October-December 1988 Volume 2. Issue 4.

# Scorpio News

Scorpio Systems
P.O. Box 286 · Aylesbury · Bucks · HP22 6PU

# Contents





# Editorial

Yes, we are a little late with this issue. There are a variety of reasons/excuses for this, but I won't bore you with the gruesome details. The main thing that I have to say is that all good things have to come to an end! I am of course referring to this illustrious publication. We've been going for two years now and two years on, the Nasbus/80-BUS market has unfortunately slid further and further into obscurity. For the number of people that we feel are likely to resubscribe for 1989, we cannot possibly justify the continuation of the mag. Material is as scarce as rocking horse manure too; since the last issue all we have received is one article and one letter.

However, I feel that it would be quite unfair to just come to a halt without any prior warning, as there are quite a number of you "out there" who are still exceedingly interested in 80-BUS. Therefore we will produce one last publication, in April/May '89. If you wish to receive this please send £5, and if you ALSO have a message that you wish to have included (i.e. "I'm particularly interested in ...., please contact .... if you are too."), or perhaps just your address, then please let us have this information with your cheque/postal order. We will also, as usual, be accepting ads – rates on application. Money/letters/ads/etc in by February 28th please. Bye. ●



# Letters to the Editor

## Overlays & Chained Files

Dear Sir

I am writing in response to the article by R. Pearce in Volume 2, Issue 3 of *Scorpio News* concerning the use of overlay and chained files.

He may like to know that at least two compilers support overlay files: Borland's Turbo Pascal and the MIX C Compiler. I have yet to use this facility with the MIX C compiler, and so cannot comment on its use, but I have used the Borland product extensively and found it entirely suitable for producing large programs of commercial quality. It also allows for the creation of chained files which are called by the procedure Chain(-.-).

The transfer of global data is also possible by reserving space between the end of the Pascal run-time library and the start of the chain file.

The creation and use of overlay files is even easier. Overlay files are produced automatically by the compiler simply by prefixing the declaration of any procedure or function with the word "overlay". Consecutive overlay sub-programs are placed in the same overlay file. Any intervening non-overlay declaration starts a new file for subsequent overlay sub- programs.

As Mr Pearce implies in his article, overlay files are the most useful method in general, but chaining has the advantage when the application consists of a series of stages which never, or rarely, return to an earlier stage.

A more important point, perhaps, is the choice of an appropriate language and compiler, with all the facilities likely to be required, before one begins writing the program. Programming ought to proceed with as few delays or resorts to machine code as possible.

Unfortunately this goal can usually only be achieved when you have access to a reasonably large number of compilers. However, from my experience I would recommend Turbo Pascal as a suitable language for writing serious applications.

Yours sincerely, Kevin R. Smith. ●

# Review of the NE898 processor board

by Peter Bell

"Review of the what board?" I hear you ask. A year ago, in Volume 1, Issue 4 of *Scorpio News*, you may remember seeing reports of two new 80-BUS processors, both with Gemini part numbers - the GM880, based on the Hitachi 64180, and the GM890, using the Zilog Z280. Both of these boards appealed to me since my interest lies in developing the operating system software – just think what could be done with the increased memory, DMA and more interrupt facilities.

It didn't take long to decide that the Z280 based board, although possibly less 'compatible', would provide the greater challenge with its dual supervisor/user modes, separate instruction and data space and exception trapping. I rang Gemini to enquire about availability. After a pause the receptionist said "There's no one here at the moment. I'll get someone to call you back." My call still hasn't been returned, eleven months later! By the next day I decided to phone Newburn, whose advert in the same magazine offered the GM890 at a special introductory price. 'Two weeks' the man said. It seemed that the GM890 development was being carried out as a joint exercise between Newburn and Gemini, with Newburn primarily responsible for the hardware.

Being an impetuous person I sent an order with a deposit cheque. Over the weeks that passed I had several conversations with the people at Newburn. They were always willing to spend time detailing the various problems they were experiencing in the development of the GM890, but delivery was always 'a few weeks away'.

In June this year I received a letter advising that development of the GM890 had been discontinued, but that an alternative board, the NE898, was almost ready, with a price tag of £439. This time they had opted for the 64180 chip, now also supplied by Zilog as the Z180. Apparently the Z280 is not fully compatible with Z80 peripherals using the mode 2 vectored interrupts, and this is what caused the development to be abandoned. Disappointed by this news I took a little time to decide to try the NE898. However, I have now been using this board for just one week, and am very impressed. Obviously this device is in almost direct competition with Gemini's own product. Anyway, let me list some of the features and facilities that the NE898 offers:-






## Features of the Z180 processor

- Z80 compatible CPU running at up to 8MHz clock
- Memory Mapping Unit giving access to a 1MByte address space
- Two vectored interrupt inputs in addition to the standard Z80 NMI and INT facilities
- Two DMA channels to provide fast memory to memory, and memory to/from I/O data transfers
- Two full duplex asynchronous serial communication interfaces
- One synchronous, or clocked, high speed serial interface
- Two programmable counter/timer channels
- Illegal opcode software interrupt (trap)
- Enhanced Z80 instruction set

## Features of the NE898 board

- Z180 with a link selectable 4 or 8MHz clock, giving access to all the above facilities
- GM813 compatibility in all respects except for the serial interfaces
- 80-BUS compatibility, running the bus at the full 8MHz
- 256Kbyte dynamic RAM (120 nanosecond)
- 32 Kbyte Monitor/Boot PROM
- Five additional byte-wide sockets each accepting 32Kbyte ROM or static RAM, with link selectable battery backup
- An SPCT chip offering, in one package, the following fully Z80 compatible 8MHz devices:
  - Z80 PIO - two parallel channels
  - Z80 CTC - four programmable counter/timer channels
  - Z80 SIO - two full duplex serial ports
- Battery backed 58274 real time clock chip
- P-Bus parallel interface to allow connection to the range of Newburn Industrial I/O modules
- LCD interface capable of driving standard alphanumeric displays
- Multi-processor facilities - two NE898s can operate in one backplane, sharing peripherals and the work-load
- Watch-dog timer to allow recovery or detection of system crash or hang-up situations

The decision to run the bus at 8MHz is a brave one. Although the possibility had been recognised in the 80-BUS specification by inclusion of the AUX CLK line, I believe this is the first time it has been done commercially. Newburn claim to have tested many of the existing boards with the NE898 at 8MHz, and have had no serious problems. The NE898 is also available without the eight DRAM chips at a price of £399. This option will appeal to those wishing to run the system on static RAM, or who might wish to populate the DRAM sockets themselves.

## What you get

- A 48 page manual.
- A floppy disk containing some test and benchmark programs
- Hard copy of the test program results
- The NE898 board

Monitor PROM source and circuit diagrams are available on request

## Documentation

I understand that the documentation is currently undergoing revision. What is currently being distributed gives sufficient information to operate the NE898, detailing all the link options, switches, connectors and the I/O port allocations. Also included are some example Pascal programs demonstrating the use of the DMA, LCD interface and both the CPU and SPCT asynchronous serial interfaces. Those people wishing to make use of the new facilities offered by the NE898 would do well to equip themselves with the 64180 User Manual, the technical documentation on the Z80 peripheral chips, and the application notes on the 58274 RTC chip.

## Visual

The board is very solidly constructed, in keeping with Newburn's Industrial systems market. It is quite densely packed, and all ICs are socketed. Most of the link options are provided with plug and jumper, some are multi-way DIL socket and solder headers. The outer edge of the board is lined with a series of four dil plugs, incorporating ejectors, - P-Bus, SPCT serial, parallel and CPU serial. According to the manual the PCB is six layer - one of the internal layers appears to be a total ground plane. A set of 20 uncommitted plated through holes are provided, ready to accept an additional IC. This will allow a limited amount of tidy customisation. There are a few minor 'fixes' on the board - the reset switch is mounted at a peculiar angle, two pull-up resistors are tacked onto the back of the board, as is one wire link. However the overall impression is one of quality. Indeed the board can be supplied with full certification of conformity of the parts used, for those customers who require evidence of Quality Control.

## In use

The board arrives ready configured as a plug in replacement for a GM813, set to 8MHz. Note that some existing boards will require a link setting to the AUX CLK position before operating in an 8MHz system. The author's system was Nascom 2 based, and so this was removed together with the GM802 64K RAM and GM803 EPROM cards. Thus the new configuration is NE898, GM829 FDC/SASI and GM832 SVC.






Cautiously the board was initially installed set to 4MHz. On power-up or reset the Monitor starts by displaying the text 'ABC' on the screen. If auto boot is selected, then both the first floppy and then the Winchester drive are polled in turn. It won't boot my winchester since I modified the SASI interface on the GM829 about a year ago in order to facilitate DMA transfers. However, booting a floppy 'sysgened' with the winnie on drive A works a treat.

If the monitor setting is selected then the Monitor announces its presence with the text 'Newburn Monitor', followed by a rather untidy menu of options on a wrapped round line. Perhaps this is attended to in the new version (1.1) of the monitor PROM, which I am told is already in the post. I gather that Newburn operate a free update service, for the first year after purchase. No problems were encountered in this first period of use, and the clock was soon re-set to 8MHz. The improvement in performance was immediately noticeable. Everything happens much more quickly and display of a full screen of text almost gives the impression of a memory mapped display!

One minor problem soon became apparent - with the floppy drive motors stopped disk access produces a 'Drive not ready' message. Immediately selecting the re-try option causes a resumption of service. The cause of this is the one second delay which the GM829 inserts after starting the drive motors, before allowing the drive ready bit to be asserted. Most of the existing 80-BUS BIOSs incorporate a software delay loop before forcing the time-out. The improved performance of the NE898 shortens this delay to the extent that it is shorter than the hardware delay. The author's BIOS source code was soon modified, assembled and the whole MOVCPM image re-linked. Users of proprietary BIOSs may not have the facilities to carry out this modification, so I have investigated an alternative solution in hardware. This appears to effect a cure for all the Gemini BIOSs. Newburn claim that no problems are encountered with the GM849, so presumably the same delay is not implemented on that board. Users of other FDC boards should beware of this possible problem.

Another problem was discovered when using the public domain utility D.COM. If the files were held in the directory in alphabetic sequence, then all was OK. However if a sort was required, then the disk access proceeded as usual but no output was produced before the next system prompt. After much investigation it was discovered that D.COM employs two undocumented Z80 opcodes in the sort. The Z180 detects these and merely sets an internal flag before trapping to address zero, hence causing a harmless warm boot. I intend to modify my OS to report detection of illegal opcodes, which will save much time in tracking down other similar problems. In this particular case the four bytes of offending code were replaced by three bytes of legitimate Z80 code, plus a NOP. Other cases may not be so obliging!

Apart from these two incidents no other incompatibilities have been encountered, although, not surprisingly, Z80A peripherals do not respond kindly to being driven

at 8MHz. A Z80A CTC on a home brew board operates perfectly when the system clock is set to 4MHz. It seems very likely that Z80B peripherals will have sufficient margin to operate successfully at 8MHz. All the NE898 facilities tried to date have worked well. These are the CTC and PIO on board the SPCT, and the real time clock. The PIO is fully software and plug compatible with that on the GM811 and GM813 processors. Swapping from the Nascom only required a change of port addresses in the BIOS to allow the printer to operate. I have run the system with Gemini BIOSs V1, V2 and V3, and no problems have been experienced.

It is worth pointing out that in order to obtain standard RS232 baud rates a separate crystal clock is employed, and speed selection is by dip switch. All serial clocks are driven at the same frequency, producing the same baud rate except that each channel can be software set to /16 or /64. This is rather limiting if the serial interfaces are used for differing purposes. A split 1200/75 baud rate is just not possible.

A four way DIL switch is provided with the intention of providing boot options, although it would be possible to use these for other purposes. However, these switches are connected via the four most significant PIO lines on port A. If these lines are used then the switches should be left open, and the state of any external device connected to these lines is liable to alter the cold boot action. One option allowed by the Newburn Monitor is to boot the system from a serial port - this would permit the operation of a diskless configuration.

## Performance

The Z180 gains in raw processing power over the original Z80 in several ways:

- Doubling the clock speed doubles the throughput
- Most instructions complete in fewer clock cycles giving up to 25% increase (Even the NOP completes in three cycles instead of four - this must be of some use!) - however the Z80 used these extra clock cycles to perform memory refresh. The Z180 requires to insert additional refresh cycles, although on a less frequent basis.
- The new instructions, in particular the MLT instruction, can be used to good effect in new software
- Use of the other facilities in particular MMU and DMA can offer very worthwhile benefits

Some impression of the performance gains with pure Z80 code can be gauged from benchmark results, although these are often criticised for not bearing much relationship to real life processing requirements. I have run some of the tests listed in Paddy Coker's articles in Scorpio News Issue 1, Volumes 1 & 2. The results are as follows:






| Test | Language | Nascom @ 4MHz | NE898 @ 8MHz | Relative speed |
|---|---|---|---|---|
| PCW BM8 | MBASIC | 51.2 | 23.0 | 2.23 |
| PCW BM8 | BBC Z80 BASIC | 34.8 | 14.8 | 2.35 |
| Dr Dobbs | Turbo Pascal | 337 | 151 | 2.23 |

(Result = 2500.004634)

All times in seconds, relative speed expressed as Nascom time/NE898 time. All tests were run on my own concoction of a complete Z80 CP/M look-alike.

## Standards

The NE898 departs from 80-BUS specification in a few minor details:

- The /NMI SW line is not implemented
- Line 7 (RSFU) is used to provide the clock for the slave processor in a dual NE898 system (This is link disabled by default)
- The /RAM DIS line is not implemented (Is this significant in a memory mapping system with 20 bit addressing, and should the bus-master generate it or respond to it?)
- The /HALT line is not implemented
- INT0 and INT1 are allocated (via link block) to DMA request lines

None of these is a serious departure, although the author is accustomed to using /NMI SW and /HALT during testing and debugging of interrupt driven software. Address bit A19 is link selectable to either line 49 or line 58, thus accommodating both the MAP80 and Gemini implementations of 20 bit addressing.

## Software support

At the time of writing, Newburn have almost finished an implementation of CP/M + specifically for the NE898. This will make use of the MMU, DMA and RTC facilities, and provide support for all the available interfaces. The price of this will be £199.

## Conclusions

It might appear from my several criticisms that I am not pleased with the NE898. This is not at all the case, all the points I raise are minor inconveniences, or matters of personal preference. The new board is a very welcome upgrade to my system, providing very real benefits in performance terms, and a richness of facilities which I intend incorporating into my operating system in the course of time.

## Appendix

The GM829 problem can be cured by adding a second 270K resistor to the disk controller, in parallel across R13. This halves the hardware delay and appears to provide a remedy without introducing any other difficulties.

The problem with D.COM can be patched out using your favourite debugging tool to alter the code thus:

| | | | |
|---|---|---|---|
| 0944 | DD | -> | DD |
| 0945 | 54 | -> | E5 |
| 0946 | DD | -> | D1 |
| 0947 | 5D | -> | 00 |

The addresses given are correct for the version of D.COM which I currently use (V1.6), other versions may be different.








# An Introduction to C

by R.C. Pearce

The C programming language is still fairly young, though by now well established as one of the major ones. Its development was very much tied in with the development of the UNIX operating system for PDP-11 and VAX mainframes, and a good C compiler will always attempt to create a UNIX-like environment. It is worth noting that much of UNIX was written in C, and all input/output in C is modelled on UNIX.

Like the earlier language B, C is derived largely from the BCPL language, though it differs from those two in supporting 'type'd data (ie. integers, reals etc.). It is a very concise language (which as we all know often equates to unreadable, though it is quite possible to write very clear, tidy and self-explanatory code) and places few restrictions on the programmer, thus allowing fast and efficient code to be written. This leeway raises some doubts as to the validity of the term "high level" when referring to C, and indeed it is commonly used for low level code (the Gemini BIOS 3, FORMAT and WHIG programs were written in C, as was GENSYS).

## Program Structure

Since C is a compiled language, programs must be entered using an editor such as WordStar or PEN. I use HiSoft C so I enter programs in ED80, the HiSoft editor. The layout of a C program does not matter, but it is obviously best to lay it out neatly and in a way which clearly shows the structure. White space can be added wherever it helps and comments can be included anywhere where white space is allowed.

A C program consists of a list of function definitions and global variable definitions. Also there may (and usually will) be some "pre-processor directives". Actual code only occurs within function definitions. There is no program in the way PASCAL has, since the main program is a function called main( ).

A function definition starts with a type declaration like that of a variable except for the parentheses which indicate a function. This is followed by type declarations for the arguments of the function and then the main code. Unlike PASCAL, the argument types are defined outside the parameter list. For example :-

```
char scrn_val ( x,y )
int x, y;
{
/* Code for the function in here */
}
```

Local variables are defined within the function body along with the code, which consists of a list of statements. There are basically four types of statement, which are:-

an expression to evaluate
a pre-defined construct (e.g. while)
a compound statement
the null statement

An interesting note here is that there is no assignment statement, and technically speaking an expression is only evaluated (and thrown away) and not assigned. However, assignment can be carried out as an operator forming part of the evaluation. This allows some very interesting code such as:

```
sum = a-- + ( b = x/y ) + ( c = 2 * d );
```

(This results in b being set to x/y, c being set to 2*d, sum being set to a+b+c (after the previous assignments) and a being decremented.) Note that expressions must be followed by a semi-colon to form a statement. A semi-colon alone is the null statement. Compound statements can be made by enclosing any list of statements within braces ("{" and "}").

## Constructs

There are five defined constructs available :-

```
if ( expression ) statement else statement
while ( expression ) statement
do statement while ( expression ) ;
for ( expression_OPT; expression_OPT; expression_OPT ) statement
switch ( expression ) statement
```

The else section of an if construct is optional. All three expressions in a for statement are optional, but the semicolons are not and neither are any of the parentheses shown above.

Within the "statement" part of some of the above there are some other items allowed. These are :-






```
case constant_expr : statement    /* within switch only */
default : statement               /* within switch only */
break ;                           /* within loop or switch */
continue ;                        /* within loop */
```

Also allowed anywhere in the function body are :-

```
return ;                          /* return from function */
return expression ;               /* return from function with value */
goto label_name ;                 /* destination in the same function */
label_name : statement            /* destination of goto */
```

You may notice that there are no "PRINT" statements or the like. The basic philosophy of C does not allow them as you would find in BASIC, but all the useful ones are supplied as functions (which can be evaluated as (part of) an expression) in the system library.

## Comments

As mentioned earlier, comments in C can occur almost anywhere, even in the middle of an expression or between arguments of a function call. A comment starts with "/*" and ends with "*/" and can contain any characters in between.

## Data Names & Types

Variables in C are given names, the number of significant characters depending on the implementation, and each has a specific type. The basic types allowed are:

| | |
|---|---|
| int : | Signed integer, usually 16 bit. |
| unsigned : | Unsigned integer of same size as int. |
| short : | Signed integer, smaller size than int. |
| long : | Signed integer, larger size than int. |
| char : | Byte value, usually a character but can be unsigned integer. |
| float : | Floating point value ( where implemented ). |
| double : | Double precision float. |

On top of these, a variable can be defined as a pointer to any specific type, or as a structure ( struct ) or union. Structures are the equivalent of PASCAL's record type, and are essentially a fixed array whose members can be of different types. For example:

```
struct obj    { int x;
                int y;
                char obj_type;
                short rotation; };
```

In this case a structure type called obj refers to a pair of integers x,y (the co-ordinates), a character obj_type (the type of object) and a short integer rotation (the direction of the object). Now we can define variables of type "struct obj", or pointers to such a structure etc.

Unions are similar to structures except that all items are at the same place and thus only one item is valid at a time. The main use of unions is for shared allocation of data to reduce overall memory usage.

Arrays of any type are allowed using square brackets ("[" & "]") around the index. Multi-dimension arrays use multiple brackets. For example:

```
char      array_of_chars[16];
int       two_dimensional_array[10][10];
```

The name of an array so defined translates as simply a constant of type pointer_to_element_type, so array_of_chars above is a constant character pointer (written as "char * array_of_chars") whose value is the address of the start of the array. This fact is not important for the use of arrays, but it does mean that a variable of a pointer type can be used to point to the base of an array, for example:

```
char * string;          /* define variable string as a character pointer */
if ( string[4] = = 'h' ) do_something( );   /* test the fifth character */
```

There is no string type in C. A string is stored in an array of chars, usually terminated by a NULL. Constant strings can be entered, e.g.

```
printf ( "Hello World" );
```

However, the string here evaluates to a constant of type "pointer to char" which is the address of an area of memory containing the twelve bytes (including a terminating NULL). Consequently a statement such as

```
if ( input_string = = "YES" ) ....
```

makes no sense at all in C. The way round this is to use one of the standard functions supplied in the library.






## Operators

The standard operators are all available in C, but there are some interesting notes. The logical operators are called & (and), | (or) and ^ (xor). Also available are && and || which perform logical (as opposed to bitwise) AND and OR. These operators have the very useful feature that they do not evaluate the second argument if there is no need. For example,

```
mflg && !timeout( )
```

will not evaluate !timeout( ) (logical not of function timeout( )'s result) if mflg is false.

The equality operator is " = = " to distinguish from the assignment operator. There are a set of useful shorthand notations for amendment assignments. For example, we need not write:

```
total = total + entry
```

because we can use

```
total + = entry
```

Similarly we can have - =, * =, / =, & = etc. Also available are + + and -- which increment or decrement the value to which they are attached. If they occur before the reference they operate before the value is taken otherwise the operate after. Thus if a = 2 and b = 5,

```
a+ + + b+ + = = 7          + +a + + +b = = 9
```

## Input & Output

As mentioned earlier, there are no input or output commands as such in C. However, the standard function library should contain a good selection. All I/O is handled through "streams" which can be the keyboard, a printer, the screen, a disk file or whatever. Under UNIX, all console I/O is treated as a group of streams which can consequently be redirected if desired. UNIX allows stdin and stdout (the default console streams) to be redirected on the command line which invokes the program. Since CP/M does not do this, the extent to which this particular feature is supported will depend on the implementation. HiSoft will operate correctly providing the program calls a standard library function "cpm_cmd_line( )" before doing anything else. The compiler used to compile the Gemini utilities I mentioned earlier does not support command line redirection.

The C I/O system starts with a set of character level functions: getc(chan), putc(chan,char), getchar( ) and putchar(char). The last two use the default streams. Unlike CP/M, UNIX uses NEWLINE (ASCII 10) to separate lines rather than CR,LF (13,10). Also unlike CP/M, UNIX knows the exact length of all files, so end of file is reported by a value of -1 (which is why getc( ) and getchar( ) return an integer rather than a character). Because of these differences, files must be declared as either text or binary type. This is not a problem since UNIX versions of C must be able to cope with standard terminal devices so the facility is available.

Built on the basic I/O above are string I/O functions: puts(str), gets(str), fputs(str,chan) and fgets(str,len,chan), and also word I/O getw(chan) and putw(word,chan) and block I/O which accounts for a number of similar functions which I don't have room to go into here. The behaviour of the string I/O is slightly different with the file functions. Specifically, puts( ) adds a newline, fputs( ) does not. Also fgets( ) has a length limit and leaves the newline in, whereas gets( ) overwrites the newline with the NULL.

The most useful functions for high level I/O are printf( ) and scanf( ). There are also file based equivalents fprintf( ) and fscanf( ), and string based versions sprintf( ) and sscanf( ). These both take a variable number of arguments (they are variadic), the first of which is a control string (with the string and file versions the control string comes second after the file-pointer or string-pointer). The control string declares how the output/input should look. For example:

```
printf ( "File %s, %d records in %dK\n", filename, record_count, size );
```

Which would give, for a file FRED with 25 records in 4K, the output

```
File FRED, 25 records in 4K    <newline>
```

The item declarators following % symbols can specify minimum and maximum field widths, left or right justify, and in the case of numeric values the leading zeros can optionally be displayed. This is best shown by a table:

| Control string | Values | Result |
|---|---|---|
| "'%c' - %d <%x>" | 48,48,48 | '0' - 48 <30> |
| "*%-15s %02d*" | "Total",5 | *Total 05* |
| "%3d %3d %c" | 12,6,83 | 12 6 S |

The scanf( ) function attempts to match the input to the control and assigns values accordingly. For example, if you expect an input like "15 : x (4,5)", you might use a control such as " %d : %c ( %d , %d )". The spaces in the control will match any space (or no space) on the input.






## Leaving The Program

Since a C program is run by effectively calling the main( ) function, any exit from this function will cause a return to system. This will occur when the code for main( ) runs out, or if a return statement occurs in main( ). There is also usually a function supplied called exit( ). This takes one argument which is usually 0 for a normal exit or non-zero for an error exit.

## Conclusions

I have obviously not had space to attempt to teach anyone to use C in this article and that was never my intention. For those of you who have not yet tried out C, I hope this little taster has whet your appetite. There are a good number of C compilers around at a range of prices. Probably the cheapest is HiSoft C++ (a misleading title as C++ is a slightly different language. HiSoft is standard C). HiSoft does not support floating point and it compiles directly to a .COM file without using a linker (which I find a disadvantage although some people prefer it), but it comes with a good editor and is a good version to start on as it is pretty much correct and standard. There are also a number of good books on the subject, but look for recommendations as there are some bad ones too. The ultimate definitive book is "The C Programming Language" by Kernighan and Ritchie, but this is probably a bit heavy for beginners. ●

# Picking up the PCs

by P.D.Coker

A couple of years ago, I was reading one of the more erudite electronic magazines (Elektor) and came across a series of articles on the construction of a PC compatible - this was some time before Mr Sugar let loose his PC 1512s on the scene. I found that a motherboard (PCB) was available for about £75, and a scrounge around produced most of the chips needed to populate it. I was even able to get hold of a PC keyboard and other requisite hardware at a very good price - the whole lot cost rather less than £250. Naturally, there were conflicts in the constructional and setting up details and even Elektor got one or two things wrong. Sadly, the machine didn't work and it was consigned to the "pending" area of my workshop until I had the time to do something about it! I carried on computing with CP/M Plus and the trusty Map-80 system.

Then, the Amstrad PCs started the rot. After some piggy-bank rattling, my wife and I decided to buy one of these, add 128k of memory and upgrade it with a V30 and

8087 to improve its already quite reasonable facilities and performance.... All was sweetness and light until my wife - one of those highly motivated people who take Open University courses seriously, decided to bite the bullet and register for the brand-new course on computing, which started in February. "I'll need the Amstrad", she said. I was stunned - bereft, even - MSDOS leaves something to be desired but the range of appropriate software available for PCs, which I had become used to was greater than I had ever found for CP/M. The prospect of returning to CP/M didn't excite me either (murmurs of "what about CCPZ then?"), since I rather liked WordStar Professional and found that even CP/M WordStar 3.3 was very slow and lacking in useful features. This is quite an important consideration when one spends a lot of one's time grinding out words rather than programs.

There was a solution - get the DTC PC-clone working. A couple of abortive sessions revealed that there were a few bridged tracks but I failed to spot a number of dodgy soldered joints (and even unsoldered pins) - typical of the sort of problems which used to beset Nascom constructors. I was talking one day to Ian Cullen and he kindly agreed to have a look at it. After quite a lot of hassle, he managed to get it working and even after the kind attentions of the Royal Mail, who managed to damage the power supply and keyboard with a well-aimed kick, it arrived and has been in regular use since. Fortunately, it is a very close clone of the IBM PC-XT and will boot up and work quite happily with any of the versions of MSDOS that I have. I've also had a go at improving its performance - the original 256k of memory was upgraded to 640k and the 8088 CPU was replaced with a second-hand Victor 80286 board which plugs into the 8088 socket - this improved its performance quite dramatically. (about 6 or 7 times better than an IBM PC, according to the infamous Norton SYSINFO program).

So why go to the bother of building a PC from scratch? Amstrad PCs are relatively cheap and are likely to become available on the second hand market as newer models become available. The answer is probably most obvious to those who built their Nascoms from kits - it's the challenge of actually getting the b****r to work and possibly the addiction that some of us have for burnt fingers and solder fumes. Regrettably, now only one 80-BUS manufacturer allows us to construct our boards from kits. The financial saving is worthwhile in some cases, as it was for me since the majority of the i.c's were salvaged from obsolete but working boards obtained from suppliers such as Greenweld for extremely low prices. The technique used for desoldering sounds horrendous - a gas blowtorch, carefully applied to the pins, a quick flick of a thin-bladed screwdriver underneath the chip and out it comes - but it is a lot easier than fiddling around with a soldering iron and solder pump, and much safer if you do it outside the house! I rescued 48 41256 DRAMs this way and all of them worked afterwards...

Bare PC boards are obtainable (if you persist) in the UK but I purchased mine from an American firm (Display Telecommunications Corporation in Texas). Bare, as






well as constructed but memory-less boards are available from a number of American firms and can be paid for with Visa or Mastercharge (Access) at extraordinarily good prices, even allowing for the imposition of VAT and, probably customs duty as well when they arrive in the UK. Service is generally quick and courteous - if you pay by credit card, delivery is very rapid (I had my board about 10 days after sending the order). If you pay by cheque, do get a quotation first, to include shipping and other charges - Lloyds Bank offer a convenient means of payment via an American Express draft which is a lot cheaper than many other methods of payment.

I must admit to being bitten by the 'construct-a-clone' bug. A browse through 'Byte' shows that JDR Microelectronics do a ready-built Turbo XT-clone main board for under $100 (supply your own RAM but it runs at 8 MHz); another firm advertises a 10MHz AT clone main board for well under $300.... Hmm. I wonder if anyone wants an XT clone with 80286 Accelerator board, SMPS etc. (yes, this is an advertisement - if Dr Dark can slip one in, I will as well!)

For the would-be clone-constructor, a major problem is the cost of memory chips. Prices have risen dramatically in recent months as a result of pressure on Far-Eastern manufacturers who have been dumping their excess chips on the Americans. The minimum memory required for most MSDOS programs is 256k (parity-checked with 36 150 nS 4164s or 9 150 nS 41256s), although 512k is more useful. Greenweld seem to obtain (from time to time) supplies of boards with one or other of these types of DRAM (usually) soldered in and removal by conventional means is straightforward but inclined to be tedious. The fact that most suppliers now sell only the assembled main boards does mean that the type of problems which Ian sorted out for me should not occur. If you are interested, it might be an intriguing and instructive route into MSDOS.

You will, of course, need floppy disk controller and video cards, and an appropriate XT or AT compatible keyboard - and at least one 40 track 5.25" drive. It may be possible for you to fiddle with a standard monitor so that it can cope with the 18 KHz or so timebase required by the IBM video card - but suitable switch mode power supplies are available from a number of suppliers for about £25. Digitask are a useful firm for IBM PC-compatible items, but a bit of browsing through electronic magazines will provide you with plenty of addresses for other items. In particular, Matmos (in Cuckfield) or their close associate Computer Appreciation (in Canterbury) are worth contacting - they seem to obtain some remarkably good bargains from time to time - particularly for stripping down, although they seem to specialise in 'end of run' or lesser-known second-hand systems. I would be very interested to hear how other constructors have fared..... ●

# Direct Keyboard Commands - Update & More GEMPEN Mods.

by C. Bowden. (Tel. 0209 - 860480)

In Issue 1 of Volume 1 of Scorpio News, I described some BIOS modifications to provide immediate acting commands from the keyboard. I have updated and refined this software. In particular I have added a routine to allow the saving of Screen display to a Disk File (From the Operating System only), and altered the Screen Dump facility to allow it to be accessed from within any program. Previously it could only be accessed from 'EDIT' mode, which meant it would not work from within some programs that redefined the 'EDIT' key. Several more commands have been added, and the BIOS code has been altered to save memory, allowing more routines to be added.

My BIOS is a home-made modification of MAP80 CP/M 2.2 BIOS with some SYS18 routines patched in. It supports a wide combination of hardware e.g.; NASCOM 2 or GM813 CPU, MAP4 or MAP32 paging, MAPRAM, GM833/NE889 Ramdisk, 809/829 FDC, 812/832 IVC/SVC, Nascom and/or Gemini Keyboards, Micropolis/Teac Floppies, Rodime and Shugart Winnies. The only thing it does not currently support is the Nascom Screen or the GM849 FDC.

It should not be too difficult to patch the necessary code into any BIOS that is similar to SYS18 or MAP BIOS. I do not use CPM 3 much, since I think that the ZCPR3 system is far superior, but it would probably be possible to patch something similar into the MAP CPM3, for which the code is supplied. If you are using a recent GEMINI BIOS, then you have had it ! (Unless you care to disassemble it).

The code to be patched into the BIOS can be supplied on a disk if anyone is interested. Contact me on the 'phone number given.

In the original article I described seven commands. The current version has 15 commands. They are accessed, as before, by entering ^T. This causes a prompt to be displayed on the top line of the screen, with the required operative key displayed in inverted video, as a reminder of the commands available. The second key may be a 'straight' key or Control key. The options available are limited mainly by the BIOS space available, and could be considerably extended if a longer BIOS is acceptable, or if some hardware options are not supported, such as RAMDISK or Winchester.






Commands that I have currently available are:

^T followed by one of –

| | | |
|---|---|---|
| P | - | Page the Printer and reset BIOS line counter. |
| N | - | Select attached printer to Normal print. |
| C | - | Select attached printer to Condensed print. |
| I | - | Initialise Printer. |
| A | - | Set paging to 64 (A4) and skip to 6 |
| J | - | Set paging to 66 (12 in paper) and skip to 6 |
| K | - | Set paging to 60 (11 in paper) and skip to 6 |
| O | - | Paging OFF. (Skip set to 0) |
| F | - | Save current screen to file S.CPY. |
| S | - | Turn screen paging ON. (Cancels ^K or ^W commands) |
| D | - | Dump current screen to Printer. |
| L | - | Lock screen Top line. |
| U | - | Unlock Screen top line. |
| Z | - | Clear Screen. |
| R | - | Reset IVC/SVC. |
| ^T | - | Pass ^T to current program. |

Any other key returns a space character to the current program. Most of the above are self explanatory, but a few comments are in order.

The ^TP command will cause the printer to advance to the top of the next page, and reset the BIOS internal counters so that the printer can always be controlled from the keyboard. (The only time that problems arise is when lines longer than the current maximum printer line are sent to the printer. The BIOS will count 1 line when it sees a CRLF sequence, even if this is after 200 characters, but the printer will put in its own every 80/132/233 or whatever, so that misalignment can occasionally occur.) I intend to add a Character per line feature to my BIOS, and make this alterable to 80/132/233 by means of another ^T command. This would solve the problem.

^TO will set the BIOS skip count to 0 lines, so that rolls of paper or non standard operations can be controlled. To restore the skip, use ^TA, ^TJ, or ^TK as appropriate.

^TF will save the contents of the screen to a disk file called S.CPY. This is a 'quick and dirty' routine that will only work from the operating system. It works by putting the command SAVE 8 S.CPY into the (ZCPR3) command buffer, copying the screen to RAM at 100H, and then rebooting. I have not tried it with standard CP/M by putting the command into the CPM command buffer, but it works well with ZCPR3.

I have modified my BIOS so that screen paging is OFF when the system boots. By typing ^TS the screen paging flag is set to restore paging, and most of the SYS paging commands are operative. The problem with the SYS method was that if paging had been turned off, a Warm or Cold boot was needed to restore it. With my method it can be turned ON or OFF at will. The BIOS screen paging routine has been further modified so that control characters are needed to control the paging options. (ie; ^C, ^S, ^R, ^K). The Space key is the only exception, and this functions as in SYS. The 'W' option is now redundant.

Screen Dump control ( ^TD) has been transferred to this function, rather than from EDIT mode, so that it is universally available.

No originality is claimed for the various support subroutines in the listing, but I have included them to show how they inter-relate with the direct keyboard commands, or how extra space can be saved.

The routine SIVCMD is used to send command strings to the IVC/SVC. It saves a considerable amount of code overall compared multiple blocks of code like -

```
LD      A,ESC
CALL    CRTA
LD      A,'M'
CALL    CRTA            10 bytes
```

The above is replaced by -

```
CALL    SIVCMD
DB      ESC,'M',0       6 bytes
```

The actual routine SIVCMD only takes 8 bytes, so if more than two blocks like the first example above are used, there is a real saving in BIOS size, allowing room for more features.

The clock routines listed are for the MAP80 R.T.C. card, which is a small card that shares the Printer Centronics port. I use several of these cards, at home and at work, and they are very successful, keeping good time, and not interacting with printer operation or being corrupted by it in any way. The main benefit of this card is that it does not require an extra port. I have also successfully used the RTC on the GM816 I/O card for this purpose. I was not able to overcome the clock corruption problems when I tried the GM822 RTC card, some years ago. I have added an extra 'JUMP' to the jump table at the start of the BIOS, so that the clock routine can always be found by external programs, by calling this address at BIOS + 33H. On EXIT from the clock routines, HL contains the address of the Date and Time string so that it can be found by external programs.






The listing contains a number of conditionals such as 'IF SSAVE' and 'IF CLOCK', and a number of definitions such as 'PCTRLB' and 'NMRREG'. These are defined in the EQUATES section of my BIOS. The purpose of each should be fairly obvious, but I would be pleased to help anyone trying to implement these routines who is in difficulty.

## More GEMPEN Modifications.

Since I submitted the article describing modifications that I have made to GEMPEN (PEN3.COM) and which was published in Scorpio News, Vol. 2, Issue 1, I have carried out some more useful alterations to GEMPEN. I call the present version PEN4.COM. The related HELP overlay is called HLP4.OVL. The extra features of PEN4.COM over those added to PEN3.COM and described in Vol. 1, Issue 4 are:

a) The top line of the screen now displays the current EDIT file name, and the current Date and time, if a system clock exists.

b) ^QH - The ^QH command updates the screen Date/Time Display if a system clock is supported and asks whether a Date and Time 'tag' should be inserted at the start of file in the format:

```
;> SAVSCRN .DOC 27/02/88 15:09 <
;>
```

If a Clock is not supported by the BIOS, the Date/Time Tag must be hand-edited. The format of the 'tag' is such that it appears as a comment in a source code file. The ^QH routine is 'called' by the ^QW and ^QE routines so that the operator is reminded to consider whether he wishes to tag the file being saved.

c) ^QN - Now allows four options. The prompt now displays:

Remove A)ll, S)tar, B)lock, N)one ?

a) 'A' will delete ALL '*' and '\' markers from the text.
b) 'B' will delete all '\' markers.
c) 'S' will delete all '*' markers.
d) 'N' will not delete either '*' or '\' markers.

The ^QN routine is called by the ^QW and ^QE commands so that the operator is reminded to remove markers before a file is saved.

d) ^QP - The Print command now supports PAGE Heading and Numbering.

This command first calls the ^QH command, to allow the operator to 'tag' the file. (N.B. This returns the cursor to the START of text, so that the tag will be printed at the start of the file). The operator is then asked whether he wants the pages numbered. If the answer is yes he must then select the title to be printed at the head of each page, which may be either:

a) Blank
b) The File name as displayed on the top line of the screen.
c) Enter his own title. (20 Characters max.)

e) 5 & 6 Commands. These have been altered again and now change the starting page number for the ^QP command so that it can have any value from 1 to 255.

f) ^A - Convert ALL text from cursor to end of text to LOWER case.

g) ^U - Convert ALL text from cursor to end of text to UPPER case.

h) D - Delete line from cursor to previous NL character. This now displays a '>' character in inverted video at the position where the delete will end. With several lines of non-formatted text, it is now obvious what will be deleted. The only exception is line 1 which can only be deleted by using the backspace key.

i) ^C, ^B - Now scrolls the screen 20 lines instead of one quarter of a page as set by the 3 and 4 commands.

In order to accommodate these changes, my customised version of Gempen has grown by about half a K. All overlays still work satisfactorily however, and the program runs on NASCOM systems with IVC/SVC and Gemini Keyboards. It has not been tried with a NASCOM keyboard, and will NOT run with a NASCOM screen. If the system does NOT support a Clock, Bit 0 of VIDMEMFLG must be set accordingly, or the program will crash. The customisation locations for Clock and Printer support are described in the notes that I have written to support the modifications.

Anyone interested in obtaining a copy of the modified version and notes can telephone me on the number at the start of this article to discuss details. I will need a disk with your copy of GEMPEN on it so that I can serialize the patched version, since I would be infringing the original authors' copyright otherwise. •






# Digital Imaging Systems – Part 3

by D.R.Hunt

## The Print System.

As the expanded image is in memory, it can be viewed by the display system exactly as a freshly scanned image. And if the compression and expansion part of the system has done its job properly, the resulting image will be a dot for dot replica of the original.

This brings us to the last part of the system. The print system. It's usual to use a Laser printer for the job and these work in yet another simple fashion. Like scanners, there are two common sizes of printer, A3 and A4. They work in exactly the same fashion, one is bigger, that's all. The Laser printer is based on the photocopier and uses a Xerographic process, which translated means 'dry photography'. (Xerographic is the original name coined by the inventor. He then sold his designs to a certain large company who formed a company with a similar name and they did very well – thank you very much – until the patents ran out.)

The Xerographic process in brief, uses a drum coated with a photo–conductor, it used to be the metallic element selenium (selenium isn't actually chemically a metal), but these days there are many other things which changes their electrical properties when light falls on them, and often the photo–conductive properties of silicon are employed. Anyway, the way it works – the photo–conductive coating is charged with a very high voltage and a static charge builds up on the drum. The drum is then exposed to the light reflected off the brightly illuminated paper original, which moves past as the drum rotates (or a moving mirror reflects the image of the paper onto the drum as it rotates). Where light falls on the drum, it changes its electrical conductance and the charge leaks away. This leaves a pattern of static charges on the drum where the black bits were but no charge in the white areas. The drum continues to rotate past a tray full off very fine soot, well not really soot, it's finely divided graphite mixed with a dry resin glue. Being a fine dust, the soot is attracted to the static charges on drum, so the drum picks up the soot in the patterns of the black bits of the image. Whilst the top side of the drum is being charged and discharged and picking up the soot, the under–side of the drum is also rolling a piece of blank paper through the works, and the pattern of soot ends up on the paper. The final touch is the paper is passed out through heated rollers which melt the resin glue and stick the soot to the paper. Funny, copier repairmen get upset when I refer to the toner as soot, but I ask you, what else is finely divided graphite !

The Laser printer works in exactly the same way, except there is no paper original as with the photocopier. It stores the image to be printed in a large area of memory of its own, and when the image is complete, it starts to print. Again the self same properties of the Laser, which make it so attractive in the optical disk come into play, namely, the ability to focus the light to a very fine spot and to switch it on and off very fast. The laser scans the drum using a swinging or rotating mirror and switches on for white dots and off for black, this way it draws lines of dots on the drum – the self same dots which went to make up the image in the computer memory. The rest of the story is the same as the photocopier.

We established that there were some 8 million dots on an A4 page scanned at 300 dpi, and that's a lot of dots. These dot patterns have to be transferred from the computer memory to the printer memory, and if left to the traditional methods of connecting printers to computers, would take until the middle of next week ! Some imaging systems do just this on grounds of cost, as the electronics to do it are all present inside the printer, it's just that they weren't designed to transfer huge lumps of graphics, and as a result, are very slow. Better systems replace the data transfer system with one of their own. This is usually built into the computer which takes the existing image in the computer memory and dumps it, very fast, straight at the Laser in the printer, by–passing the printer memory and most of the intervening printer electronics. More costly, but a lot faster.

That just about wraps up the technical side of it. I hope I've explained the workings adequately. But before going on to my pets hates and prejudices regarding the software used to drive it all, just a few words about what else can be done with the hardware.

## Other uses.

Most of the kit shares a lot in common with 'Desk–Top Publishing'. In fact, the Imaging System in it's idle moments becomes a very effective DTP (another acronym: Desk–Top Publisher/Publishing). Not that I would suggest for a moment that the optical disks be used for storing the DTP output, that would waste space and anyway, DTP output is best directed at the computers' hard disk – but it works very well indeed.

Another area is FAX, a means of communication which has been in the doldrums for the last 20 years. Now as Japan Inc. has found itself running out of sales for its photocopiers, they've bent the technology to FAX and with typical Japanese sales technique have brought the market into the 1980's. It's now 'the done thing' to have FAX in the office. The Imaging System only requires the inclusion of a suitable FAX modem (yet another acronym: modem – MOdulator/DEModulator – a device for sending digital data over the phone line), and a piece of communications software, and the Imaging System is a FAX machine in its own right. Not only can






it send external images but it can send its own stored images as well. Ok, so it's a very expensive FAX machine, but that's not why you bought it.

Yet another sneaky idea is the 'Security Photocopier'. Now in high security establishments, they tend to guard their photocopiers not so much against unauthorised use as the copying of unauthorised documents. The problem with a photocopier is that once it's done its business, that is made a photocopy, no–one knows what was photocopied. Now it's easy enough to bar access to the photocopier, using a security key is one way, but suppose someone with a security key intended to be naughty and copy things he wasn't supposed to anyway. There's no way of knowing what was copied. Alter the logic of the Imaging System, so it requires a password or key to use it – that way you know who used it – because the system logs who used it. As it copies it compresses the image(s) of what was copied to the disk and ties it to the log–on. That way you not only know who used it, but what was copied as well. The only problem with that is if it was a stolen key or password – but that's a physical security problem. Again, you wouldn't buy an Imaging System as a photocopier, but it's a useful adjunct to it's normal use.

## The Software.

At last – on to the software. Now all the hardware is well tried boxes of works, each of which has been designed to do it's job, either on its own or in conjunction with other boxes of works. The thing which hangs it all together is the computer software which runs it and if anything, it's the software which is the most important part of all. Now I have some very fierce criticisms of some approaches to the software design of some imaging systems. Most of my criticism stems from one of two (to my mind, mistaken) assumptions made by the software designers, based on the fact that as Imaging Systems are expensive and the software will inherently be a low volume dedicated market compared with, say word processing software. This means that the cost of software design which is high in any event, will be amortised across a relatively low volume of sales. The first assumption seems to be, *"As we aren't going sell many anyway, let's throw the software together as cheaply as possible."*, the second assumption, is the reverse, *"As the software is going to cost a bomb, let's make it as thorough and complicated as possible, that way the punter thinks he's getting his money's worth."*. Much software I've seen falls into the latter category, and as a result you need a Ph.D. to drive it. Of course I advocate a middle road, I'm a firm believer in KISS technology (translated in two ways, either 'Keep It Simple and Stupid' or as an instruction to the programmer 'Keep It Simple – Stupid'), I prefer, the 'Keep It Simple – Stupid' approach. That way you end up with 'bomb–proof' (another Computerese term which means 'no idiot can make it go wrong', and it's relatively free of bugs) software which is easy for an unskilled operator to use, and which covers 95% of all possible uses. It's always the remaining 5% of clever ideas which don't work and cause complication. The only trouble with KISS software is that it's

almost as hard and time consuming to write as the 'over the top, all bells and whistles' kind.

The aim with Imaging Systems is two–fold, to enter data as painlessly as possible, and once entered, to extract it with the minimum of fuss. All the software I've seen achieves these aims, it's the amount of bother attached to the tasks which annoys me, and it's not the actual data entry and extraction which is usually at fault. It's the way in which these are approached. There is a great tendency these days to go on first impressions. Perhaps, the tendency has always been there, it could be as I get older, I see it for what it is.

My point is that it's not first impressions which should count. But salesmen only have a short time to impress the customer, so any impression made has to count. Take, for instance the use of 'pull–down' or 'pop–up' menus in software, where a little window appears on the screen containing commands or instructions. Fine – nice idea. These are a salesman's dream, they look neat and tidy, very clear (if properly laid out) to the prospective buyer and so make the salesman's job easier. Pop–up and pull–down menus are, more often than not, coupled with the use of a mouse, claimed as a great ergonomic tool by Xerox, who first researched the idea. The problem with the combination is that *if* the job in hand is essentially a keyboard entry job, it means moving one hand away from the keyboard, usually the right hand, to reach the mouse to point to the menu required. This leaves the left hand for typing. Now I'm a reasonably proficient typist, not a normal feature amongst those who design and write software, or for that matter those who buy (as opposed to use) software. Now I can't type left handed – I need both hands ! To me, and to many others who can type, and many others who use computers seriously, the mouse is an absolute pain. It slows things up. Sometimes, only too rarely, the mouse commands are supplemented by a set of keyboard commands, so the hands don't leave the keyboard. But then, if keyboard commands are provided as well, why bother with the mouse in the first place ?

Don't get me wrong, Xerox weren't wrong in their research, the mouse is an invaluable tool – *when used with graphics and combinations of text and graphics.* It's simply that the mice have been breeding and have now spread to areas where they shouldn't have.

Pop–up and pull–down menus have their bad points (they must have some advantages when used with keyboard entry, but for the life of me, I can't think of any). Because they 'pop–up' and are supposed to appear (but never do) in an unused area of the screen, they are invariably quite small. This means they usually contain only half the set of commands required and so require another menu for the rest, or if they contain instructions, they go on endlessly. The guy who designs them tends to ensure they overlay each other, always slightly to the right and down from the previous one. This is supposed to represent a pile of pages on a desk. So you can






tell there are a number of different pages, they are coloured differently. This leads to the most appalling Technicolour splodge on the screen – confusing, messy and not in the least helpful. Isn't it simpler to use a larger area of screen – like all of it – for instructions and menus, and when done with them make them go away ? My preference is to use the whole screen for menus, offering all the choices at once with separate help screens (also full screen size) which come and go on request. It's not first impressions which count. Windows (as pop–ups are called) might look very clever to the prospective buyer, but pity the person who has to use them with no other choice. After a day or so, they know the commands and what the help says. How they wish they could turn it all off.

Pop–up and pull–down menus are good in their place, like the mice, with graphics, but aren't clever when it comes to anything which is predominately keyboard oriented. When someone is really familiar with something, menus and help screens aren't required at all. So the step beyond my idea of full page menus is to make command keys work direct. You simply bash a key and the software goes ahead and does what is asked. Mind you, in keeping with the KISS idea of doing things, this last approach has to be pretty clever in its own right. The user might bash keys and the software reacts, but what if the wrong keys are hit ? This is a process known as 'input validation' and that's what KISS is all about. Keep the software simple to use, but make the software check everything before it does it. This isn't so easy as it sounds, because the software writer has to anticipate everything which could go wrong. I reckon a full 30% of the software I've written using these ideas is down to input validation.

Input validation must always be polite and to the point, in fact all responses from the computer must be polite and to the point. Users don't like being insulted by a dumb machine, they don't like their mistakes pointed out to them at any time, and least of all by a smart–arse computer. The error messages I see on a lot of software make me wince. These are terse in the extreme, and although not exactly rude, are not exactly helpful either. May be you've seen the sort of thing I mean:

"INPUT ERROR IN FIND."

much nicer and more helpful to say

"You have entered the find command incorrectly, please correct and retry."

A lot more words, but words costs nothing and at least it's helpful, it tells you what you've done wrong and what to do about it. A few error messages like the first, and those who have little experience soon build up a resentment of the computer and will then use them reluctantly from then on. Some while ago, I got fed–up with writing polite software and wrote a cataloguing program where all the error messages were abusive to put it mildly and where it wasn't abusive, it was wildly

sarcastic. This program has gone done well in computing circles, but I daren't let the public at large have a go at, it might put them off the idea of computers for ever !

So less misapplied gimmicks in the way of mice and multiple menus, more straight forward common sense and appreciation of the person who has to use the software.

Having said all that, there's always the first time user. This presents a contradiction in my terms of reference. If the software is to be slick in use, it can't stop and politely prompt the first time user every time he makes a mistake. This all comes down to simplicity. If any function of a piece of software is simple enough, a totally untrained operator should get to grips with it instinctively. In other words, the software should do what the user expects it to do rather than doing what might be the easiest option for the programmer.

This requires a bit of research on the part of the software designer. There is always a temptation to do things with computers in a certain way, simply because someone has done it that way before. It's part of a software analyst's job to find out how the task is tackled, unfortunately, he often only asks those experienced in the job – fine, they tell him the way the job is done now, and in the light of past experience, it may it may not be the most efficient approach – but is it the way that a novice would approach the problem ? When presented with an analysis job, I've always taken the trouble to talk to a number of people, not just those who do the job at present, but to ask those who know something about the task, but don't actually do it, how they would approach it, could it be improved, and so on. This can be most revealing, it often shows ways in which the job may be simplified, improved, but even more important, it shows the instinctive way that a novice would tackle it. This is a good pointer. If software works the way you expect it to work, then it's so much easier to drive.

I've found that if you sit someone in front of software designed along these principles, someone who has a superficial knowledge of what the job's about, then they can usually work their way round the software using the help screens alone without reference to the manual inside an hour. This of course is no great surprise. It's simply the application of common sense to an area which to my mind is organised too much for the ease of the program designer and too little for the ease of the people who have to use it.

So to sum up, when looking at software, don't be beguiled by the salesman's blandishments that it's easy to use. If you're seriously interested in how well it all works, don't go on first impressions. Sit down in front of it, not beside the salesman with the salesman in the driving seat, and see if you can drive it yourself. If you can afford the time, spend the time and see if it works. If after a few minutes, you're starting to get the idea, then the equipment will become more important than what the salesman says about it. As a final test, take someone who has to use the kit for






real along to see it. These are the people who have to actually use it, isn't it fair to make sure they can live with it before purchasing it ?

So I hope this little paper, with it's concentration on the way things work, has helped to remove some of the mystique from the subject. A subject which at the moment is full of unpleasant surprises, it's like a mine field. This dozen or so pages won't have removed all the mines – life's too short. But if I've made a safe path through, I've achieved something.



## Private Ads.

Nascom 2 CPU board, CP/M, NASDOS, £85. 256K MAP80, £50. AVC, £50. Dual disk, £50. FDC, £25. Maplin MODEM, £25. Digitalker board, £10. Chassis style green screen monitor, £10. Kenilworth case, PSU, backplane, £25.
Phone 01-553-8765 – 8.30–5.00.

813 CPU £82, 832 SVC £82, 812 IVC £50, 829 FDC/SASI £82, 805 FDC £15, 803 EPROM card £20, 802 64K RAM £20, Nascom RAM B £18, 836 Network cards (inc software) £40. Rodime 204 (20MB) £140, CDC Wren 9415-36 30MB Wini £163. Controller cards: Xebec S1410 £138, Adaptec ACB4000 £195. Streamers: Tandberg TDC3200 £140, Archive 5945L (QIC02 i/f, QIC11/24 format, DC300/600 tapes, full spec, half height) £178. 1115F6 800K Micropolis FDD £47, 8" drive double/single sided £85. SM-PSUs: GM817 85W £45, GM850 50W £33. KBDs: GM827 cased £47, GM852 low profile case/cable £82. 3 slot card cage £30, 6 slot £43, 8 slot b/p with 5 connectors £36. Vero frame/front/back panels suit N2/Pluto £55, GM686 Workstation boxes £34. GM661 Double drive Galaxy case £62, Phoenix monitors (Grn & Amber) £47.
All bits GWO, ANY offers, phone Ian (0473) 831353.

Quantum 2000 (GM812, GM813, GM829, GM827 keyboard, 800K drive), amber screen monitor, together with two uncased Shugart 800/801 8" SSSD drives, home brew PSU for 8" drives. BIOS will run all drives (8" as SD or DD 604K), £450. 10MB Xebec half-height winchester with SASI interface, plugs direct into GM829, little used, £150. GM802, GM829, GM811, GM803, £100. 800K disk drive, £50. Two uncased SA800/801 8" SSSD drives, £30. GM827 keyboard cased, £25. Alloy case with 8-way motherboard, £25.
Phone: J.D. Wilson, Halifax (0422) 56725

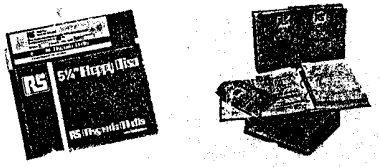
FOR SALE: Pluto card, trackerball & interface, Cotron colour monitor, CAD-8 software. Offers. WANTED: Gemini keyboard.
Phone: Mr Emptage, Staines (0784) 54373 evenings.

Rodime RO203 15MB Wini, brand new and unused. £100 CASH (£25 extra if you want to pay by cheque). P.D. Coker, 0689 58510. •

# NEWBURN

## CP/M 3 PLUS OPERATING SYSTEM

DIGITAL RESEARCH™



CP/M Plus™
(CP/M® Version 3)
Operating System

- Supports 1MByte of Memory.
- Memory transfers via DMA.
- Ram-disc transfers via DMA
- Extended 33 BIOS calls.
- Auto Bank Switching.
- Interrupt support via CP/M.
- Hard disc capacities 512MB
- Supports 2 Hard discs.
- On-board virtual "P" disc.
- Resident Extensions (RSXs).
- Static Ram-disc support.
- Multiple sector Read/Write.
- FIGSYS drive configuration.
- Support of GM873/833
- Files Time & Date Stamped.
- Files "Archive" stamped.
- Files Password protection.
- FULL directory command.
- User added Help files.
- Supports 12 I/O devices
- I/O devices by name.
- Versions for NE898 or GM813

The Digital research CP/M 3 Operating system is upwards compatible with CP/M 2 for all BDOS calls, BIOS calls, memory allocation etc.

We have yet to find a CP/M 2 program that does not run correctly in CP/M 3.

CP/M 3 maintains a large user TPA by "Banking" the operating system outside of the user area. 60K TPAs are usual. Large disc buffers are also used (outside TPA) for faster disc transfers, hashing etc.

CP/M3+ also remains disc compatible with CP/M 2. Floppies, Ram-discs &winchesters may be freely interchanged between CP/M 2.2 & CP/M Plus systems.

Software developed under CP/M3 Plus may be simply blown into eprom(s) for disc-less target system operation.

The NE898 version has full support of 4 serial I/O, Svc, RTC, LCD, keypad, P-Bus, cmos & Eprom ram-disc

CP/M 3+ is supplied ready to run on an NE898 or GM813/GM862 system (state which). 2 discs are supplied together with information on starting up, running & eprom programming.

Prices: CP/M 3 PLUS inc Guide £167, DR Manuals £60, Winchester software £42, all +vat

NEWBURN ELECTRONICS,
Tel 09603-53330

58 Manse Road, Ballycarry,
Carrickfergus. BT38 9LF

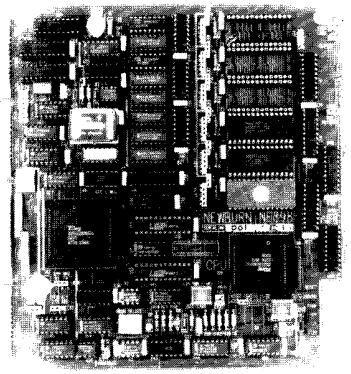






NEWBURN

## NE898 SINGLE BOARD COMPUTER



- Zilog Z180 CPU
- 8MHz, no wait states,
- 256K Bytes Dram
- 6*28 pin Eprom sockets
- 813 Parallel port
- Real time clock.
- On-board Battery
- 5 Serial Ports.
- RS232 or RS422
- Multi-drop comms.
- 6 Counter Timers
- Watchdog Timer
- 2 DMA Channels
- Memory Management (MMU)
- Fully Z80 Compatible.
- 140% faster than Z80
- LCD & Keypad Interface
- Peripheral Bus
- Full 80-Bus operation
- Resident Monitor
- Runs CP/M 2 or 3+
- Stand-Alone operation
- Multi-Processing

The Newburn NE898 CPU is a general purpose single-board computer. The board is completely Z80 compatible and will run all software developed for a Z80, but 138% faster.

All serial & parallel ports can be serviced by polling, interrupt or transparent DMA. Full support of the RS422 multi-drop serial line is provided.

For 80-Bus use, it drives all boards at 8MHz, no wait states. GM802, GM816, PLUTO, GM832, GM833, GM836, NE840, GM841, GM849, GM853, GM863, GM862, NE871,3,4,5,6,7,8, NE889. The NE898 runs standard Gemini CP/M at both 4 & 8MHz without modification. Just remove a GM813 and plug-in an NE898.

The resident monitor allows memory dump, modify, copy & fill, port I/O, console redirection, loading & saving programs between memory, serial link & Cmos ram. The monitor will boot floppy, ram, and winchester discs.

The peripheral bus drives a range of Newburn Industrial I/O modules including digital, analogue and frequency.

The board is supplied with a comprehensive manual, test programs and application examples.

Priced at £439 +vat

NEWBURN ELECTRONICS,
Tel 09603-53330

58 Manse Road, Ballycarry,
Carrickfergus. BT38 9LF