user are the weekly receipts for each stand — other figures are then adjusted automatically.

So far, this is all standard spreadsheet material, but what if the owner wishes to put the stands in order of sales, so that the location with the highest sales is at the top of the list? These stands would initially be entered in alphabetical order, but will need re-sorting each week on receipt of the new sales figures. With Lotus 1-2-3 this may be done quickly and easily. The newsstand owner may require a weekly chart that shows how each stand has performed; a sequence of keypresses will allow this information to be retrieved for the spreadsheet/database, displayed in graph form, and printed out.

LOTUS SYMPHONY

Lotus's follow-up to 1-2-3 is called Symphony, and follows the same principle of basing applications on the spreadsheet format. However, Symphony allows the user to divide the screen display into separate windows, each of which focuses on a different part of the spreadsheet. Each window is formatted in a manner appropriate to the information it displays.

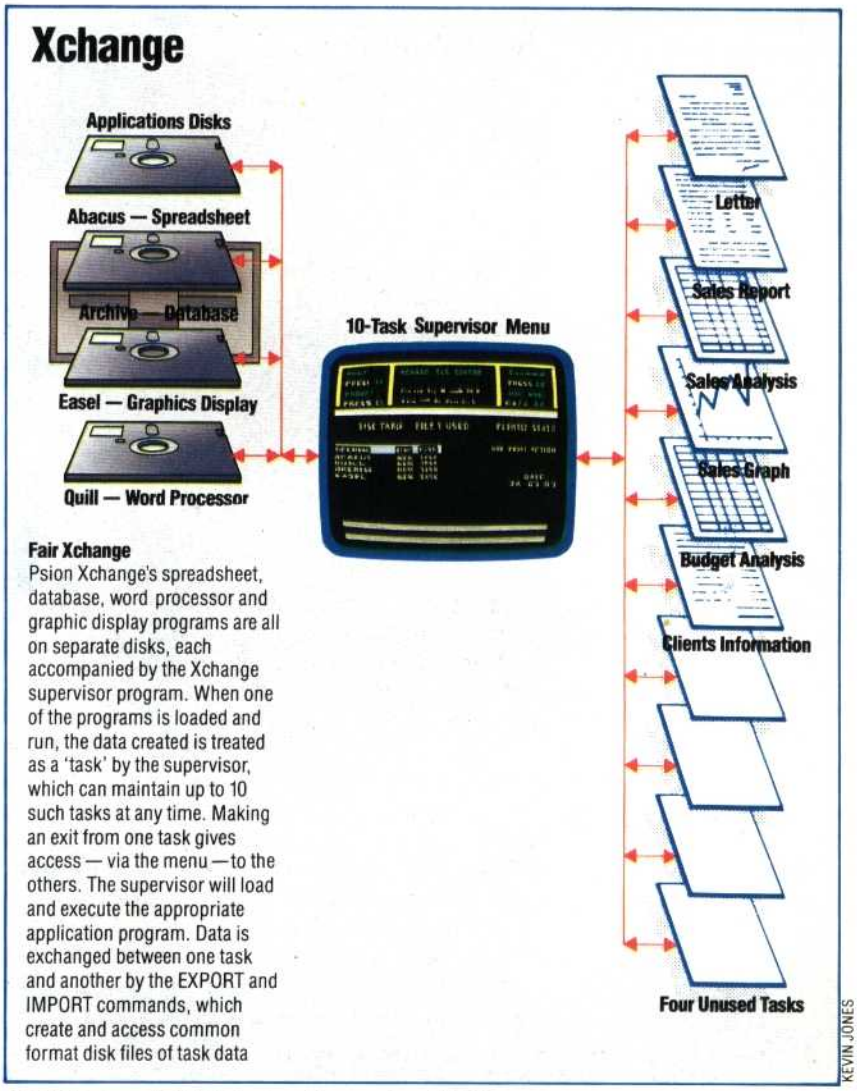
If the information to be displayed is held as text, the window takes the form of a small word processor screen, with margins and tab stops clearly marked. If a graph display is required, the window will show the labelled and scaled axes. Database information is displayed with each entry having its own screen; this looks like a card-index record. So, although Symphony is really an overgrown spreadsheet, it gives the impression of having four major applications all onscreen and working at the same time.

Like 1-2-3, Symphony can 'learn' particular sequences of keystrokes so that the user can automate any operations that are carried out frequently. The small programs that activate the sequence are called 'keyboard macros'. Symphony also includes its own high-level programming language. Programs are stored on the worksheet in the same way as all other data, and have access to all the operations available: so, if you have a task such as an invoicing or stock control system, you can write the program in Symphony's programming language and it will automatically be part of all the applications in the Symphony 'environment'. Once you are familiar with Symphony, you will find it easier to write programs in its command language than it is to use a separate programming language such as BASIC because Symphony already deals with such tasks as drawing graphs or searching for and organising data.

Symphony is just one of several similar systems that are now on the market. Ashton Tate's Framework is a strong competitor — this provides a similar range of functions but hides its underlying data structures to an even greater extent. Both Symphony and Framework are expensive (around £500 each) and require large amounts of memory. Symphony will work with 320 Kbytes of RAM but really requires 512 Kbytes to make the most of its facilities, while Framework needs a minimum of 256 Kbytes. As a result of these demands, the packages will run on 16-bit microcomputers only.

Interestingly, neither Symphony nor Framework requires information or portions of program to be swapped between disks and main memory, as is the case with most business programs. In theory, of course, computer memory continues to become cheaper and cheaper, so it is not unreasonable for software developers to assume that most users will have large amounts available. In practice, however, this is not yet the case and it will be some time before such memory-intensive integration becomes commonplace. Although a program such as Symphony sets new standards of performance, such software is still constrained by hardware limitations — it's only by being such a large, carefully crafted program that Symphony manages the things it does.

An alternative method of providing integrated software has been developed over the last 20 years, and packages that use this method are now starting to appear on the market. In the next instalment we will consider possible future developments in integrated software.



Fair Xchange
Psion Xchange's spreadsheet, database, word processor and graphic display programs are all on separate disks, each accompanied by the Xchange supervisor program. When one of the programs is loaded and run, the data created is treated as a 'task' by the supervisor, which can maintain up to 10 such tasks at any time. Making an exit from one task gives access — via the menu — to the others. The supervisor will load and execute the appropriate application program. Data is exchanged between one task and another by the EXPORT and IMPORT commands, which create and access common format disk files of task data