THIS ISSUE ...

Subjects featured in this issue include the language of the HP94, a generous helping of HP28/48 Miscellany, details of the winning entry in the 'Area of a Triangle' challenge, news of the latest HP48 developments and a very thorough review of the Sparcom Personal Information Manager.

HP NEWS ...

The tiny new hard disc drive reported in my last Editorial is rumour no more - the Kittyhawk has arrived! What price that HP announce an HP95 in a slightly larger format with an 80*25 screen, a JEIDA/PCMCIA card slot and a Kittyhawk drive inside the next 6 months?

THE MEMBER PACK ...

You should receive a copy of the 1992 Member pack with this issue of DATAFILE. I hope you find it of use. If you have any comments on its contents, or what should be its contents, then do write a letter to the Editor.

If you pick something out of the list of Articles with a view to purchasing a back number of the journal, please remember that a follow-up article or an errata may well have been published later. Do read as much of the list as you have time for. Those who would wish to have this material sorted alphabetically or sorted by machine, should write to me with an S.A.E (UK Members) or International Reply Coupons (Overseas Members).

THE INCENTIVE SCHEME ...

At the beginning of this year the Committee decided to revise the Incentive Scheme in order to make it more attractive to members. Full details of the new scheme were published in DATAFILE V11N1 on page 19. In accordance with the rules laid down therein, a draw to cover the first half of the year was conducted just a few minutes ago and the lucky winners of a year's subscription to DATAFILE are:-

RON COOK, Member No. 644, and CRAIG FINSETH, Member No. 745.

NEW MEMBERS ...

HPCC welcomes the following new members to the club :-

Desmond FOX	Dartmouth, Nova Scotia	[48SX]
George ZINTILLIS	Ashtead, Surrey	[48SX]
Gary GOSLICKI	Kentish Town, London	[48SX]
M. Didier JOURDAN	Bruay-La-Bussire, France	
Pall GESSTSON	Reykjavik, Iceland	[12][17][19][32][38] [42][48S][95LX]
Peter JERABEK	Wien, Austria	[15][25][28S][41] [48SX]
Philip J. ROGER	Jakarta, Indonesia	[48SX]

Signed:-
Mark Cracknell, Editor,
11 August 1992

THE COLLECT 94 LANGUAGE

Craig A. Finseth (member #745)

(This is article number three of a series of four.)

When a user first picked up an HP-94 running a Collect-94 application, he or she typically saw one of the main menus. There could be up to ten of these menus and they were arranged in a ring. Pressing the right or left arrow key scrolled to the next menu in that direction, and a repeated press would bring the user back to where he or she started.

Each menu had a title line and up to three choices. The title line blinked, alternating between displaying the title itself and the menu number with arrows to the next menus. When the user pressed the left or right arrow key, the menu scrolled to the next position, thus making it feel like a ring.

Ten menus with three choices each gave a maximum of thirty "commands" that users could direct the data collector to do. (Typical applications had from seven to nine such commands.)

When a user selected a choice from the main menu ring, he or she would see a series of screens, some of which called for input and others merely gave information. One of the hallmarks of a Collect-94 application was that it standardized the ways that the screens appeared to you. There was pretty much exactly one way to:

- Display a single line message.
- Display a screen.
- Pause and continue execution.
- Pause and wait for a key to be entered.
- Ask a yes/no question.
- Select from up to three choices.
- Select from an arbitrary list of choices.
- Request arbitrary keyboard input.
- Request keyboard input or a wand scan.

Each of these interactions was visually distinct from all others: an operator could tell at a glance exactly what input was expected.

We also provided two additional mechanisms for unifying the user interface:

First, some of the screens accepted only a few keys (for example, the yes/no question accepted only the y(es) or n(o) keys). If the user pressed another key, the HP-94 simply beeped. A second bad key press would cause a second beep. A third bad key press would cause a "only YES or NO" message to blink on the bottom line, alternating with the prompt. The blinking message mechanism was also made available to the programmer so that all messages would have the same appearance.

Second, the user could, at any time, press a "Return to Menus" key. Pressing this key caused the current "command" to immediately and safely exit. (The "safe" portion required a modicum of discipline on the programmer's part.) The presence of this key encouraged the users to experiment with the device and program as they knew that they could safely get out of any situation.

(Actually, we found that the operators never had any problems with learning the device. It was their managers and the managers of the programming staff that needed to have this "safety valve.")

So how did all of this look to the programmer?

THE LANGUAGE

The Collect-94 language was based on Basic, but one tailored for compilation. We drew from the C language for any syntax and semantics that Basic did not cover. Some specifics:

- There were no line numbers and line breaks were not significant.
- Space, however, was, and you had to space out constructs like "NEXTI".
- The "Let" keyword was required.
- You could have as many lines as you liked between a "Then" and "End" (or "Else"), but you couldn't nest If-statements.
- The data types were integer, fixed-point number, string (up to 32 characters), and True/False.

Odd as it may sound, the Collect language wound up being closer to ANSI standard Basic than many common Basic interpreters. We found that rather amusing.

All variables had to be declared with a both a type and initial value. We came up with this syntax:

```
Variable <name> <value>
```

with the type determined from <value>.

Values and intermediate results were typed and automatically converted among types as required. For example, if you said:

```
StrLen(12)
```

The language converted the number 12 to a string, "12", and return the length of that string. In a similar fashion, if you wanted to concatenate the numbers 12 and 34, you could do:

```
StrCat(12, 34)
```

You could not use the tradition "+" syntax for concatenation as that operation would, of course, convert any string operands to numbers and add them.

We had two types of numbers: integers and fixed-point numbers. Integers were the usual 32-bit numbers. Fixed-point numbers were integers with one or more places after the radix point. Internally, they were stored as 32-bit integers and a one-byte count of decimal places. This type was carried through all arithmetic, so if you added 13 and 4.50 your answer was 17.50. We selected this storage method because it was both space and time efficient and well-suited to the early applications. It also only took me a short time to write and debug the arithmetic operators. Our next release would have switched to HP-71B-format BCD arithmetic. We would have retained the implied display format, however.

The language supported the usual range of file operations. The programmer had eight files (numbered 0 to 7) which always existed and would expand to hold whatever you stored in them (memory permitting). We were strong believers in preserving the integrity of the data, and the interpreter came equipped for both Kermit and Xmodem transfers. For example, to send a file you would say:


```
<status> = Send(<which>, <port>, <protocol>)
```

<which> gave the file number, <port> referenced a declared combination of character format and speed, and <protocol> specified either Kermit or Xmodem. The function returned an empty string on success or a string describing what when wrong. This was all very simple and easy to use.

(In addition, the interpreter used Kermit for loading programs over the serial port. We had quickly got fed up with having to "dump" a program over the serial port and hope that the recipient didn't drop anything. Hence, the _only_ way to load an application through the serial port into the interpreter was to use Kermit. I like to think that this example helped pave the way for the use of Kermit by the HP48SX.)

Aware of our focus on data collection, we provided quite a few functions that were designed to make it easy to write programs:

- We provided Between, Outside, and Within functions which took a low bound, a test value, and an upper bound and indicated whether the test value was between, outside of, or within the bounds (Between used < and <; within used <= and <=).

- We provided OnlyAlpha, OnlyDigit, OnlyAlnum, and OnlyNum functions which determined whether the argument had only alphabetic, digit 0-9, alphanumeric, or number (0-9, ., and -) characters.

- We provided routines to convert strings to all uppercase, all lowercase, from ASCII to EBCDIC, and from EBCDIC to ASCII.

- We provided a routine to check a sting using the "mod 7" algorithm. More of these checking functions would have been provided as needed.

The other major feature of the interpreter was the way that it used format-directed input and output. We had noticed how the HP-75's clock and appointment setting worked, with the prompt being displayed as:

```
##/##/##....
```

and the cursor automatically skipping over the punctuation. We wrote a general-purpose input routine which accepted a format string and handled all details of the user interaction. For example, if you were going to ask for a date and time, you might use the format string:

```
"DDDD BBB DD DD:DD:DD"
```

With a prompt, it might look like:

```
Enter the date/time
  YYYY MMM DD HH:MM:SS
  ____ ___ __ __:__:__
```

As the user entered the response, the program automatically skipped the punctuation. The "D" characters indicated that only digits were valid responses and the "B" characters indicated that only alphabetic characters were valid. As the HP-94 had the alphabetic and digit keys on different keyboards, the program automatically switched into the alphabetic or digit keyboard as appropriate. The cursor was an inverse "#" character if the digit keyboard was active or an inverse "a" character if the alphabetic keyboard was on.

Similar format strings were used to read data from a file or get data from the serial port.

Output also used format strings, but in a different way. The data to be output (into a file or out the serial port) was first turned into a string according to the usual conversions. Again, letters were used to indicate what action to take. However, all letters had to be the same, the number of letters indicated the field width, and the letters themselves indicated whether to center, left justify, right justify, or right justify and zero fill the data. There was also a method to indicate that numbers should have thousands separators placed if required.

Finally, format strings were also used in getting and setting the time. In this case, the letters are used to indicate how to format and place the time in a string or how to extract the time from a string.

[[Editor's Note : For space reasons, I was not able to include the example which followed in Craig's Original Article. Those of you who wish to have a copy should write to me enclosing an S.a.e. (UK members) or International Reply Coupons (Overseas members).]]

MEMBERS' LETTERS

Dear Mark,

High Powered Calculators: Tools, Toys, or simply defunct?

I would like to present my answer to the interesting question posed by Ron Cook and Michael Brown in HP48 Miscellany, V11N4, as to whether the HP48 breed of calculator is justifiable as a tool, or whether it is essentially a hobbyists machine.

First, though, I would like to answer a comment made in an earlier issue of Datafile (was it the editorial?) in which the writer postulated that the role of high powered calculators may be taken over in the near future by hand held computers, which are able to do more, and it seems, faster.

I myself do not believe that this could ever be so, because the role of calculators, however powerful, is quite distinct from the role of computers, however small. A computer is a universal Turing machine first, a specific Turing machine second; while a calculator has its priorities reversed; it is a specific Turing machine first, a universal one second. In other words, a calculator has been designed from the bottom up to be a calculator; its general - and necessary - computing capabilities have been fashioned around this function. A computer has no such luxury when considered as a calculator, and can never be a true calculator - and hence can never be as useful as a true calculator when used as a calculator - because the other uses to which it may be put take at least an equal priority when designing the physical and electronic structure of the machine. So far, HP seem to be the only manufacturer who fully appreciates this.

I think this, and the fact that there is currently a healthy demand for the 48, answers the question of whether there will be a demand for future machines more powerful than the 48, but which retain its essential design philosophy and build upon its pedigree.

Where this demand actually comes from is quite another question, and follows on from the question of the actual role of the machine as far as the users are concerned. I have emphasized its role as a calculator, but what does this actually mean? For the 48, it means that it is a highly specialised personalisable small portable tool, designed primarily to extend a technical person's capabilities (as any tool does) in his or her field, and perhaps to enhance their understanding and command of, and provide greater insight to, fundamental concepts. But it is the means to an end that the calculator provides, and the most significant way it provides this is by being programmable. The fact the people will find enjoyment largely in the means, and may therefore call their involvement a hobby, is neither here nor there. After all, the machine is as I said, a universal Turing machine, and may be put to a huge variety of uses, just as a computer can. In Michael Brown's analogy of a person building a church from matchsticks, the hobbyist is putting the matchsticks to a use that the 'designers', if I may call them that, of the matchstick never envisaged, or at least never intended. This is just as true with 48 hobbyists.

Speaking from the point of view of a student engineer, I personally get huge enjoyment out of my 48. But I dislike the word hobby; I say it is an interest of mine. In my view, hobbyists are only ever concerned in simply getting some kind of satisfaction or enjoyment out of their hobby; while I find that not only can you find enjoyment, but you can gain insight and understanding of a whole new world, and do something useful to boot. And there lies the source of yet more enjoyment and fulfilment! If the matchstick model maker was gaining an understanding of the architectural nuances of church design, or providing a service to architects who wanted scale models of their designs built, then the activity would no longer be a hobby. A hobby is, by its very nature, a self contained world.

The only case that Michael Brown makes for the 48 being not useful (and therefore in his own words justifiable only as a hobby) is that (1) "programming them for long-term or third-party use is neither quick or instinctive," and (2) "many of the applications would be better suited to a desk-top PC." However, he doesn't actually say why this might be so. Firstly, programming for third party use doesn't have to be quick, since generally the only people doing that are going to be commercial software developers, or dedicated enthusiasts. In any case, it is quicker if anything than programming a PC - RPL (system or user) is enormously flexible and provides a multitude of ways of doing any one thing. Secondly, as for being instinctive, well, personally I find RPL more instinctive and much more suited to the type of programming likely to be done on the 48 than any PC languages I know of. Thirdly, if all the applications would be much better on a desk-top PC, why has no-one bothered to write them for a PC? I do not generally play chess on my 48, do word-processing on my 48, program in Fortran on my 48, or draw on my 48; nor do I work out engineering problems with units on my desk-top PC whilst sitting on the bus, or look up someone's phone number on my desk-top PC whilst in a friend's house, or solve an equation with my desk-top PC whilst sitting in a tutorial, or time events and enter the data into a matrix in the lab with my desk-top PC...need I go on?

To summarise, then, I do think that the HP48 type of machine, and more powerful, will remain in demand in the future, and while there will always be 'hobbyists' who program their machines only for fun, they will not be using the machine for what it was essentially designed to be: an extension to oneself to interact with the real world, not an imaginary one. Of course, this is an engineer speaking!

If there is one fault with the HP48 - and it is not one which is particularly relevant to my argument, nor is it an inherent design flaw - it is that its processor is far too slow. Fix that, and I would be happy.....

Yours, Aaron Lambert
(#786, EXUSOC18@UK.AC.HW.CLUST)