Article for Practical Computing : 10/05/81
P.J.Blower 54,Cedar Walk, Hazel Hempstead, Herts. HP3 9ED. 01-953-2030 x3446 : 0442 62480

Prestel & Telesoftware, where do we go from here?

Prestel hasn't done as well as British Telecom had hoped. Telesoftware might be the vital injection - for everybody.

Ceefax and Oracle have now become familiar to most of the population as a method of putting written text on to domestic television screens via the broadcast television signal. They offer from 100 to 800 pages of what their editors think is useful information.

Prestel is the logical extension of the Ceefax/Oracle idea. Instead of picking the information out of the broadcast signal, it comes directly from a computer over the public telephone network with hundreds of thousands of pages at our bidding.

Prestel has not, so far, been as successful as British Telecom had hoped. One answer is for the Prestel service to offer more than just the normal information pages. Telesoftware could make the whole system cost effective to all concerned. It could even become the Prestel lifesaver.

The Prestel service is actively encouraging interest in the telesoftware concept. This is where we, the potential users, have the best chance of saying what we want and laying down standards to make telesoftware simple, cost effective and foolproof.

Who is going to want to use telesoftware? At first, mainly schools, colleges and universities, followed by small business and home computer users. In time, the business users will probably come to the fore with growing confidence in the system, followed closely by the scientific establishment, with home computer users bringing up the rear.

One prime example of telesoftware's potential is in the field of educational programs. A student or teacher could produce a program to MUSE standards and submit it to a MUSE approved Information Provider. In turn, the IP would either buy the copyright or pay royalties to the author. The program is offered as telesoftware pages to anyone who wishes to purchase it - anywhere in the world where international Prestel operates. If the program is good, more and more places of learning will find themselves teaching to a common standard. In the meantime, if someone else can write an improved program, then this will replace the original. A recipe for ever improving programs.

(MUSE: Microcomputer Users in Secondary Education)

Incidentally, the programs should be inexpensive enough to be accessed by many users so as to make the problems of copyright infringement virtually disappear.

Telesoftware is a general term. It does not specify whether, for example, BASIC, PASCAL or machine code programs will be contained in the telesoftware pages, but it must cater for all. Clearly, problems of page formatting and interpretation must be considered.

A comparision of 3 systems

At present, there are three versions of telesoftware on the Prestel system. All three are using the BASIC language as the telesoftware testpiece and since BASIC is not the only language in the world, I shall review them as if it were not. Also, the methods of implementation should allow the receiving software to be reliable, flexible, compact, efficient, inexpensive and as far as possible independent of machine or monitor.

CET (Council for Educational Technology) - page 21143

This organisation invited computer manufacturers, software agencies and representatives of Prestel to discuss the formulation of recommendations for the telesoftware format. I note that no representatives for the users were invited.

Their version has all ineffective spaces removed allowing greater packing density. Spaces are replaced by ¾ on the assumption that some spaces are stripped off and replaced by CRLF (carriage return, linefeed) by the Prestel system. Both British Telecom Research Department and BEC seem to disagree, (their version leaves spaces intact). Even if spaces are stripped off, the frames can be edited to correct this. Space replacement has to be reconverted back to spaces at the receiving end requiring extra software, extra machine time and extra expense. Furthermore, those users who need to examine the Prestel frames directly will find them practically unreadable.

An escape character is defined by CET as character 7C hex (124 decimal) in the Prestel G0 character set. This is not to be confused with the Prestel escape control character (1B hex) which is used to access character subsets.

Escape L is used to denote the end of each statement line and it is this which the receiving software uses as CRLF. Transmitted CRLF's are ignored.

Escape A & escape Z denote start and finish of the telesoftware block. Appended to the block is a block check sequence of up to 3 decimal characters. The block check is based on a calculation using XOR & ASC functions. These are recognisable as BASIC commands. This is unfortunate, as the local calculation will have be performed using either the local BASIC interpreter (of which there are many different versions) or special software for the purpose.

Finally, escape F is used to indicate the end of the program.

CET's version is only really suited to a fast decimal orientated computer running BASIC. The format is unnecessarily complex and expensive to implement. In conclusion, this version should be avoided.

(British Telecom Research Dept.) - page 64025

GEC (General Electric Co.) - page
338252 These organisations have worked together to produce a far more sensible approach. Again extensive use has been made of character 7C hex (124 decimal) which I will call the "escape" character in the absence of anything better.

A header page provides a list of programs and the page numbers where they can be found. It is set out in such a way that the user can type the name of the wanted program, the local software will check the name against the list and automatically call the page where the program starts.

As with the CET, this format uses statement compression. The end of each statement line is terminated with an "escape" character (in place of CRLF) followed immediately by the next statement. Line 23 (of the Prestel page) is reserved for page pointers and page checksum. This line gives the local software the present page number and the next page number followed by a two digit checksum. The checksum uses the 16 characters A to P (41 hex to 50 hex) as an elegantly simple means of providing 256 checksum combinations suitable for all 8 & 16 bit computers. Where there is a checksum mismatch, the current page would be recalled, otherwise the software calls the next page as a unique page number. This method does not use the more usual single digit access to the next frame, (each page being divided into 26 frames (a to z)), but uses direct page access. The virtues are difficult to find and must cause more problems than they solve, particularly if a corrupted page number is received at the Prestel computer.

When these programs are viewed as Prestel pages, they are not easy to read, but, they are much better than the CET version. In my view, it is a reasonable compromise between space saving and readability.

IPC (Practical Computing) - page 45631838

The presentation in this approach is completely different from CET; GEC & ResD. To quote their appletel telesoftware page for Apple Basic:-

@PROGNAME@c/f

PROGNAME is the name of the program and the @ signs delimit it. The c/f means that this is frame c while the whole program occupies frames a to f. The last line of the program is followed by @END@ .

There is no statement compression in this version. Each statement appears on a different line from the rest. However, some statements are longer than 40 characters and have to be continued on the next line. In this case a % sign precedes the continuation line. The % sign instructs the receiving software to ignore itself and the preceding CRLF.

Some BASIC interpreters and compilers use the @ symbol in PRINT @ statements, also @ and % are valid characters inside PRINT statements and therefore, they should not be used as control characters, which is the case here.

To be fair, the appletel telesoftware pages are very easy to read and do demonstrate a very simple method of down loading software from Prestel to remote computers.

A more complete approach

In all three versions, use has been made of valid ASCII characters as telesoftware control codes. Where these codes come naturally in a program, they will cause interpretation errors. A method which overcomes this and makes the program more readable is to use the Prestel control and extension codes. For example, a statement line would begin with Escape (1B hex) D followed by the line number; the rest of the line would be preceded by Escape G. Visually, the line number is shown blue whilst the rest of the line is white. CRLF is inserted when any new escape character combination follows a text line, all transmitted CRLF's are ignored. Similarly, frame checksums would be preceded by Escape A (colour red) and would also serve as end of software frame marker.

The header page, the page that lists all the available programs and their respective page numbers, should be retained. There would be an advantage in displaying the page numbers in different colours, a different colour for each program or listing type. In this way, the receiving software can perform special handling of some program listings or future listing formats, if required to suit particular machines.

Because of the many variety of programs and program types, a preheader page would be useful, classifying programs by type. These may be:- 8 bit machine code, 8 bit assembly programs, 16 bit machine code, 16 bit assembly programs, high level programs (1), high level programs (2), etc.

For example:-

- header page escape codes for 8 bit assembly programs.

Escape A (red)	- CP/M, MP/M
Escape B (green)	- 8080 (Z80)
Escape C (yellow)	- Z80
Escape D (blue)	- 6800
Escape E (magenta)	- 6502
Escape F (cyan)	- PLM/80
Escape G (white)	- Spare

The header page for machine code would follow a similar format.

- header page escape codes for high level programs (1).

Escape A (red)	- BASIC
Escape B (green)	- PASCAL
Escape C (yellow)	- ALGOL
Escape D (blue)	- COBOL
Escape E (magenta)	- CORAL
Escape F (cyan)	- FORTRAN
Escape G (white)	- APL

Each frame of software will need a start of frame character. This will guard against rubbish listings being entered into the receiving computer if an incorrect frame is transmitted or the Prestel system fails. It becomes a convenient marker for the receiving software to start its checksum calculations and it allows the telesoftware content of the frame to start on any line. The 7C hex character (mentioned earlier) is one of the most infrequently used codes in the Prestel system and, as such, is the most suitable character to perform this task.

It is almost essential to have some form of checking ability against wrongly received characters. The checksum process adds the hexadecimal value of the character just received to the previous character value. It goes on doing this for every character until it told to stop. The number remaining is compared with the number set aside at the end of the frame. If the two numbers are not the same then the receiving computer must assume that there was a transmission error and ask for the page to be transmitted again, free of charge. If the two numbers match, then the computer automatically calls for the next frame. The checksum calculation adds up to 256 (8 bits) then starts again from zero. In this way, fast calculation from two characters (A to F) will give error detection in the ratio 1 to 256. The two checksum characters can be combined with either the end of frame marker or the end of program marker.

The end of frame marker (escape A) instructs the software to suspend transfer and check for errors. If the transfer was successful, the software has time to place it wherever it wishes and calls for the next frame by transmitting the hash character to the Prestel computer. The software will then lie idle until character 7C hex is received. If that character is not recieved before 23 CRLF's have been transmitted, then the software must assume a failure and call for a frame repeat, the same procedure as the checksum mismatch.

If, say, five successive frame repeats have been called without success, then the software must assume a system failure and abort. The receiving computer reverts to a simple Prestel terminal.

The end of program marker (escape B) instructs the software to suspend transfer and perform all the tidy up operations as in the end of frame procedure, but instead of calling for the next frame and waiting for character 7C hex, it calls the next frame and then immediately reverts back to the Prestel terminal mode.

The end of frame marker only occupies three character positions at the tail end of the software block and can appear anywhere on the screen. It also allows the last line (line 23) to be used, like the rest, for telesoftware information. Compared with the ResD/BEEB version, this makes extra characters available and more than makes up for any overheads incurred.

Example of escape codes when used in telesoftware frames.

```
Character (7C hex) - start of frame & checksum calculation
Escape A (red)     - end of frame & checksum
Escape B (green)   - end of program & checksum
Escape C (yellow)  - spare
Escape D (blue)    - line number (field 1)
Escape E (magenta) - field 2
Escape F (cyan)    - field 3
Escape G (white)   - text line
```

Codes for extra fields have been included for use inside the text line. These could be used for indicating indents, tabs, two part text lines, space compression, etc.

In high level languanges such as BASIC, the text is preceded by a line number. The total line length would not exceed 128 bytes.

```
[escape D] 10000 [escape G] LET N=20 :FOR Y=2TO15 :FOR X=0TO7
:PRINT N :LET N=N+1 :NEXT X :PRINT :NEXT Y
```

Visually, the line would appear;-

```
 10000 LETN=20:FORY=2TO15:FORX=0TO7:PRIN
TN:LETN=N+1:NEXTX:PRINT:NEXTY 10010 REM
```

The line numbers (10000 & 10010) are coloured blue and the rest coloured white. The escape and the following character (A to G) appear on the screen as a single space.

Assembly listings have two common formats. A short version giving line number followed by mnemonics.

e.g.

```
 0210 LABEL PUSH HL ;Save it
```

Again, the line number is coloured blue and the rest of the line coloured white.

The longer version gives a hex memory location and a hex representation of its contents (OP code) in addition to the short form.

e.g.

```
 1065 E5 0210 LABEL PUSH HL ;Save it
```

Use can now be made of the field 2 and field 3 codes. The field 2 code (escape E) shows the the number 1065 in magenta and indicates to the receiving software that this number is a hex memory location. The field 3 code (escape F) shows E5 in cyan and indicates that E5 is the contents of memory starting at 1065. The rest of the line is exactly as for the short version.

Machine code would have to be dealt with in a special way. It uses all 256 combinations of 8 bits. These codes cannot be transmitted directly by the Prestel system. By representing each byte as two hexadecimal digits, the problem could be overcome. The string would have to start with a memory address to enable direct memory loading.

e.g.

```
 AAAA BBbbBBbbBBbbBBbbBBbbBBbbBBbbBBbb

where AAAA    = memory address
      BB = bb = 1 byte (hexadecimal notation)
```

or more practically:-

```
 1DE0 F5DB04E84720FA7E57280503052318F1
```

In this case, there are two logical ways of assigning escape codes. The first, is to treat the hex listing as an ordinary listing and assign escape 3 (blue) to the memory location and escape 5 (white) to the rest. The second, is to regard the hex listing as the first part of the longer assembly listing where escape E (magenta) was used for memory location and escape F (cyan) for the successive contents. Of the two, I prefer the second, mainly because the escape codes have a clear meaning and allow the receiving software to differentiate between hexadecimal addresses/code and text in any listing.

The receiving software

The receiving computer would have to operate in two modes. First, a normal Prestel terminal. Second, a Prestel terminal capable of displaying Prestel frames, recognising software blocks, checking and resolving errors, disposing of the received software and calling for more software blocks. At this level, the software is not concerned with the type of program being received, only where the software starts and finishes in a frame, the checksum calculation and where it should put the software.

Once the receiving computer has been disconnected from the Prestel system, the software can then process the telesoftware program into a form that will mate with other programs. Where listings are concerned, the software will probably try to simulate the console keyboard and so gain access to the mating programs without having to alter them. (This is a section of software that is monitor dependent, but in tradeoff terms, it is the lesser evil.) Machine code programs, however, need to be translated from the hexadecimal representation back into machine code and be loaded directly into memory.

The major problem now is to create software that will allow the majority of small computer systems to behave as Prestel terminals and receive telesoftware. The early beneficiaries will be those users running the CP/M operating system, Apples and PETs. The software will be in modular form and conceptually very similar to the CP/M structure. The core of the software (i.e. main program and various subroutines) would be completely machine independent. The remaining I/O subroutines, which must communicate with the monitor, keyboard, screen, storage device, etc., would have to be tailored to suit each type of machine.

Hardware considerations

Many small computers have screens of 48 to 64 characters by only 16 lines. The Prestel page is 40 characters by 24 lines. The cheapest way to overcome this problem is to write software that allows split scrolling of the screen. Later on, if the user wishes, he could probably purchase hardware which outputs a 40 x 24 screen in colour. Personally, I would wish to see hardware designed for 80 x 24 (industry standard) for normal computer work, being software switchable to double width character mode (40 x 24) for Prestel use.

A simple modem would be required to convert RS232C/V24/current loop signals used by the computer to tones used by the Prestel system. The computer would have to be able to receive and transmit at different speeds.

Another method would be to connect to the cassette/printer interface of a Prestel adaptor such as Tantel currently priced at £170 + VAT.

The typical cost for a Prestel/Ceefax/Oracle colour receiver is about £1000 to buy and £300 a year to rent. A small computer costs from £400 to £2000. At present price levels for small computer systems, one could expect to pay the following to convert the small computer to receive telesoftware.

Hardware:		
	Prestel telephone modem	£20 - £60
	screen hardware	£100 - £250
	Prestel adaptor	£170 - £300
Software:		
	for terminal only	£20 - £60
	terminal & telesoftware	£30 - £150
	telesoftware only	£20 - £90

If you already have a computer system, the cost to receive telesoftware could be as little as £50 but could set you back as much as £350 or more, depending on the type of system. A standard computer system connected through a Prestel adaptor may prove to be the most versatile choice.

Small computer manufacturers would find themselves at an advantage if they could offer a dual purpose product. Standard screen implementation between manufacturers would be even better.

Conclusions

Sometime in the near future, many people, each with different equipment and needs, will access telesoftware frames. Safety has to be the keyword. In any telesoftware format, control codes are unavoidable. The three commercial versions all use valid ASCII characters as control codes. CET use a painful circumvention process and ResD/GEC & IPC ignore the problem and have to assume these characters do not occur naturally within their programs. The only suitable non ASCII character available for use on the Prestel system is the escape character (1B hex). It has the added advantage of changing text colours and helping to improve readability. When this argument is accepted, then there are few logical choices left of how to format telesoftware frames. I hope this article has shown just that.

Clearly, the whole concept of telesoftware is too important to be reduced to a number of incompatible 'hotch potch' systems. As with any new idea, the setting of standards is paramount. All of us now have the opportunity to put forward our views, and this is the time to put pen to paper and write to this magazine. Over the next few months I hope to hold a running discussion on the whole subject. I also invite British Telecom to join in. With effort, all interests should be satisfied.