and runs on an 8K IBM 1130 with Card Reader, Printer, Disk and Calcomp Plotter.

C. A. THROWER
Area Systems Manager (Engineering)
C. A. W. WELLS
Senior Systems Analyst

Derby Engine Division
Rolls-Royce (1971) Limited
PO Box 31
Derby DE2 8BJ
19 April 1974

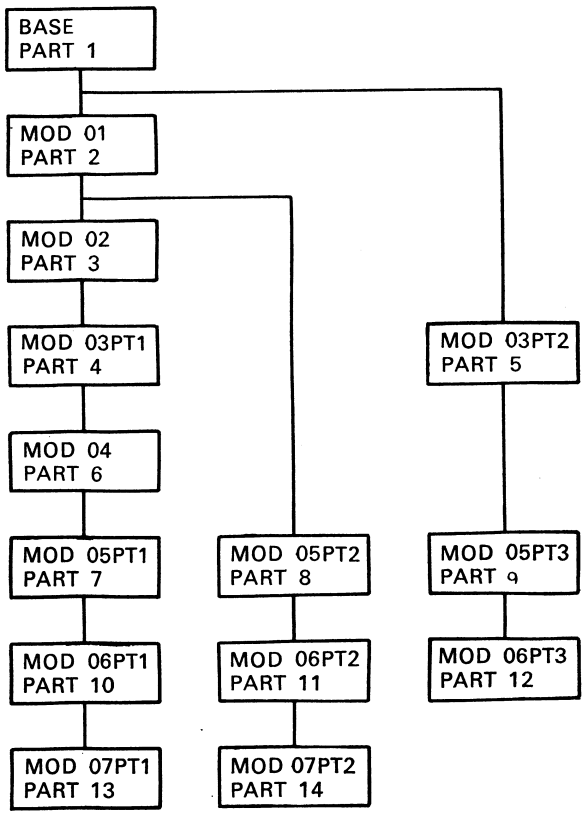

Fig. 1

To the Editor
The Computer Journal

Sir

The validity of D. W. Barron's criticism of Job Control Languages (Discussion, Job Control Languages and Job Control Programs, *The Computer Journal*, August 1974) depends upon just what we mean by 'Job Control'. I question whether JCLs should duplicate the facilities of high-level programming languages—all this IF, THEN, ELSE, GOTO &c—in an area which is, surely, concerned with the definition and allocation of the appropriate physical resources required to run a job. What a job does with these resources is a programming consideration, and as such correctly comes under the responsibility of the Programming Languages, not of the Job Control Languages. Looked at in this way it is quite the wrong approach to suggest that 'job control is just another sort of programming'. The two titles identify distinct tasks which should be understood to accord reasonably with the common language meaning of words.

If we were to completely separate these two tasks, no job would ever have more than one job step. But the need for flexibility has led to the provision of program hook-up facilities at the job control level, because of the specialised nature of the individual programs available to users; to provide complete flexibility in the field would anyway involve the use of a 'command language' which is again a hook-up facility (though this certainly has its place, e.g. in conversational programming systems). Unfortunately it is the gross misuse of the multi-step facilities provided by JCL which leads to the 'thick overgrowth of JCL' which Professor Barron compares with the 'massive pollution of Lake Erie'. Pollution is a consequence of misuse, not an intrinsic fault in things (incidentally it is the JCL we are polluting by the invasion of programming, not the other way round).

This is not to say that JCLs are wrong to provide for program hook-up; rather we should beware of criticising those who have not merely recognised a real practical problem but have provided a convenient if primitive solution which should properly be dealt with elsewhere. But the hook-up features already provided could resolve into a dangerous precedent: by pandering to the demands for yet more programming-like features JCLs would indeed 'lose sight of the indispensable maxim that simple things should remain simple'. Fortunately this has not yet happened; the very awkwardness of IBM's 'COND' parameter, for example, does have the advantage that we will prefer to avoid its use, and so discover the proper solution for ourselves.

We all know that practical programming involves the use of control features—the IF, THEN, ELSE, GOTO &c of the modern high-level language—and Professor Barron's concept of the 'Job Control Program' fits so easily into modern languages by virtue of their subroutine CALL facilities that there is scarcely any need to distinguish between a 'Job Control Program' or any other sort of program. The stand-alone program of the multi-step job becomes just another subroutine in the comprehensive system (incidentally it is at this point that the linkage editor shows its continuing value; it is premature to claim its persistence 'in large systems is to a large extent a triumph of faith over reason'). It is quite straightforward (or ought to be!) to look up the system name and parameter list of any stand-alone program in the relevant literature and hence invoke it as a subroutine, even if for the present this may involve low-level language programming. Successors are nominated by ordinary programming within the invoking control routine and, if necessary, core storage is managed by linkage editor overlay facilities (and/or other facilities such as PL/I 'FETCH' and 'RELEASE' statements). Everyday programming practices such as Update + Compile + Link-Edit have responded as readily to this treatment as have applications systems (for whose benefit Sort utilities may be invoked directly from PL/I and COBOL). So there is no 'necessity' for a program to delete itself and nominate a successor.

JCL should enable programming languages to refer to system resources symbolically, so machine and installation dependent odds-and-ends of information can be filled in later without the need for familiarity with internal program design. Hence a generally available program (e.g. a compiler) may be tuned on-site to make efficient use of the local hardware by the most appropriate choice of secondary storage media, data blocking factors &c. It is a programming design challenge to ensure that only those items which ought to be tuned need to be tuned. JCL should also enable access to computing facilities via the minimum of administrative parameters. Thus a well planned installation will construct JCL procedures to relieve users even of the above tuning details. Stripped of elegant programming-like features in order to avoid an 'obsession with generality' JCL may look like 'an implementation in search of a language' because it can merely assign values to parameters, but should we not rather be judging the implementation upon its conciseness and logical integrity?

Yours faithfully,
J. A. TEMPLEMAN

79 Piccadilly
Bulwell
Nottingham
2 November 1974

To the Editor
The Computer Journal

Sir

Management in the Computer Business

May I, through your columns, congratulate Mr. H. N. Coates on his excellent and lucid article. Whilst under the heading 'Systems Analysis and Design' the rather abrupt transition to the Liaison Section makes one suspect a printers' error or the editorial scissors, the exposition of the functions of this section is most noteworthy. Too few organisations have such a section and, in my experience, it is they who are generally the most successful in receiving the full co-operation of satisfied users.

In this context it is perhaps a pity that Mr. Coates does not stress