

# 80-BUS NEWS

MAY-JUNE 1984

VOL. 3 ISSUE 3

- The 80-BUS 800 Series
- IVC/SVC Insight - Part 2
- Improved Nas-Sys 3



The Magazine for  
**GEMINI & NASCOM USERS**

£1.50

May-June 1984.

# 80-BUS NEWS

Volume 3. Issue 3.

CONTENTS

Page 3	Editorial
Page 4	GEMPEN/DISKPEN - a Review of the New Version
Page 10	64K, or 128K on a RAM A Board
Page 16	An Insight into the Gemini IVC and SVC - Part 2
Page 23	NASCOM BASIC Disassembled - Part 7 - The End
Page 31	Dave Hunt's Bits
Page 38	The 80-BUS 800 Series - Part 1
Page 44	An Improved Nas-Sys 3
Page 47	Private Sales
Page 48	Random Rumours (and Truths?)
Page 49	Lawrence
Page 50	Ads

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

SUBSCRIPTIONS

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.

PRIVATE SALES

Free of charge to Subscribers by sending to:

ADDRESS: 80-BUS News,  
c/o Gemini Microcomputers Ltd.,  
Unit 4, Springfield Road,  
Chesham, Bucks. HP5 1PU.

## EDITORIAL

### **Questionnaire**

Well, I hope that it all went to plan, or else I'm about to make a fool of myself - but then that's nothing new! The theory is that with the last magazine a Questionnaire went to all of the subscribers to 80-BUS News. (The reason for my uncertainty is that this is being written before the other mag. has come back from the printers.) Now, the purpose of the Questionnaire (from now on referred to as 'Q') is many-fold. Firstly it is to find out what equipment you are using - this will then let us know whether an article about, for example, the Polydos operating system, will be read with interest by the 25 per cent (say) of the readership that runs Polydos, or just by the author! Secondly there is a section asking what type of articles you like - this will influence whether we write more about hardware, software or more reviews etc. Thirdly, the section on your likely expansion and product desires will give us an indication of what general direction we should be heading in with you. Fourthly, depending on how modest you are (or aren't) in answering the 'knowledge' part, we can establish the average level of competence, which will let us know how simple or complex we should get. And finally it will keep someone here (yes Dave, you've been volunteered!) out of mischief for a considerable time - assuming we actually get some response.

### **The Prize**

When the 'Q' went to press we decided that, as we usually get a pitifully low response to requests for input, we needed to offer a prize as an incentive. At the time of writing the 'Q' we didn't know what this should reasonably be. Since then Gemini has stepped in and offered a prize of £100 (inc. VAT) of Gemini 'Goodies' in the form of a voucher that can be redeemed at any Gemini dealer. In addition to this we will give 10 lucky people a free years subscription to 80-BUS News. So now you have 12 amazing reasons for sending in that 'Q' that is still sitting next to your computer:

- 1 - the chance of £100 worth of goodies,
- 2-11 - the chance of a free sub.,
- and 12 - the opportunity of being able to influence the contents of the mag. (Yes YOUR opinion really does count.)

Being the eternal pessimist what I am, I personally doubt that we'll receive more than a 20 per cent return rate, so PROOVE ME WRONG. After all, what weight will "70 per cent (of the 3 per cent of the readership that actually replied) wants the entire mag. to be about Comal" carry? Anyway....

### **This Issue**

Somewhere in this issue you will find one of David Hunt's ramblings. Now the layout used is an experiment, and we would like your reaction to it. Please just scribble a note on your still un-returned 'Q', on your re-subscription, on an article, or on a post-card, saying what you think. The cost of printing this mag. has slowly been creeping up and up, and we need to cover this. There are several choices: a) put up the subs (never a very popular idea), b) get the magazine typeset and thus smaller (difficult to justify for the relatively low number of magazines produced and for the additional cost, delays and loss of control incurred), or c) reduce the number of pages. I favour the latter, but one way of keeping the actual quantity of material the same would be to treat most of the magazine the same way as Dave Hunt's article (i.e. fit two pages onto one). Now even I don't know exactly what it'll look like until the magazines are returned from the printer, so I hope it is legible - please let us know what you think. Be seeing you.

P.S. For those of you puzzled by this issue's front cover, the answer is really quite simple. Read David Parkinson's article and think about one of the new features of the SVC!! Your answers with the next article you submit please.....

**GEMPEN/DISKPEN - A Review of the New, Improved Version****By C. Bowden**

A short while after I got my NASCOM 2, I saw NASPEN advertised, and I bought a copy. Having no Disks or Printer at that time, I really did not need a Text editor/formatter and NASPEN remained largely unused. The built in editors of ZEAP and BASIC fulfilled all of my needs. When I upgraded my system to IMP Printer, Disks and CP/M though, I required an editor to write Assembly Language source files. I had to learn to use ED for this. ED is very powerful, but also very clumsy to use, after cursor type editing. (At least both the Gemini CP/M and SYS allowed me to screen edit MBASIC source code.) Then DISKPEN, a revised NASPEN running under CP/M, became available. I bought one, and soon learned to use it to write my Source files, and odd letters, and also started to do some real Text editing. After a while I added an IVC card to my system, and upgraded to GEMPEN at a very reasonable cost, to take full advantage of the 80 wide screen.

This situation continued until a few months ago when a new improved GEMPEN/DISKPEN was advertised. (From now on I will call it 'PEN'.) The cost of upgrading was again very reasonable, so I got the new version. The improvements are certainly worth while. Whilst previous versions will do most of the things that the average user needs, there were certain bad points. For example use of "D" instead of "d" could have fairly disastrous consequences. Such 'CASE' related commands have now been removed. Another big improvement has been in the provision of a 'HELP' command. Customization information is also provided. Another feature that greatly increases the scope and power of PEN is that it is also possible to load and run 'Overlay' files.

In 80BUS NEWS Vol 2. Iss.5 the Editor refers to WORDSTAR which is a very well known text processor program that runs under CP/M. I have used WORDSTAR, and it is a VERY powerful program. It can deal with Files longer than RAM Memory, which PEN cannot. Due to its power and complexity, there is quite a lot to learn if one is to become reasonably proficient, and I have not really devoted that much time to this, probably because PEN can do just about all that I require anyway, and is much more suited to system hardware, unless a lot of customizing is done to Wordstar. It is really a case of 'Horses for Courses'. If one needed to write anything longer than about 30-40K, or do something really 'Fancy', then WORDSTAR would be the better choice. The real 'Clincher' for most users would be the price, which is in favour of PEN by about 5 to 1.

PEN is available in a number of versions, but those to suit GEMINI or NASCOM Hybrid systems are the same. Minor differences can be patched as required. The PEN module is supplied on disk together with a number of other files. These are:

- |                                  |         |                            |
|----------------------------------|---------|----------------------------|
| 1) GEMPEN.COM or DISKPEN.COM     | (10K)   | Main Module.               |
| 2) PEN1.DOC, PEN2.DOC & PEN3.DOC | (98K)   | Program Documentation.     |
| 3) CONFIG.DOC                    | (22K)   | System Configuration Docs. |
| 4) SPOOL/MAXIFILE/.DOC; etc.     | (26K)   | DOC's on OVERLAYS.         |
| 5) EPSON/CENT/NEC.LST            | (8K ea) | .PRN's for Printer Config. |
| 6) HELP.OVL                      | (14K)   | HELP Overlay.              |
| 7) OVERD/MENU.OVL                | (2K ea) | More overlays.             |

It will be seen from the above that extensive .DOC files are provided. Persons experienced in assembly language will be able to alter and Patch the main module to support other printers, or to change other system defaults. These may be patched using a debugging tool such as ZSID/DDT/GEMDEBUG. For example,

I quickly altered my main module to support a GM827 keyboard, attached to my IVC Card. This is supported by PEN, but is not 'switched on' in PEN when supplied. I have also altered the default 'lines per page' to 20, and 'Line width' to 76 as I find 20 line pages ideal for 'skipping through' [Ed. - Mr. Bowden hasn't found the ^B and ^C commands.] and 76 wide does not seem to stop for Hyphenation so often. At the time of writing this, I have just bought an EPSON FX80 Printer, and I hope to change a few things in the printer tables. PEN contains tables for Proportional Printers but like the option for the GM827 keyboard, this is also switched off when supplied, also the tables are for Centronics and NEC printers, which will probably turn out different to those used by the FX80. To fully support reconfiguration though, it is necessary to alter HELP.OVL, and this is not easy, as the small bits of machine code in the file prevent it loading with an Editor. To make the support 'suite' complete, I feel that the source for HELP.OVL ought to have been supplied, since all of the other 'tools' to make alterations are supplied.

PEN can be entered in several ways. If the command PEN (Enter) is given, then it is assumed that a new file is being started and a default name \$\$.PEN will be allocated. If a name is given, e.g. PEN STD.MAC - then the named file will be loaded if found on the logged in disk. If not found then the question NEW FILE ? Y/^C is asked. ^C allows recovery from a spelling mistake, whilst a "Y" will open a new file of the given name. Normal CP/M conventions are supported so that:

A>B:PEN A:TEST.DOC is quite legal.

Note however that unless HELP.OVL is on B:, current USER or USER 0, or on A: USER 0, it cannot be loaded, so it is best to put PEN, and any .OVL files onto the default or A: disk before starting. Another good feature of PEN is that it much more forgiving than most Programs if a Disk suddenly 'fills up' since PEN allows one to change Disks without rebooting.

The top line of the screen displays the version number, current cursor line, and Bytes free. If the cursor is off the screen, an indicator shows whether it is before or after the displayed data. The bottom two screen lines are command status lines. This leaves 22 lines (IVC Version) free for text display.

In common with most text Editors, PEN operates in two modes - Command and Insert, and will be in Command mode when run up. Over 60 commands are available, although many are infrequently used. The commands fall into several groups, relating to specific functions such as Cursor movement, text formatting or disk access. The various groups may be displayed on the screen by calling up the HELP Overlay, by means of the "?" or ^E command. When this is done the lower part of the screen is cleared of text and a 'MASTER MENU' of 12 groups is displayed. Entry of the key letter will then call up a second, more detailed and specific display describing commands relevant to that group. It is thus possible to refresh one's memory on all commands at any time. When the HELP display is no longer needed, it may be removed from the screen.

The Insert mode is entered by either the I (i) - Insert, or A (a) - Append command. If I is used the cursor does not move within the text, whereas the 'A' command moves it to the 'end of text' position so is more suited to adding text at the end of an existing file. Either command causes the word 'Insert' to be displayed on the status lines at the bottom of the screen. The cursor is displayed as a flashing Left arrow in Inverted video and it is non-

destructive. It is best to consider that ALL operations will occur at the place pointed to by the arrow. e.g. BS will delete the letter pointed to and characters under/after the cursor will move left/up. Any typed Characters will be inserted in the 'gap' pointed to, and text under/after the cursor will move right/down. This is different to editing in BASIC etc. where characters will 'overtype' unless space is created by shift/right arrow. The Insert mode is terminated by a ^Z.

A number of commands (also used in Insert mode) are available to move the cursor forwards or backwards through the text by units of one character/word/line/paragraph/Quarter page. The main cursor commands are Right and Left arrow, to move one character, Shift Right or Shift Left arrow to move one Word, Up or Down arrow to move one (logical) line, Shift Up or Shift Down arrow to move one Paragraph. In command mode, text can also be scanned in 'Pages'. The length of a page can be changed from the default value by using the '3' & '4' commands, to suit screen or Printer output. Page markers can then be inserted into the text, or removed at will, by simple commands. If Pages are marked forward and backward scanning is supported (8 or 9 commands). Any page may be printed (P command) and if the cursor is not moved, repeated printing of a Page is possible. More sophisticated printing is allowed under the Spool overlay.

It is important to realize that a line MAY NOT mean one line as seen on the screen. One normally enters text into a file by just typing words, spaces and punctuation, but WITHOUT pressing ENTER at the end of each screen line. The text will 'wrap around' onto the next line of the screen, etc., although some words may appear broken when the end of screen line is reached. In memory though, the text will be continuous. Thus a paragraph of say 200 words may be entered, and the cursor may be half way down the screen but the line counter will still say line 1, because the logical end of line is marked by a CR,LF sequence, which results from pressing ENTER, or from formatting. Half a dozen such paragraphs may occupy several K. of memory and take two or three screens to display fully, but may only register six lines on the status display.

When the text is formatted to the desired length however, the program will place a CR,LF at the end of every line of 76 (or whatever length set - N.B. These are .1" steps, NOT characters), and the line counter will display the correct count. Since the screen can only display 48 or 80 chars. on a line, any lines formatted longer than this will 'wrap around', and this will result in the line count indicator not agreeing with lines as counted on the screen.

Another consequence of the lack of CR,LF before formatting is that the UP/DOWN arrow keys will cause the cursor to jump the whole length of the logical line. This makes reaching the middle of a paragraph by cursor character or word commands tedious. The answer is to format frequently, or to press ENTER or RETURN fairly often when near the Right hand side of the screen. Formatting takes longer as the amount of text grows, and it is not necessary to put the cursor to the start of the text each time a format is done. It can be placed at the end of the previous formatted part. If formatting is done on the last one or two paragraphs entered, it is very quick. The whole text only need be reformatted if changes are made near the start, or some other change is needed such as a different line length.

A minor criticism of PEN as far as I am concerned is that after formatting the cursor always moves to the start of the text. I would have preferred it to display from the position at which the format command was given. This is usually what I want to see after formatting so I have to step through the text until I find the required position again, which is a little annoying, although use of '\*' and '=' commands (which are cursor position markers) would probably help. Source Code files are not formatted of course, and normally ENTER would be pressed at the end of each line.

In Insert mode anything typed except cursor commands and BS will be placed into the file. This includes a number of non Alpha-Numeric characters, which have a special meaning, and are 'embedded' commands to control formatting or printing. It is thus possible to indent, exclude from formatting, cause a printer to change character set or print mode and so on. PEN has around 16 commands to control printing.

A nice feature of PEN is that TAB characters show on the screen as a row of little dots. Since 'TABs' can upset formatting, it is useful to be able to 'see' them. (A formattable TAB character (^A) is available as an alternative.) Any area of text may be marked (^U) to exclude it from formatting, so that tables or lists that would be destroyed by normal formatting may be included. The ^U command is cancelled by a ^F command, to allow formatting to recommence. PEN also supports an Indent command (^R), which indents in steps of 4 spaces, and appears on the screen as (different) dots. This can help in building tables or lists, or just to indent the whole text. The Indent is reduced by the ^X command. If the commands are placed into the text in insert mode, variable amounts of indent are possible. In command mode, the ^R and ^X commands operate on the whole text, so could be used to set a print margin, for example.

#### Some Other Commands

It is impossible to describe all of the commands in a review, so only a few will be described in any detail. Many commands are straightforward, no option commands. Potentially fatal commands are protected by 'Are you sure?'.

#### Block Move/Delete

Any block of text may be marked with a 'backslash' character at start and end. If the cursor is then moved to another place, the 'M' command will cause the marked block to be inserted at the cursor position. The original block is not deleted. The markers may be removed by the 'N' command. Alternatively a '^D' command will allow block(s) to be deleted, but PEN first checks that it has the right block, as there could be several marked blocks throughout the text. Using backslash for the move markers can be inconvenient to the C programmer as these are used as remark delimiters within C. The information supplied with PEN makes it an easy task to choose some other less used character instead.

#### Change Keyboard Case

The 'U' command will reverse the Keyboard case, which is very useful since this saves much operation of the Shift key, depending on the type of file being edited. With the GM827 keyboard, which has a 'CAPS LOCK' key, it is possible to permanently set Upper case or Lower case, or to have Lower case/Shift upper or Upper case/shift lower. With a NASCOM and GEMINI keyboard in use together, a number of parallel options are possible!!

Kill Text

'B' will kill all text from cursor to the start of text, while 'K' will kill from cursor to the end of text. Smaller amounts may be removed by using the Block Delete command. The 'D' command will kill text from cursor back to the previous 'New line', and so could kill several displayed lines. (See comment on lines above.)

Disk Commands

At any time a copy of the text in memory can be saved to disk by use of the 'W' command. Control returns to PEN. When the edit session is finished, the 'E' command will write the file to disk, and exit to CP/M. Editing can be terminated without writing to disk, by use of the 'Q' Quit command, but in this case the question 'Are you sure' is posed. In common with most programs of this type, two disk copies of a file are maintained, the most recent always becoming the Named file, whilst the other becomes the .BAK - Backup file. When the next edit is completed, the old .BAK will be deleted, and the previous master file will be renamed FILENAME.BAK. The newly edited file will saved as the new master, under the correct FILENAME.TYP. If the user wishes to retain the .BAK file on the disk before an edit, then it can be renamed. i.e. -  
REN MYFILE1.DOC=MYFILE.BAK.

An unusual and very useful feature is the 'O' command, which change the name of the output file from within PEN. very useful for breaking bits of source code up into constituent parts, etc.

Another very useful command is the 'R' command. This will read another file from disk, and insert it into the file currently in memory, at the Cursor position. This is useful for things like Letter Heads, or EQUATES Source file listings.

Find Commands (with Replace option)

When activated, PEN prompts for the 'Find' string on the status lines. The required string should then be entered, and terminated with ^Z or ^C. A ^Z causes PEN to immediately start searching. If ^C is used, a second String is expected, and should be entered, terminated by ^Z. If a second string is entered, it can be used as the replacement string as described below.

The 'F' command will search the entire file, ('f' from current cursor position) and display any text that matches the 'Find String'. It will then stop. If 'ENTER' is pressed the next occurrence is found. If the 'Replace' option is active, then pressing 'ENTER' will cause the String to be changed for the replacement option, and a further 'ENTER' will find the next occurrence. Whenever a match is found, if 'SPACE' is pressed instead of 'ENTER' the substitution is NOT made, and the next match is sought. The last entered Find command can be re-activated, even after further editing, by use of the '^F' command. Apart from changing spelling mistakes, or replacing abbreviations, one nice use of this command is to rewrite 8080 Source files to Z80 Mnemonics.

Overlays

Overlays are a new feature of PEN. I am unable to comment on these in much detail because I have only had experience of the two supplied with PEN. The idea is good though, and it adds greatly to the power and versatility of PEN. The OVERLAY 'Menu' is activated by the '&'amp; command, which displays a list of overlays that may be available and used. The 'OVERD' overlay for example, will display the Directory of a Disk, or allow deletion of files.

Some Other Commands

'1','2'	- Alter line length - .lin steps (for formatting).
'V','v'	- Change screen format (48/80 col)/Remove 'Menu'.
'*'	- Place a marker in text.
'='	- move cursor to marker and delete it.
'0'	- Allows renaming of the output file.
'J'	- Insert single Character. Avoids 'I','^Z' sequence.
'P'	- Print File. (To next Page marker, if any exist.)
'G','^G'	- Insert (Remove) Page Markers.
'Z','(Y)'	- Move to Start (End) of text.

N.B. A few commands do not apply to all Versions of PEN, due to possible Hardware differences, particularly Screens and Keyboards.

Formatting

Although only a few commands relate to formatting, this is one of the main features of PEN. With a Formatter, one can type away without attempting to regulate the Right hand edge of the display, or relating to Compressed Print on a 15 inch Printer. One simply continues typing until say, a paragraph has been entered. At this stage, the screen display will be 'ragged' with some words broken at end of (screen) line. If the formatter is now invoked, the text will be adjusted so that all lines are of the same (preset) length, without any broken words. To achieve this PEN will add some extra 'space' characters to the line, at positions where a single space already exists, until the total count is correct. The net result is a very pleasing display. Occasionally PEN cannot continue because a word at end of line is too long, and there are not enough single spaces in the line for formatting. The text is displayed with the cursor pointing to the offending word, and one can hyphenate ('-' command), or edit to get around the problem. PEN is not permitted to automatically hyphenate as this could result in some odd effects. Three or more adjacent spaces are not adjusted, as these could be Indents, Tabs or similar required gaps. Two adjacent ENTER characters are not altered as this is taken to indicate end of paragraph.

The commands associated with formatting are 'S', 'L' and 'X', and they are protected by 'Are you sure' to avoid making a mistake such as formatting Source Code. The use of these commands is fully described in the .DOC's and there is little point in going into the differences here. Normally it is only necessary to place the cursor before the part of the text to be formatted and to issue the 'S' command, and confirm with 'Y'.

There are still a number of commands that have not been mentioned in this short review, largely concerned with setting up system parameters for screen display, formatting or printing, but it is hoped that what has been written will give the reader a good idea of the main features of the PEN family of text Editor/Formatter programs. While it is easy to criticise any Software, and I have mentioned a couple of things in this article, I think that the current issue of PEN will satisfy the Text Editing and Formatting needs of most Home and Small Buisness users in all but the most exacting of tasks, and is well worth the cost.

**64k, or even 128k, on a RAM A Board?!****by P. A. Forrester**

There are probably many Nascom owners who would like to have a full complement of RAM but who jib at paying over £100 for one of the 64k boards which can be purchased, and so most of us soldier on with our old RAM A boards. Some may have added another set of 8 4116s, using the 'piggy-back' method described in INMC-80 (1). However, now that 4164 dynamic RAMs can be obtained for less than £5, I started to ponder whether a better solution might not be to abandon the 4116s and add a block of 8 4164s at a cost of about £35. After I had worked out how to do this, it seemed a shame to leave the other block of 8 sockets unfilled, so I splashed out yet again and added another set of 4164s to give me a total capacity of 128k. Since the Z80 can only address 64k at any given time, I had to find out how to page the additional memory in and out of the address space. This note describes how all this can be achieved. I will not attempt to give a detailed recipe of which tracks to cut where, but rather, will try to give sufficient information so that anyone with a bit of knowledge of digital logic should be able to follow what I have done. The conversion to 64k can be done without the addition of extra support chips, apart from a couple of inverters, and only requires simple mods to the circuitry, but the use of two 64k blocks imposes the need for some means of controlling which parts of the 128k of available RAM is coupled into the memory-map at any given time and this requires some additional chips; the way I have done it only takes 7 extra TTL parts costing in total under £2.

Let us start with a little background on dynamic RAM. This differs in several respects from static RAM, such as the 6116 discussed in 80-BUS News (2). Firstly, the chips we are interested in use address multiplexing; secondly, they have only two data pins, an input and an output; and, thirdly the data must be periodically refreshed. Also, because they are based on NMOS, rather than CMOS, the 4116 consumes rather more power than the equivalent-capacity 6116. Let us briefly consider the implications of these statements. Our understanding can be aided by Fig. 1, which gives the pin layout of the 4116 and 4164.

You see that the 4116 has only 7 address pins and the 4164 has 8. To address 16 kilobits, we need 14 address lines, and so these have to be fed to the chip in two parts. First the lowest 7 lines of the address bus are fed to the address pins of the chip and these are latched internally by applying a low on the /RAS (row address select) pin; then, the next 7 bits are fed to the chip and these are latched by pulling the /CAS (column address select) pin low. It is one of the functions of the board designer to

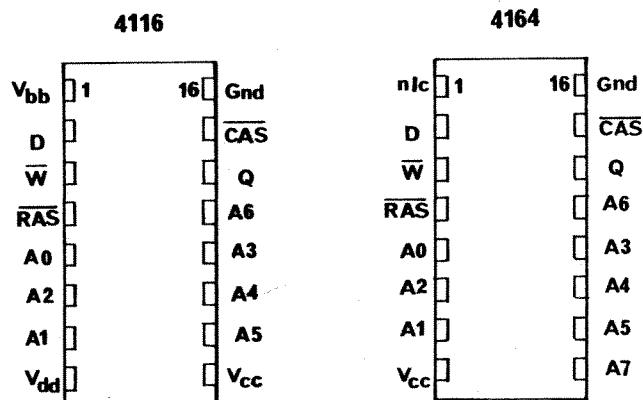


FIGURE 1

ensure that the addresses and their appropriate latching signals are fed to the chip in the correct sequence. The 4164 operates in a similar manner but requires that all sixteen address lines are coupled in two groups A0 - A7, and A8 - A15. Setting up an address on the input latches gives access to one cell so that we can read or write one bit. When R/W goes low, the bit presented at the input pin D is latched provided /CAS is also low. The stored bit then

appears at the output pin Q when R/W goes high. /CAS also acts as an output enable; when it is high, Q switches into the high-impedance state. Because only one bit is stored for each address pattern, 8 chips are needed to service the full data bus. Thus a block of 8 4116s give 16 kbytes and 8 4164s give a full 64 kbytes. This is in contrast to the byte-mapped 6116, where a single chip can be used to provide a block of 2 kbytes.

The other main feature of dynamic RAM is that the charge in the memory cells slowly leaks away and has to be replenished periodically by rewriting. Each cell has to be refreshed within a time less than 2 mSec for the 4116 and less than 4 mSec for the 4164. This is another feature which has to be provided by the memory board designer. Fortunately, the problem is not as complex as it first appears since cells in the same column can be refreshed in parallel by applying an address to the rows only which is strobed by /RAS. Provided all row addresses are strobed during the stated period, refresh is assured. During this process, /CAS is held high which minimises power consumption and ensures that no data is placed on the data bus during refresh cycles. The refresh process occurs during the second half of the opcode fetch cycle, and the Z80 provides a 7-bit refresh counter which makes this part of the memory board design fairly straight-forward. However, when selecting 64k chips a little care has to be taken since, for example, the Texas 4164 requires an 8-bit refresh cycle and therefore presents problems when used with the Z80. Also, some chips, such as the Motorola MCM6664, have an internal refresh counter but require a strobe pulse to be applied to pin 1 to clock the counter on. You should therefore use 7-bit refresh chips which do not utilise pin 1, such as the Mostek MK4564, the Fujitsu MB8264, the NEC uPD4164, etc. The ones I actually used were Motorola MCM6665AP20s.

The 4164 is simpler than the 4116 in that it can operate from a single 5 volt supply, so the -5 and -12 volt supplies will not be required in the new layout. A further advantage of the 4164 is that it actually consumes less power than the 4116; the 4116 typically consumes only 125 mW when running and 17 mW on standby, whereas the corresponding figures for the 4116 are 460 and 20 mW respectively. Let us turn now to the circuit modifications required to add a single set of 8 4116s to the RAM A board, and after that we will go into the additional circuitry to achieve paging for the 128k RAM. Incidentally, if your copy of the circuit diagram is as minute as mine, you will probably find the use of a magnifying glass helpful to see the fine details of the circuit.

The first item which must be attended to is the address multiplexing. For the 4116s, A0 to A6 are multiplexed with A7 to A13 by IC20 and IC21, which are 74157 quad 2-to-1 line data selectors. The two remaining lines, A14 and A15, are decoded by IC24 to produce 4 16k decodes - we will utilise these for paging the 128k RAM. For the 4164s, A0 to A7 must be multiplexed with A8 to A15. It does not actually matter which lines are multiplexed together as long as the components of each pair come from the two different groupings, and that A0 to A6 are coupled as a unit to the RAM chips and latched by /RAS so that refreshing is performed correctly. Unfortunately, in the original design, A0 is switched with A7 which is not allowed, so A7 must be disconnected and replaced by A14, which can be picked up from pin 15 of the 74LS75 address latch IC25. All the remaining pairs can be left unaltered, which leaves A7 and A15 to be dealt with. As currently configured, the remaining inputs on IC21 toggle between 0 and +5 volts to provide the /CAS signal via three buffers in IC 3 which provide a short delay to allow the address lines to settle. It seemed to me to be more straightforward to use these inputs to toggle between

A7 and A15 to provide the additional RAM address line A7', and find some other means of generating /CAS. Therefore A7 is removed from IC20-pin3 and taken to IC21-pin 13 and the connection to earth removed; A15 is picked up from IC25-pin 16 and taken to IC21-pin 14, and the 4K7 resistor removed. The complete set of 16 address lines are then multiplexed by IC20 and IC21.

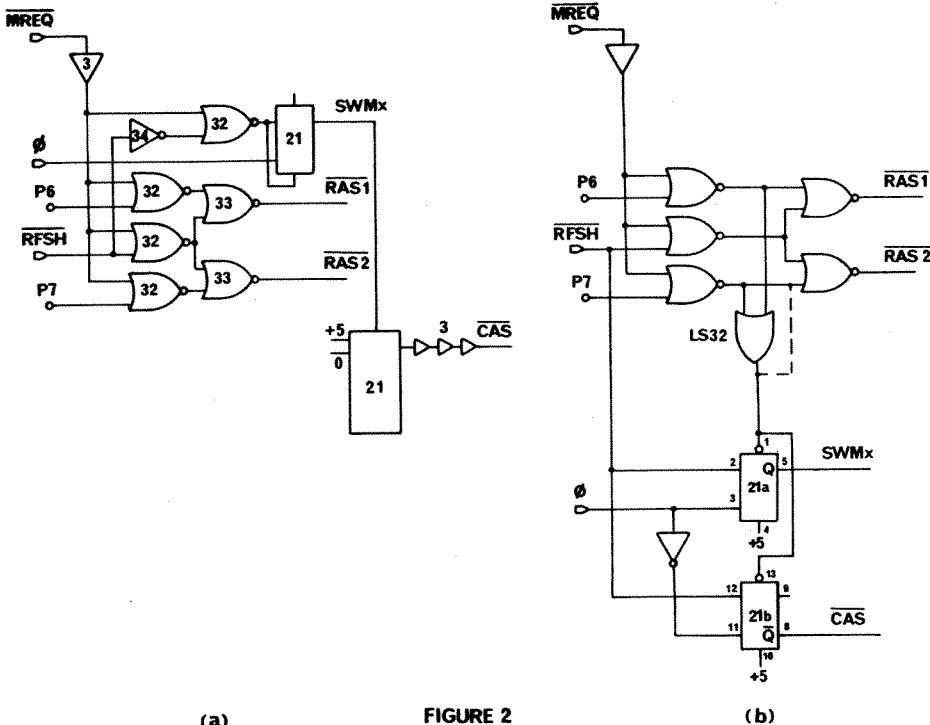


FIGURE 2

Now we have to produce a /CAS signal with an appropriate delay to replace the one generated by the multiplexer. There are various ways in which this could be done but an elegant solution, using only a single 74LS74, was shown recently by D. Allen (3). His circuit is shown in Fig. 2(b), and the relevant part of the original RAM A circuitry in Fig. 2(a). The latter uses only one half of the 74LS74, but by rewiring IC21 to the form of Fig. 2(b) both /CAS and the switch signal SWMx for the multiplexer can be generated. This works as follows: when /MREQ and /CS go low, /RAS is generated, thus latching the row address A0 to A7. At the same time, the reset line of both flip-flops goes high. The D input is /REF, which is high during memory access cycles, and so on the first positive edge of the system clock after /MREQ, IC21(b) flips, which switches the A8-A15 to the RAMs. On the next negative going clock edge, IC 21(a) flips /CAS low, latching the column addresses. During refresh cycles, the D input to both flip-flops is low and so neither switches and /CAS remains high, although /RAS goes low to strobe the row addresses providing refresh.

Fig. 3 shows waveforms derived from my modified board using the shortest program I know which is xx: JR xx in assembler, or 18 FE in hex. When loaded anywhere in memory, this program repeatedly accesses that address. The waveforms were obtained with an 8-channel Tektronix 7D01 Logic Analyser with a time resolution of 20 nSec. Two complete cycles are shown; note the sequence of the /RAS, SWMx and /CAS signals and how the A0 line toggles during each refresh period. In this arrangement, /CS, which is applied to either P6 or P7 depending upon which block of sockets is used, can, in principle, be tied to earth; however, see below. Also, the 74LS32 OR gate, which drives the 74LS74 flip-flop reset-pin high when either P6 or P7 and /MREQ are low, is not

required if only the 64k version is to be used. It is then sufficient to tie the reset pin to the OR gate output as indicated by the dotted line. The inverter for the clock input to the flip-flop can be taken from pin 9 (input) and pin 8 (output) of IC 34 which does not appear to be used in the original design. Both the extra address line A7' and the /CAS signal should be coupled to the RAM via 33R resistors to reduce electrical noise and ringing.

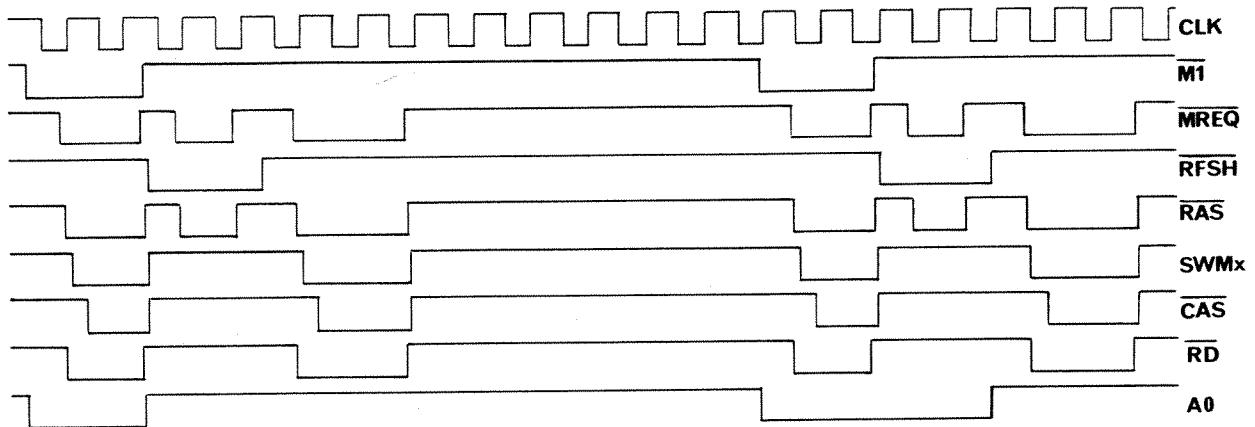


FIGURE 3

When it was first tested by writing a block of identical bytes using the NasSys C command, the RAM showed a lot of corrupted bytes. The reason for this took me many hours to uncover. Eventually I discovered that the read/write line on pin 1 of the data bus buffer IC 2 suffered from spurious negative pulses of about 50 nSec duration. These pulses occurred every time the /RAMDIS signal was activated. This line goes low about 50 to 60 nSec after /MREQ; thus, with /CS on P6 or P7 held low, the buffer is put into the read state as soon as /RD and /MREQ go active. Now if this is the start of a read-cycle of memory on the main board, the read buffer on the RAM board should not be activated, but it does not know that for 50 nSec or so, until /RAMDIS goes active. As designed, P6 or P7 are fed from the decoders IC 22 and IC 23, or IC 24, and these, together with the address latch IC 25, produce sufficient delay that /RAMDIS goes low before /CS and hence there is no timing problem. The solution which seemed to be most simple was to produce a version of /MREQ delayed by about 60 nSec, and apply it to the NASCOM SEL pin (which is coupled directly to bus line 11); this stops IC 36 from going low until after /RAMDIS goes active, if it is going to. I used two gates from a CMOS 4049 inverter connected in series, which produces sufficient delay. This is the only additional chip which is required to complete the 64k conversion. I must confess that I thought I still had problems, even after applying the delay, but this turned out to be nothing to do with the modified board, but rather, that I was testing it using a 10 inch extension board to give me more ready access to the RAM card; it turned out that this introduced sufficient spurious fluctuations to the bus lines to cause corruption of some of the data. Finding this also wasted a considerable amount of time - so be warned. With all the bugs removed, the board functioned perfectly without wait-states at 4 MHz.

If you do not like the thought of cutting up the tracks of your board until you are sure that these modifications will work, you can get round this by putting the ICs which require circuit changes to be made into new sockets; they should be the type with the pins oriented so that they can be plugged directly into the original IC sockets (you may have some difficulty finding this type). The new sockets have those pins which require modifications bent out so that they are accessible for soldering; they are then inserted into the original sockets and the rerouting of the wiring made to the bent-out pins. This technique can be used for a permanent modification and leads to quite a neat job but do not forget that the 5 volt line should be decoupled at each memory chip.

So that is all you have to do to produce a full 64k conversion job. Adding another 64k follows along the same lines, with the /CS for that block being taken from P7. The 74LS32 OR gate needs to be added to ensure that the flip-flop reset pin goes high when either block 1 or block 2 is selected. However, adding this additional gate led to another problem; on my board, the rising edge of the clock occurs almost simultaneously with the generation of the /RAS signal so that SWMx is produced within a few nanoseconds after /RAS. Even though the RAM specification states that the address should remain stable for 20 nS following /RAS, the unmodified board and the 64k version appear to work happily. However, the delay introduced by the OR-gate (even using the higher speed 74S32) causes the flip-flop reset to go high after the clock transition, which upsets the correct phasing of SWMx and /CAS. The solution is to delay the clock signal applied to the flip-flop by inserting four 74LS04 inverting buffers coupled in series between the clock signal from the bus and the flip-flop clock pin, which introduces a delay of about 30 nSec; this appears to be sufficient. I believe that similar modification to the original board was suggested in an early issue of the INMC to delay the generation of SWMx and /CAS, but I was unable to find a reference to this.

We now have to select which block, or part of a block (which we will call a page, following the usual custom) is coupled into the memory map. If you are content to couple in either one block or the other, you only have to toggle one bit derived from the PIO, so that when P6 is up P7 is down and vice versa. Provided you have NasSys overlaying the RAM you can keep control by using the 0 command. However, greater flexibility is achieved if you are able to couple in any 16k page from either block. One way of doing this is shown in Fig. 4; the 74LS75 4-bit latch is partly decoded as port 3 - this fits in with the partial decoding of the other ports used on the N2 main board; you can, of course, fully decode it to be any other number between 9 and 255, but then /IOEXT must be generated too. The four Q outputs from the latch are ANDed with the four 16k decodes which appear at the points labeled 0 to 3 on the RAM board, and the outputs ORed to produce a /CS for the first block of RAMs on P6. The /Q outputs are treated similarly and fed to P7. This ensures that there can be no contention between the two RAM blocks. Since /CS acts as the output buffer control line, it must only activate one block at a time; this is ensured using the 74LS32 to create logic-low AND gates with the inputs P6 and P7 to give two /CS lines, one for each block. Control of the paging can then be done by writing to Port 3, either statically by using the NasSys 0 command or as part of a program by using the out instruction. 0 3 0 selects the whole of block 1, and similarly 0 3 F selects block 2, while 0 3 3, for example, maps pages 1 and 2 of block 2 and pages 3 and 4 of block 1 into the memory map.

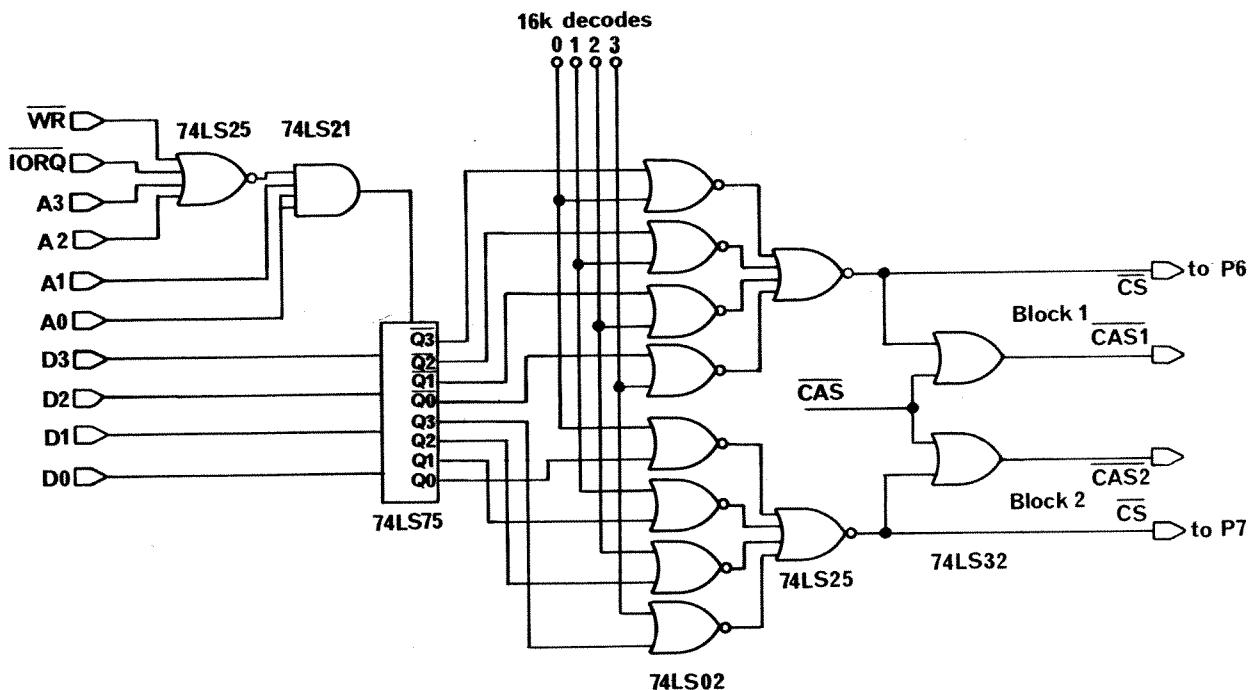


FIGURE 4

I built the additional circuitry on a 3 by 4 inch piece of Vero card which was mounted about 2 cm. above the main board; it was held at one end by wire-wrap pins which plugged into the empty EPROM sockets and was supported at the other end by two insulating pillars. These were held in place by fine insulated wire threaded through convenient holes in the boards. Connections to the various signal lines were then made by soldering to the IC socket pins on the underside of the main board. This made a reasonably neat arrangement of what is essentially a bodge job.

256k RAM chips are now becoming available, but unfortunately they currently cost about £50 each, so even a single block would cost £400, which is beyond the reach of most of us. However, the price projection is that they will be under £10 by mid-1985. These chips require a full 8-bit refresh cycle and a little thought is required to toggle the A7 line at the appropriate times during refresh; a number of ingenious solutions have been suggested in various electronic journals. I wonder who will be the first to upgrade his RAM A board to half a megabyte?!

- [1] INMC 80 News Vol 3, Feb-Apr 1981, page 16-17 see also 80-BUS News Jan-Mar 1982, page 12.
- [2] 80-BUS News Vol. 1 Issue 3, July-Oct. 1982, page 21-23.
- [3] D. Allen New Electronics Vol. 17 No. 5, 6th. March 1984, page 26.

## An Insight into the Gemini IVC and SVC Part 2

By D. W. Parkinson

### THE GEMINI SVC

To continue from where we left off ... The SVC is a close relative of the IVC that is upwards compatible with it, but offers extended features and greater power. The extra features that have arrived with the SVC are summarised in table 1.

I'll start by taking various of the hardware changes that have been made and indicating the software benefits that have been gained.

### SCREEN MEMORY/CHARACTER GENERATORS

The IVC had a 2k x 8 static RAM for the screen memory, a 2k x 8 EPROM for the main character generator, and a 2k x 8 static RAM for the alternate character generator. With the SVC design these have been combined into a single 8k x 8 static RAM. The 8k of RAM is partitioned in various ways depending upon the selected display mode (see Figure 1). When in alpha mode the RAM is actually accessed twice per character. The first time is to pick up the character from somewhere in the lower 4k, and the second time to look-up the appropriate row of dots for that character from the upper 4k.

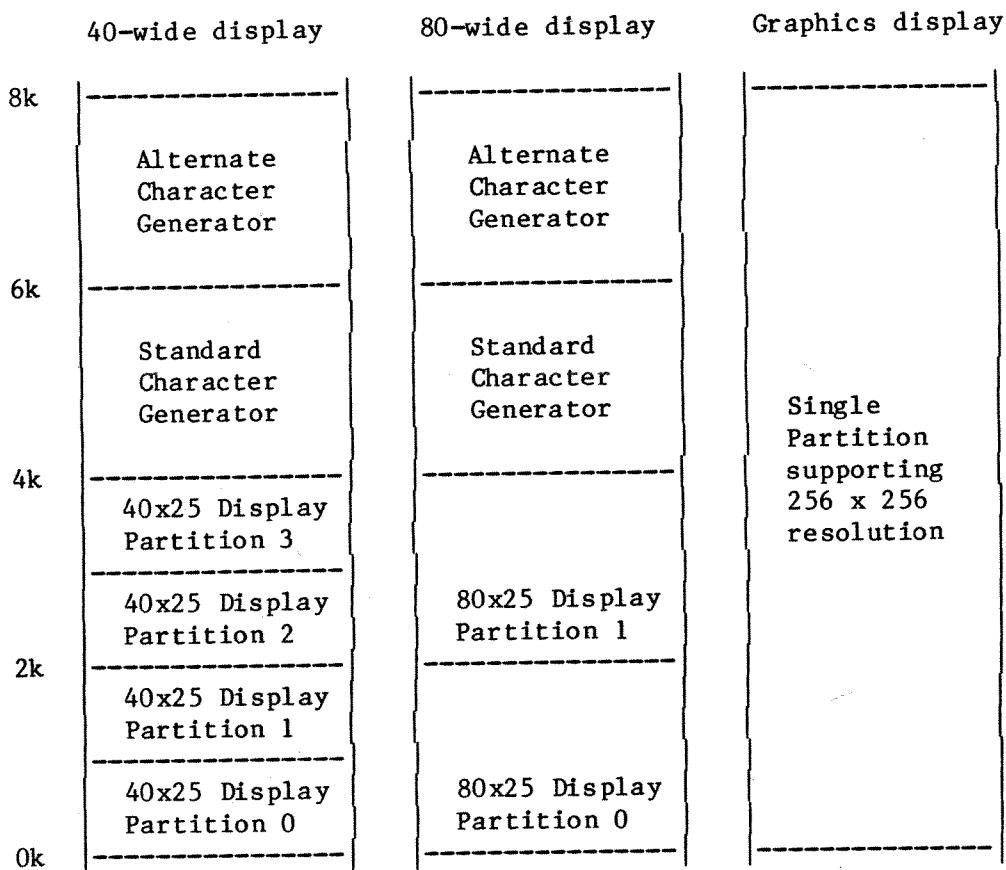


Fig 1. Display memory image

With the SVC the character set is now totally programmable. The initial character set is held within the monitor EPROM, and is copied out into the top half of the screen RAM on power-up, or whenever alpha-mode is reselected following a graphics display. The SVC-MON also holds various foreign character

sets that can be selected by means of an on-board DIL switch. (The character generator is actually initialised with the English character set, and then, depending on the DIL switch settings, a small sub-set of characters may be overwritten by their foreign language versions.)

The two standard alpha display modes are now 80 x 25, (80 characters by 25 lines) and 40 x 25. With the increase in the size of the alpha screen memory from 2k to 4k, (remember the other 4k is used as a character generator), it was decided to add the concept of multiple 'screens'. I felt that this offered certain advantages over treating the 4k of memory as a single screen, although the latter would offer the possibility of being able to scroll back through the last 50 lines (or 100 lines in 40-wide format) that had been displayed. With the 'multiple screen' concept you can select (via an escape sequence) which of the available screens is to be displayed, and which is to be updated by the incoming data/commands from the Host. My reasons for doing this I list below, but I must confess that reason (1) was perhaps a little selfish!

1. It required a very small change to the existing software to support it.
2. This feature had zero impact on the current performance of the SVC. (The alternative approach would have meant scrolling 4k of data, and also added the problem of handling any 'locked' screen lines.)
3. It allows programs to set up hidden 'help' screens that can be switched to instantly.
4. It simplifies (possibly) the implementation of concurrent tasks. (If you want to have two tasks running concurrently with independent screens on a system with an IVC there is the problem of how the undisplayed screen is handled. There are two options open: a) Queue all console output for the non-displayed task, wait until its screen image is swapped into the IVC, and then send it. Or b) Duplicate all the necessary parts of the IVC monitor in the system software so that the screen image can be updated immediately.)

When set into the graphics mode the SVC uses the entire 8kx8 screen memory as a 256 x 256 pixel display. (No colour, it's black-and-white only I'm afraid!) As a result the RAM based character set is lost, although the default character set is still present within the monitor EPROM. The SVC will still display text in this mode, but the format is now 32 characters/line and you are restricted to the English character set. (Each incoming ASCII character is used to index into the EPROM-based character generator table from whence the bit patterns are copied out into the screen RAM. I have not as yet included any check for special sub-set characters, nor the subsequent check of the DIL switch settings to see if these should be modified.) Any enterprising Dane or German who is irritated by this could swap the relevant dot patterns of the subset characters in the default table with those in their language table, not forgetting to exchange the words 'English' and 'Danish' (or whatever) where necessary in the manual.

#### **ATTRIBUTES**

The IVC only offered one attribute - the selection of the alternate character generator. This is normally loaded with the complement of the main character set to give video-inverse characters.

The SVC offers a variation on this theme. For various reasons the screen memory word width remains at 8 bits. This, as before, leaves a single bit that can be used as an attribute bit. But the SVC, unlike the IVC, allows this bit to select a variety of attributes. So setting bit 7 of a character can select one or more of the following attributes:- use the alternate character generator, blink the character, half intensity character, and, display the character with a half-tone background. Note that I said 'one or more'. The SVC supports an escape sequence that is used to select which attributes you want. You can select any combination of the above, e.g. blink + half-intensity, but the one thing to remember is that with only one attribute control bit available, all characters on the screen with this bit set will respond in the same way. Therefore it is not possible to have inverse characters on one part of the screen, and half-intensity characters on another part.

One apparent omission from the attribute list above is that of inverse video. This was originally included as a specific attribute, but was later dropped in favour of the half-tone background. It is assumed that inverse video is provided in the usual way via the alternate character generator.

#### **TRANSPARENT ACCESS & ESP**

With the SVC, a hardware technique is used to achieve transparent access. On the face of it all that has to be done is to connect the 'display enable' (or 'display active') signal from the display controller via a few gates to the Z80 /WAIT input. Thus, if the processor attempts to access the screen memory while the display is active, it is made to wait until the end of the display period, after which the Z80 is free to continue with its read/write cycle. However a problem arises when the Z80 finds, on going to access the display memory, that the display is inactive. The dilemma here is that although the display memory is free at that instant, the display controller may start a new display line at any moment (like in the middle of the forthcoming read/write cycle of the Z80). If it does, this will lead to one of two outcomes: a corrupted read/write cycle, or interference down the left hand side of the display depending on which controller gets priority.

This is where ESP comes in. The arbitration circuit has to know whether there is time for the Z80 to complete its read/write cycle without interruption, or whether the /WAIT signal should be applied immediately as the display controller is about to start a new display line. Those of you who have followed the description so far are probably thinking - 'ah! you just generate or tap off an earlier version of the Display Enable signal and use that'. Unfortunately this is easier said than done as the 'display enable' signal generation circuits are buried deep inside the 40-pin plastic package of the display controller and only the final signal emerges on one of the pins. (That's what modern LSI does for you!) There is a way around this problem and that is to apply a classic wartime (and peacetime) ploy of mis-information. What happens with the SVC is that some external delay has been added to the 'Display Enable' signal, and in setting up the internal registers of the controller, the SVC monitor informs it that the first address of the display is a byte earlier than in fact it is.

The following simplified description assumes a screen format of 80 x 25: When the display controller displays a TV line it loads its display memory address register with the starting address of where the current line of characters is stored in the memory (lets call it N). The address register is then incremented at the character rate for the entire TV line width (64uS)

after which it is reset for the next line. If it is still working through one of the 25 'character' lines on the screen then it is reset to N and the character generator 'line' address is incremented by one, otherwise it is reset to N+80 to select the next line of characters. When the display line starts the controller turns ON the 'display enable' signal. This is turned OFF after address N+79 is passed. Then, as further pre-programmed character positions are passed, the controller generates the Horizontal Sync pulse, and finally the internal reset signal for the counter.

From the information we have given it the controller thinks it is displaying characters N to N+79, but by playing around with the external delays we actually display N+1 to N+80, and in the process gain sufficient advance warning of an active display line to prevent any unseemly clash between the Z80 and the display controller. As the controller has been fooled over the start of the display, it also has to be mis-informed about the cursor position, a task that the NMI routine carries out as it updates the internal registers of the controller.

#### **INTERRUPTS**

With the display contention problem solved by hardware, various other changes could be made. One was the use of the interrupt system. Previously, with the transparent screen access being determined by software loops, interrupts could not be used as they could interfere with critical parts of the loop. With the SVC, keyboard input is now done under interrupt. Also, while the SVC-MON is scrolling the screen (with an LDIR instruction), it enables an interrupt routine to collect and buffer incoming characters from the Host. Normally the SVC-MON polls for input characters.

#### **KEYBOARDS**

An option for a serial keyboard has been included in the SVC. This is an either/or option with the parallel keyboard, not an AND one, as they both share the same IO port (on the SVC) and some of the interface circuitry. The serial keyboard requires fewer wires to interface it, and this leads to lower cost. (See the latest Gemini price list - The multi-core 'curly-wurly' cables for the parallel keyboard cost a fair bit! The difference in price solely reflects the cabling costs. There is no difference in keyboard costs as the change required on it was only a small one in the on-board software. (The keyboards use an 8035 single-chip microprocessor to scan the key matrix and generate the appropriate ASCII codes.)

#### **NMI**

As with the IVC, on the SVC the vertical sync output of the display controller is connected to the NMI input of the on-board Z80. However the SVC NMI service routine performs different functions. It no longer scans the keyboard as that has been transferred to a conventional interrupt routine (see above). It handles the timing of various attributes. The tone generator for the on-board beeper is directly enabled and disabled by the SVC monitor and thus the duration (but not the frequency) of the 'bleep' is fixed at an integral number of NMIs. Similarly the low frequency waveform used to 'blink' characters is also generated by this software. One other feature, mentioned by Dave Hunt in the last issue, is the implementation of a software 'clock'.

One point to bear in mind when you start thinking 'wouldn't be nice if.. is the fact that the NMI routine is executed once every 20ms. The larger and more cumbersome it becomes, the more it is likely to have an impact on the

general performance of the SVC. Thus the NMI software is written for speed, and tries to avoid specialised checks for unlikely occurrences. (See below for the ramifications.)

### THE SOFTWARE CLOCK

The SVC supports a software clock that works on the basis that NMIs are occurring at the rate of one every 20ms. (This assumes that the CRT controller's registers have been set up correctly.) A software counter repeatedly counts down from 50, and every time it reaches zero (once a second) an internal ASCII digital clock display is updated. After updating the clock the software checks a flag to see if it is to be displayed. If the flag is set, the clock is copied to a predetermined position on the screen. The default display position for the clock is at the top right of the display, but, by sending the appropriate escape sequence, it may be positioned anywhere on the screen.

Note that the clock software in the NMI routine only spends time updating the screen once per second when the internal ASCII clock is advanced. For the other 49 'ticks' of the clock no time is wasted updating a (probably) unchanged display. One other 'feature' that resulted from the basic NMI philosophy of minimum overhead is that the SVC supports a 60 hour clock rather than the more usual 24 hour clock! The updating software does not include a check for the special case of the hours. This fact will only be noticed by those who burn the candle at both ends. (It would have been interesting to have kept quiet about this, and seen if it was ever reported as a bug. My feeling is that it would have been a year or two before anyone noticed it, and then it would probably have been by accident.)

### GRAPHICS SOFTWARE

My intention with the initial graphics software was to provide the fundamental routines on-board where the greatest speed advantage could be obtained, and to leave the neat user interface and extended features to an intermediate interface program. (e.g. The Gemini/CCSOFT GRAPHPAC package.) Thus for example there are no Escape sequences for 'pen-up' and 'pen-down', or for relative plotting. The basic escape sequences deal with absolute coordinates, and include the pen-up-down command within them.

The routines included offer: Pixel set, reset, and test. Line drawing. Circle drawing. Flood fill.

The algorithms I used were simple and orientated towards speed. They only use addition, subtraction. There is no multiplication or division involved other than by a simple shift for division by 2. In this first release of the SVC-MON I concentrated on simplicity, and so certain possibly desirable features have not been included. (e.g. All the line drawing is done with a single line type - a solid line. There is as yet no option for dotted lines, dashed lines, or whatever.) The circle routine draws a true circle, there is as yet no support for drawing ellipses. (Depending on how your monitor is adjusted you might have to draw an ellipse in order to get a circle displayed!)

Internally the SVC graphics routines use 16-bit arithmetic. Although this does have a small impact on performance it also has two advantages.

- i) It offers a possible upgrade path sometime in the future if Gemini should ever produce a card with higher resolution.

ii) Any over-sized plots that go off the screen re-appear in the correct place when they return within the bounds of the display. There are no sudden 'wrap-around' problems that wreak havoc on some displays.

The graphics origin has been placed at the bottom left of the screen to line up with the pencil-and-paper convention. This also aligns with the approach taken by Hewlett-Packard in their extensive range of desk top computers, (98XX series, series 200, etc). These are widely used throughout Industry mainly in engineering applications.

There is also a 'flood fill' routine that, given starting coordinates X,Y, will fill the enclosed area that includes X,Y. This routine will fill polygons of any shape, and is fast in operation. The hardest part of developing the 'fill' algorithm was reducing the amount of workspace required by it. In essence a fill routine is very simple if recursion is allowed and a simple 'C' implementation is shown below.

```
fill(X,Y)
int x,y;
{   if(test(x,y) return;      /* stop if point is already set */
    set(x,y);                /* ..else turn point ON          */
    fill(x+1,y);              /* Now check adjacent points    */
    fill(x-1,y);
    fill(x,y+1);
    fill(x,y-1);
}
```

This routine will do the job. The drawbacks of it as it stands are various. If it fills an empty 256 x 256 screen in the worst possible order the routine FILL will call itself 65,535 times, requiring a stack depth of about 128k together with another 256k required for the new values of X and Y that are generated on each CALL to FILL. The problem can be eased by making FILL fill in a line at a time instead of a point, but it still requires a fair amount of memory.

I ended up using an algorithm that utilised a FIFO (First in First Out) buffer rather than recursion. (You could regard recursion as a LIFO - Last In First Out - buffer.) This reduced the workspace requirements considerably. In fact I found that I could fill very complex shapes with only 128 bytes of workspace. However the FILL routine did drop out when I was filling a particularly complex shape with the software clock also present on the display. (It gave up as it was filling around the numbers of the clock display.) Up to then I had managed to keep my hands off the 1k of memory on the SVC that was reserved for downloading user programs into. As this memory sits there unused in 99% of cases I decided to grab it for the FILL routine, but to temper this impudence by implementing an 'intelligent' grab. What actually happens is this:

If no user program has been downloaded the FILL routine will use the full 1k as its workspace.

If a user program has been downloaded, then the FILL routine will use 3/4 of the remaining space as workspace. (i.e. If a program 227 bytes long is downloaded, the FILL routine will grab the last  $(1024-227)*3/4$  bytes of the 1k RAM area for its buffer.)

This I believe is an acceptable approach as it is only likely to affect the 0.000001% of users who download 1k user programs while using the graphics fill command at the same time.

As a brief aside on FILL routines - You can always tell a recursive based FILL routine, as, when it is filling a complex area, there are often pauses while nothing apparently happens on the screen. What is actually happening is that the FILL routine has just filled to a 'dead end', and there then follows a period where the recursive structure 'unwinds' itself, returning from layer to layer back up the stack until it finds an area that it has not yet filled.

#### BUG TIME

Well folks, it had to happen. Between writing the first part of this article and completing this one someone has unearthed a bug that is present in both the IVC and SVC monitors. That fact that it has taken so long to surface is indicative that none of you are likely to have met it yet (otherwise you would have reported it - wouldn't you?). It is in the ESC "f" ... sequence that allows you to redefine the function keys from a program. If you try and 'undefine' a key using this sequence the key is not actually fully deleted from the function key table, and if the key is subsequently pressed something will be returned to the host system. What the 'something' is cannot be easily predicted. Key definitions can be removed totally from the table, but only by using the Shift/Edit feature direct from the keyboard.

TABLE 1: Extra features of the SVC

HEX	ASCII	DECIMAL	COMMAND
<u>General</u>			
07	~G	7	Bell - sound on-board buzzer
<u>Cursor Movement</u>			
1B 0C	<ESC> ~L	27 12	Cursor home
<u>Screen format</u>			
1B 34	<ESC> "4"	27 52	Select graphics mode
1B 70...	<ESC> "p"...	27 112...	Select display/update partition
<u>Character set</u>			
1B 67...	<ESC> "g"...	27 103...	Set language
1B 62...	<ESC> "b"...	27 98...	Read back character set
<u>256x256 graphics</u>			
1B 52	<ESC> "R"...	27 82...	Reset point X,Y
1B 53...	<ESC> "S"...	27 83...	Set point X,Y
1B 54...	<ESC> "T"...	27 84...	Test point X,Y
1B 6C...	<ESC> "l"...	27 108...	Draw/Erase/Complement a line
1B 6F...	<ESC> "o"...	27 111...	Draw/Erase/Complement a circle
1B 6D...	<ESC> "m"...	27 109...	Move cursor to X,Y
1B 77	<ESC> "w"...	27 119	Fill a polygon
1B 64...	<ESC> "d"...	27 100...	Graphics screen dump
<u>Keyboard</u>			
1B 3E...	<ESC> ">"...	27 62...	Download soft-key display
<u>Clock</u>			
1B 74...	<ESC> "t"...	27 116...	Set clock time
1B 74 3D...	<ESC> "t="...	27 116 61	Position clock display
1B 74 45	<ESC> "tE"	27 116 69	Enable clock display
1B 74 44	<ESC> "tD"	27 116 68	Disable clock display
1B 74 3F	<ESC> "t?"	27 116 63	Return clock time
<u>Miscellaneous</u>			
1B 61...	<ESC> "a"...	27 97...	Set/reset attributes

Dis-assembly of NASCOM ROM BASIC Ver 4.7		PAGE 91
FB8B CD13F8	RND:	CALL TSTSZN ; Test sign of FPREG
FB8E 211910		LD HL,SEED+2 ; Random number seed
FB91 FAECFB		JP M,RESEED ; Negative - Re-seed
FB94 213A10		LD HL,LSTRND ; Last random number
FB97 CD51F8		CALL PHLTTP ; Move Last RND to FPREG
FB9A 211910		LD HL,SEED+2 ; Random number seed
FB9D C8		RET Z ; Return if RND(0)
FB9E 86		ADD A,(HL) ; Add (SEED+2)
FB9F E607		AND 0000011B ; 0 to 7
FBA1 0600		LD B,0 ; Re-save seed
FBA3 77		LD (HL),A ; Move to coefficient table
FBA4 23		INC HL ; 4 bytes
FBA5 87		ADD A,A ; per entry
FBA6 87		ADD A,A ; BC = Offset into table
FBAB 4F		LD C,A ; ADD HL,BC
FBAB 09		ADD LOADFP ; Point to coefficient
FBAC CD62F8		CALL COEFFICIENT ; Coefficient to BCDE
FBAC CD08F7		CALL FPMULT ; Multiply FPREG by coefficient
FBAB AF 3A1810		LD A,(SEED+1) ; Get (SEED+1)
FBB2 3C		INC A ; Add 1
FBB3 E003		AND 0000011B ; 0 to 3
FBB5 0600		LD B,0 ; Is it zero?
FBB7 FF01		CP 1 ; Yes - Make it 1
FBB9 88		ADC A,B ; Re-save seed
FBBB 321810		LD (SEED+1),A ; Addition table
FBBB 21FF0B		LD HL,RNDTAB-4 ; 4 bytes
FBC0 87		ADD A,A ; per entry
FBC1 87		ADD A,A ; BC = Offset into table
FBC2 4F		LD C,A ; Point to value
FBC3 09		ADD HL,BC ; Add value to FPREG
FBC4 CD1DEF5	RND1:	CALL ADDPHL ; Move FPREG to BCDE
FBC7 CDFFF8		BCDEFPL ; Get LSB
FBCA 7B		LD A,E ; LSB = MSB
FBCB 59		LD E,C ; Fiddle around
FBCC EE4F		LD 0100111B ; New MSB
FBCF 4F		C,A ; Set exponent
FBD1 2B		LD (HL),80H ; Point to MSB
FBD2 46		DEC B,(HL) ; Get MSB
FBD3 3680		LD (HL),80H ; Make value -0.5
FBD5 211710		LD HL,SEED ; Random number seed
FBD8 34		INC C ; Count seed
FBD9 7E		DEC D ; with the
FBDA D6AB		LD A,(HL) ; number
FBDC C2E3FB		SUB 171 ; Normalise number
FBDF 77		JP NZ,RND2 ; Save random number
FBE0 0C		LD (HL),A ; Move FPREG to last and return
FBE1 15		INC C ; FPTHL
FBE2 1C		DEC D ; Move FPREG to last
FBE3 CD1EF6	RND2:	INC E ; Save random number
FBE6 213A10		LD HL,LSTRND ; Move FPREG to last
FBE9 C36BFB8		JP

# NASCOM ROM BASIC

## DIS-ASSEMBLED

## PART 7

BY CARL LLOYD-PARKER

## Dis-assembly of NASCOM ROM BASIC Ver 4.7

## Dis-assembly of NASCOM ROM BASIC Ver 4.7

## PAGE 93

RESEED: LD (HL),A ; Re-seed random numbers  
 DEC HL  
 LD (HL),A  
 LD (HL),A  
 LD (HL),A  
 LD RND1 ; Return RND seed  
 JP 068H,0B1H,046H,068H ; Table used by RND  
 099H,0E9H,092H,069H  
 010H,0D1H,075H,068H

COS: LD HL,HALFP1 ; Point to PI/2  
 CALL ADDPHL ; Add it to FPREG  
 CALL STAKFP ; Put angle on stack  
 LD BC,8349H ; BCDE = 2 PI  
 LD DE,ORDBH  
 CALL FPBCDE ; Move 2 PI to FPREG  
 POP BC ; Restore angle  
 FC12 C1  
 FC13 D1  
 FC14 CD69F7  
 FC17 CD44F8  
 FC1A CDE6F8  
 FC1D C1  
 FC1E D1  
 FC1F CDC4F5  
 FC22 214ERC  
 FC25 CDC4F5  
 FC28 CD13F8  
 FC2B 37  
 SCF  
 FC2C F236FC  
 FC2F CDBBF5  
 FC32 CD13F8  
 FC35 B7  
 SIN1: PUSH AF  
 CALL P,INVSGN ; Test sign of value  
 LD HL,QUARTR ; Flag positive  
 SUBCD E ; Positive - Ok  
 CALL TSTSIN ; Add 0.5 to value  
 ROUND ; Test sign of value  
 CALL TSTSIN ; Flag negative  
 OR A ; Save sign  
 PC36 F5  
 SIN1: PUSH AF  
 CALL P,INVSGN ; Negate value if positive  
 LD HL,QUARTR ; Point to 0.25  
 SUBCD E ; Add 0.25 to value  
 ADDPHL ; Restore sign  
 AF NC,INVSGN ; Negative - Make positive  
 CALL HL,SINTAB ; Coefficient table  
 SUMSER ; Evaluate sum of series  
 JP 0DBH,00FH,049H,081H ; 1.5708 (PI/2)

HALFP1: DEF8 SINTAB: DEF8 5 ; Table used by SIN  
 OBAH,0D7H,01EH,086H  
 0E4H,026H,099H,087H  
 058H,034H,023H,087H  
 0E0H,05DH,0A5H,086H  
 ODAH,00FH,049H,083H

DEFB CD44F8 TAN: CALL STAKFP ; Put angle on stack  
 FCCA CD06FC CALL SIN ; Get SIN of angle  
 FCBD C1 POP BC ; Restore angle  
 FC6F EI POP HL ; Save SIN of angle  
 FC6F CD44F8 CALL STAKFP ; BCDE = Angle  
 FC72 EB EX DE,HL ; Angle to FPREG  
 FC73 CD54F8 CALL FPBCDE ; Get COS of angle  
 FC76 CD00FC CALL COS ; TAN = SIN / COS  
 FC79 C36/F7 JP DIV ; Test sign of value  
 FC7C CD13F8 ATN: CALL TSTSIN ; Negate result after if -ve  
 FC7F FCA7FA CALL M,NEGAFT ; Negate value if -ve  
 FC82 FC3CF8 CALL M,INVSINP ; Get exponent  
 FC85 3AE710 LD A,(FPEXP)  
 FC88 FE81 CP 81H ; Number less than 1?  
 FC8A DA99FC JP C,ATNL ; Yes - Get arc tangent  
 FC8D 010081 LD BC,8100H ; BCDE = 1  
 FC90 51 LD D,C  
 FC91 59 LD E,C  
 FC92 CD69F7 CALL DVBCDE ; Get reciprocal of number  
 FC95 21C4F5 LD HL,SUBPHL ; Sub angle from PI/2  
 FC98 E5 PUSH HL ; Save for angle > 1  
 FC99 21A3FC ATN1: LD HL,ATNTAB ; Coefficient table  
 FC9C CD55FB CALL SUMSER ; Evaluate sum of series  
 FC9F 214AFC LD HL,HALFP1 ; PI/2 - angle in case > 1  
 FC9A C9 RET ; Number > 1 - Sub from PI/2

ATNTAB: DEF8 9 ; Table used by ATN  
 FC94 4AD73B78 DEF8 04AH,0D7H,03BH,078H ; 1/17  
 FC98 026847B DEF8 002H,06FH,084H,07BH ; -1/15  
 FCAC FEC12F7C DEF8 0FEH,0C1H,02FH,07CH ; 1/13  
 FCBD 74319A7D DEF8 074H,031H,09AH,07DH ; -1/11  
 FC84 843D5A7D DEF8 084H,03DH,05AH,07DH ; 1/9  
 FCBB C87F917E DEF8 0C8H,07FH,091H,07EH ; -1/7  
 FCBC E4BB4C7E DEF8 0E4H,0BBH,04CH,07EH ; 1/5  
 FC00 6CAAA7F DEF8 06CH,0AAH,0AAH,07FH ; -1/3  
 FCC4 00000081 DEF8 000H,000H,000H,081H ; 1/1

Dis-assembly of NASCOM ROM BASIC Ver 4.7

FCG8 CD39FE	CASFFW: CALL	FLPLED	;	Turn on cassette
FCCB 0600	LD B,0	DELAY	;	Set 1 second delay
FCCD CD9BF0	DELAYB: CALL	B	;	Wait a bit
FD00 05	DEC B	NZ,DELAYB	;	Count
FD01 C2CDFC	JP RET		;	More delay needed
FD04 C9			;	
FC05 C339FE	CASFF: JP	FLPLED	;	Flip tape LED
FC08 C9	ARET: RET		;	A RETURN instruction
FC09 E5C5D5F5	COMMON: PUSH	HL,BC,DE,AF	;	Output character to screen
FCDD CD6DFE	CALL MONTST	N2,NASOUT	;	NAS-SYS - Output ASCII
FCE0 C2BFBC	POP AF	AF	;	Get character
FCE3 F1	PUSH AF	AF	;	And re-save
FCE4 F5	LF CP	Z,IGCHR	;	ASCII Line feed?
FCE5 F00A	CP	BSKP	;	Yes - Ignore it
FCE7 CA00FD	JP	NZ,CON01	;	ASCII back space?
FCE8 FB08	CP	LD A,TBS	;	No - Test for CR
FCEC C211FC	JP	CR	;	NASBUG back space
FCEF 3E1D	LD CON01: CP	N2,OUTCHR	;	ASCII CR?
FCF1 FE0D	JP	A,TCR	;	No - Output character
FCF3 C2DFDC	LD	OUTCHR	;	NASBUG CR
FCF6 3E1F	JP		;	
FCF8 C3DFDC			;	
FCFB F1	NASOUT: POP AF	POP AF	;	Get character
FCFC F5	PUSH AF	MONOUT AF	;	And re-save
FCFD CD45FE	OUTCHR: CALL AF,DE,BC,HL	RET	;	Output it
FD00 F101C1E1	IGCHR: POP AF		;	
FD04 C9	RET		;	Restore character
FD05 E5C5D5	GETINP: PUSH CALL	HL,BC,DE	;	Get an input character
FD08 CD6DFE	MONST	MONTST	;	See if NAS-SYS
FD0B CA13FD	JP Z,GETIN	JP Z,GETIN	;	"T" monitor - Get input
FD0E DF7B	SCAL	BLINK	;	
FD10 C319FD	JP CONVIN	CONVIN	;	Convert to ASCII
FD13 CD490C	GETIN: CALL TIN	TIN	;	"T" input a character
FD16 D213FD	JP NG,GETIN	NG,GETIN	;	No input - wait
FD19 FE1D	CONVIN: CP NZ,CNVIN1	CP NZ,CNVIN1	;	NASBUG back space?
FD1B C220FD	LD A,BSP	LD A,BSP	;	No - Test for break
FD1E 3E08	CNVIN1: CP TBKR	CP TBKR	;	ASCII back space
FD20 FE1C	JP NZ,CNVIN2	JP NZ,CNVIN2	;	NASBUG break?
FD22 C227FD	LD A,CTRLC	LD A,CTRLC	;	No - Test for control Z
FD25 3E03	CTRLZ	CTRLZ	;	Control C
FD27 FE1A	NZ,CNVIN2: CP NZ,CNVIN3	CP NZ,CNVIN3	;	~?
FD29 C228FD	LD A,DEL	LD A,DEL	;	No - Test for escape
FD2C 3E7F	ESC	ESC	;	Delete
FD2E FE1B	CNVIN3: CP NZ,CNVIN4	CP NZ,CNVIN4	;	"ESC" ?
FD30 C235FD	LD A,CTRLC	LD A,CTRLC	;	No - Test for CR
FD33 3E03	TCR	TCR	;	Control C
FD35 FE1F	CNVIN4: CP NZ,CNVIN5	CP NZ,CNVIN5	;	NASBUG CR?
FD37 C23CFD	LD A,CR	LD A,CR	;	No - Return character
FD3A 3E0D	POP DE,BY,HL	POP DE,BY,HL	;	ASCII CR
FD3C DIC1E1	RET	RET	;	
FD3F C9			;	

Dis-assembly of NASCOM ROM BASIC Ver 4.7

PAGE 94	PAGE 95		
FD40 AF	CHKBRK: XOR A	;	Check for break
FD41 CD70FD	SFTENT CALL JP	;	Test for shift/enter
FD44 CA50FD	Z,TBK2 LD A,(BRKF1G)	;	Yes - Test for second break
FD47 3A4D10	OR A	;	Get break flag
FD6A B7	JP NZ,TBK2	;	Break flag set?
FD4B C250FD	XOR A	;	Yes - Test for second break
FD4E AF	RET	;	Flag no break
FD4F C9		;	
FD50 CD53FE	TBK2: CALL LD A,-1	;	Second break?
FD53 3EFF	RET	;	Flag break
FD55 C9		;	
FD56 DB02	GUART: IN RL A,(UARTS)	;	Get UART status
FD58 17	RLA JP NC,GUART IN A,(UARTD)	;	Any data ready?
FD59 D256FD	FD5C DB01 IN RET	;	No - wait until there is
FD5E C9		;	Get data from UART
FD5F D001	UARTOT: OUT (UARTD),A	;	Send data to UART
FD61 DB02	URTOFLP: IN A,(UARTS)	;	Get status
FD63 87	ADD A,A	;	Byte sent?
FD64 F8	RET M	;	Yes - Return
FD65 C361FD	URTOLP	;	Keep waiting
FD68 F5	UART: PUSH AF	;	Save A
FD69 CD5FFD	CALL AF	;	Send it to UART
FD6C F1	POP AF	;	Restore A
FD6D C9	RET	;	
FD6E 00	NOP	;	
FD6F 00	NOP	;	
FD70 E5	SFTENT: PUSH LD	;	Test for Shift Enter from KBD
FD71 3E02	LD HL,PORT0	;	Reset KBD counter mask
FD73 21000C	XOR (HL)	;	Get old contents
FD76 AE	OUT (O),A	;	Toggle bit
FD77 D300	OUT (O),A	;	Reset KBD counter
FD79 E001	XOR 0000001B	;	Toggle bit
FD7B D300	OUT (O),A	;	Next row
FD7D E002	XOR 0000000B	;	Clear "clear" strobe
FD7F D300	OUT (O),A	;	Get old value
FD81 7E	LD A,(HL)	;	Original contents
FD82 D300	OUT (O),A	;	?? WHAT ??
FD84 19	ADD HL,DE	;	Restore HL
FD85 E1	POP HL	;	Read in row
FD86 DB00	IN A,(O)	;	Mask SHIFT and ENTER
FD88 B612	AND RET	;	
FD8A C9		;	
FD8B CD6DFF	CLS: CALI	;	See if NAS-SYS
FD8E CA96FD	Z,TCLS	;	"?" CLS
FD91 3E0C	LD A,CS	;	ASCII Clear screen
FD93 C3D9FC	JP CONMON	;	Output character
FD96 3E1E	TCLS: LD A,TCS	;	NASBUG Clear screen
FD98 C3D9FC	JP CONMON	;	Output character

## Dis-assembly of NASCOM ROM BASIC Ver 4.7

## PAGE 96

```

FD9B AF      DELAY: XOR A          ; Delay routine
FD9C F5      DELAY1: PUSH AF       ; PUSHes and POPS delay
FD9D F1      POP AF             ; AF
FD9E F5      PUSH AF             ; AF
FD9F F1      POP AF             ; AF
FDA0 3D      DEC A              ; Count delays
FDA1 C29CFD   JP NZ,DELAY1     ; More delay
FDA4 C9      RET               ;
```

```

FDA5 CD84F4   WIDTHH: CALL LD    ; Get integer 0-255
FDA8 7B      LD A,E             ; Width to A
FDA9 324210   LD (LWIDTH),A     ; Set width
FDAE C9      RET               ;
```

```

FDBA CD41ED   LINES: CALL DEINT  ; Get a number
FDB0 CD88BE9  DEEK:  PUSH HL    ; Get integer -32768 to 32767
FDB3 ED534610  LD (LINESC),DE   ; Save number
FDB7 ED534810  LD (LINESN),DE   ; Number to HL
FDBB C9      RET               ; Get LSB of contents
```

```

FDBC CD88BE9  DOKE:  CALL GETNUM ; Get a number
FDBD D5      DEINT: CALL DEINT  ; Get integer -32768 to 32767
FDDC CD90E6   PUSH DE             ; Save address
FDD1 2C      CALL CHKSYN        ; Make sure "", follows
FDD2 CD41ED   CALL GETNUM        ; Get a number
FDD5 CD88BE9  CALL DEINT        ; Get integer -32768 to 32767
FDD8 E3      EX (SP),HL         ; Save value, get address
FDD9 73      LD (HL),E           ; Save LSB of value
FDDA 23      INC HL             ; Save MSB of value
FDDB 72      LD (HL),D           ; Restore code string address
FDDC E1      POP HL             ; RET
FDDD C9      RET               ;
```

```

FDE0 2A180C   FE02 E1          ; Disable interrupts
FDE1 22180C   FE09 E1          ; Flag cold start
FDE2 365F     FE0D 365F        ; Go and initialise
FDE3 10FD     FE36 10FD        ; ADD64X: ADD DJNZ ADD64X
FDE4 21FFFF   FE38 C9          ; RET
```

```

FDE5 10FD     FE38 C9          ; Column to DE
FDE6 2C      FE22 D5          ; Test it
FDE7 240E9   FE18 CAA0E9        ; Zero - ?FC Error
FDE8 1600   FE23 1600          ; 16 lines
FDE9 5F      FE25 5F           ; > 16 - ?FC Error
FDEA B7      FE26 B7           ; RETurn address
FDEB 2A0E9   FE20 D1           ; Get column
FDEC 19      FE21 F1           ; Re-save RETurn
FDEF 10FD     FE22 F2A0E9        ; Add column to address
FDE0 1600   FE24 F2A0E9        ; Column to DE
FDE1 59      FE25 1600          ; Test it
FDE2 0640   FE26 B7           ; Zero - ?FC Error
FDE3 19      FE27 CAA0E9        ; 48 characters per line
FDE4 10FD     FE28 0640          ; > 48 - ?FC Error
FDE5 10FD     FE29 F2A0E9        ; Add line
FDE6 240E9   FE30 1600          ; SIXTY FOUR TIMES!!!
FDE7 10FD     FE31 F2A0E9        ; Add line
FDE8 1600   FE32 59           ; Line to DE
FDE9 0640   FE33 0640          ; 64 Bytes per line
FDEA 19      FE34 0640          ; ADD line
FDEB 10FD     FE35 19           ; RET
```

## PAGE 97

## Dis-assembly of NASCOM ROM BASIC Ver 4.7

```

FDE6 CD84F4   SCREEN: CALL GETINT ; Get integer 0 to 255
FDE9 F5      PUSH AF             ; Save column
FDEA CD90E6   CALL CHKSYN        ; Make sure "", follows
FDEB 2C      CALL GETINT        ; Get integer 0 to 255
FDEE CD84F4   CALL BC             ; Column to B
FDF1 C1      POP BC             ; Save code string address
FDF2 E5      PUSH HL             ; Save column
FDF3 C5      PUSH BC             ; Calculate screen address
FDF4 CD1FF  CALL SCRADR        ; Save screen address
FDF7 E5      PUSH HL             ; See if NAS-SYS
FDF8 CD6DFF  CALL MONST          ; "T" monitor - "T" cursor
FDFB CA04FE  JP Z,TMNCUR        ; Restore screen address
FDFE E1      POP HL             ; Set new cursor position
FDF9 22290C   LD (CURSOR),HL    ; Rstore code string address
FDFC 21C907   SCRADR: LD HL,YDU+10-65 ; Get address or cursor
FDFD 0600   LD B,0              ; Remove cursor
FDFE 4F      LD C,A             ; Set new cursor address
FDFG 1600   LD (TCUR),HL        ; Put it on screen
FDFH 2A0E9   LD (HL),"—"        ; Restore code string address
FDFI 0600   RET               ;
```



## Dis-assembly of NASCOM ROM BASIC Ver 4.7

PAGE 100

Dis-assembly of NASCOM ROM BASIC Ver 4.7

PAGE 101

```

FF02 D5 LINTBF: PUSH DE ; Line length to BC
FF03 C1 POP BC
FF04 03 INC BC
FF05 116110 LD DE,BUFFER
FF06 C1 INC BC
FF07 E1 POP HL
FF08 C5 PUSH BC
FF09 09 PUSH BC
FF0A EDB0 LDIR A,0
FF0C 3E00 LD (DE),A
FF0E 12 POP BC
FF0F C1 LD B,C
FF10 41 LD HL,BUFFER-1
FF11 216010 RET
FF14 C9

FF15 CD90E6 GETXYA: CALL CHKSYN
FF18 28 DEFB "("
FF19 CD41ED CALL GETNUM
FF1C CD88BE9 CALL DEINT
FF1F D5 PUSH DE
FF20 CD90E6 CALL CHKSYN
FF23 2C DEFB ","
FF24 CD41ED CALL GETNUM
FF27 CD90E6 CALL CHKSYN
FF2A 29 DEFB ")"
FF2B CD88BE9 CALL DEINT
FF2E E5 PUSH HL
FF2F FDE1 POP IY
FF31 CD96FP CALL XYPOS
FF34 F5 PUSH AF
FF35 CDC2FF CALL ADJCOL
FF38 CD11FF CALL SCRADR
FF3B F1 POP AF
FF3C 06C0 LD B,11000000B
FF3E B0 OR HL
FF3F C9 RET
FF40 CD15FF SETB: CALL GETXYA
FF43 F5 PUSH AF
FF44 7E LD A,(HL)
FF45 FEC0 CP 11000000B
FF47 D250FF PUTBT: JP NC,SETOR
FF4A F1 POP AF
FF4B 77 PUTBT: LD (HL),A
FF4C FDE5 RESCSA: PUSH IY
FF4E E1 POP HL
FF4F C9 RET
FF50 C1 SETOR: POP BC
FF51 B0 OR B
FF52 C34BFF JP PUTBIT

```

; Line length to BC  
; Length +1  
; Input buffer  
; Line start  
; Save length  
; Move line to buffer  
; Mark end of buffer with 00  
; Restore buffer length  
; Length returned in B  
; Point to start of buffer-1  
; Make sure "(" follows  
; Get a number  
; Get integer -32768 to 32767  
; Save "X"  
; Make sure ")" follows  
; Get integer -32768 to 32767  
; Save code string address  
; In IY  
; Address and bit mask  
; Save mask  
; Adjust column  
; Get VDU address  
; Restore bit mask  
; Block graphics base  
; Set bits 7 & 6  
; Get co-ords and VDU address  
; Save bit mask  
; Get character from screen  
; Is it a block graphic?  
; Yes - OR new bit  
; Restore bit mask  
; Put character on screen  
; Restore code string address  
; From IY  
; Get co-ords and VDU address  
; Save bit mask  
; Get character from screen  
; Is it a block graphic?  
; Yes - OR new bit  
; Restore bit mask  
; Put character on screen  
; Restore code string address  
; From IY  
; Get co-ords and VDU address  
; Save bit mask  
; Get character from screen  
; Is it a block graphic?  
; Yes - Leave it  
; Six bits per block  
; Clear bits 7 & 6  
; Get bit mask  
; Test for common bit  
; None - Leave it  
; Get byte from screen  
; Isolate bit  
; Clear that bit  
; Is it a graphic blank?  
; No - Save character  
; Put a space there  
; Save the space  
; Drop bit mask  
; Restore code string address  
; Get co-ords and VDU address  
; Get character from screen  
; Test if bit is set  
; Different - Return zero  
; Integer AB = 1  
; Drop return  
; PUSH code string address  
; To return a number  
; Save for return  
; Return integer AB  
; Set zero  
; Return value  
; Get return address  
; Get column  
; And re-save  
; Put back return address  
; Get column  
; 2 bits per character  
; Odd or even bit  
; Save it  
; Get row  
; to HL  
; Zero line count  
; 3 blocks per line  
; Subtract 3  
; Count the subtractions  
; Exactly - Exit  
; More to do

Dis-assembly of NASCOM ROM BASIC Ver 4.1		ABPASS	F0F2	ABS	F838	ACCSUM	E9
ADDEXP	F7D1	ADDIG	F977	ADDPLH	F51C	ARET	F0
ANTVLU	EC65	ANTNAM	0C1B	ARLDV	F0A5	BSRLL	F0
ARG2	0C0E	ARRSV1	E8BB	ARYLP	E917	CFEVAL	EF
ARRFLP	F2B8	ATN	FC7C	ATNL	E3	CHKSTK	E3
ASPCS	EBC7	BADINP	EBEC	BAKSTK	E3	CHSUNS	E9
BAD	F59D	BFRINE	EB07	BKSPW	00	CLOUD2	F6
BCDFEP	F85F	BREAK	FE53	BRKFLG	10	CLRPT	E4
BREAK	FDE	FEDE	BS	BSERR	0010	CMPRES	EE
BRKET	EB01	CASFNW	FCC8	CFVNL	FD	CNVIN3	FD
CASFK	FCDF	CHKLTR	E977	CHKSTK	FD27	COMPL	F6
CHKBK	FD40	CHR	F3A2	CHSUNS	FD	CONOP3	FD
CHKTP	ED46	CLOUD1	F52B	CLRPT	FD19	COPY	EC
CLOUD	FE49	CLRFST	E4DF	CR	00	CPYLIT	EE
CLOSET	E626	CMPPNM	F88E	CRLIN1	FF77	CRUNCH	EE
CMPLOG	EEBA	CRTSTE	F1E4	CTRLC	00	CTRLU	00
CNVINI	FD20	CTLFG	1043	DD	00	DATA	00
CNVNUH	F926	COMMON	FCF1	DEINT	00	DEINT	EE
COMMON	FCD9	CONN01	FD13	DELCHR	00	DELSRC	EE
CONVAR	E222	CONVNTN	FD19	DIR	00	DISSRC	EE
CPDEHL	E68A	COPYLT	E5B1	DIM	00	DISVOL	EE
CRESTR	EA9C	CRTLINE	FB0D	DIV10	00	DIVA	14
CRESTR	F1CE	CRTSTE	FF6D	DIVEL	00	DODBL	E
CSTART	ED12	CTLFG	1045	DOSPC	00	DOSPC	EE
CTRLR	0012	CTRLS	0013	DTBDE	00	DTBDE	F
CURPOS	10AB	CURSOR	0C29	DTBLU	00	ENDBUF	E
DATOSNR	EA37	DCBDE	FBDF	DD	00	ENDBUF	E
DEF	F106	DEFNSIZ	FF6D	DEINT	00	ERRMSG	E
DELAY1	FDFC	DELAYB	FFCD	DELCHR	00	EVAL1	E
DEPTHBL	F870	DIGITXT	FA20	DIM	00	EXECUTE	E
DIV	F767	DIVI	100A	DTBLU	00	EXTIG	E
DIV3EX	FFB1	DIV3LP	FFA8	DTBLU	00	FILE	F
DOAGN	E4F8	DOCOM	EB98	DTBLU	00	FLGVER	F
DOKE	DC7C	DONULL	EB86	DTBLU	00	FNDELP	F
DPOINT	F952	DTSTR	F1D2	DTBLU	00	FNDRWD	F
ECHDEL	E5D5	EDIGIT	F999	DTBLU	00	FOPRD	E
ENDINP	E7C7	ENDLN	FEF1	DTBLU	00	FOUND	F
ERRIN	E3E1	ERRLIN	10D2	DTBLU	00	FPMULT	F
ESC	001B	EVAL	ED5A	DTBLU	00	FRE	F
EVLPAR	EE09	EVNTC	EEF8	DTBLU	00	GETCHR	E
EXPTAB	F3A4	EXPEN	FA72	DTBLU	00	GETLEN	F
FCERR	E9A0	EDTLP	ECD2	DTBLU	00	GETSTR	F
FLGDF	F81E	FLGREL	F825	DTBLU	00	GOFUNG	F
FNCFTAB	E10F	FNDAY	FF18	DTBLU	00	GRBLP	F
FNTHKR	E71B	FNDAYR	EE8F	DTBLU	00	GTEFLNM	F
FOREND	E7A9	ENVAL	EE5F	DTBLU	00	GTFNAM	F
FPEXP	10E7	FPIINT	F8BB	DTBLU	00	HALFW	F
FPSINT	E97F	FPTRL	F86B	DTBLU	00	IDTEST	F
GARBLP	F253	GARBLP	F256	DTBLU	00	INCLEN	I
GETINP	FD05	GETINT	F484	DTBLU	00	INCNLH	I
GETNUM	ED41	GETNKT	E557	DTBLU	00	INCNLH	I
GETXKA	FF15	GNAKAY	F28A	DTBLU	00	INCNLH	I
GRBARY	F2AA	GRBDON	F22B	DTBLU	00	INCNLH	I
GSTRHL	F556	GTEFLNM	F395	DTBLU	00	INCNLH	I
GTVLUS	EC3D	GUARD	FD56	DTBLU	00	INCNLH	I
ID	0016	IDTEST	F17B	DTBLU	00	INCNLH	I
INCGRH	FD00	INCNLH	F869	DTBLU	00	INCNLH	I

## Dis-assembly of NASCOM ROM BASIC Ver 4.7

## Dis-assembly of NASCOM ROM BASIC Ver 4.7

PAGE 105

INIT	E01.9	INITAB	E2DF	INITBE	E33F	INITST	FEBB	INLINE	FE88	S1N1	FC36	SINTAB	FC52	SIXDIG	F9D5	SMOTOR	FFD1	
INNSG	E346	INP	E441	INPBIN	E48F	INPBK	E877	IMPORT	103F	S1SER1	FB6A	SN	0002	SNDARY	E91D	SNDHDR	E906	
INPSUB	103E	INPUT	E4F0	IRNG	F9F3	INT	F8E8	INTVAR	E4C9	S1CFST	F9C6	SPCLP	EBCB	SQR	0002	SRCHLP	E49C	
INVSFN	F84C	ITMSEP	ECS0	JJUMP	FFF4	JJUMP1	FDD6	JPLDSV	E4C9	S1TSAA	F33D	ST	001E	STACK	E066	STAKPP	E866	
JSTZER	F47C	KILFOR	ED31	KILLIN	E5EC	LCRFLG	10AC	LDNM1I	E73C	START	E003	STKTHS	EDBA	STOOLK	115D	STMON	Q00D	
LEFT	F3B2	LEN	F3B2	LET	E857	LETIN	EADA	LETSTR	EAA2	STORED	E9ED	STPOOL	F2B8	STR	F19A	STRADD	F2B8	
LEN	0004	LFRGMN	F437	LINEAT	E05C	LINEIN	F9A5	LINES	FDAD	STRENT	E070	EC83	STRSPC	105A	STTLIN	E874	SUART	FD68
LF	LINESC	1046	LINENM	1048	LINEND	E3E	LINTBF	FF02	LIST	E6DD	SUBCDE	F1CA	SUBPHL	F5C4	SUMSER	E937	SUMOFF	FB58
LISTLP	E6E9	LOADFP	F862	LOG	F6C7	LOGTAB	F6BA	LOKFOR	E35A	S1PTLZ	F454	SYNAME2	F1C8	T2DUMP	03D1	T4READ	070C	
LS	001C	LSTC	10C7	LSTBIN	E0CC	LSTLIP2	E709	LSTLIP3	E70C	T4WR	0400	TAN	F067	TBK2	FD50	TBS	001D	
LSTRND	10AF	LSTRND	103E	LSTND	E09A	LWIDTH	1042	MAKINT	F487	TCHINP	F662	TCLS	FB96	TCLR	001F	TCS	OC18	
LSTRAM	10AF	MANIP	F92E	MATCH	E589	MEMMSG	E103	MFLP	0051	TESTOS	F247	TMERR	F548	TLS	0040	TILOAD	FE99	
MAXNUM	FA0F	MID	F3EC	MID	F3B8	MIDNUM	F43C	MINGDE	F0D	TM	0018	TMERR	F642	TMNCUR	FE04	TMOUT	FE4F	
MIDBLP	F1BF	MKTMST	F907	MLDIBC	F90F	MLOOP	E049	MLSPL0	F7FC	TMSTR	10B3	TMSTPL	10B1	TOPOLL	F3AE	TOSTRA	F346	
MON	0024	MONITR	FEA2	MONLUD	FEE8	MONSTT	0000	MONSTT	F45F	TOUT	0C4A	TRYAGN	FA2F	TSAVE	F347	TSTBIT	FB7F	
MONST	F6ED	MONTYP	008D	MONVNE	FEAA	MORDT	EC46	MORINP	E610	TSPRK	E861	TSYMEM	E05B	TSTOPL	ED44	TRSTRED	EEA8	
MOVBUF	E474	MOVEDR	E591	MOVULP	E34F	MOVSTR	E37C	MOVUP	E379	TESTRM	E5A2	TSESGN	F813	TTYLIN	E607	TYPE	10AD	
MRPRNT	EB1F	MSIZE	E036	MULBLP	F733	MULLN2	F6FF	MULT	F706	UARTOT	0001	UARTD	FD5F	UARTL	0022	UFERR	E3B9	
MUL18	F72A	MULTEV	10F6	MUSTPT	EAD1	NASPUT	F4CB	NASPUT	FB1F	TMSTR	10B2	TMSTPL	10B1	URTOLP	FD61	URTOPL	FB61	
NEDMOR	E339	NEGART	FAA7	NEW	E4B9	NEKITM	E070	NEKITM	E070	OCIA	0C4A	TRYAGN	FA2F	TS47	FB7F	WORDTB	E2A	
NEXT1	E4F9	NF	0000	NFEAR	E3B3	NMI	0C7E	NMIFLGL	FB4C	WAITLP	F466	WARMST	ED45	WORDTS	E143	WORDTB	E2A	
NOCHNG	E581	NOEND	FA7F	NOFLD	E8BD	NONADD	F744	NONMLD	F915	WRKSPC	1000	WUART	F4B4	WT4BT2	F4B7	ZDATA	0083	
NOOMP1	E517	NOREM	FFBA	NORES	FF75	NORMAL	F638	NOSPC	E578	ZDIV	00AF	ZEND	0080	ZEQUAL	F0B3	ZEBBYT	E3AA	
NOSWAP	F5E7	NOTSTR	EF65	NOXOR	F467	NSCEOR	EF75	NULFLG	1044	ZEROLP	EFCE	ZFN	00A7	ZFOR	ZGOSUB	008C	ZGOTO	0088
NULL	E8B1	NULLP	EB8D	NULLS	1041	NUMASC	F9B3	NXTARY	FC0C	ZGTR	00B3	ZJUMP	FB4D	ZMINUS	ZLTH	ZMINUS	00B5	
NXTBYT	E567	NXTCSR	E5A8	NXTDAT	10DC	NXTEDTA	E6F	NXTITM	FC31	ZNOTE	00AA	ZNOT	00AA	ZONELP	EBA6	ZPLUS	00B2	
NXTOPR	10D0	NXTSTL	EA76	NXTSTT	EAT9	OD	0006	OKNSIG	E3AB	ZPOINT	00C7	ZPRINT	009E	ZREM	008E	ZSPC	0048	
OM	000C	OMER	E3A2	ON	EAE1	ONGO	EAFO	ONGOLP	EAF1	ZSTEP	00AB	ZTHEN	00A5	ZTIMES	00A9	ZTO	00A6	
ONJMP	E81E	OPNPAR	ED56	OPRNDR	EDD1	OS	001A	OTKLIN	E5E9	OUTCH	001A	OUTCP	FA0	OUTEXP	FA0	OUTF	FB45	
OPPORT	1007	OUTBAD	F56B	OUTC	E69B	OUTNC	F7F4	OUTSUB	1006	OUTWRD	E725	OUTVTD	F7F5	OUTVST3	F7F5	OUTVTD	E725	
OUTIT	E67C	OUTNBS	E3BC	OUTNR	E071	OUTNST1	F7EF	OUTST2	F7F4	OUTVST3	F7F5	OUTVST3	F7F5	OUTVST3	F7F5	OUTVST3	F7F5	
OV	000A	OVERL	EE82	OVERL	E3BC	PASSA	F101	PBUFF	10E9	PEEK	F5A3	PRINT	E2B3	POWER	E2A5	POWER	E2A5	
PADD	F994	PAND	EE81	PASSA	F672	PLUCDE	E672	PNORM	F640	POINT	1051	PRNTBL	EB69	PRNTOK	E3F8	PRNTOK	E3F8	
PEND	E872	PHLTTP	F851	POINTR	F7F9	POINTER	F8F7	POKE	F5AA	PROGND	10D6	PROGST	10F9	PROGST	10F9	PROGST	10F9	
POINTO	FF91	POPHRT	F754	POPNOK	E3F7	PORNOK	E3F7	POR	EE80	PORT1	EEA3	PORT1	EEA3	PORT1	EEA3	PORT1	EEA3	
POPHL	F63F	POPHRT	F754	POFE	F982	POINT	E982	POUT	F44D	POUTBUF	E66D	POUTBUF	E66D	POUTBUF	E66D	POUTBUF	E66D	
PORTO	0000	POS	F8F0	POFE	F982	POWERS	F985	POWERS	F985	PRINT	E2B3	POWER	E2A5	POWER	E2A5	POWER	E2A5	
POWER1	FAC5	POWER2	FAD0	PRNTBL	F9AD	PRNTBL	E826	PRNTBL	E826	PRNTBL	EB69	PRNTOK	E7A1	PRNTOK	E3F8	PRNTOK	E3F8	
PRNTCR	E8B1	PRNTBL	F9AD	RESET	1057	RESETC	FF55	RESETL	FF55	RESTNL	E85B	RESTOR	E846	RESTOR	E846	RESTOR	E846	
PRNTST	E8B2	PRNTST	F20F	PRNS	F210	PRNS1	F213	PRSLP	F21A	PSET	1054	PSET	1054	PSET	1054	PSET	1054	
PROMPT	E4FC	PRNS	F210	PRNU	F6B2	PRNU	F6B2	PRNU	F6B2	PRNU	E81C	RETREL	E81C	RETREL	E81C	RETREL	E81C	
PSUB	F5C8	PTRLP	F481	PUTBUL	F64F	PUTBUF	E66B	PUTBUF	E66B	RINPUT	104E	RINPUT	104E	RINPUT	104E	RINPUT	104E	
PUTFID	E75E	QTSTLP	F1D5	QTSTR	F1D5	QUATR	F1D5	QUATR	F1D5	QUATR	FC4E	READ	E2C2	READ	E2C2	READ	E2C2	
READFG	10CD	RDO	EBD9	REM	E7A2	RESCSA	FF4C	RESCSA	FF4C	READY	F7A1	READY	F7A1	READY	F7A1	READY	F7A1	
REFED	F8E5	REFED	F20F	RETCF	F629	RETCF	F629	RETCF	F629	RETNL	E846	RETNL	E846	RETNL	E846	RETNL	E846	
RESER	F633	RETADR	F4B4	RETNU	F4B4	RETNU	E81C	RETNU	E81C	RETREL	E81C	RETREL	E81C	RETREL	E81C	RETREL	E81C	
RETNA0	E58D	RETNU	E4F0	SAVE	FE73	SAVEXP	F634	SAVEXP	F634	SAVSTP	E7E5	SAVSTP	E7E5	SAVSTP	F1AA	SAVSTP	F1AA	
RG	0004	RIGHT	F3E2	RIGHT1	F3E2	SCALLP	F634	SCALLP	F634	SCALMI	F959	SCALMI	F959	SCALMI	F96F	SCALPL	F96F	
RND	FBB8	RND1	FBC7	RND2	FBC7	RND2	FBC7	RND2	FBC7	RNDTAB	F8F4	RNDTAB	F8F4	RNDTAB	F8F4	RNDTAB	F8F4	
RONDB	F654	RONDU	F653	ROUND	F653	RSCALE	F98E	RSCALE	F98E	RSCALE	F98E	RSCALE	F98E	RSCALE	F98E	RSCALE	F98E	
RSTSTR	F405	RHART	F4B4	RUN	EAI0	RUNCT	E7F2	RUNCT	E7F2	RUNCT	E7F2	RUNCT	E7F2	RUNCT	E7F2	RUNCT	E7F2	
RUNLIN	E42C	SAVE	FE73	SAVEXP	F634	SCALRP	F634	SCALRP	F634	SCALMI	F959	SCALMI	F959	SCALMI	F959	SCALPL	F96F	
SBSOPT	EFEA	SCALE	F692	SCALE	F692	SCRADR	F611	SCRADR	F611	SCRADR	F611	SCRADR	F611	SCRADR	F611	SCRADL	F611	
SCNEND	F2E1	SCPPLP	EFF0	SETIO	F471	SETIO	F471	SETIO	F471	SETLIN	E733	SETLIN	E733	SETLIN	E733	SETLIT	E59F	
SEED	1017	SETB	FF40	SETOP	E0D	SETOP	E0D	SETOP	E0D	SETOP	F070	SETOP	F070	SETOP	F070	SFTPRG	E446	
SETOR	F5F0	SETPTR	E47C	SIGNR	E088	SIGNR	E088	SIGNR	E088	SHTBT	F6A1	SHTBT	F6A1	SHTBT	F6A1	SHTBT	F6A1	
SGN	F822	SGNEXP	EE70	SIGNON	E0C5	SIGNON	E0C5	SIGNON	E0C5	SIGNON	F6A4	SIGNON	F6A4	SIGNON	F6A4	SIGNON	F6A4	
SHRLP	F6A4									SIN	F6A4	SIN	F6A4	SIN	F6A4	SIN	F6A4	

THE END

Dave Hunt's BitsIntroductory Wafflings

Time flies, it seems like only yesterday I was writing my last piece in a panic to get something into the mag. Yet, in the intervening time I have been on holiday, come back again, sat around, another mag. has come out, I have done this and that and I still don't feel inclined to start setting anything down for the next issue. Still having read the letters in the last issue, and having a distinct feeling that I'm being got at, I have summoned up the energy to put fingers to keyboard, and pour out more of the same old gibberish I'm famous for.

As I seem to have got away with some rude remarks about our Editor in my last piece, we'll try again. He's had a new toy for the last six months!! No, it's not Viv, as mentioned in my last article ... but read on. If any one in the readership happens to be a traffic cop on the M1 or M6, en route from Chesham to Blackburn, and if you happen to stop a white TVR doing what it's supposed to (about 150, that is), don't put him away for too long, or we'll not see another mag until you let him out again .... That's if you can catch him in the first place!!

Computer Illiteracy

Now there seems to be an interesting case of computer illiteracy in our local junior school, and I hasten to add that this school is one of the better thought of schools in our area, progressive but sensible with it. Anyway the story goes like this:

They recently spent some of the PTA loot, extorted from mugs like me who have to spend good money going to school fetes and things, on four computers of the type very popular in schools right now. That's four computers between 400 kids, a pretty poor ratio for a start. Then they sent one class teacher off for a one day course on teaching the kids the mysteries of the computers. Now I believe you can learn a lot in one day, but the mysteries of computers ... in one day? Well when I was dragged along to the last open evening by the Mrs (I had to go, duty or something, as she's on the Board of Governors), I decided to find out what they did with these computers. I chatted up the teacher in charge of the computers and tried to find out what was taught. Well it wasn't programming as he didn't know a PEEK from a POKE, it wasn't how it worked, because he didn't know what a processor was, yet alone what type, or what the accumulator was, or anything. I was told they had 32K of RAM whatever that was, but he seemed to somewhat wooly about how much of that was workspace, video RAM, etc. What they did have was some "educational" software, but the purposes of this were somewhat obscure, despite the teacher's obvious enthusiasm.

In all quite a waste of twelve hundred quid if you ask me, but I haven't finished yet. Number one daughter, who is twelve, shows no inclination towards computers apart from playing "Wheelie" on a Spectrum. She's more into carpentry at present and spends her time murdering pieces of wood and mangling my tools.

Number two daughter is different, ten years old and altogether a quieter and more studious character. She wants to learn about computers, and fully appreciates that if you can program them, they can do very clever things far faster than you can do them. So after seeing the computer teacher, she went to the school library and borrowed some books. Rather good books they are, with serious intent, but amusing with it. Partially in comic strip, with cartoon bugs upsetting the programs, and written in the most generalised way. They are written for Microsoft BASIC, so what is so syntactically different, that you can't get them to work? The type of computers they use in schools of course.

Pitiful isn't it. Kids with interest can't get past the first few hurdles because the machines they have don't match the books in the library, while the teachers whilst keen, don't know what it's about. Number two daughter is now allowed on my machine, and is bashing in all the programs from David Ahl's book "101 BASIC Computer Programs". Good practice is this book, it's written for Microsoft BASIC and because many of the programs don't work properly they require modification to make them go. She's actually getting very good, and is writing a database to keep track of her bird watching activities.

Altogether not bad for a ten year old (Dads are allowed a little pride in their kids, aren't they?), but an indictment on the current way computers are being taught in our local junior school. I hope this isn't par for the course in other junior schools, but I rather suspect it might be.

The printer revolution (horrid pun, you work it out)

A couple of years ago daisy wheel printers were priced such that you could only afford one if you were the sort of chap who could afford to buy gold plated ashtrays for the spare Roller. Certainly not the sort of thing to be purchased by the home user (to write poshly typed pleading letters to the Bank Manager), nor the sort of thing which may be used by smaller companies to write a couple of dozen letters a day, or to bash off the odd quote or two. In other words they were expensive!

Despite their price what you got for your money was something which was built like a battleship (so it didn't wear out churning out thousand upon thousand of circular letters which looked hand typed) and that went at a fair old speed (so it didn't take a month of Sundays to churn out the thousand upon ....). You know the sort I mean, the Readers' Digest ones which get consigned to the round filing cabinet unread. Qume and Diablo were the byword for these printers. As such, these beasts were (and still are) worth the money. And if you can lay hands on one at a reasonable price, they would still do for the better quality print job around the home.

However, over the last year a number of "cheaper" daisy wheel printers have appeared. Some of these are simply office typewriters fitted with computer interfaces, some are purpose built computer printers. The office typewriter types tend towards the lightweight, both physically and also in longevity.

Two or three years ago one famous typewriter manufacturer got very cross with some enterprising people who had fitted an interface to one of their lightweight typewriters which was not designed to pound out text, sheet after sheet at a continuous 12 characters per second without stopping every now and then for coffee or to get to the loo (the typist, not the typewriter; before I get any more letters about DRH's funny little mistakes). The reliability problems started to get the manufacturer a bad name. Things have improved now, and the interfaced office typewriters are now robust enough for the average low duty cycle of the typical one person word processor user. Some even have the keyboard arranged to be a serial output device so it can be connected to the computer as the main keyboard. These also act as stand alone office typewriters, and so are seeing increasing use in offices around the country. As far as I'm concerned the only snag with these is the size. With the built in keyboard they are too big for my situation (and I dare say most other home users). I need something I can stick in the corner, and as I already have a keyboard for my computer I don't need (nor want to pay for) another.

When I started playing with computers the only choice at a price I was prepared to pay, was an antiquated IBM golfball. Solid (if its weight was anything to go by, made of lead), reliable, horribly mechanical, noisy and slow. These could be purchased for two or three hundred pounds and provided hours of innocent amusement trying to interface them to a Nascom. I wonder how many readers remember the early issues of the INMC magazine, that was printed on the old IBM golfball, which I might add is still giving sterling service, although no longer in my ownership. Somewhat dearer at that time were surplus Diablo and Qume daisy wheels, equally as heavy and reliable, less mechanical and more electronic, almost as noisy, a lot faster and a nightmare to interface as a lot of them were not fitted with serial or parallel interfaces as we know them today. Software and the necessary hardware drivers for these were diabolical, hampered by the total lack of cooperation on the part of the manufacturers in supplying interface information.

Whilst thinking of interface and service information, I remember IBM computers were totally unhelpful when it came to details on their golfball printers. However, in this instance, a point worth noting was that the IBM typewriter

division didn't exactly go a bundle on the 'stuck up twits' in the computer division (their description, not mine). When we told the typewriter division just how unhelpful the computer lot were, the typewriter guys bent over backwards to provide all the stuff we needed.

More recently, a number of medium priced (500.00 to 600.00 price bracket), medium duty daisy wheel printers have become available, mostly from the Far East. Brother, the Japanese typewriter people (is it the same as the sewing machine and knitting machine company?) have a number of daisy wheel computer printers with or without keyboards in all price brackets except the very lowest. Of the Brother machines the best way to describe them is 'average', you seem to get what you pay for. None are exceptional for the price, but all seem to be provided with the facilities that you might expect from the competition at a similar price. Qume, Olivetti, Triumph Adler, Tec and many others have followed the lead with machines in this bracket. But these are still too expensive for the home and small business user.

The Juki 6100 (I might have the number wrong, but the one you can get for about 300.00) is a very nice machine. Not fast, but a well built mechanism and very precise character registration. The only thing I don't like is the choice of wheels and ribbons. Now I'm all for standardisation in this area. I recently came across a 'Kores' printer ribbon catalogue (trade bible), and there are about 600 different types of cassette ribbon in the catalogue. About half of these are visually similar, but bet your bottom dollar, a ribbon for one machine which looks almost like a ribbon for another won't fit. The same applies to daisy wheels, there is a bit more standardisation here, but a growing trend for each printer manufacturer to make his own unique wheel for his printer. Like I says, the Juki have chosen Triumph Adler wheels and ribbons for their printer. On the face of it a very sensible idea, as these are the wheels and ribbons used by the Triumph Adler office typewriters, and so, should be available at your local stationers. Well I tried our local branch of a big chain stationer, and yes, they had them. The snag? The price. 6.00 for a ribbon and 22.00 for an ordinary plastic daisy wheel. I tried the local Indian stationers, he was a bit cheaper for the ribbons, 5.50, but didn't have the wheels. What a rip-off!!.

Two daisy wheel printers have recently come under my scrutinization, and having given them an intense scrute, I will now pronounce myself!! The Sample Daisy Step 2000 and the almost identical (but not quite) Quen-Data printer. Both printers cost about 225.00 and both are Qume clones in all but speed and quality of construction. Actually, Quen might just be a Far Eastern misspelling for Qume, who knows. Anyway, both use Qume wheels, which are cheap at about 5.50 to 6.00, and available in a vast range of type styles and pitches (if you can only find someone who actually keeps more than just a couple of types), and the carbon multistrike ribbons are cheap at about 3.00 and seem to last for ever.

Both printers have a Centronics interface with a 256 character buffer. The Sample has a serial RS232 option at about 55.00 with a 2K'buffer. The Quen probably has a serial option available, but I haven't yet discovered for sure or how much. The printer control protocols are a superset of the Qume Sprint 5, and being a superset have a few extras thrown in, for example software selectable bidirectional logic seeking printing, selectable hammer strike pressure, etc. Printing speeds are about 20 characters per second with high speed skipping of white space. All the internal software seems to work without bugs, and although slow, the print output is of very good quality.

What about the mechanism? Well it's Ok so long as you keep the lid shut and don't look too closely. Pressed steel chassis, guides and runners where the Qume has a diecast aluminium chassis and precision ground steel guides and runners. The carriage is poorly located on the guides with nylon carriage slides. By contrast the Qume has phosphor bronze bushings for the front guides and sprung ball races for the rear. A not so subtle difference between the Qume mechanism on the one hand, and the Sample and Quen on the other. Mind you the price difference isn't so subtle either, the Qume costs nearly ten times as much as the other two.

The point is that both these cheap printers work, and work well. There can be no doubt that they won't stay working as long as the Qume, but then in a home environment, that sort of longevity isn't required. These two are DRH's best buy at the moment.

#### And so on to other things

I've spent some time doing some late spring cleaning amongst my disks and in the process came across a nice little program by Ward Christensen. Now for those who aren't devotees of the various CP/M User Groups, Ward Christensen is a software writer of some renown and greater output. You could be forgiven for thinking that he has single handedly written all the software on all the couple of hundred US CP/M User Group disks. He hasn't, but his name must certainly appear more often than most. Anyway, SCRAMBLE is a neat little encryption program for scrambling the text or code of a disk file. I liked the program so I spent a little time doing a little tidy here and a little tidy there just to provide fodder for those who like to find DRH's deliberate mistakes. Now I'll not take the blame for all the mistakes, only the ones I've committed, as the source code came to me as a Z80 source program. Now I haven't yet seen a Ward Christensen program written in Z80, it's always 8080 assembler, so someone has translated the file, and possibly done goodness knows what else in the process. So with no more said, I'll re-print Ward's original DOC file here, and the source will appear somewhere else in this mag.

"SCRAMBLE is a command used to encode a CP/M f"

The format of the command is:

"SCRAMBLE filename.type password"

where "password" is an 8 character password made of characters permissible in a file name (i.e. no ".", etc). To obtain a good "initial seed" for the scrambling process, no character in the password may appear more than twice.

The requested file is scrambled, and re-written in place. To unscramble the file, the IDENTICAL command is issued, i.e:

>SCRAMBLE filename.type password

This is because SCRAMBLE does an "exclusive-or" type modification to the file, and doing two identical exclusive-or's to data result in the same data being returned.

I feel a scrambled file is quite secure. Given that a file was scrambled and the password forgotten, I know of no way to determine what the original file was. Even a file which is all binary-0's, is sufficiently scrambled to defy finding out what the password or original data was. ...But I assume no responsibility for the "security" of files scrambled with SCRAMBLE as I am not a "student of cryptology".

Note also, that if an attempt is made to unscramble a scrambled file, using the WRONG password, then the file is technically "double scrambled" and SCRAMBLE would then have to be executed TWICE, once with the original password, and once with the erroneously-used password. Because of the exclusive or-ing process, either password may be used either time.

03/11/79 Ward Christensen"

#### Comms Software

Following Ward's Scramble, I've had a letter from Arto Liimatta in Finland, who has thrown all the beautiful and complicated comms software out of the window and come up with a very simple patch to PIP.COM to make it communicate with another PIP on another machine, suggested by an article in "Microsystems" July 1983. As he writes,

"I also put here one program PIP, and how to use it for transferring data between machines. It uses some kind of handshaking in the transmitting procedure, so the receiving machine won't lost data, while it is saving it to disk. The transmitting computer sends a character and when the receiving computer gets it, it echoes it back to the transmitter who then knows the receiver is ready. When the receiver is not ready it does not echo until it is ready to receive. I think it has been used for many years, but I haven't seen it yet in 80-BUS News."

Now I haven't seen this one either and the code consists of a simple patch in the front of PIP. Once changed, this version of PIP can't be used for anything else, so Arto's suggestion is that the patched PIP be called IOPPIP. The code came as a hard copy dump and wasn't quite complete. Fortunately Arto also sent a disk with IOPPIP.COM on it, so I disassembled it and then re-assembled it. I then checked the newly assembled code was the same as the code on disk, just in case. Now if you look at the patch area in PIP as supplied, you'll see that there is the start jump, two fake RETs and a great wedge of bytes which read (INP:/OUT:SPACE). This routine neatly fits in this space with acres to spare, which now I know the space is there, I can think of a number of things which could be shoved in. Anyway, Arto's source code is shown later.

All very simple and straight forward, this program should work with a direct wired connection or through a modem via the telephone, but there have to be snags. One problem is that you can not assume an 8-bit data transfer, some machines only have 7-bit serial I/O implemented, so transferring object code files is a problem. Apart from that Arto's letter implies that the [0] option with PIP doesn't work as the receiver can not see a ^Z under this option and therefore doesn't know when the file is finished (I know this to be true, and the [B] option doesn't work either). Hitting RESET does not close the file, so some or all the file will be lost. His suggestion is to transmit all files in HEX in the Intel HEX format and use the LOAD.COM program provided with CP/M to convert back it into a .COM file. This is alright if you have the CP/M User Group utility UNLOAD.COM which converts a file into Intel HEX format. If you haven't tough! Well perhaps not tough, just tough work, you see, I've just used UNLOAD to unload itself, so if you are really masochistic, you can sit down and type the following in using a word processor and then use LOAD to turn it into a .COM file. If you use Wordstar, do this in the non-document mode, otherwise you'll have soft carriage returns, and that will probably upset LOAD.

Title SCRAMBLE subttl Program to scramble CP/M files using an 8-bit password.

```

        day    equ   13      ; Last..
        month  equ   10      ; ..modification..
        year   equ   84      ; ..date

```

Comment "SCRAMBLE" by Ward Christensen  
Minor assembly changes. D. R. Hunt 13/10/84

Scrambling is done in place, i.e. the file is modified on top of itself. The same password used to scramble the file is used to unscramble it, using the exact same command. This is because the scrambling code is exclusive-or-ed with the data file, and two same exclusive-or's result in the original value being returned.

Command format:

scramble filename.type password

where password is any 8 character string which is allowable as a file name (i.e. no ~, ., etc).

```

; C/M equates
0009         print    equ     9
000F         open    equ     15
0010         close   equ     16
0014         read    equ     20
0015         write   equ     21
0015         bdos   equ     0005h
0005         fcb    equ     003ch
003C         fcb2   equ     006ch
0068         fcboxt equ     fcbl+12
007C         fcbrno equ
000D         cr     equ     0dh
000A         lf     equ     0ah
0000         mf     defl   0      ` ; Show move not requested
0000         cf     defl   0      ` ; Show comp not requested
0000         mf     defl   0
0000         cf     defl   0

; Define some macros to make things easier
; Define data move macro: move from,to,length
; from may be addr, or quoted string
move        macro ?f,?t,?l
if          not nul ?f
?rpc       ?c,?f
defl       -&?c&?c`-
?q          extm

```

SCRAMBLE M-80 14 Oct 1984 20:45 PAGE 1-1  
Program to scramble CP/M files using an 8-bit password.

```

if    ?q eq `~~`                                ; Init. local stack
      local ?b,?z
      call ?z
      ?b: defb
      ?f: pop hl
      ?z: pop bc,?z-?b ; Get from
           ld ?t ; Get len
      else
           ld hl,?f
      endif
      endif
      if not nul ?t
           ld de,?t
      endif
      if not nul ?l
           ld bc,?l
      endif
      endif
      call mover -1 ; Show expansion
      mF
      defl
      endm

; Define CP/M macro - cpm fnc,parm
cpm macro ?f,?p
      bc
      push de
      push hl
      if not nul ?f
           ld c,?f
      endif
      if not nul ?p
           ld de,?p
      endif
      call bdos
      pop hl
      pop de
      pop bc
      endm
      0000`                                     ; Start of program execution
      call start
      defb cr,lf,`SCRAMBLE as of `

0100  CD 0122                                     ; Start
0103  0D 0A 53 43
0107  52 41 4D 42
010B  4C 45 20 61
010F  73 20 6F 66
0113  20
0114  31 33 2F
0117  31 30 2F
011A  38 34 2E
011D  0D 0A 0D 0A
0121  24

0122  D1
0123  0E 09
0125  CD 0005

; Start: pop de ; Get id
        ld c,print
        bdos
        call
; Print id

0128  21 0000 ; Scramble a while to mix up the seed
012B  39 add h1,0
012C  22 0407 add (stack),hl
012F  31 0407 add sp,stack

0132  26 00 ; See that the password is 8 characters
0134  CD 02F8 mixup: call pseuan
0137  25 dec h
0138  C2 0134 nz,mixup jp
; More? exit
; Loop if so

013B  3A 0074 ; Get 256 F's
013E  FE 20
0140  C2 0162
0143  CD 03A8
0146  2B 2B 20 50
014A  61 73 73 77
014B  6F 72 64 20
0152  6E 6F 74 20
0156  38 20 62 79
015A  74 65 73 2E
015E  20 2B 2B 24

0162  21 0060 ; Save the password
0162  11 03B7
0165  01 0008
0168  CD 039D
; is repeated more than 2 times
016B  21 03BF
0171  06 08
0173  0173 duptest: call ckdup
0176  23 inc inc
0177  05 dec dec
0178  C2 0173
; Password is 8 bytes, now make sure no character
; is repeated more than 2 times
017B  C5
017C  D5
017D  E5
017E  0E 0F
0180  11 005C
0183  CD 0005
0186  E1
0187  D1
0188  C1
0189  3C
018A  C2 01A4
; See that the input file exists
017B  + open,fcb
017C  + push bc
017D  + push de
017E  + push hl
017F  + open c,open
0180  + 1d de,fcb
0183  + call bdos
0186  + pop hl
0187  + pop de
0188  + pop bc
0189  + inc a
018A  + inc nz,scramlp
; Ok?
; Yes, scramble the file

```



```

SCRAMBLE M-80 14 Oct 1984 20:45 PAGE 1-5
Program to scramble CP/M files using an 8-bit password.

open,fcb ; Re-open
        cpm
        push bc
        push de
        push hl
        + + +
        ld c,open
        ld de,fcb
        call bdos
        pop hl
        pop de
        pop bc
        inc a
        nz,open2ok
        jp exit
        call crlf
        '+'+ Re-opening extent failed -',cr,lf
        defb

026F      C5      + + +
0270      D5      + + +
0271      E5      + + +
0272      0E 0F    + + +
0274      11 005C  + + +
0277      CD 0005  + + +
027A      E1      + + +
027B      D1      + + +
027C      C1      + + +
027D      3C      + + +
027E      C2      + + +
0281      CD 0348  + + +
0284      2B 2B 20 52
0288      65 2B 6F 70
0289      65 6E 69 6E
0290      67 20 65 78
0294      74 65 6E 74
0298      20 66 61 69
029C      6C 65 64 20
02A0      2D 0D 0A
02A3      20 20 20 66
02A7      69 6C 65 20
02AB      69 73 20 63
02AF      6C 6F 62 62
02B3      65 72 65 64
02B7      20 2B 2B 24
02BB      3E 7F
02BD      32 007C
02C0      C9

; Get a pseudo-random 8 bit number using the password
; as a seed.
; For speed, this routine does no register
; pushes and pops, however hl isn't used.
; Get a pseudo-random 8 bit number using the password
; as a seed.
; Shift thru 8 bytes
; pseuar:
; pseuar: 02F8 0E 04    c,4      ; Grab every 4th pseu.
; pseuar: 02F8 0E 04    b,8      ; Shift thru 8 bytes
; pseuar: 02FA 06 08    de,password
; pseuar: 02FA 06 08    a          ; Clear initial carry
; pseuar: 02FC 11 03BF 02FF B7   or
; pseuar: 0300 1A      a,(de)   ; Get a char
; pseuar: 0300 1A      rra     ; Shift
; pseuar: 0301 1F      (de),a  ; (de),a
; pseuar: 0302 12      inc de  ; inc de
; pseuar: 0303 13      dec b   ; dec b
; pseuar: 0304 05      jp nz,pseupl ; Exclusive-or the last few bits into the first one
; pseuar: 0305 C2 0300 0308 1B      dec de  ; Back up to last
; pseuar: 0309 1F      rra     ; Shift a few more
; pseuar: 030A 1F      rra     ; Shift a few more
; pseuar: 030B EB      ex de,hl ; Isolate single bit
; pseuar: 030C AE      xor (hl) ; Get first byte
; pseuar: 030D OF      rra     ; or in the bit
; pseuar: 030E E6 80    and 80h   ; Move it back
; pseuar: 0310 21 03BF 0313 B6    1d (hl),a ; Restore hl
; pseuar: 0311 0D      ex de,hl ; Loop if more passes
; pseuar: 0312 77    c      nz,pseup0 ; Loop if more passes
; pseuar: 0313 C9      ret

; Routine to check for duplicate chars in password
ckdup: 0E 03    c,3      ; Dup char counter
        031B 0E 03    c,3      ; Dup char counter
        031D 11 03BF 0320 3E 08    a,8      ; Char count
        0322 0E 04    af      ; Save count
        0323 1A      a,(de)  ; Get char
        0324 BE      (hl)    ; Dup?
        0325 C2 0396 0328 0D      nz,c,kndup ; Count dups
        0329 C2 0396 032C 32 0381 0342 6D 61 79 20
        0330 CD 0348 0342 6D 61 79 20
        0331 C2 0396 0346 61 70 70 65
        0332 CD 0348 0346 61 72 20 6D
        0333 CD 0348 0346 61 72 61 63
        0334 CD 0348 0346 61 74 65 72 20
        0335 CD 0348 0346 61 74 68 61 6E
        0336 CD 0348 0346 61 74 77 69

; Write back the sector
wrsect: cpm
        write,fcb
        + + +
        push bc
        push de
        push hl
        + + +
        ld c,write
        0E 15    + + +
        11 005C  + + +
        02C6 10 005C  + + +
        02C9 CD 0005  + + +
        02CC E1      + + +
        02CD D1      + + +
        02CE C1      + + +
        02CF B7      + + +
        02D0 C8      + + +
        02D1 CD 0348  + + +
        02D4 2B 2B 20 57
        02D8 72 69 74 65
        02DC 20 65 72 72
        02EO 6F 72 20 20
        02E4 20 66 69 6C
        02E8 65 20 63 6C
        02D9 6F 62 62 65
        02EC 72 65 64 2E
        02F4 20 2B 2B 24

; Write error - file clobbered. +-+
defb

; than twice in the password.",cr,lf

```

**SCRAMBLE** M-80 14 Oct 1984 20:45 PAGE 1-7  
Program to scramble CP/M files using an 8-bit password.

**SCRAMBLE** M-80 14 Oct 1984 20:45 PAGE 5  
Program to scramble CP/M files using an 8-bit password.

**The 80-BUS 800 Series****by P. A. Greenhalgh**

In these hallowed pages in the past you must have come across various 800 series numbers (e.g. "I use a GM802 plus EV814.....") and wondered what the writer was referring to. Well, fear no longer. This is the first part of an 'article' describing what physical products these magic incantations refer to.

So what is an '800 Series' product? Well, the 800 series was started by Gemini Microcomputers some considerable time ago, and consecutive numbers were given to each product as it was developed. An 800 number is allocated by Gemini only to major 80-BUS related items, and this therefore includes all of their 80-BUS boards, power supplies, keyboards, assembled motherboards etc. All of these Gemini numbers are given a 'GM' prefix. Gemini also decided to include other 'preferred' products from other manufacturers, and therefore the 'EV' referred to above means that the product comes from EV Computing. Similarly Belectra, Climax Computers, IO Research, and Microcode Processes have all had 800 series products.

So what is special about 800 series products? Well basically these are all built and tested products (there are one or two exceptions, and these have a 'K' suffix, meaning 'kit'), any 800 product should run with any other 800 product, and software support is available for most permutations. This is a somewhat over-simplified view, as, because of the exceedingly high number of permutations, there will inevitably be some exceptions, but for example to the question "Will a GM813+GM832+GM829+GM888+GM833+I0828+I0830+EV814+GM862+GM825+GM835+GM816+GM818+MP840 system run, and is there software for this?" the answer would be "Yes, if you can afford it, and if you have Battersea Power Station at your disposal!!!!" However, if the question substituted that string of 800 numbers for just "GM813+MAP256+GM812+NASCOM-AVC" then the answer would be a resounding "NO", as the AVC and 812 are set to conflicting ports, the AVC memory has to overlay the MAP256, the GM813 and MAP256 have different uses of port OFEH, etc. Now the problem is not, at the end of the day, totally insoluble, but you would have to have a very good working knowledge of all of these products, and you are certainly going to face a severe challenge in finding any software that will run straight away, even if you overcome these 'slight' problems.

So the point is that if you have an 800 series based system then you will certainly find life much easier if you stick with 800 series add-ons, and hopefully the following list will give you a much greater insight into what these are..... (Please note that the following information is NOT guaranteed to be correct, although it hopefully is! Contact your dealer if you have a specific requirement.)

**GM801**

A complete 64K twin disk CP/M computer based on a single board designed by Gemini and sold to British Micro, who later released it as the 'Mimi'.

**GM802**

80-BUS 64K dynamic RAM board from Gemini incorporating Nascom style 'page-mode' to allow up to four of these boards to be used in a system. Also available as a kit in 16K form and minus page-mode circuitry. Does not support the 'Extended Addressing' mode later introduced by Gemini (see GM813).

GM803

80-BUS EPROM/ROM board from Gemini taking up to 16 EPROMs, either 2708 (1Kx8) or 2716 (2Kx8) types. Also has socket for Nascom 8K BASIC ROM. Incorporates 'page-mode' but not 'Extended Addressing mode'. Was also available as a kit. Discontinued, to be superceeded by GM853.

GM804

A linear power supply for powering up to two 5.25" floppy disk drives. Used inside GM805, GM815 and GM825. Only ever sold as a kit. Discontinued (although still used by Gemini in the GM825).

GM805

A single/twin 5.25" disk drive unit incorporating Henelec single density disk controller board and GM804 PSU. Pertec FD250 double sided 48tpi drives used giving 160K bytes formatted per drive. For use with Nascoms, connected to the system via the Nascom's Z80 PIO. Discontinued.

GM806

A combined 5A power supply board, buffer board with 'Reset jump' circuitry, and 5 slot backplane for expanding Nascom 1 computers. Sold as kit only. Also sold in 'A' version without PSU components. Only 'A' version now available.

GM807

A 3A linear power supply unit suitable for powering small systems. Was also available as a kit. Now discontinued and replaced by various switch-mode units (see on).

GM808

An EPROM programmer for 2708 and 2716 EPROMs. Originally produced by Bits and PCs and later taken over by Gemini. Attaches to the system via a Z80 PIO. Supplied with software for use with Nascom and Nas-Sys. CP/M software for Gemini systems available on request. Only available as a kit.

GM809

80-BUS single/double density floppy disk drive controller board. Supplied suitable for 5.25" use only, but small modification possible to support 8" only. Superceeded by GM829.

GM810

80-BUS motherboard incorporating 8 80-BUS positions (and connectors) and 5A linear power supply. Discontinued.

GM811

80-BUS CPU board. 4MHz Z80A processor, Z80A PIO, 8250 UART providing programmable baud rates, stop bits, parity, etc., RS232 interface, 1200 baud CUTS cassette interface, 8-bit input port. Most powerful feature of this board is the four 'byte-wide' 28 pin sockets that may be populated with EPROMs, ROMs or static RAMs from 2Kx8 to 16Kx8 (32Kx8?) per socket, thus making the board suitable for many control applications with, for example, 48K of EPROM and 8K of RAM on board. On-board memory may be switched out of memory map under software control. Memory decoding is provided by PROM which must be replaced if the standard 4 by 4Kx8 decoding is unsuitable. Normally supplied with RP/M monitor that emulates CP/M in a cassette based environment, and can also 'boot' GM809/GM829 based floppy disk system.

GM812

80-BUS IVC (Intelligent Video Controller) board. Has own on-board 4MHz Z80A to control all video functions. Interfaces to the host computer via three 80-BUS I/O addresses, thus not encroaching on any host memory. Normal mode 80 characters per line by 25 lines, with optional second mode (supplied set to 48x25, but user alterable). System keyboard (parallel type) is normally connected via the IVC, which thus provides 'type-ahead' facility, and with certain keyboards also provides user-definable function keys (see GM827 and GM852). Light pen input provided. 256 different characters may be displayed, 128 fixed in EPROM and 128 user-programmable, being held in on-board RAM. Many control sequences include cursor addressing, partial screen scroll lock, pixel graphics (160x75), clear to end of line, clear to end of screen, define function keys, define programmable characters, insert or delete character from line, etc, etc. Now replaced by GM832 SVC.

GM813

80-BUS CPU/RAM board. Similar overall specification to GM811 above, but without 8-bit input port and with the 'byte-wide' sockets replaced by a 2K/4K EPROM socket (removable from memory map under software control) plus 64K of dynamic RAM. This RAM is switchable as Page 0 under the Nascom style 'page-mode' operation. This board also includes the Gemini 'Extended Addressing' mode by utilising memory mappers. This allows the user to select the 64K of memory that the Z80 'sees' at any time as any 16 blocks of 4K each, out of a total memory space of 512K. This 'Extended Addressing mode' may also be used in conjunction with 'page-mode' to provide up to 4 pages of up to 512K, i.e. 2MBytes total. However note that very specialised software would be required to do this. Normally supplied with RP/M (see GM811), but in complete systems (e.g. Gemini Galaxy) supplied with an auto-boot EPROM that detects whether Winchester hard disks or 5.25" floppy disks are present on a GM809/GM829 and automatically 'boots up'.

EV814

80-BUS IEEE 488 interface board from EV Computing for (surprise, surprise) connecting to other IEEE 488 devices. Incorporates on-board control software under the 'page-mode' scheme that can be brought in and out of the system memory map. Does not support 'Extended Addressing'.

GM815

Single/double 5.25" disk drive unit for use with GM809/GM829, incorporating 1 or 2 Pertec FD250 double sided 48tpi drives and GM804 PSU. Capacity, in double density mode, 350K (formatted) per drive. Discontinued - replaced by GM825.

GM816

80-BUS Multi-I/O board. Originally designed by Quantum, and then taken over by Gemini, this board provides 3 Z80A PIOs, a Z80 CTC, and a National Semiconductor 58174 Real-Time-Clock chip with battery back-up. An interesting feature of this board is the internal expansion bus - this allows additional boards to be 'piggy-backed' onto the GM816 and make use of the I/O decoding and buffering that it provides. Currently available 'piggy-backs' are the GM818 (see below) and the GM663 prototyping board.

GM817

An 85 Watt switch-mode power supply unit. Suitable for systems with about 5 boards and 2 5.25" drives.

GM818

A serial daughter board for the GM816 (see above) providing two 8250 UARTs with RS232 interfaces. These UARTs have programmable baud rates, stop bits, parity, stop bits etc, and also spare user-programmable input and output bits.

AM819

80-BUS Speech board, using the National Semiconductor Digitalker chip set. This card has a limited on-board vocabulary contained in ROM, and making it say anything outside of this limited library is extremely difficult. Produced by Arfon Microelectronics, who unfortunately went into receivership. Boards MAY still be available from one or two dealers.

AM820

Light pen, suitable for use with the GM812, GM832 and GM837 boards. Again, as AM819, this was an Arfon product, and so unfortunately it may prove impossible to obtain any more.

GM821

Original 59 key parallel interface keyboard from Gemini for use with the GM812/GM832 video boards. Although only 59 physical keys, all 128 possible 7-bit ASCII codes can be generated. Discontinued (see 827 and 852).

GM822

A small board containing a Nat. Semi. 58174 Real Time Clock chip with battery back-up. Connected to the system via a Z80 PIO. Only available in kit form.

GM823

Special version of the Gemini GM827 keyboard for the Danish market. Replaced by GM852 low profile version.

GM824

80-BUS A/D convertor board. Originally produced by IO Research, but later taken over by Gemini. This board provides 8 channels, each of 8 bit resolution. Input range is 0 - 5 volts with over-voltage protection. Conversion time is approx. 30 uS, including sample and hold phase. Support for vectored interupts, and on-board prototyping area.

GM825

A single/twin 5.25" disk drive unit using one or two Micropolis drives and GM804 PSU. For use with GM809/GM829. Available in single and double sided 96tpi versions. Capacity in double density mode is 400K single sided, 800K double sided (formatted).

MP826

80-BUS CMOS RAM board from Microcode Processes. Contains 32K of battery-backed static RAM to give 1000 hours of memory retention during power-down periods. Flexible address decoding and Page Mode operation, but does not support Extended Addressing.

GM827

Gemini's 87 key keyboard (parallel interface) for use with the GM812/GM832 video boards. The keyboard has a 57 key main cluster, plus 11 function keys, 4 cursor keys, and 15 key numeric pad. Theses latter 30 keys may ALL be user-defined (via the G812 or GM832 video boards) in both their normal and shifted modes. Replaced by GM852 low profile versions.

I0828

80-BUS 'Pluto' colour board from IO Research. The number of permutations of this board should be the subject of an article of its own! Started off with on-board 5MHz 8088 processor, 640x288 resolution, 8 colours, two screens. Then there were 640x576 options, 8MHz options, and the current product is normally 768x576, 8 colour, 8MHz 8088, although the 640x576 (640x288x2) version can also be ordered! Also see other IO 800 products. Any volunteers for a 'Pluto Family' article?

GM829

80-BUS floppy disk controller board. Fully compatible with the GM809 that it replaced, but with a previously unused bit of the control port being used to switch the board between 5.25" and 8" operation, allowing both types (and 3.5" drives that electrically 'look' like 5.25" drives) to be attached to the GM829 at the same time. In addition the board contains a SASI interface that is used for the connection of Winchester disk controller boards. For an example of the flexibility of this board look at the Gemini Galaxy M-F-B family, the top model of which has a 16 Mbyte Winchester, double-sided 96tpi 5.25" drive, double-sided 48tpi 5.25" drive, double-sided 8" drive, and double-sided 3.5" drive all connected via a GM829.

I0830

Mini-palette board from IO Research, available as an add-on for the I0828 'Pluto' (although the latter then needs a new monitor EPROM). Gives the user the ability to select any 8 screen colours out of a Pallette of 256. This can be particularly impressive if 8 grey levels are selected, the resulting picture almost looking as good as a B&W photograph. Two look-up tables are supported, and the user may switch rapidly between them.

GM831

5 slot 80-BUS backplane (with 1" pitch between boards) including 77-way 80-BUS edge connectors. Discontinued as a built product, but the PCB continues to be available.

GM832

80-BUS SVC (Super Video Controller) board. Fully compatible with the GM812 IVC that it replaces, but with the following enhancements. 6MHz Z80B CPU used, which in conjunction with other hardware and software changes provides far greater speed. Second text mode is now 40x25. Graphics mode added of 256x256, with on-board monitor containing line and circle drawing routines, polygon fill routines etc. Buzzer added. Attributes added - half-intensity characters, half-tone background, blinking. All 256 characters are now user-definable - on power up they are down-loaded from the on-board monitor, and this includes switch options for French, Danish, German, Swedish, American and English characters. Serial keyboard input option added - although this is of Gemini's own design (see GM852).

GM833

Gemini's 80-BUS RAM-DISK board. Provides 512K bytes of RAM driven via a 'disk-like' 80-BUS I/O port interface of 'track', 'sector' and 'data' ports. On-board switch allows up to 16 boards to be decoded at the same port addresses, theoretically providing up to 8MBytes!

GM834

Special version of the Gemini GM827 keyboard for the German market. Replaced by GM852 low profile version.

GM835

5.25" Winchester hard disk drive sub-system. Contains a Rodime drive, along with hard-disk controller board and switch-mode power supply. Currently available in 5.4, 10.8 and 16.2MByte (formatted) capacities. Connects to SASI interface on GM829 board.

GM836

MultiNet interface board. This small board is connected to the system via a Z80A PIO. An on-board DIL switch provides a unique station address to allow up to 32 systems to be connected together. The board transmits data onto the Network at 250 kbaud, at distances up to 600 metres using RS422 differential transmission.

GM837

80-BUS Colour board. Previously produced by Climax Computers, production of this board has now been taken over by Gemini. The board uses the Thomson EF9365 controller chip and provides high speed vector graphics of 256 x 256 in 16 colours. The chip also has a built-in character generator, and can write text in various orientations. Output is PAL UHF and analogue RGB TTL.

GM838

Special version of the Gemini GM827 keyboard for the French market. Replaced by GM852 low profile version.

GM839

80-BUS prototyping board. One of the great attractions of 80-BUS is the ease with which one-off boards can be interfaced to it, and this board provides an ideal base for one of these, incorporating extensive power rail connections, and laid out to allow high density IC packing.

MP840

14 slot 80-BUS backplane from Microcode. NOT supplied with any 80-BUS 77-way edge connectors. All active BUS signals are terminated into a potential balanced RC filter and are interleaved with ground shield tracks. The board can be easily 'cut-down' if required for smaller systems.

GM841

This 80-BUS extender board allows 80-BUS boards to be brought outside of a frame for testing and debugging purposes. Test point pins are provided on all lines, and all lines have their 80-BUS nomenclature.

Well, that's enough for this time. Next issue I hope to continue through to GM888, and possibly beyond !! I hope that the above descriptions bring out a few points that you were possibly unaware of before, and thus help you to find your way through the growing maze of products. 80-BUS provides a very flexible means of producing so many possible self-tailored systems, and if the information above in some way helps just one reader from making a potentially costly incorrect decision then it has been justified.

---

**AN IMPROVED NAS-SYS?**

Here are some suggested modifications to NAS-SYS 3. NAS-SYS 3, while an excellent piece of software, has a couple of shortcomings from my point of view. In particular:

- It does not support a parallel printer.
- Access to the ASCII characters "{", "|", and "}" is awkward.

The latter point is perhaps of most concern to Scandinavians, as these characters are redefined as special Scandinavian letters. To ease word processing, the requirements are that the small "letters" { | } shall correspond to the capital "letters" [ \ ], respectively, each pair of "letters" assigned to its own key on the keyboard.

Both these shortcomings are corrected by these modifications to NAS-SYS 3. The requirements are:

- NASCOM 2 keyboard (57 keys)
- Centronics standard printer connector
- PIO port A (8 lines) to printer DATA lines
- PIO line B0 to printer STROBE line
- PIO line B2 to printer BUSY line
- PIO connector ground to printer LOGIC GROUND

The PIO is initialized during power-up, and reset to the printer requirements; this must be taken into account if the PIO is to be used for other purposes.

One problem with NASCOM standard software is differences in supplying LF or not after each CR, this gives problems when using printers. On EPSON printers this is easily fixed, as the CR is not needed, only LF may be used to separate lines. Consequently, this NAS-SYS version converts all CRs to LFs, and ignores all LF codes sent from some software packages. If your printer insists on CRs, it should be possible to convert all CRs to a CR and a LF, still ignoring incoming LFs.

To make room for the extra facilities inside the allotted NAS-SYS space, it was necessary to remove some existing code. I chose to attack the register display routine, shortening it so that now it will only display the contents of the Program Counter. After all, if you have DEBUG, you never use it anyway.

**Description of new NAS-SYS facilities****a) Keyboard changes**

```
{ and [ are on the "[\" key
| and \ are on the "CH/LF" key
} and ] are on the "]_" key
@ is now unshifted, repeating
_ is Shift/@
```

Remarks: The @ key is no longer available as an extra CTRL key (this facility is only needed on NASCOM-1 keyboards). CH and LF are not very important control codes, consequently they do not need their own key. CH is still

**by Olav Lerflaten**

available as CTRL/W, and LF is available as CTRL/J. Whether [ \ ] and { | } are shifted or unshifted depends on the Keyboard option. With these changes, the full ASCII character set is still available from the keyboard.

#### b) Parallel printer support

Upon power-up or reset, the PIO is initialized. Printer output is activated by the U command, and deactivated by the N command. This is made possible by a change in the workspace initialisation, \$UOUT will initially point to the printer driver routine. Initial contents of \$UOUT: C3 CB 04

#### c) New NAS-SYS commands

Two new NAS-SYS commands are provided:

##### L PRINT

The contents of the accumulator are sent to the printer, using the printer driver routine. Please note that this command is independent of the N and U commands, working always. If the accumulator contains code f0A, data will not be sent to printer, code f0D will be sent to the printer as f0A. The printer must be on line, or the routine will make the Nascom hang up. This command is best used as a subroutine call in programs:

fDF f4C PRINT Sends contents of A to printer.

##### F aa bb cc dd ....

##### SEND ARGUMENTS TO PRINTER

Up to ten hexadecimal arguments following the command are converted to 8-bit data and sent to the printer. If arguments >fFF are specified, only the least significant byte is sent. Please note that this command is independent of the N and U commands, working always. The printer must be on line, or the command will make the Nascom hang up. As this command uses the printer driver routine, arguments of f0A will be ignored, and arguments of f0D will be converted to f0A. This command is very useful for sending control codes to the printer, for example to activate or deactivate emphasized or enlarged print mode.

#### Using modified NAS-SYS and printer with standard Nascom firmware

##### a) NAS-SYS

Printer on - U command  
Printer off - N command

The user is in direct communication with the printer, that is, everything that is typed on the keyboard will be sent to the printer. Line feeds are ignored. Carriage returns will be sent as line feeds.

##### b) NASPEN

For the P command to function, the printer reflection must be changed:  
101D DF 4C C9 SCAL "L RET

##### c) DEBUG

DEBUG uses the NAS-SYS \$UOUT jump, and therefore turns off the printer when started. Use these DEBUG commands to control the printer:

Printer on - :C 04CB  
Printer off - :C C143

; NAS-SYS 3 modifications  
; by Olav Lerflaten.  
;  
; Bergen, Norway  
; Nov.27, 1983  
;

ORG E0005 ;Link PIO init routine  
JP PRINIT  
ORG E00B6 ;Change kbd mask (@ repeating)  
LD D,EE7  
ORG E012F ;Support lower case [ \ ]  
CP "]+1  
ORG E0145 ;@ as CTRL key not supported  
JR E0155  
;  
;Rest of PRARGS code  
CONTUE RET Z  
LD B,A  
LD HL,E0C0C  
NXTARG LD A,(HL)  
SCAL "L  
INC HL  
INC HL  
D,NZ,NXTARG  
RET  
DEFB 0  
ORG E0183 ;Workspace init \$I\$OUT  
JP DRIVER  
ORG E0408 ;New commercial  
DFRM /\* NAS-SYS 3.1 \*/  
ORG E04A9 ;Shortened PREGS routine  
PREGS RST E28 ;PRS  
DEFB F18 ;CCR  
DEFB 0 /PC=/  
LD HL,(E0C69) ;HL=(PC) before brkpt  
SCAL E66 ;TBCD3  
SCAL E6A ;CRLF  
RET ;End of PREGS routine  
;  
;PIO initialization routine  
PRINT LD A,FF  
OUT (6),A  
XOR A  
OUT (6),A  
DEC A

46

;Printer driver routine  
DRIVER PUSH AF  
BUSY IN A,(5) ;Save AF  
AND E04 ;Line B2 high if busy  
JR NZ,BUSY ;Mask  
POP AF ;Try again if busy  
PUSH AF ;Data restored in A  
CP EA ;Is it LF?  
JR Z,FIN ;If so, ignore  
CP E0D ;Mask  
JR NZ,NOCHG ;Is it CR?  
SUB 3 ;If not so, don't modify  
NOCHG OUT (4),A ;Make a LF from the CR  
XOR A ;Data to printer  
OUT (5),A ;Save AF  
INC A ;Strobe low  
OUT (5),A ;ARGN works loc  
OR A ;Set Z flag if no arguments  
JP CONTUE ;Continue elsewhere  
;  
;Start of NAS-SYS F command  
PRARES LD A,(E0C0B) ;ARGN works loc  
OR A ;Set Z flag if no arguments  
POP AF ;Restore AF before return  
RET ;End of printer driver  
;  
;Changes to the keyboard table  
ORG E05C4 ;Changes to the keyboard table  
DEFB FF  
ORG E0D1 ;No CH on CH key  
DEFB FF  
ORG E0616 ;CH/LF key gives "\n" and "|"  
DEFB FE  
ORG E0619 ;Shift/@ gives "\_" char  
DEFB FD  
;  
;Changes in subroutine table  
ORG E08C  
DEFW E04E8 ;F command, args to printer  
ORG E0798 ;L command, printer driver  
DEFW E04CB

## d) NAS-DIS

Use in Direct mode (NAS-SYS U command).

As NAS-DIS asks for memory addresses, unpredictable things may happen to the printer. This is because the subroutine outputs an ESC code, which the printer takes as the start of a control code. The cure:

- Either turn the printer momentarily off when NAS-DIS signals "Go?"
- Or change these memory locations in NAS-DIS:

CD00	00	NOP
CD01	00	NOP
CD02	00	NOP

## e) ZEAP

Change printer reflection:

```
OF04 DF 4C C9 SCAL "L RET
```

ZEAP U command will now list the source file, and ZEAP A command will, with option 04, give assembly listing.

## f) BASIC

Use in direct mode (NAS-SYS U command). It is possible to turn printing on and off from BASIC with DOKE statements:

Printer on - DOKE 3187,1912

Printer off - DOKE 3187,1913

---

**Private Sales**

Nascom 2 in Kenilworth case with fan; 40K RAM, I/O board with UART & 1 PIO. Cassette deck and display monitor. Nas-Sys 3, BASIC toolkit in ROM. Hullforth, Wordease, V&T Assembler and games on tape. Manuals and large number of Nascom related magazines. £200 ono, Tel: 029668-651 Bucks.

Nascom 2 in Vero case, with Nas-Sys 3, toolkit, NasPen, ZEAP, Debug, Nas-Dis all in ROM. RAM B with 32K. Holmes/R&EW Colour card with real time clock, 8 port A-D, CMOS RAM, CTC and sound generators. Offers around £275. R.Dickenson, Otley (0943) 464147.

Nascom 2 with 64K RAM (GM802), spare RAM A board (32K), Kenilworth case, GM805 disk system with CP/M, 3 assemblers, 6 languages (including C and Fortran), Soundbox, GM822 real time clock, Sargon Chess and many other games, Nascom I/O board (2 x PIO + CTC), IMP printer and paper. £475 or offers. Call Julian on 01-634-7658 between 10 a.m. and 5 p.m.

64K Gemini MultiBoard with GM812 IVC, GM809 FDC, GM802 RAM and GM811 CPU cards. All in aluminium case with PSU, fan and 5 slot motherboard. GM827 keyboard in sloping case. GM825 dual disk drive (800K each). Reasonable offers for the lot (or might haggle over individual units). Dr P.D.Coker, 23 Darwin Close, Farnborough, Orpington, Kent BR6 7EP. (0689 58510)

Nascom 2 with Graphics ROM, ZEAP(2.1), Castle cassette interface, mini-motherboard, Nas-Sys 3. 32K RAM A board. Nascom 3A power supply + digital filter. Gemini GM808 EPROM programmer plus software. Winchester Technology WT910 Nasbus sound board with an A-D convertor, onboard speaker, plus audio/video outputs. Complete documentation for the above plus a few games tapes, leads to cassette etc + serial and parallel port leads. £299 ono for quick sale. Dr M.R. Goulding Towcester (0327) 52476 eve, 50581 ext. 366 day.

---

**Random Rumours (and Truths?)****by S. Monger**

Let's to report this time. For example, IO Research...Pluto 2 is here!! Yes, fed up with reading in this column about how little they do, IO Research have produced a super-duper new Pluto. There is the inevitable catch, the unreal price, but let's forget that while I lay down the mouth-watering fax:

"Pluto II is a brand new powerful single board graphics display system which has been designed to be completely hardware and software compatible with the original Pluto board. [Pluto I + £50.] Re-designed to use up-to-date technology [add £200] the board replaces 3 boards from the original Pluto range [add £1000] with one compact unit giving high reliability [+£150] and ease of use [+£80]. Based upon the highly successful Pluto concept [+£25], Pluto II's more advanced [+£50] video circuitry produces truly advanced colour graphics [+£100]. The board format is a compact 12" by 8" [-£500], multi-layer, 80-BUS compatible [+£300]. The processor is an 8MHz 8088 [+£0]. Half a megabyte of memory as standard [+£50] etc, etc, etc."

Anyway, the press release goes on to say that Pluto II is basically Pluto I plus Pluto Pallete (256 colours from 16.7 million), hardware pan and zoom, fast text scrolling and smooth shading (what's that?), and optional 50Hz frame grabber. The final figure?? Well over £2000 (two thousand pounds). Oh well, I didn't really want one anyway as I can't fit a 12"x8" board in my system!

Whilst on the graphics topic it may be worth mentioning that Gemini have implemented GSX on Pluto. (Why not on their own GM837 board???) GSX is supposed to do for graphics what CP/M did for disk drives. When I've found out what that means I'll let you know.

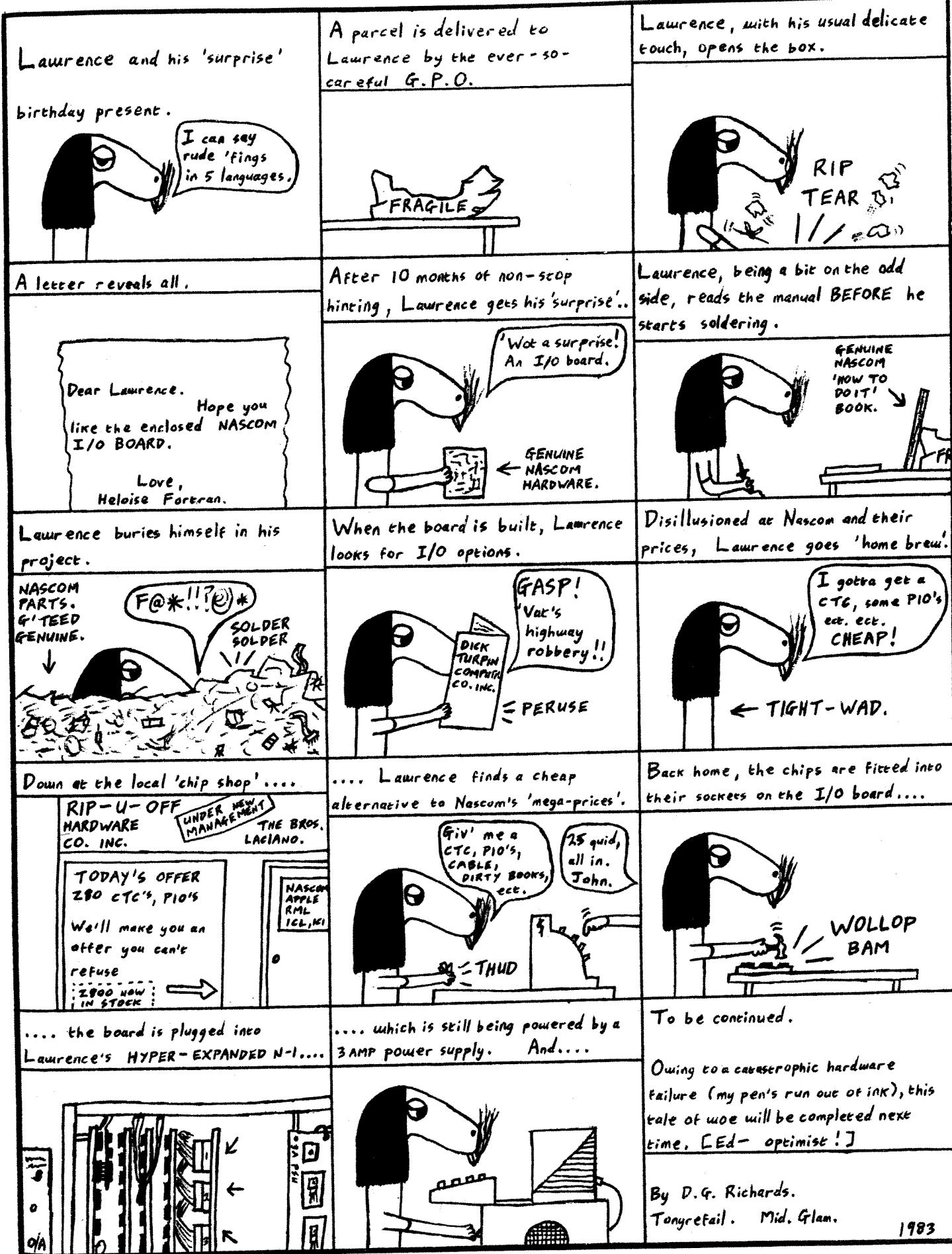
And what else has appeared? Yes, Gemini have been reading my bit as well as IO, as they have at last released the almost forgotten about GM848 Multi-Serial Board. This board has two Z80 SIOs, each providing 2 serial channels, and each direction of each channel of each device (phew) can be set to a different baud rate (must be some use). There is also a PIO thrown on the board for good measure, and the board occupies a switch-selectable block of 16 I/O ports. The SIOs of course cater with both synchronous and asynchronous protocols. Now where did I leave that mainframe? The price? £125 + VAT to you. Big 'G' are also shipping their new whizzo EPROM basher with some success, although I'm not quite sure where I'd put the 27256 that I could then program. (No, that is not a request for suggestions, thank you all the same.)

In my last bit (80-BUS Vol 3, Iss 1) I didn't have the price of the GM888, 8Mhz 8088, 8"x8", 8oz. board. that is now being delivered. £888? No, £190 actually, and CP/M-86? £175. (All plus VAT.) The CP/M-86 is currently for Gemini based systems only - will a Nascom version follow?

EV Computing have 're-vamped' their EV814 IEEE 488 board. I'm uncertain as to what 're-vamped' means, but it gives them reason to also 're-vamp' (i.e. raise) the price to, I think, £195 + VAT.

Lucas are apparently (and totally contrary to popular belief) still alive. It seems that from time to time the odd (very odd) Nascom 2 or 3 manages to escape from Warwick. But where oh where are there any new products? Or how about some advertising to show that they still exist??

And finally back to Gemini, the only great white hope where other manufacturers seem to have become stale in their lack of new products. Is the fact that they are a signed-up 'EasyLink' (some sort of computer telephone/telex/electronic mail system) customer an indicator that we may see an approved modem from them soon? Or is that a forelorn hope? And do the GM853 and GM863 boards described in their 'New Products' data sheet exist as much as/the same as/more than (delete as appropriate) the GM848 did a year ago?



## CMOS RAM BOARD RB32KC

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

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

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

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

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

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

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

**FEATURES** HARDWARE AND/OR SOFTWARE READ/WRITE PROTECTION.  
FULLY 80-BUS AND NASBUS COMPATIBLE.

## BACKPLANE BP14C

### BP14C 14 SLOT 80-BUS & NASBUS TERMINATED BACKPLANE

(Bare Board Weighs 15 oz!)

- \* Ground plane on one side of double sided 2.4mm thick printed circuit board.
  - \* All active BUS signals interlaced with ground 'shield' tracks.
  - \* All active BUS signals terminated into a potential balanced R.C. filter.
  - \* Proper implementation of both the interrupt and BUS request daisy chains.
  - \* Large tracks for power lines.
  - \* Size 15" x 8". Fits neatly into 19" rack.
  - \* Easily cut for smaller systems.
- Supplied built and tested without connectors.

#### PRICES:

RB32KC 32K Bytes £225.00	77 way connectors for uses with BP14C £4.30
16K Bytes £193.00	Few left - Visually imperfect BP14C £47.00
2K Bytes £165.00	Few left - RB32KC Bare Boards Rev. D £62.50
BP14C Backplane £57.00	Memory - Hitachi HM6116LP-3 £5.75

Plus £1.50 per board post and packing, plus VAT.



### MICROCODE (CONTROL) LTD.

40 WELL TERR., CLITHEROE,  
LANCS, U.K. 0200 27890

### GEC LTU-11 PRETEL MODEM KIT

The GEC LTU-11 PRETEL modem kit consists of two components, the modem card and the direct telephone coupler. The modem card accepts I/O at TTL levels at 75 BAUD and returns data at TTL levels at 1200 BAUD. The modem card produces the necessary modem tones which are fed to the direct coupler. The coupler consists of the line isolation transformer which sends and receives the modem tones and has TTL level inputs for the control relays for line seize and auto-dial. Power supplies are +5 volts and -5 volts. The units individually have BT approval, but as the unit is supplied as a kit, the BT approval is void. Supplied complete with connection data and auto-dial software.

GEC LTU-11 modem card 14.95 inc. VAT. (13.00 + VAT).  
GEC LTU-11 direct coupler 14.95 inc. VAT. (13.00 + VAT).  
Carriage & packing 75p for each unit.

## STC NOVATEL PRETEL TERMINAL

#### Features

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

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



Fully tested as new and guaranteed  
**£129.95** Inc. VAT  
(£113.00+VAT) (UK C/P + ins. £3.55)

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

### TRANSDATA MODEM CARDS.

Brand new, tested and aligned 300 BAUD answer and originate acoustic telephone modem cards manufactured by TRANSDATA for their model 317 acoustic modem. Full CCITT specs. Requires only a few components to complete (no case available). Computer interface is 300 BAUD RS232 I/O. The power supplies of +12 volts at 180mA and -12 volts at 170mA are required. Card size 350mm x 105mm. May be directly connected using the GEC LTU 11 coupler (see below). Full circuit description, drawings and connection data supplied. Suitable for use with computers fitted with 300 BAUD RS232 I/O. Software for Nascom & Gemini published in the last issue. Card only at 29.95 inc. VAT. (26.04 + VAT) Supplied with LEDs connector, switches and transducers at 34.95 inc. VAT. (30.39 + VAT) Supplied with LEDs connector, switches and GEC LTU 11 direct coupler at 45.95 inc. VAT. (39.96 + VAT) Carriage & packing 1.00

### MINOR MIRACLES 2000 MODEM

The Minor Miracles 2000 direct connect answer and originate modem is an all 'bells and whistles' modem capable of Bell and CCITT protocols at 300/300, 600/75, 600/600, 1200/75 and 1200/1200 BAUD receive/transmit with optional auto-dial and auto-answer. Automatic mode select under software control. Requires RS232 I/O for data transmission and an optional 4-bit port for mode control. Uses the AMD 7910 'world series' modem control chip. BT approved (until you remove the link to activate the Bell protocols). Separate 220-240V AC mains powered. Size approx. 140mm x 70mm x 180mm. Supplied complete with lead terminated in BT620 (the new rectangular type) plug, manuals, etc.

Modem 148.35 inc. VAT. (129.00 + VAT).

Auto-dial option 34.50 inc. VAT. (30.00 + VAT).

Carriage & packing 1.00

### TELEMOD 2 MODEM

The Telemod 2 direct connect answer and originate modem uses the European CCITT protocols and offers 1200/75 and 1200/1200 BAUD receive/transmit. Requires RS232 I/O for data transmission. Separate 220-240V AC mains powered. Supplied with lead terminated with BT620 plug and manuals. Size 165mm x 55mm x 235mm.

Modem 99.95 inc. VAT. (86.90 + VAT).

Carriage & packing 1.00

<b>P R E S T E L T E L</b>	<b>ON A GEMINI IVC OR SVC</b> The new Henelec Prestel Terminal software turns your CP/M based Nascom or Gemini, fitted with the GEMINI GM812 or GM832, into a full blown PRETEL terminal. Automatic character sets give proper prestel characters and mosaics. Automatic select of 40 by 24 screen size, full prestel screen wraparound. Message facilities allowing full keyboard entry to the bill boards. Downloading of prestel pages to disk. Optional auto-dial (with suitable modem) and ID transmit. Suitable for most 1200/75 BAUD modems, but designed around the Minor Miracles 2000, the Telemod 2 and the GEC LTU-11. Try cracking the PRETEL access codes for yourself!!!
--	--

Supplied on disk (please state format) at:  
29.95 inc. VAT. (26.04 + VAT). Carriage & packing 50p

### BIG BIG BIG POWER SUPPLIES

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

Type 1 +5 volts at 20 amps. 7" x 4" x 3.5" o.a.

Switch mode. 220-240 V AC in.

25.95 inc. VAT. (22.56 + VAT)

Carriage & Packing 2.00

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

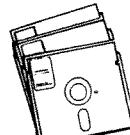
-5 volts at 1 amp, -12 volts at 1 amp

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

Thyristor switch mode. 220-240 V AC in

29.95 inc. VAT. (26.04 + VAT)

Carriage & packing 4.00



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

**HENRY'S**

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

OPEN 6 DAYS A WEEK

Credit Sales available ask for details.

Official orders welcome.



Order by Post with CHIEQUES-ACCESS/VISA or you can telephone your orders.

