IAN McKINNELL

ASCII International
To deal with the specific character sets employed by different languages Lotus has developed LICS (the Lotus International Character Set), in which there is a code value for each foreign character. On a French keyboard, for example, pressing the é key (as in café) might generate a code of 156; this corresponds to 173 in LICS. When this character is to be sent to the screen LICS decodes it back into 156; the printer may understand the code 156, or it may need to be sent a control sequence such as <e>,<ESC>,<BSPACE>,<'>. Text can be saved in native ASCII codes, LICS or the ASCII codes of another country.

numbers is a very complicated task. Secondly, if the translated text is longer than the English — almost invariably the case — the program will need more bytes to store it. This will alter the addresses of all the subsequent code, thus making a nonsense of the loops and subroutine calls.

Another problem is syntax. When an English user wishes to operate on a file the syntax is COMMAND followed by FILENAME. However, this approach is not standard in other European languages. In German, for example, it is logical to enter the filename first, followed by the command. A similar problem is encountered in the method of entering the dates, which has caused problems even in Britain. Both 1-2-3 and Symphony permit the use of dates in formulae to calculate changes in values over time. In America, the normal method of entering the date is Month/Day/Year. However, in Britain and many other parts of Europe the standard date format is Day/Month/Year. Unless a software package can be manipulated to take account of differences in the way commands and data are entered, the result will be at best confusing and at worst complete nonsense.

Software publishers must also consider the different European character sets. The French alphabet includes letters such as é and à, whereas the German and Scandinavian languages include the letter ä in their alphabets. To complicate matters different alphabets place these letters in a different order, creating havoc with any sort routine unable to cater for these differences.

PROGRAM DESIGN

In translating Symphony into the major European languages, Lotus decided that the only reasonable way to set about the problem was to design the program in such a way as to allow for easy translation. This approach was not adopted with the earlier Lotus 1-2-3 package, and as a result the company has had great difficulty in translating this. However, Symphony has been successfully translated into French, German and the Scandinavian languages and the company is working on an Italian version.

In order to overcome the problems of locating the text within the code and trying to squeeze the new words into the available space, Lotus has adopted a modular construction of the program. There are two divisions within the program: the source code containing the program routines, and a data segment containing the text area. This system of isolating the text from the source code is known as *localisation*. Organising the program into this format resolves two of the main difficulties. Firstly, having the text in a separate segment means that extra space can be set aside for any differences in word lengths and text may be extracted from the program much more easily. A utility extracts the text areas from the code; these can then be translated and dropped back into the data segment. As an added bonus, the text can be rearranged within the data section to take account of the differences in syntax required for each language.

While translating the text itself there is a further point to consider. Symphony allows commands to be entered simply by pressing the first character of the command. Thus each command must start with a different letter. Additionally, space limitations mean that in packages where the literal translation is too long a compromise may have to be made.

Problems may also arise when translating between national character sets. We have already seen how different countries have different letters in their alphabets. What makes the problem worse is that there is no internationally agreed standard for the codes. In the days of paper tape, when seven-bit codes were common, the ASCII standard was used almost without exception throughout the world. With the advent of the microcomputer and eight-bit codes, this standardisation broke down as each manufacturer produced its own version of the ASCII 'standard'.

This practice has also been adopted by different countries. In customising the computer keyboard for their own character set, many nations have replaced some familiar English characters with letters of their own. Thus communication between computers configured for different languages is becoming an enormous problem. An ASCII code in German may mean something completely different in Spanish.

This confusion was made even worse for the translators of Symphony by the fact that many of the single keypress commands used by the program were characters such as @ that are absent on non-English keyboards. IBM's own solution to the problem on the PC is to hold down the ALT key and type in the decimal ASCII code on the keypad, thus ruining any advantage gained by having a single keypress command!

Lotus decided that the only way around this obstacle was to develop its own set of codes, known as LICS (Lotus International Character Set). This set of 250 characters contains all the letters used in the main European languages and is held in every copy of Symphony. Translation involves configuring the program so that the code received by a foreign language keyboard is translated into LICS and can thus be understood by the program. To simplify the process of printing characters that do not appear on the keyboard, Lotus has managed to reduce these characters to a single press of the ALT key and a single number between 0 and 9.

The translation process for a business package is a time-consuming and costly operation. Translation takes an average of nine months and can cost anything from $10,000 to $100,000. However, software houses can no longer afford to ignore a market of 300 million people in continental Europe. Despite the investment required to translate a software package, the benefits to both customer and developer alike make it well worth the effort.