Application Form

Application generators are similar to automatic program generators, but they have games as well as business applications

Pinball Construction Set
This package is a kind of application generator for games. The user designs the layout and logic for a pinball game using a menu of objects, and various graphically represented tools to fix them on the board

IAN McKINNELL

In the last instalment of THE HOME COMPUTER COURSE, we looked at a type of computer program that would, given a set of specifications by the user, produce a program capable of performing the intended application. Such program generators can be purchased for most business micros, and a few packages are available for home computers, though the type of applications to which they are suited mean that at least one disk drive is mandatory.

A far more common way of generating programs to perform specific requirements involves using packages called 'applications generators'. Unlike program generators, these produce programs that are not free standing, but require the original applications generator package in order to run. Let's consider the creation of a program to handle invoicing using both of these types of generator, in order to highlight their differences.

If we were to use a program generator, the software would first be loaded from disk into the computer. When the user had answered all the questions relating to the files, records, fields, mathematical relationships, screen layouts and printed reports required (i.e. had specified the required applications program), the generator would ask for a blank disk to be inserted into the

disk drive. It would then save the new program it had generated on this second disk. This process could be repeated, and a copy of the invoicing program made for each branch of the company.

By contrast, an applications generator initially seems less satisfactory. When you have completed the specification stage, the necessary routines will be recorded on the same disk as the generator. Alternatively, it could record the program on a separate disk, but it would do this in such a way that the original generator disk will still be needed in order to run the application. Although a single copy of the original package could be used to produce an unlimited number of different applications, it follows that they must all be used in the same physical location as the generator disk. If you want to make your application available to others, they will need to purchase a copy of the generator, as well. Of course, such generators employ several methods of program protection, in order to make unauthorised copying very difficult.

An application generator is really just a sophisticated general purpose program. When you specify your application, you are simply assigning values to a number of important variables within the generator, called 'parameters'. These control the flow of the program, the structure of the data, and layouts for screen and printer. When the application is saved on disk, what is actually being stored is a list of these variables or parameters. This list — sometimes referred to as an 'application module' — is therefore just like a set of instructions that tell the application generator how to perform a particular application.

Some packages take this a stage further, and allow you to specify your application in a form of very high level language (similar to the pseudo-language that we first use when developing a new routine in the Basic Programming course). This listing will be interpreted by the generator; and this may in turn be interpreted by the BASIC interpreter if the generating program is written in BASIC, which creates an interesting case of software hierarchy (see page 66).

It is not uncommon for applications modules to be created and marketed by companies other than the authors of the original generator. For example, dBase II (the most popular of the sophisticated database packages available for microcomputers) can really be regarded as an applications generator, containing modules