PLAN OF ACTION

Our examination of programming techniques has so far concentrated on documentation, and the need to make each section of a program clear and understandable. Here we look at the wider implications of program design and consider the questions that should be asked before any code is written.

Program design is seen by those involved — the designer/programmer and the user — as a grand and formal exercise in applied problem-solving. Unfortunately, the problems to be solved are always assumed to be of the technical programming kind — how to format the screen, how to make this loop faster, where to fit everything into RAM, and so on — whereas the real problems are present from the start of the project, and are usually created at the first meeting of the user and the 'expert'. Users are rarely very clear about the true nature of their problems — they hope the expert will tell them what the problem is and how to solve it — and experts very often think they know the problem and the solution before the user even begins to state it. The result is bad initial communication, leading to an incomplete description of the problem and the user's requirements. Working from this specification is bound to produce an unsatisfactory system that the user may be pressurised into accepting.

For home computing projects the programmer is usually also the designer (or 'systems analyst') and the consumer. This should mean that communication problems are lessened considerably. Nonetheless, as a combined user/designer, you should always make the effort to explain problems, solutions and requirements to yourself as clearly as if you were talking to another person.

Let's consider an imaginary user and his problem: he's a keen aircraft modeller who also owns a cassette-based microcomputer. He wants to store fairly detailed descriptions of materials used in the construction of each model that he makes so that when working on later models he can search his records for previous use of this kind of glue, or that type of joint. What the designer must therefore get from the user is a clear statement of the following:

■ The program function. This can start off as a vague statement of intent such as 'It should store my model records', but it must be refined by the designer's persuasion and interrogation into something more like a requirement specification, such as 'It should store my descriptions of the model and its construction and materials, as typed in at the keyboard, and display them when I type in the model's name, or some aspect of its construction.' This states the user's needs rather more clearly, and points to some of the specific programming tasks involved (storing, searching, indexing, retrieving, etc.).

■ How the program will be used. Some of the physical details of typical usage may be clear from the function description, but these may not be complete. For example, the user may not want the model details displayed on the screen because he works in a shed without a television set. In this case, a 'hard copy' print-out of selected details may be required.

■ What it will look like — input and output formats. The professional programmer will often use pre-printed charts representing the screen to draw each display that the user will see during input/output phases. Such elaborations are not often necessary for home use, although high resolution graphics may be an exception to this. Screen formats are a very important aspect of the *user interface* — the interactions between user and machine — and deserve the sort of close attention and discussion that is sometimes given to more obviously ergonomic aspects of computing, such as the positioning of keyboards and monitors, height of the table, and levels of illumination, etc.

■ How it should be organised — file and program formats. The user may feel that he needs to store at least 100 aircraft descriptions, and anything less will be useless. On the other hand, he may only ever build half a dozen more or less standard models. The size of a program's data files has serious implications for their format and access methods. A serial scan through six model descriptions on cassette taking, say, five minutes may be quite acceptable to the user, whereas waiting for 100 to be searched would be out of the question. A solution might be to put the program and description index file on one tape, and the descriptions themselves on 20 other tapes classified by aircraft type, for example.

The size of the program itself can also become a problem: if the text input section requires a complex text editor, if the program is fat with menus and heavy with significant messages, if the file-handling sections employ complicated searching and indexing routines, then the program may have to be split into several separate programs in order to fit into the available RAM.

■ What it should do — special procedures and calculations. In the model aircraft example, these