Version 3, November 2011 **E3X4**

Programming and software for the Elliott 800 series and 503 computers.

1. Early 802 and 803 software.

Until the end of the 1950s all programming for Elliott computers was carried out in machine code or assembler. In this respect, Elliotts lagged behind many other computer manufacturers. Roger Cook, who joined the company in 1954 and became, in due course, Manager of Elliott's Scientific Computing Division and later Assistant General Manager of the Computing group remembers that: "during the 1950s there was no operating system as such. A set of library routines and programs were provided but all customers' programming was in machine code written with the minimum of mnemonic naming, translated and loaded via a simple program" – (see reference 4). Then in about 1960 Elliott Autocode was introduced.

2. Elliott Autocode for the 803 computers .

The Autocode allows simple arithmetic expressions to be written in algebraic form. The following description and examples are taken from the 803 FACTS booklet (reference 14).

In these examples:

A, B, C and D represent floating-point variables;
I, J, K and L represent integer variables;
l, m and n represent positive integer constants;
p, q and r represent any integer constants;
x, y and z represent floating-point constants.
@K represents a label, K, corresponding to a memory address

Any variable except the one before the = sign may be replaced by a constant.

Allowed arithmetic expressions:

A = B	A = -B	I = J	I = -J
A = B + C	A = -B + C	I = J + K	I = -J + K
A = B – C	A = -B –C	I = J – K	I = -J – K
A = B*C	A = -B*C	I = J*K	I = -J*K
A = B/C	A = -B/C		

Standard functions:

A = SIN B	A = LOG B	A = FRAC B	
A = COS B	A = EXP B	A = INT B	I = INT A
A = TAN B	A = SQRT B	A = STAND I	
A = ARCTAN B	A = MOD B	I = MOD J	

Control transfers:

JUMP @K

JUMP IF A = B@K JUMP IF I = J@K

JUMP UNLESS A = B@K JUMP UNLESS I = J@K
(K may not have any form of suffix. Any permitted instruction or function instruction may replace A = B or I = J, and > or < may replace =).

Other control operations.
SUBR n EXIT STOP WAIT

Vary and cycle.
Vary A = B: C: L VARY I = J:K:L
CYCLE A = B:C:D CYCLE I = J:K:L
CYCLE A = x, y ,z, … CYCLE I = p, q, r, …
REPEAT A REPEAT I
(B, C, D, J, K and L may have simple suffixes only).

Input
READ A READ I INPUT I

Output
PRINT A, n:m PRINT A, n PRINT A, n/ PRINT A
PRINT I,n PRINT I OUTPUT I
(In OUTPUT I, I may have a numerical suffix only).
LINE LINES I SPACES I TITLE
CHECH A CHECK I

Setting and start.
SETS (integer variables)
SETV (floating-point variables)
SETF (functions)
SETR n (maximum reference number)
START m (starting reference number)
In SETF, note that: (i) TRIG covers SIN, COS and TAN;
(ii) MOD and STAND need not be mentioned;
(iii) FILM allows use of magnetic film instructions;
(iv) CARD and PAR allow use of the card reader instructions.

Magnetic film.
FILM(I) SEARCHJ(K)
FILM(I) TO J(K) or FILM(I) TO A(K)
FILM(I) FROM J(K) or FILM(I) FROM (A)K
JUMP IF FILM(I) SEARCHING @ L
JUMP UNLESS FILM(I) SEARCHING @ L
FILM(I) BLOCK NUMBER TO J(K)
FILM (I) ALLOW WRITE
FILM(I) PREVENT WRITE
(K is any form of suffix; L cannot have any form of suffix).

Punched card.
J = PAR I, m, n
(I, m, n may be replaced by integer variables having numerical suffices only).

A = CARD 1, A, J, K or I = CARD 1, I, J, K
A = CARD 2, A, J, K or I = CARD 2, I, J, K
(K may be replaced by an integer constant).

3. ALGOL 60 for the 803 computer.

The team that developed the ALGOL compilers for the 803 and 503 computers was led by C A R Hoare. In reference 18 (published in July 1962), Hoare states that: "The programming team consists of three people, two of whom have been engaged on the project since April 1961, and the third since November 1961. The first ALGOL program was run on 15th February 1962". Scanned images of the Elliott 803 Algol manual (*January 1965, Issue 4*) have been made available by Bill Purvis at:
http://www.billp.org/ccs/A104

Operating systems.

Towards the end of the 1950s experiments with multiprogramming began to appear. Roger Cook remembers (reference 4) that "The Elliott 802 had an interrupt line enabling one program to be orderly shut down and another started. During a computer exhibition [probably at Olympia, London, in November 1958], we amused ourselves by demonstrating a user participating system while a background job continued. I wrote a naive article on multiprogramming on the 802 which was published by the British Computer Society (reference 10)".

Although all Elliott 803 programs worked unchanged on the 503, in 1963 Borehamwood began planning an ambitious Elliott 503 Mark II software system (reference 4). It was to comprise:

a). An assembler for a symbolic assembly language in which all the rest of the software was to be written.
b). A scheme for automatic administration of code and data overlays, either from magnetic tape or from core backing store. This was to be used by the rest of the software.
c). A scheme for automatic buffering of all input and output on any available peripheral device - again, to be used by all the other software.
d). A filing system on magnetic tape with facilities for editing and job control.
e). A completely new implementation of Algol 60, which removed all the non-standard restrictions which had been imposed on the first Elliott implementation.
f). A compiler for Fortran.

After much effort and difficulty, these ambitious plans were abandoned – due mainly to memory-allocation problems. A reduced-facility system was then developed for the Elliott 503. This included:

ALGOL, SAP (Symbolic Assembly Program), FORTRAN, AUTOCODE;
The STAR operating system;
A central package of programs (PCP and SPAN) which controlled all peripheral devices.

5. Modern simulators for the Elliott 803.

There are two known software simulators for the 803: one written by Bill Purvis and the other by Peter Onion. Both are Linux based. As at January 2005, it is not known whether either is fully-operational. A more recent, but incomplete, version of Bill Purvis's simulator, which runs as a Java Applet within a user's web browser, may be found at: http://www.billp.org/ccs/sim803/ . It should be pointed out that this new version is still under development.