 HEAD   ODDA LINE1  080A LINE2  084A
  LOOP10 OD38 LOOP20 OD5A LOOP30 OD62 NAME   OE1F
  NASSYS 0018 NEXT   OD72 NNOM   0077 NOM    0071
  NXTSEC C414 OUTTAB OEE6 PRINT  OE01 PRS    0028
  PRT10  OE65 PRT20  OE6F PRT30  OE8D PRT40  OEA1
  RDEL   0038 RETNAS 005B ROUT   0030 S1FCB  C055
  S2FCB  C069 SOUT   006F STA10  OC87 STA20  OD03
  STA30  OD27 STA40  OD32 STA50  OD79 START  0C80
  TAB    0009 TABLE  OEE9 TBCD3  0066 TOPLN  OBCA
  WAIT   OEDD WAIT10 OEDF ZCOV   0088 ZDRD   0081
  ZRDIR  0083
```

```
DUMP V1.0            Disk - Tape Backup V1.0                  PAGE 01

0C80  AF 32 E5 OE CD DA OD 21  4A 08 22 29 OC EF OD 09  .2.....!J.")....
0C90  4C 4F 41 44 20 20 44 69  73 6B 20 74 6F 20 62 65  LOAD  Disk to be
OCA0  20 44 75 6D 70 65 64 OD  OD 09 50 72 65 73 73 20   Dumped...Press
OCB0  45 4E 54 45 52 20 57 68  65 6E 20 52 65 61 64 79  ENTER When Ready
OCC0  2E OD OD 09 20 20 20 20  20 20 45 53 43 20 20 20  ....      ESC
OCD0  74 6F 20 53 74 6F 70 2E  OD OD 09 45 6E 74 65 72  to Stop....Enter
OCE0  20 52 65 71 75 69 72 65  64 20 4F 70 74 69 6F 6E   Required Option
OCF0  20 3D 3D 3D 3E 00 DF 7B  FE OD 28 07 FE 1B C2 80   ===>..{..(.....
OD00  OC DF 5B CD DD OE 3E FF  32 01 CO OE 00 DF 83 28  ..[...>.2......(
OD10  16 EF OD 09 45 52 52 4F  52 20 54 52 59 20 41 47  ....ERROR TRY AG
OD20  41 49 4E 00 C3 87 OC 21  00 C4 11 1F OE 01 14 00  AIN....!........
OD30  ED B0 21 18 C4 22 E3 OE  2A E3 OE ED 5B 16 C4 B7  ..!.."..*...[...
OD40  ED 52 19 CA 80 OC 11 55  CO 01 14 00 ED BO 22 E3  .R.....U......".
OD50  OE 06 OD DD 21 E9 OE 11  OA 00 C5 06 OA 21 55 CO  ....!........!U.
OD60  DD E5 DD 7E 00 BE 20 OA  DD 23 23 10 F5 DD E1 C1  ...~.. ..££.....
OD70  18 C6 DD E1 DD 19 C1 10  E1 CD DD OE CD DA OD CD  ................
OD80  01 OE 21 00 10 ED 5B 61  CO 3A 63 CO 47 3A 01 CO  ..!...[a.:c.G:..
OD90  4F DF 81 C2 A7 OD 21 00  10 22 OC OC OE 00 09 22  O.....!.."....."
ODA0  OE OC DF 57 C3 38 OD F5  EF OD 09 45 72 72 6F 72  ...W.8.....Error
ODB0  20 3D 3D 3D 3E 20 00 F1  DF 88 45 6D 73 67 EF OD   ===> ....Emsg..
ODC0  44 49 53 4B 20 49 2D 4F  20 45 52 52 4F 52 20 53  DISK I-O ERROR S
ODD0  54 4F 50 50 45 44 20 00  DF 5B EF OC 00 21 D4 OB  TOPPED ..[...!..
ODE0  22 29 OC EF 54 41 50 45  20 42 41 43 4B 55 50 20  ")..TAPE BACKUP
ODF0  46 4F 52 20 44 49 53 4B  53 00 21 OA 08 22 29 OC  FOR DISKS.!..").
OE00  C9 3E OC DF 6F CD DD OE  21 E6 OE DF 71 EF OD OD  .>..o...!...q...
OE10  20 20 44 49 53 4B 20 4E  41 4D 45 20 3A 2D 20 20    DISK NAME :-
OE20  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
OE30  20 20 20 OD 20 20 46 49  4C 45 20 4E 4F 20 20 20     .  FILE NO
OE40  3A 2D 20 00 3A E5 OE 3C  27 32 E5 OE DF 68 EF OD  :- .:..<'2...h..
OE50  20 20 46 49 4C 45 20 4E  41 4D 45 20 3A 2D 20 00    FILE NAME :- .
OE60  21 55 CO 06 08 7E F7 23  10 FB 3E 2E 06 03 2B F7  !U...~.£..>...+.
OE70  23 7E 10 FB 3A 5F CO FE  01 20 12 EF 20 3C 3C 20  £~..:_... .. <<
OE80  4C 6F 63 6B 65 64 20 20  3E 3E 00 18 14 FE 02 20  Locked  >>.....
OE90  10 EF 20 3C 3C 20 44 45  4C 45 54 45 44 20 3E 3E  .. << DELETED >>
OEA0  00 EF OD 20 20 4C 4F 41  44 45 44 20 41 54 20 3A  ...  LOADED AT :
OEB0  2D 20 00 2A 65 CO DF 66  EF OD 20 20 45 58 45 43  - .*e..f..  EXEC
OEC0  20 41 54 20 20 20 3A 2D  20 00 2A 67 CO DF 66 EF   AT    :- .*g..f.
OED0  OD 00 3E 52 DF 6F 3E OD  DF 6F DF 77 C9 AF 47 FF  ..>R.o>..o.w..G.
OEE0  10 FD C9 00 00 00 65 6F  00 45 78 65 63 20 20 20  ......eo.Exec
OEF0  20 4F 56 45 6D 73 67 20  20 20 20 4F 56 44 66 75   OVEmsg    OVDfu
OF00  6E 20 20 20 20 4F 56 45  63 6D 64 20 20 20 20 4F  n    OVEcmd    O
OF10  56 45 64 69 74 20 20 20  20 4F 56 49 6E 66 6F 20  VEdit    OVInfo
OF20  20 20 20 49 4E 42 53 66  68 20 20 20 20 4F 56 42     INBSfh    OVB
OF30  53 75 74 20 20 20 20 4F  56 42 53 64 72 20 20 20  Sut    OVBSdr
OF40  20 42 52 46 4F 52 4D 41  54 20 20 47 4F 42 41 43   BRFORMAT  GOBAC
OF50  4B 55 50 20 20 47 4F 53  5A 41 50 20 20 20 20 47  KUP  GOSZAP    G
OF60  4F 50 5A 41 50 20 20 20  20 47 4F 00 00 00 00 00  OPZAP    GO.....
OF70  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

Chronology, time and dates and a review of the Gemini GM822

D. R. Hunt

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 0B4H. 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 desparation, 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 0FH 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 it's 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 that 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 trucate 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                       09/09/82'

                   .Z80

                   GLOBAL  RTC

001C          BASE     EQU     1CH          ; Base address of control PIO

              ; CP/M equates
0001          CONIN    EQU     1            ; Console input routine
0002          CONOUT   EQU     2            ; Console output routine
000A          LININ    EQU     10           ; Line input
000D          CR       EQU     0DH          ; Carriage return
000A          LF       EQU     0AH          ; Line feed
0005          BDOS     EQU     0005H        ; BDOS jump vector

              ; CP/M work spaces
0082          ARGS     EQU     0082H        ; Input arguments
0082          IHOURS   EQU     ARGS+0       ; Input hours
0085          IMINS    EQU     ARGS+3       ; Input minutes
0088          IDAY     EQU     ARGS+6       ; Input day
008B          IMONTH   EQU     ARGS+9       ; Input month
008E          IDYWK    EQU     ARGS+12      ; Input day of week

              ; Port assignments
001C          PDATA    EQU     BASE         ; Data port for commands
001D          PADDR    EQU     BASE+1       ; Data port for commands
001E          PCDATA   EQU     BASE+2       ; Control port for data
001F          PCADDR   EQU     BASE+3       ; Control port for data

              ; Control instructions
00FF          CMODE    EQU     0FFH         ; Control mode
00FF          ALLIN    EQU     0FFH         ; All inputs
0000          ALLOUT   EQU     0            ; All outputs
0010          INTOUT   EQU     10H          ; All out except INT input

              ; Addresses for RTC
0000          TEST     EQU     0
000C          TMONTH   EQU     12
000E          START    EQU     14
000F          INSTAT   EQU     15
00E0          NOSTRB   EQU     0E0H         ; No strobe or select
0080          RDSTRB   EQU     80H          ; Read strobe on
0040          WRSTRB   EQU     40H          ; Write strobe on
```



```
                        ; **************************************************
                        ; ****                 Messages                 ****
                        ; **************************************************
0000'   0D 0A           TTLMES: DEFB    CR,LF
0002'   20 20 20 20             DEFB    '                ***  RTC TIME SET  ***'
0006'   20 20 20 20
000A'   20 20 20 20
000E'   20 2A 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 0A 0D 0A             DEFB    CR,LF,CR,LF
0029'   53 65 74 20             DEFB    'Set time in the form:- HH:MM DD:MM.D'
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             DEFB    ' or T for time.'
0051'   54 20 66 6F
0055'   72 20 74 69
0059'   6D 65 2E
005C'   0D 0A                   DEFB    CR,LF
005E'   45 6E 74 65             DEFB    'Enter time ? ',0
0062'   72 20 74 69
0066'   6D 65 20 3F
006A'   20 00
006C'   48 69 74 20     STMES:  DEFB    'Hit any key to start clock ',0
0070'   61 6E 79 20
0074'   6B 65 79 20
0078'   74 6F 20 73
007C'   74 61 72 74
0080'   20 63 6C 6F
0084'   63 6B 20 00
0088'   45 72 72 6F     ERMES:  DEFB    'Error'
008C'   72
008D'   0D 0A 00        CRLF:   DEFB    CR,LF,0

                        ; Output string
0090'   30 30 3A 30     HOURS:  DEFB    '00:00:00.0
0094'   30 3A 30 30
0098'   2E 30 20 20
009C'   58 58 58 20     DAYMN:  DEFB    'XXX '
00A0'   30 30 20        DAY:    DEFB    '00 '
00A3'   58 58 58 20     MONTH:  DEFB    'XXX 1982',CR,LF,0
00A7'   31 39 38 32
00AB'   0D 0A 00
```

```
                                ; Data strings
00AE'    33 3A 36 3A    INVAL:  DEFB    '3:6:4:2:8'             ; Input validation
00B2'    34 3A 32 3A
00B6'    38
00B7'    53 75 6E 4D    DYSTRG: DEFB    'SunMonTueWedThuFriSat'
00BB'    6F 6E 54 75
00BF'    65 57 65 64
00C3'    54 68 75 46
00C7'    72 69 53 61
00CB'    74
00CC'    4A 61 6E 46    MNSTRG: DEFB    'JanFebMarAprMayJunJlyAugSepOctNovDec'
00D0'    65 62 4D 61
00D4'    72 41 70 72
00D8'    4D 61 79 4A
00DC'    75 6E 4A 6C
00E0'    79 41 75 67
00E4'    53 65 70 4F
00E8'    63 74 4E 6F
00EC'    76 44 65 63

                        ; ************************************************
                        ; ****          PIO control subroutines          ****
                        ; ************************************************

                        ; Set data port to control mode, all lines input
00F0'    3E FF          INDATA: LD      A,CMODE         ; Control mode
00F2'    D3 1E                  OUT     (PCDATA),A      ; Set mode word
00F4'    3E FF                  LD      A,ALLIN
00F6'    D3 1E                  OUT     (PCDATA),A      ; Set direction
00F8'    C9                     RET

                        ; Set command port to control mode, all lines input
00F9'    3E FF          INADDR: LD      A,CMODE         ; Control mode
00FB'    D3 1F                  OUT     (PCADDR),A      ; Set mode
00FD'    3E FF                  LD      A,ALLIN
00FF'    D3 1F                  OUT     (PCADDR),A      ; Set direction
0101'    C9                     RET

                        ; Set data port to control mode, all lines output
0102'    3E FF          OUDATA: LD      A,CMODE         ; Control mode
0104'    D3 1E                  OUT     (PCDATA),A      ; Set mode
0106'    3E 00                  LD      A,ALLOUT
0108'    D3 1E                  OUT     (PCDATA),A      ; Set direction
010A'    C9                     RET

                        ; Set command port to control mode, all lines output
                        ;       EXCEPT the INT line (0B4H)
010B'    3E FF          OUADDR: LD      A,0FFH          ; Ensure all data lines ...
010D'    D3 1D                  OUT     (PADDR),A       ; ... will be 1's
010F'    3E FF                  LD      A,CMODE         ; Set mode
0111'    D3 1F                  OUT     (PCADDR),A
0113'    3E 10                  LD      A,INTOUT        ; Set direction
0115'    D3 1F                  OUT     (PCADDR),A
0117'    C9                     RET
```



```
                        ; ************************************************
                        ; **** Subroutines to read/write to the RTC      ****
                        ; **** Called with:-                             ****
                        ; ****          C=PADDR - data port for command  ****
                        ; ****          D=RTC register address           ****
                        ; ****          E=RTC reg. + strobe (R or W)     ****
                        ; ************************************************

                        ; Read a RTC register
0118'    ED 51          READ:   OUT     (C),D           ; Set up address
011A'    ED 59                  OUT     (C),E           ; Read strobe on
011C'    DB 1C                  IN      A,(PDATA)       ; Read a nibble
011E'    ED 51                  OUT     (C),D           ; Strobe off
0120'    C9                     RET

                        ; Write <A> to a RTC register
0121'    D3 1C          WRITE:  OUT     (PDATA),A       ; Put data in PIO register
0123'    CD 0102'               CALL    OUDATA          ; Set to output
0126'    ED 51                  OUT     (C),D           ; Address the RTC
0128'    ED 59                  OUT     (C),E           ; WR strobe on
012A'    ED 51                  OUT     (C),D           ; WR strobe off
012C'    18 C2                  JR      INDATA          ; Set back to input

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

012E'    CD 010B'       RDRTC:  CALL    OUADDR          ; Set command port to o/p
0131'    CD 00F0'               CALL    INDATA          ; Set data port to i/p
0134'    21 02BB'               LD      HL,WSPACE       ; Read into here
0137'    06 0C                  LD      B,12            ; 12 registers to read
0139'    16 EC                  LD      D,TMONTH+NOSTRB ; 10's of months + no strobe
013B'    1E 8C                  LD      E,TMONTH+RDSTRB ; 10's of months + read strobe
013D'    0E 1D                  LD      C,PADDR         ; C = control port address
013F'    CD 0118'       RDRTCL: CALL    READ            ; Get the data
0142'    E6 0F                  AND     0FH             ; Isolate the lower nibble
0144'    FE 0F                  CP      0FH             ; Valid?
0146'    28 E6                  JR      Z,RDRTC         ; No, so start again
0148'    C6 30                  ADD     A,30H           ; Yes, so make it ASCII
014A'    77                     LD      (HL),A          ; Store in workspace
014B'    23                     INC     HL
014C'    15                     DEC     D               ; Next RTC register
014D'    1D                     DEC     E
014E'    10 EF                  DJNZ    RDRTCL          ; Loop if more
0150'    C9                     RET                     ; Done

                        ; ************************************************
                        ; ****          Write workspace to RTC           ****
                        ; ****     (Used when initializing the RTC)      ****
                        ; ************************************************

0151'    CD 010B'       WRRTC:  CALL    OUADDR          ; Set command port to o/p
0154'    CD 0102'               CALL    OUDATA          ; Set data port to o/p
```

```
0157'   21 02B8'            LD      HL,WRKSPC         ; Write from here
015A'   06 0C               LD      B,12              ; 12 registers to write
015C'   16 EF               LD      D,INSTAT+NOSTRB   ; Interrupt status + no strobe
015E'   1E 4F               LD      E,INSTAT+WRSTRB   ; Interrupt status + write strobe
0160'   0E 1D               LD      C,PADDR           ; C = control port address
0162'   7E          WRRTCL: LD      A,(HL)            ; Get the data
0163'   CD 0121'            CALL    WRITE             ; Put in the RTC
0166'   23                  INC     HL
0167'   15                  DEC     D                 ; Next RTC register
0168'   1D                  DEC     E
0169'   10 F7               DJNZ    WRRTCL            ; Loop if more
016B'   C9                  RET                       ; Done

                    ; **************************************************
                    ; **** Display the current date and time in the ****
                    ; **** form:-  19:00:56.6  Mon 21 Sep 1982      ****
                    ; **************************************************

016C'   CD 00F0'    TIME:   CALL    INDATA            ; Set PIOs to RESET state
016F'   CD 00F9'            CALL    INADDR
0172'   CD 012E'            CALL    RDRTC             ; Get data into workspace
                    ; Get the month
0175'   21 02BB'            LD      HL,WSPACE         ; Point to output string
0178'   CD 01D1'            CALL    PLSTRG            ; Calculate offset
017B'   16 00               LD      D,0
017D'   5F                  LD      E,A
017E'   21 00CC'            LD      HL,MNSTRG
0181'   19                  ADD     HL,DE             ; Add it to <HL>
0182'   11 00A3'            LD      DE,MONTH
0185'   01 0003             LD      BC,3
0188'   ED B0               LDIR                      ; Copy across
                    ; Get the day of the week
018A'   3A 02BD'            LD      A,(WSPACE+2)      ; Get day number
018D'   D6 30               SUB     30H
018F'   CD 01E9'            CALL    TIME3             ; Place in string
0192'   D6 03               SUB     3
0194'   16 00               LD      D,0               ; Calculate offset
0196'   5F                  LD      E,A
0197'   21 00B7'            LD      HL,DYSTRG
019A'   19                  ADD     HL,DE             ; Add it to <HL>
019B'   11 009C'            LD      DE,DAYMN
019E'   01 0003             LD      BC,3
01A1'   ED B0               LDIR                      ; Copy across
                    ; Get the date and time .
01A3'   21 02BE'            LD      HL,WSPACE+3
01A6'   11 00A0'            LD      DE,DAY            ; Now the day
01A9'   ED A0               LDI
01AB'   ED A0               LDI
01AD'   11 0090'            LD      DE,HOURS          ; Now the hours/mins/secs
01B0'   3E 03               LD      A,3
01B2'   ED A0       HMS:    LDI
01B4'   ED A0               LDI
01B6'   13                  INC     DE
01B7'   3D                  DEC     A                 ; More?
```



```
01B8'   20 F8               JR      NZ,HMS            ; Yes, so loop
01BA'   ED A0               LDI                       ; Copy across the tenths
01BC'   21 0090'            LD      HL,HOURS          ; Now display it
01BF'   CD 01C3'            CALL    PRINT
01C2'   C9                  RET                       ; Done

                    ; Print the string
01C3'   7E          PRINT:  LD      A,(HL)            ; Get the byte
01C4'   B7                  OR      A                 ; Done?
01C5'   C8                  RET     Z
01C6'   23                  INC     HL
01C7'   5F                  LD      E,A
01C8'   E5                  PUSH    HL                ; Save <HL>
01C9'   0E 02               LD      C,CONOUT
01CB'   CD 0005             CALL    BDOS
01CE'   E1                  POP     HL
01CF'   18 F2               JR      PRINT

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

01D1'   7E          PLSTRG: LD      A,(HL)
01D2'   23                  INC     HL
01D3'   D6 30               SUB     30H
01D5'   CD 01E3'            CALL    TIME10
01D8'   47                  LD      B,A
01D9'   7E                  LD      A,(HL)
01DA'   D6 30               SUB     30H
01DC'   80                  ADD     A,B
01DD'   CD 01E9'            CALL    TIME3
01E0'   D6 03               SUB     3
01E2'   C9                  RET

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

01E3'   07          TIME10: RLCA                      ; Multiply by 2
01E4'   47                  LD      B,A               ; Save the partial product
01E5'   07                  RLCA                      ; Multiply by 2 again
01E6'   07                  RLCA                      ;  .. and again
01E7'   80                  ADD     A,B               ; Add the partial product
01E8'   C9                  RET

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

```
                       ; ***************************************************
                       ; ****     Set RTC from ASCII values at ARGS     ****
                       ; **** Arguments are in the form HH:MM DD:MM.D  ****
                       ; ***************************************************

01ED'   CD 00F0'       SET:    CALL    INDATA          ; Get PIOs out of reset state
01F0'   CD 00F9'               CALL    INADDR
01F3'   AF                     XOR     A               ; Set non test mode
01F4'   16 E0                  LD      D,TEST+NOSTRB
01F6'   1E 40                  LD      E,TEST+WRSTRB
01F8'   0E 1D                  LD      C,PADDR
01FA'   CD 0121'               CALL    WRITE
01FD'   21 008B                LD      HL,IMONTH       ; Get the month across ...
0200'   11 02BB'               LD      DE,WSPACE       ; ... to the register workspace
0203'   ED A0                  LDI
0205'   ED A0                  LDI
0207'   3A 008E                LD      A,(IDYWK)       ; Get day of the week
020A'   12                     LD      (DE),A
020B'   13                     INC     DE
020C'   21 0088                LD      HL, IDAY        ; Now the day
020F'   ED A0                  LDI
0211'   ED A0                  LDI
0213'   21 0082                LD      HL,IHOURS       ; Now the hours ...
0216'   ED A0                  LDI
0218'   ED A0                  LDI
021A'   23                     INC     HL              ; ... and the mins
021B'   ED A0                  LDI
021D'   ED A0                  LDI
021F'   CD 0151'               CALL    WRRTC           ; Write out the work space
0222'   21 006C'               LD      HL,STMES        ; Wait for a key press ...
0225'   CD 01C3'               CALL    PRINT
0228'   0E 01                  LD      C,CONIN
022A'   CD 0005                CALL    BDOS
022D'   21 008D'               LD      HL,CRLF
0230'   CD 01C3'               CALL    PRINT           ; ... to start
0233'   3E 01                  LD      A,1             ; Start the RTC
0235'   16 EE                  LD      D,START+NOSTRB
0237'   1E 4E                  LD      E,START+WRSTRB
0239'   0E 1D                  LD      C,PADDR
023B'   CD 0121'               CALL    WRITE
023E'   C9                     RET

                       ; ***************************************************
                       ; ****      Read / Set RTC to / from console     ****
                       ; ***************************************************

023F'   ED 73 02C7'    RTC:    LD      (STK),SP        ; Save the stack
0243'   31 02D9'               LD      SP,STACK
0246'   3A 0080                LD      A,(ARGS-2)      ; Any ARGS given?
0249'   B7                     OR      A
024A'   28 0D                  JR      Z,RTC2
024C'   CD 0261'       RTC1:   CALL    VALID           ; Yes, get and validate input
024F'   30 08                  JR      NC,RTC2
0251'   21 0088'               LD      HL,ERMES        ; Error, so tell 'em
```



```
0254'   CD 01C3'               CALL    PRINT
0257'   18 F3                  JR      RTC1
0259'   CD 016C'       RTC2:   CALL    TIME            ; Else, read clock
025C'   ED 7B 02C7'            LD      SP,(STK)        ; Get the stack back
0260'   C9                     RET                     ; Back to CP/M

                       ; ***************************************************
                       ; ****          Get and validate input           ****
                       ; ***************************************************

0261'   21 0000'       VALID:  LD      HL,TTLMES       ; Put up the caption
0264'   CD 01C3'               CALL    PRINT
0267'   21 0080                LD      HL,ARGS-2       ; Put the max string length at ARGS
026A'   36 19                  LD      (HL),25
026C'   EB                     EX      DE,HL           ; Point to the input buffer
026D'   0E 0A                  LD      C,LININ
026F'   CD 0005                CALL    BDOS            ; Get the input
0272'   21 008D'               LD      HL,CRLF
0275'   CD 01C3'               CALL    PRINT
0278'   21 0082                LD      HL,ARGS         ; Test for time
027B'   7E                     LD      A,(HL)
027C'   FE 54                  CP      'T'
027E'   20 02                  JR      NZ,VALID1       ; No, so validate input
0280'   AF                     XOR     A               ; Yes, so clear any <C> flag
0281'   C9                     RET                     ; Done

                       ; Validate the input string, and convert to nibbles
                       ;       Do hours and mins, then days and months, then day of week
0282'   EB             VALID1: EX      DE,HL           ; Point to ARGS in <DE>
0283'   21 00AE'               LD      HL,INVAL        ; Point to validation table
0286'   06 04                  LD      B,4             ; Do it 4 times
0288'   1A             VALID2: LD      A,(DE)          ; Get the input
0289'   FE 30                  CP      '0'             ; Test if less than 0 ...
028B'   38 25                  JR      C,ERR
028D'   BE                     CP      (HL)            ; ... or greater than (HL)
028E'   23                     INC     HL              ; Point to next (HL) for next time
028F'   30 21                  JR      NC,ERR
0291'   D6 30                  SUB     30H             ; Ok so convert to BCD nibble
0293'   12                     LD      (DE),A; Put it back in workspace
0294'   13                     INC     DE              ; Point to next input ...
0295'   1A                     LD      A,(DE)          ; ... and do it all again
0296'   FE 30                  CP      '0'
0298'   38 18                  JR      C,ERR
029A'   BE                     CP      (HL)
029B'   23                     INC     HL
029C'   30 14                  JR      NC,ERR
029E'   D6 30                  SUB     30H
02A0'   12                     LD      (DE),A
02A1'   13                     INC     DE              ; Skip the interdigit gap
02A2'   13                     INC     DE
02A3'   10 E3                  DJNZ    VALID2          ; More ?
                       ; Do the day of the week, same as before
02A5'   1A                     LD      A,(DE)
02A6'   FE 30                  CP      '0'
```

```
DRIVE ROUTINES FOR THE GEMINI GM822 RTC MODULE    M-80  14 Sep 1982  14:48    PAGE 1-8

02A8'   38 08                JR      C,ERR
02AA'   BE                   CP      (HL)
02AB'   30 05                JR      NC,ERR
02AD'   D6 30                SUB     30H
02AF'   12                   LD      (DE),A
02B0'   18 02                JR      VALID3
                     ; Error trap
02B2'   37           ERR:    SCF                     ; Set the <C> flag to say error
02B3'   C9                   RET
                     ; All valid, so set clock
02B4'   CD 01ED'     VALID3: CALL    SET
02B7'   C9                   RET                     ; All done

                     ; ***************************************************
                     ; ****                Work spaces                ****
                     ; ***************************************************

02B8'   00           WRKSPC: DEFB    0               ; Interrupt / status
02B9'   00                   DEFB    0               ; Stop / start
02BA'   02                   DEFB    2               ; Leap year status
02BB'                WSPACE: DEFS    12              ; Copy of RTC registers

02C7'                STK:    DEFS    2               ; Stack save space
02C9'                        DEFS    10H             ; Stack space
02D9'                STACK:

                             END     RTC
```

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

Macros:

Symbols:
00FF    ALLIN           0000    ALLOUT          0082    ARGS
001C    BASE            0005    BDOS            00FF    CMODE
0001    CONIN           0002    CONOUT          000D    CR
008D'   CRLF            00A0'   DAY             009C'   DAYMN
00B7'   DYSTRG          0088'   ERMES           02B2'   ERR
01B2'   HMS             0090'   HOURS           0088    IDAY
008E    IDYWK           0082    IHOURS          0085    IMINS
008B    IMONTH          00F9'   INADDR          00F0'   INDATA
000F    INSTAT          0010    INTOUT          00AE'   INVAL
000A    LF              000A    LININ           00CC'   MNSTRG
00A3'   MONTH           00E0    NOSTRB          010B'   OUADDR
0102'   OUDATA          001D    PADDR           001F    PCADDR
001E    PCDATA          001C    PDATA           01D1'   PLSTRG
01C3'   PRINT           012E'   RDRTC           013F'   RDRTCL
0080    RDSTRB          0118'   READ            023FI'  RTC
024C'   RTC1            0259'   RTC2            01ED'   SET
02D9'   STACK           000E    START           02C7'   STK
006C'   STMES           0000    TEST            016C'   TIME
01E3'   TIME10          01E9'   TIME3           000C    TMONTH
0000'   TTLMES          0261'   VALID           0282'   VALID1
0288'   VALID2          02B4'   VALID3          0121'   WRITE
02B8'   WRKSPC          0151'   WRRTC           0162'   WRRTCL
0040    WRSTRB          02BB'   WSPACE

No Fatal error(s)
```

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

                             TITLE READ ROUTINE FOR THE GM822 RTC

                             .Z80
                             GLOBAL  TIME

001C                         BASE    EQU     1CH     ; Base address of control PIO

                             ; Port assignments
001C                         PDATA   EQU     BASE    ; Data port for commands
001D                         PADDR   EQU     BASE+1  ; Data port for commands
001E                         PCDATA  EQU     BASE+2  ; Control port for data
001F                         PCADDR  EQU     BASE+3  ; Control port for data

                             ; Control instructions
00FF                         CMODE   EQU     OFFH    ; Control mode
00FF                         ALLIN   EQU     OFFH    ; All inputs
0010                         INTOUT  EQU     10H     ; All out except INT input

                             ; Addresses for MM58174

000C                         TMONTH  EQU     12
00E0                         NOSTRB  EQU     OE0H    ; No strobe or select
0080                         RDSTRB  EQU     80H     ; Read strobe on

                             ; ***************************************************
                             ; ****          PIO control subroutines          ****
                             ; ***************************************************

                             ; Set data port to control mode, all lines input
0000'   3E FF                INDATA: LD      A,CMODE         ; Control mode
0002'   D3 1E                        OUT     (PCDATA),A      ; Set mode word
0004'   3E FF                        LD      A,ALLIN
0006'   D3 1E                        OUT     (PCDATA),A      ; Set direction
0008'   C9                           RET

                             ; Set command port to control mode, all lines input
0009'   3E FF                INADDR: LD      A,CMODE         ; Control mode
000B'   D3 1F                        OUT     (PCADDR),A      ; Set mode
000D'   3E FF                        LD      A,ALLIN
000F'   D3 1F                        OUT     (PCADDR),A      ; Set direction
0011'   C9                           RET

                             ; Set command port to control mode, all lines output
                             ;       EXCEPT the INT line (OB4H)

0012'   3E FF                OUADDR: LD      A,OFFH          ; Ensure all data lines ...
0014'   D3 1D                        OUT     (PADDR),A       ; ... will be 1's
0016'   3E FF                        LD      A,CMODE         ; Set mode
0018'   D3 1F                        OUT     (PCADDR),A
001A'   3E 10                        LD      A,INTOUT        ; Set direction
001C'   D3 1F                        OUT     (PCADDR),A
001E'   C9                           RET
```

```
                        ; ***********************************************
                        ; **** Subroutine to read to the MM58174     ****
                        ; **** Called with:-                         ****
                        ; ****         C=PADDR - data port for command ****
                        ; ****         D=RTC register address        ****
                        ; ****         E=RTC reg. + strobe (R or W)  ****
                        ; ***********************************************

                        ; Read a MM58174 register

001F'   ED 51           READ:   OUT     (C),D           ; Set up address
0021'   ED 59                   OUT     (C),E           ; Read strobe on
0023'   DB 1C                   IN      A,(PDATA)       ; Read a nibble
0025'   ED 51                   OUT     (C),D           ; Strobe off
0027'   C9                      RET

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

0028'   CD 0012'        RDRTC:  CALL    OUADDR          ; Set command port to o/p
002B'   CD 0000'                CALL    INDATA          ; Set data port to i/p
002E'   2A 007E'                LD      HL,(PTR)
0031'   06 0C                   LD      B,12            ; 12 registers to read
0033'   16 EC                   LD      D,TMONTH+NOSTRB ; 10's of months + no strobe
0035'   1E 8C                   LD      E,TMONTH+RDSTRB ; 10's of months + read strobe
0037'   0E 1D                   LD      C,PADDR         ; C = control port address
0039'   CD 001F'        RDRTCL: CALL    READ            ; Get the data
003C'   E6 0F                   AND     0FH             ; Isolate the lower nibble
003E'   FE 0F                   CP      0FH             ; Valid?
0040'   28 E6                   JR      Z,RDRTC         ; No, so start again
0042'   77                      LD      (HL),A          ; Store in workspace
0043'   23                      INC     HL
0044'   15                      DEC     D               ; Next RTC register
0045'   1D                      DEC     E
0046'   10 F1                   DJNZ    RDRTCL          ; Loop if more
0048'   C9                      RET                     ; Done

                        ; ********************************************
                        ; ****  Convert nibbles into single byte  ****
                        ; ********************************************

0049'   7E              BCD2BN: LD      A,(HL)          ; Get 10's
004A'   23                      INC     HL
004B'   CD 0054'                CALL    TIME10
004E'   46                      LD      B,(HL)          ; Get units
004F'   23                      INC     HL
0050'   80                      ADD     A,B
0051'   12                      LD      (DE),A          ; Save binary
0052'   13                      INC     DE
0053'   C9                      RET
```



```
                        ; ***********************************************
                        ; ****          Multiply <A> by 10           ****
                        ; ***********************************************

0054'   07              TIME10: RLCA                    ; Multiply by 2
0055'   47                      LD      B,A             ; Save the partial product
0056'   07                      RLCA                    ; Multiply by 2 again
0057'   07                      RLCA                    ; Multiply by 2 again
0058'   80                      ADD     A,B             ; Add the partial product
0059'   C9                      RET

                        ; ***********************************************
                        ; **** Get the current date and time in the  ****
                        ; **** form:- Mth Dy Hr Mn Sc                ****
                        ; ***********************************************

005A'   5E              TIME:   LD      E,(HL)          ; Get the address of TM$
005B'   23                      INC     HL
005C'   56                      LD      D,(HL)
005D'   EB                      EX      DE,HL
005E'   22 007E'                LD      (PTR),HL        ; Save it in PTR
0061'   CD 0000'                CALL    NDATA           ; Set PIOs to RESET state
0064'   CD 0009'                CALL    INADDR
0067'   CD 0028'                CALL    RDRTC           ; Get data into workspace
006A'   2A 007E'                LD      HL,(PTR)        ; Get back PTR
006D'   54                      LD      D,H             ; Put HL in DE
006E'   5D                      LD      E,L
006F'   CD 0049'                CALL    BCD2BN          ; Do months
0072'   ED A0                   LDI                     ; Do DYWEEK
0074'   06 04                   LD      B,4
0076'   C5              TIME1:  PUSH    BC
0077'   CD 0049'                CALL    BCD2BN          ; Do days, hours, mins, secs
007A'   C1                      POP     BC
007B'   10 F9                   DJNZ    TIME1
007D'   C9                      RET                     ; All done

007E'                   PTR:    DEFS    2

                                END
```



```
Macros:

Symbols:
00FF    ALLIN           001C    BASE            0049'   BCD2BN
00FF    CMODE           0009'   INADDR          0000'   INDATA
0010    INTOUT          00E0    NOSTRB          0012'   OUADDR
001D    PADDR           001F    PCADDR          001E    PCDATA
001C    PDATA           007E'   PTR             0028'   RDRTC
0039'   RDRTCL          0080    RDSTRB          001F'   READ
005AI'  TIME            0076'   TIME1           0054'   TIME10
000C    TMONTH          000E    TSTART

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 When the random digit appears,press the same digit on the keyboard. Better than 0.6 seconds is good - how good are you?	0C80-0FAD	0.20
S3	Bouncing Beastie A character bounces around the screen leaving a trail behind it.	0C80-0CFC	0.05
S4	Reverse Arrange the randomly generated list of numbers in order in the fewest moves by reversing any number of them at a time.	0C80-0F99	0.25
S5	Double Mastermind Try and crack the Nascom's four digit code while it tries to crack yours. (It won't let you cheat!)	0C80-0FB9	0.35
S7	JJ Enter the machine code of this before looking to see how it works. Written to demonstrate machine code string handling.	0C80-0E0F	0.20
S8	Unizap Recreate the life of the Universe. Take it from the 'big bang' to its 'ultimate destiny' by shooting stars.	0C80-0ED9	0.25
S9	Random Buzz-word Linker Short of jargon? Then use the 'Functional flexibility' of this program to give you 'limited transitional communication' in your 'balanced digital engineering' !!	0C80-0F3E	0.20
S10	Quiz Program (+Italian file) What is Italian for 'please' or English for 'Fratello'? Find out with this quiz. Also additional files (see below).	0C80-0DCB (+0DD0-0FD5)	0.25
S11	Cockney Rhyming Slang (for S10)	0DD0-0FD6	0.10
S12	Currency Types (for S10)	0DD0-0FC0	0.10
S13	Capital Cities (for S10)	0DD0-0FE1	0.10
S14	Ship Game Blow up ships using your torpedoes. Ships come at random speed, distance & direction.	0C80-0FA4	0.30
S15	Hangman Can you guess the word before you are hung? Can add own words.	0C80-0FA4	0.25
S16	Memory Test 1 Tests for addressing faults	0F00-0F6F	0.10
S17	Memory Test 2 Tests for bit faults/pattern sensitivity.	0F00-0F61	0.10
S24	Octal-Hexadecimal Conversion	0C80-0CE8	0.10
S28	Decimal-Hexadecimal Conversion	0C80-0D2B	0.15
S34	Random I-Ching Characters Great mystical significance (for those who understand these things).	0C80-0D25	0.10
S36	Sub-Search Ship traverses screen dropping random depth charges on randomly moving submarines.	0C80-0DA3	0.20

Nasbug Programs

T1	Crash Land the spacecraft. Real time. Can you make NASA proud of you.	0C50-0FC9	0.35
T2	Reaction Test (as S2)	0C50-0FCC	0.25
T3	Bouncing Beastie (as S3)	0D00-0DAB	0.10
T4	Reverse (as S4)	0C60-0F88	0.25
T5	Double Mastermind (as S5)	0C50-0FD5	0.30
T6	Possum 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.	0D00-0D84	0.10
T7	JJ (as S7)	0C50-0DE5	0.20
T9	Random Buzz-words (as S9)	0C50-0F35	0.20
T10	Quiz Program + Italian (as S10)	0C50-0DAA	0.25
T21	TV Test Pattern Displays test pattern to set up TV.	0D00-0D19	0.05
T22	NIM The classic game of NIM.	0D00-0FC4	0.45
T23	Attack Shoot the descending aliens before they get you!	0D00-0E7F	0.20
T24	Octal-Hexadecimal (as S24)	0D00-0DBC	0.15
T26	24 Hour Digital Clock	0D00-0D7B	0.15
T28	Decimal-Hexadecimal (as S28)	0C90-0D07	0.10
T29	Life The ubiquitous simulation of Life, using special characters to give an expanded universe. Very fast.	0C50-0FA0	0.20
T31	Compact Editor	0F70-0FDF	0.05
T32	Carre Chinois We have no idea of what this game is or what it does. But it is totally written and extensively commented in French!	0C50-0F3F	0.40
T33	Lollypop Lady Trainer A classic! See if you can get the chickens across the road without them getting run over.	0C60-0F95	0.50
T34	Random I-Ching (as S34)	0C50-0CF3	0.15
T35	Walled Chase. One player chases another, but there are invisible walls in the way which appear when you hit them. Compulsive.	0C50-0E27	0.50
T36	Sub-Search (as S36)	0E0B-0F31	0.25
T37	Random Display Random patterns of asterisks.	0C60-0C84	0.05
T38	Submarines Hit the randomly displayed submarines with your steerable depth charges.	0C60-0E50	0.40
T39	Burst The Balloon Shoot the balloons as they make their way up the screen at the mercy of random breezes. Frustrating.	0C50-0E02	0.50

Nascom 8K BASIC Programs

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

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 pigmies. They want it back and are in hot pursuit. Can you reach safety before the pigmies (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

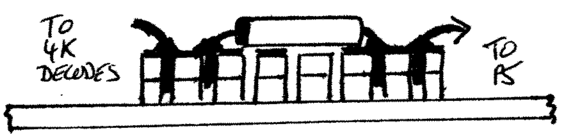
John Harrison

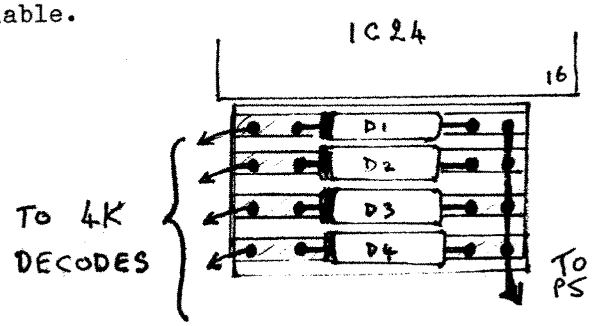
I decided to modify my system to take 4K EPROMs, and having read the two articles in INMC80-5 I chose the RAM board as an easier mod. than the main board. In practice, it turned out to be rather more complicated by the time I had it working.

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 slects (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 slect 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.

IC24 is made redundant by the mod.





REMOTE TERMINALS AND THE GALAXY

D. W. Parkinson

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 succesfully 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 transfered 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!

```
        .z80
        title Remote terminal emulation
;
;       Remote terminal Emulation for GEC 4065.
;

escape  equ     1bh
uartd   equ     0b8h            ; 8250 data port
uarts   equ     uartd+5         ; 8250 status port
ivcd    equ     0b1h            ; IVC data port
ivcs    equ     0b2h            ; IVC status port

;       **** Uart Drivers. ****
;
; POLLU - polls the Uart and moves any waiting
;       character to (DE)

pollu:  in      a,(uarts)       ; Check status
        and     01h
        ret     z               ; Return if no character
        in      a,(uartd)       ; Get it
        and     7fh             ; Strip msb
        ld      (de),a          ; Store in the buffer
        inc     de              ; Bump pointer
        ld      a,d             ; Check for limit
        cp      2
limit   equ     $-1
        ret     nz              ; Return if not there
        dec     de              ; Else don't advance pointer
        ret

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

serout: push    af
sero0:  in      a,(uarts)       ; Check ready to accept
        and     20h
        jr      z,sero0         ; wait if not
        pop     af              ; Get character back
        or      a               ; Check parity
        jp      pe,sero1
        or      80h
sero1:  out     (uartd),a       ; Send character
        ret

;       **** 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
        rlca
        jr      c,getvid        ; Loop if busy
        in      a,(ivcd)
        ret

putvid: push    af
pv0:    call    pollu           ; Service the serial port
        in      a,(ivcs)        ; See if IVC ready
        rrca
        jr      c,pv0
        pop     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
        ld      a,escape        ; Send ESC "K" to get it
        call    putvid
        ld      a,'K'
        call    putvid          ; Read the character
        call    getvid
        cp      'C' and 1fh     ; ^C?
        ret     nz              ; No, return with NZ set
        jp      0               ; 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
loop0:  ld      de,buffer
        ld      hl,buffer
loop1:  call    const           ; Check keyboard....
        call    nz,serout       ; ...and echo any character
        push    hl              ; See if anything in the buffer
        or      a
        sbc     hl,de
        pop     hl
        jr      z,loop0         ; No, zero pointers
        ld      a,(hl)          ; Yes, get character...
        inc     hl
        call    putvid          ; ...and echo it
        jr      loop1

;       Workspace

        defs    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.

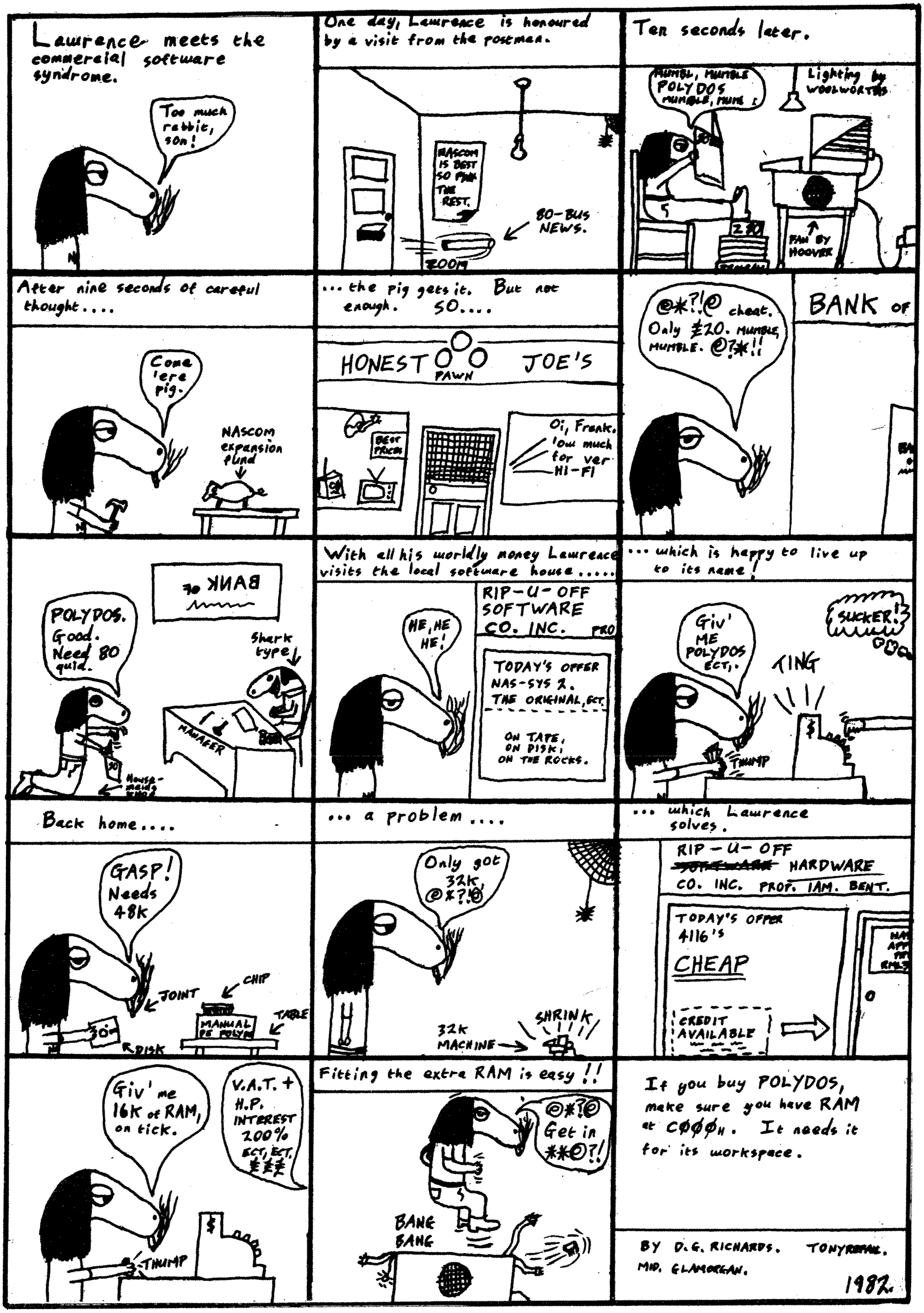
Unfortunately, for a number of reasons, NM went into receivership. Under receivership there was little advertising or production done, as a buyer was sought. Consequently the market waned considerably, and a number of dealers (including one owned by John Marshall (JM), the ex-MD of NM) got together to 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 subsription 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!!


Lawrence meets the commercial software syndrome.
Too much rabbit, son!
One day, Lawrence is honoured by a visit from the postman.
80-BUS NEWS.
ZOOM
Ten seconds later.
MUMBL, MUMBLE POLYDOS MUMBLE, MUMB
Lighting by WOOLWORTHS
FAN BY HOOVER
After nine seconds of careful thought....
Come 'ere pig.
NASCOM expansion fund
... the pig gets it. But not enough. SO....
HONEST JOE'S
PAWN
BEST PRICES
Oi, Frank. 'ow much for yer HI-FI
@*?!@ cheat. Only £20. MUMBLE, MUMBLE. @?*!!
BANK OF
POLYDOS. Good. Need 80 quid.
Shark type
MANAGER
With all his worldly money Lawrence visits the local software house.....
RIP-U-OFF SOFTWARE CO. INC.
HE, HE HE!
TODAY'S OFFER NAS-SYS 2. THE ORIGINAL, ECT.
ON TAPE, ON DISK, ON THE ROCKS.
... which is happy to live up to its name!
Giv' ME POLYDOS ECT.,
SUCKER!
TING
THUMP
Back home....
GASP! Needs 48k
JOINT
CHIP
TABLE
DISK
... a problem....
Only got 32k @*?!@
SHRINK
32k MACHINE
... which Lawrence solves.
RIP-U-OFF ~~SOFTWARE~~ HARDWARE CO. INC. PROP. I AM. BENT.
TODAY'S OFFER 4116's CHEAP
CREDIT AVAILABLE
Giv' me 16K of RAM, on tick.
V.A.T. + H.P. INTEREST 200% ECT, ECT, £££
THUMP
Fitting the extra RAM is easy!!
@*?@ Get in **@?!
BANG BANG
If you buy POLYDOS, make sure you have RAM at C000H. It needs it for its workspace.
BY D.G. RICHARDS. TONYREFAIL. MID. GLAMORGAN.
1982.

80-BUS NEWS SERVICES

BACK 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.

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 — phone 01-402 6822
404, Edgware Road,
London W2 1ED — 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.

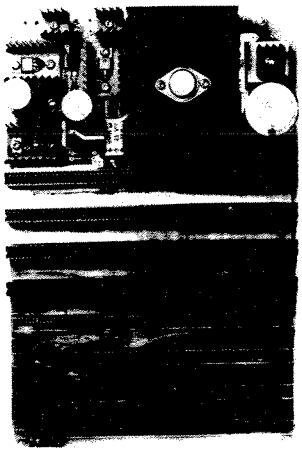
COMING SOON
Low cost 80 column video card for Nascom.

MAP 80
Systems. Ltd.

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

Gemini Microcomputers Ltd

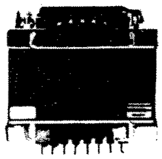
GEMINI 'SUPERMUM AND 'HYPERMUM'



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	Eprom/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 TP1	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 Processer	£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.

Access

VISA

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