INITIAL EXPERIENCE OF THE USE OF INTERPRETIVE SOFTWARE FOR BEAMLINE CONTROL

P Adams, R H C Morgan, W A Smith, F Atchison, J C Kerr

Rutherford High Energy Laboratory, Chilton, Didcot, Berkshire, England
(presented by J. Dickson)

Abstract

A computer control system using a high level language interpreter has recently been installed on the K9 beamline at the Rutherford Laboratory. This paper describes the initial experience gained with this system from the users' point of view, with a description of the facilities that the system provides. The flexibility of the system is discussed with reference to the various modes of running the beamline. Finally there is a description of the performance of the interpreter itself with particular reference to program execution time.

Introduction

This paper deals with some of the software aspects of the K9 computer-assisted beamline at the Rutherford Laboratory. The beamline and the associated computer hardware are described in another paper[4] being presented at this Conference, and therefore they will not be dealt with here. The software system which makes use of a real time interpreter (RTI) to execute user programs has also been described elsewhere[1,2,3] so only the basic points will be summarised. The purpose of this paper is to present the initial experience gained by users of the system and to make some general comments on its performance comparing this approach with the more conventional assembly language system. We would like to point out that the system being described does, by no means, represent the ultimate that can be achieved using the interpretive approach, nor have we explored more than a small fraction of possible lines of development for such systems. Our aim was to produce a basic working system which would give us experience in implementing and using an interpreter for this type of application. It is our experience so far that we now describe.

Initial Experience

It is generally agreed that a system which allows programs to be written in a high-level language is easier to use than an 'assembly-only' system, because program development time is greatly reduced. However, this advantage is only apparent to the programmer. With an interpretive system the user and the programmer can be the same person (although not necessarily so). In fact one of the aims of the system is to allow the physicists running the beamline to use the computer to compute, a rare facility on most beamline computers[5]. With these facilities the system can be developed on-line so that, although initially only single control routines are available (e.g. magnet current scan, collimator set, etc.), more and more complex setting-up programs can be developed until parts of the tuning process are completely automated. This continuing development is entirely in the hands of the users.

Initially we envisaged that the basic sub-routines, such as that to set a collimator slit to a given position, would be programmed in assembly language because it involves a reasonable amount of 'bit-checking' to test status, etc. Because of this, the system allows machine code sub-routines to be called from disk by a RTI routine. In fact, this facility has not yet been used; the basic sub-routines have been written in RTI. This has brought home to us one simple fact - the logic of these basic control routines is no less complex when programmed in a high-level language. The problem of analysis and flow-charting still remains. This can easily be forgotten when one has a conversational programming system at one's finger-tips. The time taken to produce a program is mainly governed by the 'thinking' time. The time taken to code it and type it into the computer is comparatively insignificant. For example, the program to set collimators took five days to produce the final working routine. The coding time involved was about 45 minutes!

Another noticeable feature of these routines is that they contain more diagnostic checks of the state of the hardware than the corresponding assembly language routines. Not only are checks made to detect errors but, if at all possible, the programs attempt to recover or re-adjust the hard-ware to obtain the desired effects. This extra sophistication is undoubtedly due to the fact that the overall logic of a program is much easier to follow in a high-level language, and also to the fact that such additions increase the size of the final routine very little.

The very powerful debugging facilities that an interpreter offers are extremely useful. Errors in the program are trapped by the interpreter and reported to the operator. Both the program and the variables can be examined from the keyboard. The program can be partially executed and also executed in 'slow motion'. Errors in the text can be corrected immediately and on-line using the edit features of the interpreter. Since control of the computer resides with the interpreter and not with the user programs it is very rare for a program error to cause the system to 'crash'.

Basic Software System[1,2,3]

The computer software can be divided into four main parts:

a. System sub-routines, interrupt handler and peripheral drivers.

b. Alarm scan program.

c. Interpreter and executive.

d. High level language (RTI) programs (user programs).

The first three groups are written in assembly language and are core-resident. The alarm scan routine (b) runs on interrupt every accelerator cycle, that is, about every 3 seconds, and checks that the beamline hardware is set and functioning correctly.

The RTI programs, written by the users, are stored on disk in source form and can be called from the keyboard or by a program. They are executed interpretively. The user communicates with the interpreter via the keyboard from which he is able to write or modify his programs or ask for programs to be executed. Other facilities available to the user include recursive functions to perform CAMAC operations and a table that maps the hardware configuration and characteristics onto the beamline. This enables the user to write the names of the beamline elements in his program or data, and the appropriate addressing and checking information is automatically acquired from the table.

Performance

One could discuss under this heading many of the items mentioned in the last section, but the main point of interest is that of execution speed. Slow execution time is the price that must be paid for all the advantageous features that an interpreter offers. In the data processing field execution speed is of paramount importance, but in many control applications slow execution can be tolerated. After all, the speed of a control computer is a relative concept. It is measured in relation to the response time of the devices it controls, and also to the response time of the operator.

The basic rate of the PDP-8 implementation of the interpreter is about 20 ms per statement. This is greatly influenced by the amount of mathematics involved in the statement because the system uses floating point arithmetic realized entirely through sub-routines. However, the position of the variables in the variable stack has little influence on the execution time since the time taken to search this stack is quite short.

The other basic time quantity that affects the running time of a program is the time taken to overlay one program segment with another. This occurs, for instance, when a routine calls a disk-resident sub-routine. The average time to perform the overlay is 80 ms, with an absolute maximum of 160 ms. These times are almost entirely governed by the characteristics of the type of disk drive used. (The DEC DF32 fixed-head disk).

So far the slow execution speed of the interpreter has caused us no embarrassment. Even with this system the computer spends most of its time waiting for the external devices.

Future Developments

The possibility of using a system similar to that of K9 for counter beam experiments is being investigated. This will entail translating the interpreter for use on a Honeywell DDP-516. It is also intended to translate the interpreter for the IBM 1800 used to control the PS here at CERN[6].

Recently much interest has been shown in the use of remote access terminals to computer systems and an interpreter, such as RTI, offers a simple and effective way of achieving this on a beamline computer. One experiment that has been performed with the K9 system was to operate the computer over the switched telephone network. This was done by using a CAMAC modem module, developed at RHEL, and an acoustic coupler. The computer was operated successfully and in spite of occasional periods of high transmission error rates during three days of the trial the computer system did not 'crash'. This is yet another indication of the high degree of protection that an interpreter provides.

The possible applications for interpreters are many, both inside and outside the fields of accelerator and beamline control.

References

1) A Real-Time Interpreter for Computer Control. P Adams. Ruth. Lab. Report RHEL/R207 Oct. 70.

2) Beamline Computer Control by Interpreter. P Adams. Submitted to 1971 Particle Accel. Conf. Chicago, March 1971.

3) Interpretive Software for Computer Control. P Adams. Presented at the European Seminar on Real-Time Programming, AERE, Harwell, April 1971.

4) Hardware for a Computer Control System for the 1.5m Bubble Chamber Beamline at Nimrod. R H C Morgan et al, for presentation at this Conference.

5) On-line Experiments in High Energy Physics. B C Levrat. Proc. 1970 CERN Computing and Data Processing School. CERN 71-6, p.333.

6) G Shering, CERN PS Injector. Private communication.