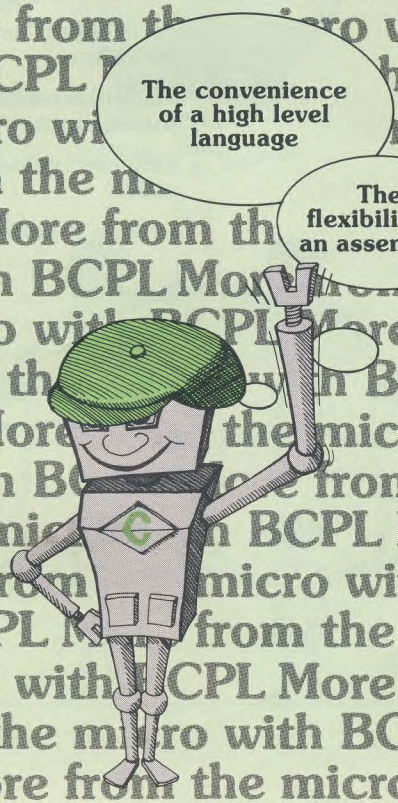


BCPL – large systems for small computers

This implementation of BCPL for the BBC Microcomputer makes use of a compact interpretive code, CINTCODE.

CINTCODE is a major advance in software for microcomputers, allowing larger and more effective systems to be developed and maintained. It has been carefully designed to meet the needs of system builders on 8-bit microcomputers.

BCPL on the BBC Microcomputer is a full and extended implementation of the popular systems programming language, BCPL. It is very effective for the construction of large systems using microcomputers, but it is also very convenient for developing smaller programs.

Support for large systems

CINTCODE gives dramatic reductions in the space required for the program. This is a major factor in permitting larger systems on small computers. However CINTCODE combines many other features which are important in the development of larger systems. These include:

– A modular structure both of procedures and compilation units.

– The ability to load and unload parts of the program as required, providing simple and effective overlaying.

– Linkage on loading, avoiding an offline linkage stage.

– Linkage to machine code.

– Multitasking on the BBC Microcomputer.

– A large number of general-purpose procedures.

– A flexible method of free store management.

– Excellent debugging facilities on the final code.

– Files in store to minimise disc transfers.

More from the micro

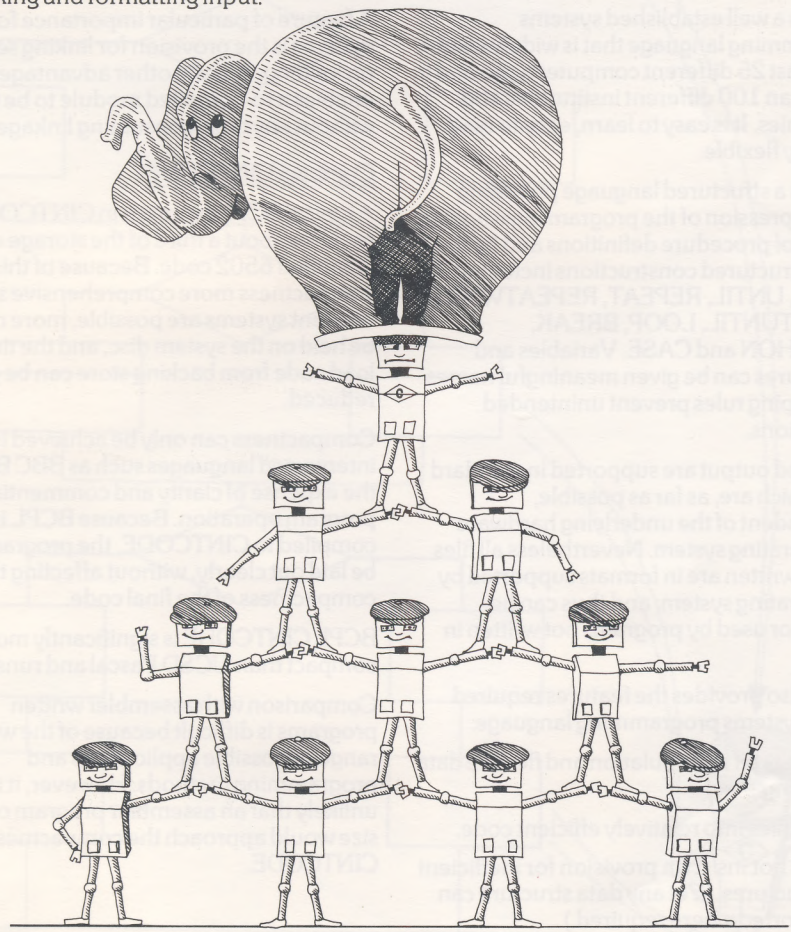
Rapid development of small programs

Many of the features which are so useful in the creation of large systems can also be very helpful in the rapid development of smaller utilities.

The large number of standard tested procedures are particularly convenient with input, output, files and text. For example the procedure RDARGS is a very powerful tool for checking and formatting input.

The comprehensive test aids often avoid any need to develop a special test environment.

A program is automatically linked to the standard procedure library whenever it is run. Other special libraries can be loaded when required. This avoids storage of the same code again and again with each program.



Support for large systems

BCPL

BCPL was developed by Dr Martin Richards, starting in 1966 at the Massachusetts Institute of Technology, and later at the University of Cambridge Computer Laboratory. BCPL pioneered some of the concepts of portable languages. After many years of use it has become standardised in a developed form, with very considerable commonality between the implementations on many different computers.

BCPL is a well established systems programming language that is widely used on at least 25 different computer types and in more than 100 different institutions and companies. It is easy to learn, easy to read, and very flexible.

BCPL is a structured language providing clear expression of the program logic both by the use of procedure definitions and by the use of structured constructions including IF, WHILE, UNTIL, REPEAT, REPEATWHILE, REPEATUNTIL, LOOP, BREAK, SWITCHON and CASE. Variables and procedures can be given meaningful names and scoping rules prevent unintended interactions.

Input and output are supported in standard ways which are, as far as possible, independent of the underlying hardware and operating system. Nevertheless all files read or written are in formats supported by the operating system, and thus can be created or used by programs not written in BCPL.

BCPL also provides the features required from a systems programming language:

– It permits bit manipulation and flexible data addressing.

– It compiles into relatively efficient code.

– It does not insist on provision for inefficient data structures. (Yet any data structure can be supported where required.)

– It provides for easy conversion between different representations of data (for example from characters to integers).

Although the BCPL compiler provides considerable optimisation of the instructions used to execute a program, they are designed to follow the program logic as expressed by the programmer. Thus they do not introduce hidden processing overheads, nor do they cause unexpected reorganisations of the order of execution.

A feature of particular importance for small systems is the provision for linking segments on loading. Among other advantages this enables a re-compiled module to be tested without any time-consuming linkage stage.

Compact CINTCODE

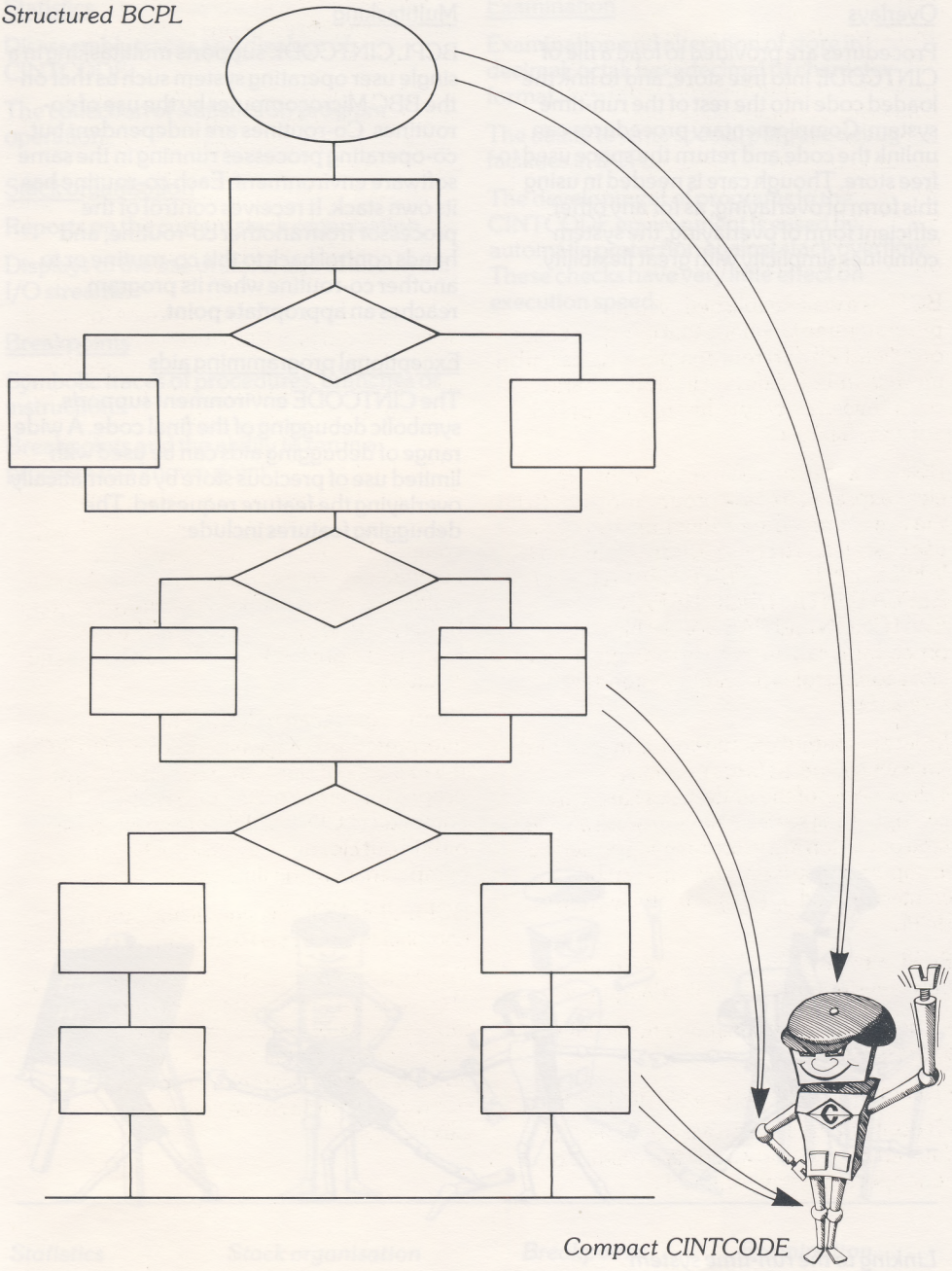
A typical BCPL program in CINTCODE requires about a third of the storage of fully compiled 6502 code. Because of this compactness more comprehensive store resident systems are possible, more code can be held on the system disc, and the time to load code from backing store can be greatly reduced.

Compactness can only be achieved in interpreted languages such as BBC BASIC at the expense of clarity and commenting of program operation. Because BCPL is compiled to CINTCODE, the program can be laid out clearly, without affecting the compactness of the final code.

BCPL CINTCODE is significantly more compact than UCSD Pascal and runs faster.

Comparison with assembler written programs is difficult because of the wide range of possible applications and programming methods. However, it is unlikely that an assembler program of any size would approach the compactness of CINTCODE.

Structured BCPL



Compact CINTCODE

More from the micro

Overlays

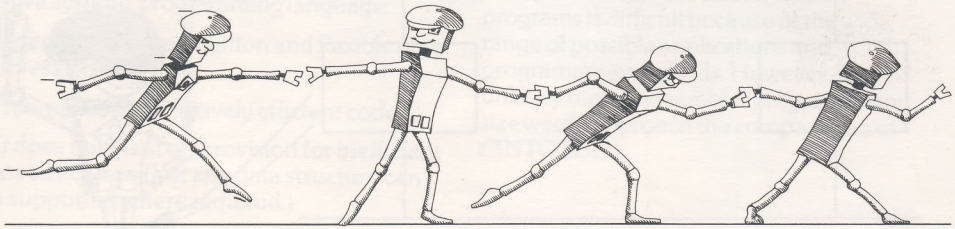
Procedures are provided to load a file of CINTCODE into free store, and to link the loaded code into the rest of the run-time system. Complementary procedures can unlink the code and return the space used to free store. Though care is needed in using this form of overlaying, as for any other efficient form of overlaying, the system combines simplicity with great flexibility.

Multitasking

BCPL CINTCODE supports multitasking in a single user operating system such as that on the BBC Microcomputer by the use of co-routines. Co-routines are independent but co-operating processes running in the same software environment. Each co-routine has its own stack. It receives control of the processor from another co-routine, and hands control back to this co-routine or to another co-routine when its program reaches an appropriate point.

Exceptional programming aids

The CINTCODE environment supports symbolic debugging of the final code. A wide range of debugging aids can be used with limited use of precious store by automatically overlaying the feature requested. The debugging features include:



Linking to the run-time system

More from the micro

Statistics

Disassembly traces and displays of CINTCODE.

The collection of statistics on program operation.

Stack organisation

Reports on the current stack organisation.

Displays of the use of store, and of all current I/O streams.

Breakpoints

Symbolic traces of procedures, branches or instructions.

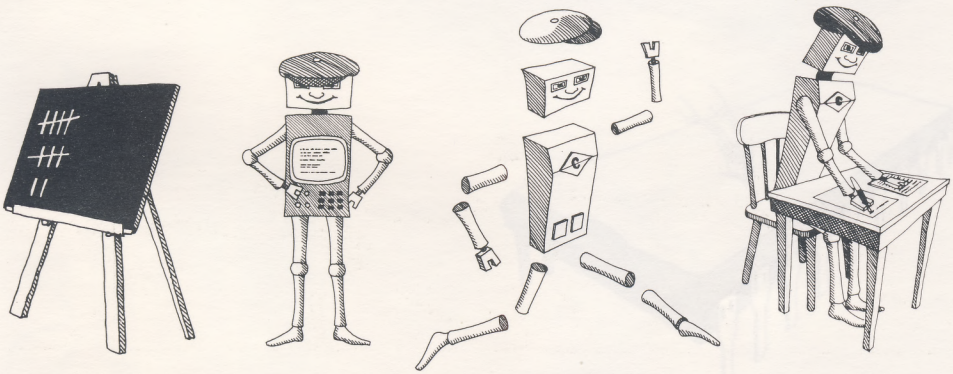
Breakpoints and the ability to rerun a program to a known point.

Examination

Examination and alteration of store in decimal, octal, hexadecimal or character format.

The ability to write special purpose debug facilities in BCPL.

The development of programs in the CINTCODE environment is eased by automatic protection against stack overflow. These checks have very little effect on execution speed.

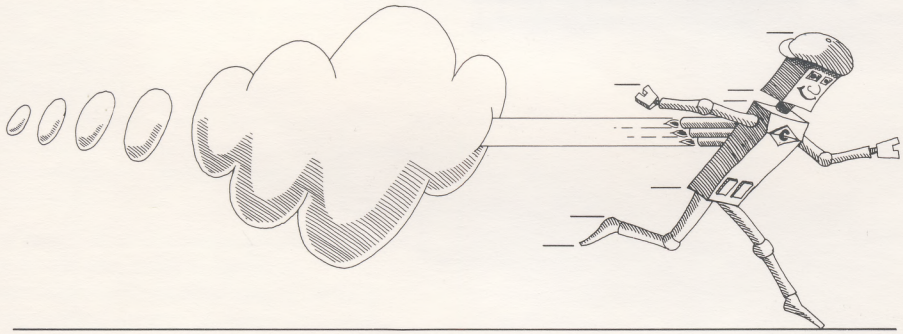


Statistics *Stack organisation* *Breakpoints* *Examination*

Speed

Because CINTCODE instructions are powerful and because the instructions to execute have been chosen by an optimising compiler, the speed of execution is many times higher than that of the BASIC interpreter working on source text. BCPL is therefore suitable for most single user applications on micros. Indeed in many cases a BCPL system will run faster than a machine code system because less code has to be loaded from a slow device. When input or output speeds are important, the system provides procedures for rapid transfer of data between store and disc.

However, there will be cases in which the greater processing speed provided by native machine code is required. Normally only a small proportion of the application requires the maximum speed. The BCPL system can include machine code subroutines as required.



Speedy CINTCODE

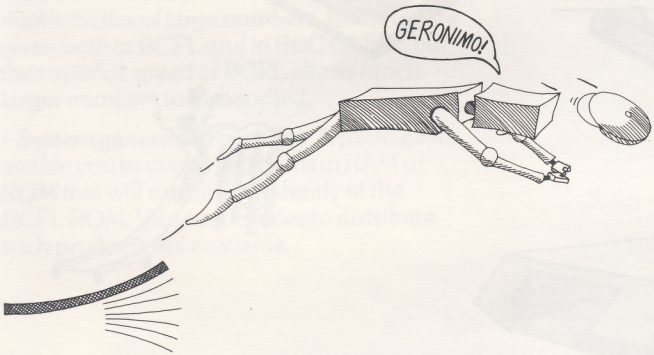
More from the micro

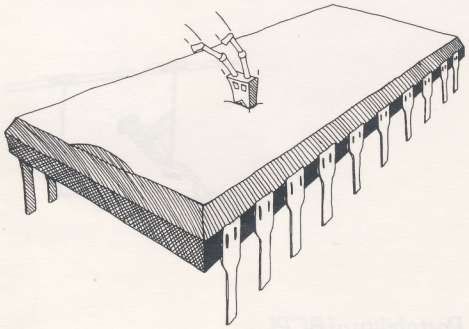
Calculations

BCPL does not aim to be an ideal language for scientific calculations, and is designed to use integer arithmetic. Greater precision is often best provided on micros by fixed point representations using two or more words for each number. Support for such formats is eased by a procedure MULDIV.

Read Only Memory

Provided a few minor restrictions are observed, BCPL CINTCODE segments can be held and executed on Read Only Memory (ROM).





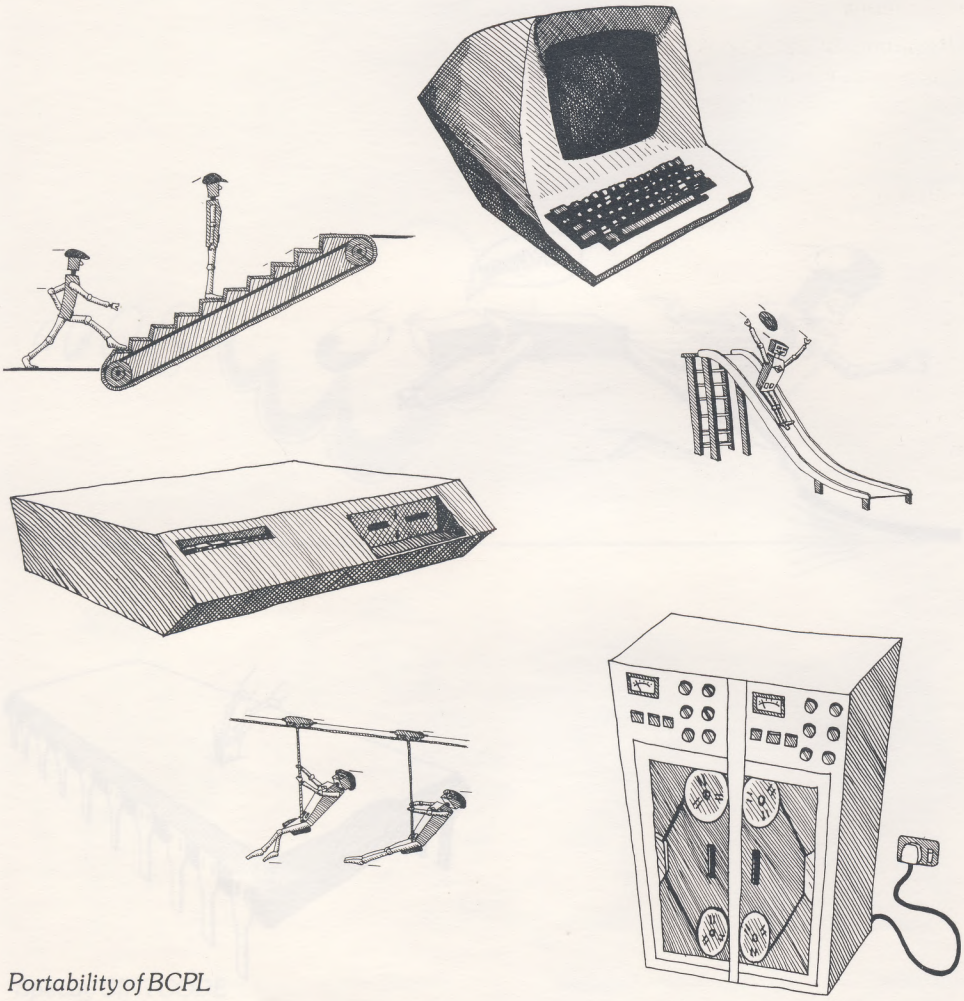
More from the micro

Portability

Ease of transfer of software to new systems is often very desirable. It is particularly important with microcomputer systems because of the variety of microcomputers and the rapid changes of technology.

BCPL has been designed to be a portable and standardised language, with very few of the incompatibilities found between different versions of BASIC.

BCPL CINTCODE is intended to have an even greater level of compatibility. As well as this implementation for the BBC Microcomputer, a number of other versions already exist on other micros. In many cases these will even permit the compiled CINTCODE to be moved from one type of computer to another without change.



Portability of BCPL

Further products for use with BCPL

A number of additional products are being brought out by Acornsoft in the near future. These include:

– BCPL calculations package. This consists of a set of routines which handle fixed point arithmetic, and provide floating point arithmetic and I/O within BCPL. In addition, it contains a set of trigonometric and other fast routines, intended for use in computer graphics. Together these sets of calculations meet the requirements of a wide range of applications.

– *Beginning BCPL*. A book for complete beginners by Paul Martin. This covers all aspects of programming in BCPL.

– *Multiple precision arithmetic*. A book of routines to handle integers of arbitrary size, written by Robert Macmillan. These routines will be of value to anyone interested in the mathematics of large numbers. Routines are given both in BCPL and in BBC BASIC, but the superior speed of BCPL allows much larger numbers to be handled.

– System generation package. A package to enable you to create programs in RAM or ROM that will run independently of the BCPL ROM. Various licences to distribute such products are available.

A brief specification

Features

Binary I/O, compilation to a very compact interpreted code, fast disc I/O, free store allocation, linkage to machine code, loading and unloading CINTCODE, multitasking by co-routines, symbolic testing aids including breakpoints, traces, profiling, displays of store, stacks, and I/O in progress, temporary files in store.

The BCPL language

A standard implementation with the '%' character operator and with optional compilation.

Procedures

ABORT, ADVAL, APTOVEC, BACKMOVE, BACKMVBY, CALL, CALLBYTE, CALLCO, CAPCH, COMPCH, COMPSTRING, COWAIT, CREATECO, DELETECO, DELFILE, DELXFILE, ENDPROG, ENDREAD, ENDWRITE, ENVELOPE, ERRORMSG, EXTSFILE, FILETOVEC, FINDARG, FINDINPUT, FINDOUTPUT, FINDXINPUT, FINDXOUTPUT, FREEVEC, FSTYPE, GETBYTE, GETVEC, GLOBIN, GLOBUNIN, INPUT, LEVEL, LOADSEG, LONGJUMP, MAXVEC, MODE, MOVE, MOVEBYTE, MULDIV, NEWLINE, NEWPAGE, OPSYS, OUTPUT, PACKSTRING, PUTBYTE, RANDOM, RDARGS, RDBIN, RDCH, RDITEM, READ, READN, READVEC, READWORDS, RENAME, RESUMECO, RUNPROG, SAVE, SAVEVEC, SELECTINPUT, SELECTOUTPUT, SHUFFLE, SOUND, SPLIT, STACKSIZE, START, STARTINIT, STOP, TESTFLAGS, TESTSTR, TIME, TRAP, UNLOADSEG, UNPACKSTRING, UNRDCH, VDU, VDUINFO, VECTOFILE, WRBIN, WRCH, WRITEA, WRITEBA, WRITED, WRITEDB, WRITEF, WRITEHEX, WRITEN, WRITEOCT, WRITES, WRITET, WRITEU, WRITEWORDS.

A brief specification

Utilities

BCPL	The compiler
ED	The editor
TED	A small editor
EX	To execute a command file
JOIN	To join files
JOINCIN	To join CINTCODE segments
NEEDCIN	Extract sections from a library
RAS	The relocatable assembler

The distributed system

The system is issued with the BCPL run-time system on ROM, a disc of utilities, suitable for 40 track or double sided 80 track drives and a User Manual of 450 pages. It also includes a number of examples of BCPL source text.

Computer requirements

The BCPL system requires a BBC Microcomputer Model B with disc interface. Econet may also be used – please contact Acornsoft for advice.

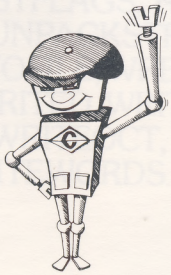
BCPL can also be used to considerable advantage on the 6502 second processor. The ROM automatically reconfigures to use the extra memory available.

Developed programs may be used without the ROM present, using the RAM based run-time system. This requires the use of the system generation package described above. Contact Acornsoft for details of licensing agreements.

About the authors

BCPL was developed for the BBC Microcomputer by Richards Computer Products Ltd in conjunction with Acornsoft. Richards Computer Products Ltd was founded in 1980 by John Richards and specialises in the production of system software and other aids for the development of computer systems, particularly for microcomputers.

BCPL CINTCODE has been developed in consultation with the originator of BCPL – Dr Martin Richards.



Acornsoft Limited, 4a Market Hill, Cambridge CB2 3NJ, England
Telephone (0223) 316039