Operation

This example shows the dialogue used with the machine code converter to convert a G·code program called GPROG into a machine code program called MPROG.

```
>*RUN "CONVERT"
MACHINE CODE CONVERTER
G-code filename:GPROG
Library filename:L3&l900
Machine code filename:MPROG
Sideways ROM format (Y/N):N
(Program is converted)
Execute program (Y /N):Y
(Program is executed)
BASIC
>
```

The library filename is formed from the level and execution address required (in the example, a level 3 library with an execution address of &1900 is used).

Sideways ROMs

This example shows the dialogue used with the machine code converter to convert a G·code program called GPROG into a sideways ROM image called SIDEPRG.

```
>*RUN "CONVERT"
MACHINE CODE CONVERTER
G-code filename:GPROG
Library filename:L3&8100
Machine code filename:SIDEPRG
Sideways ROM format (YIN): Y
Invocation command:BEGIN
Title string:EXAMPLE ROM
HELP message:Sample ROM 1.00
Variable address:&l900
(Program is converted)
>
```

Notice that a library execution address of&8100 is used. Once the ROM is installed, it can be executed by typing **'*BEGIN'**. The operating system will then print the title string and jump into the ROM. **'*HELP'** (with no parameters) will include the string 'Sample ROM 1.00'.

Relocator

Operation

This example shows how to use the Relocator to change the execution address of a G·code program to &E00. The original program is called GPROG and the new version is called GPROGL.

```
>*RUN "RELOC" G-CODE RELOCATOR
Object filename:GPROG
New object filename:GPROGL
New address:&E00
(Program is converted)
>
```

Library generator

Operation

This example shows how to create a level 2 library with an execution address of &3000. The library will be stored on drive 1.

```
>CHAIN "LIBGEN"
LIBRARY GENERATOR
Execution address:&3000
Level:2
Destination drive: 1
(Library is constructed)
>
```

Accelerator

The BBC BASIC Compiler

Quick Reference Card

Computer Concepts

G-code Compiler

Restrictions

LOMEM, PAGE, TOP, EVAL, TRACE, assembly language - Not implemented.
RND - Returns result as an integer, except when used with a literal numeric argument of zero or one, when the result is real. For example, **'RND(A%)'** is an integer (even if **A%** is zero or one), and **'RND(I)'** is real.
CHAIN - Refers to another G-code program.
FN -When calling a user defined function, the result is assumed to be real unless a '%' or a '$' is appended to the name. For example, a function called **'WORD'** ending with **'=ANS$'** must be called with **'FNWORD$'**.

Operation

This example shows the dialogue used with Accelerator to compile a program stored on disc/ cassette called MYPROG and store the compiled version in a file called GPROG, as well as leaving it in memory.

```
>* ACCELERATOR
(C) 1985 Computer Concepts
Source filename:MYPROG Object
filename:GPROG
3450
3450
3450
Execute program (Y IN):N
BASIC
>
```

If the source filename is omitted, the compiler attempts to compile the program currently in memory at the default **PAGE** address. If the object filename is omitted, the compiler will not save the G-code program it generates.
If the **'Execute'** prompt is answered with 'V' or carriage return, the G-code interpreter is entered directly.

G-code Interpreter

Operation

This example shows the dialogue used with Accelerator to execute a G-code program stored on disc/cassette called GPROG. .

```
>*GRUN
G-CODE INTERPRETER
Object filename:GPROG (Program is
executed) Re-execute (Y IN):N BASIC
>
```

If the object filename is omitted, the G-code interpreter attempts to run the G-code program stored in memory above the BASIC program at the default **PAGE** address.
If the **'Re-execute'** prompt is answered with a 'V', the program is re-executed.

Machine code converter

Restrictions

Real variables, real arithmetic, **SIN, COS, TAN, ASN, ACS, ATN, DEG, RAD, EXP, LN, LOG, INT, SQR, PI, VAL, LOMEM, PAGE, TOP, EVAL, TRACE**, assembly language - Not implemented.
GOTO and **GOSUB** cannot be used with computed destinations. Integers are stored in 16 bits, with a range of -32, 768 to 32,767. **RND** cannot be used with a literal numeric argument of zero or one. A '%' or a '$' is required at the end of each user defined function name to indicate the desired type of the result. This is only required where the function is called, and will cause an error if used in the definition.
CHAIN refers to another machine code program.
Escape is not automatically trapped.
Line numbers are not included in error reports.

Libraries

Level 1 libraries are about 3K long and support the following routines:
***commands, ADS, ADVAL, CALL;CLEAR, CLG, COLOUR, DIM, DRAW, END, ENVELOPE, ERR, FALSE, FOR, GCOL, GET, GOSUB, GOTO, HIMEM, IF, INKEY, LET, MODE, MOVE, NEXT, ON ERROR, PLOT, POINT, POS, PRINT, REPEAT, REPORT, RND, SGN, SOUND, SPC, STOP, TAB, TIME, TRUE, UNTIL, USR, VDU, . VPOS, WIDTH, I,?, *, -, +,DIV,MOD,EOR,AND, OR, NOT, < ,> , < > ,<=** and **>=.** (N.B. Strings are not supported).

Level 2 libraries are about 4.5K long and support everything in level 1 with the addition of the following routines:
ASC, BGET~, BPUT~, CHR$, CLOSE~, COUNT, EOF ~, EXT~, GET$, INKEY$, INPUT, LEFT$, LEN, MID$, OPENIN, OPENOUT, OPENUP, OSCU, PTR ~ , RIGHT$, STR$, STR$- and **VAL.** (N .B. Strings are supported).

Level 3 libraries are about 6K long and support everything in levels 1 and 2, with the following additions:
INSTR, STRING$, READ, DATA, RESTORE, PRINT$, INPUT$, PROC, FN, DEF, ENDPROC and **LOCAL.**