Programming and software for the Elliott 152, 153 and Nicholas computers.

Software and sample programs for the Elliott 152.

The 152 was designed to solve the fundamental fire-control equations of motion iteratively, without reference to conventional range tables. The method took into account “factors such as ballistic coefficient, tenuity of the air, temperature, speed of sound, wind resistance, muzzle velocity, number of rounds fired, drift, motion of gun during firing, and any variation of these quantities with height, elevation, etc." (reference 9).

Besides sending and receiving signals to/from the MRS5 radar and to the guns, the Elliott 152's main computational sequence was summarised as:

(1). Transform the input data from the radar into co-ordinates with respect to a stable set of reference axes derived from the ship's gyro platform.

(2). Smooth each co-ordinate and determine the constants defining the predicted course – (also known as the ‘course-fitting’ stage, and involving the derivation of the target’s velocity and acceleration).

(3). Determine the time of flight of the shell (for fuse-setting) and the co-ordinates of the future position of the target.

(4). Transform to co-ordinates necessary for gun aiming.

Great emphasis was placed on the provision of reliable input data to the Elliott 152 from the radar unit (the MRS5 Director). Since the requirement was for an angular accuracy of one minute of arc, no backlash in the position sensors could be tolerated. Reference 1 explains the Borehamwood invention of three-function binary-coded discs that gave 14-bit values of elevation and bearing angles, plus their sines and cosines. The complete solution time including axis conversion, the iterative process of obtaining the time-of-flight balance and the conversion of future position to gun orders could all be done “in about 130 millisecs”. It was assumed that numbers would be read from the radar and stable (gyro) element about once every millisecond and that the axis conversion routine then generated the converted axes of the target once every 4 milliseconds. Details of the algorithms are given in reference 10.

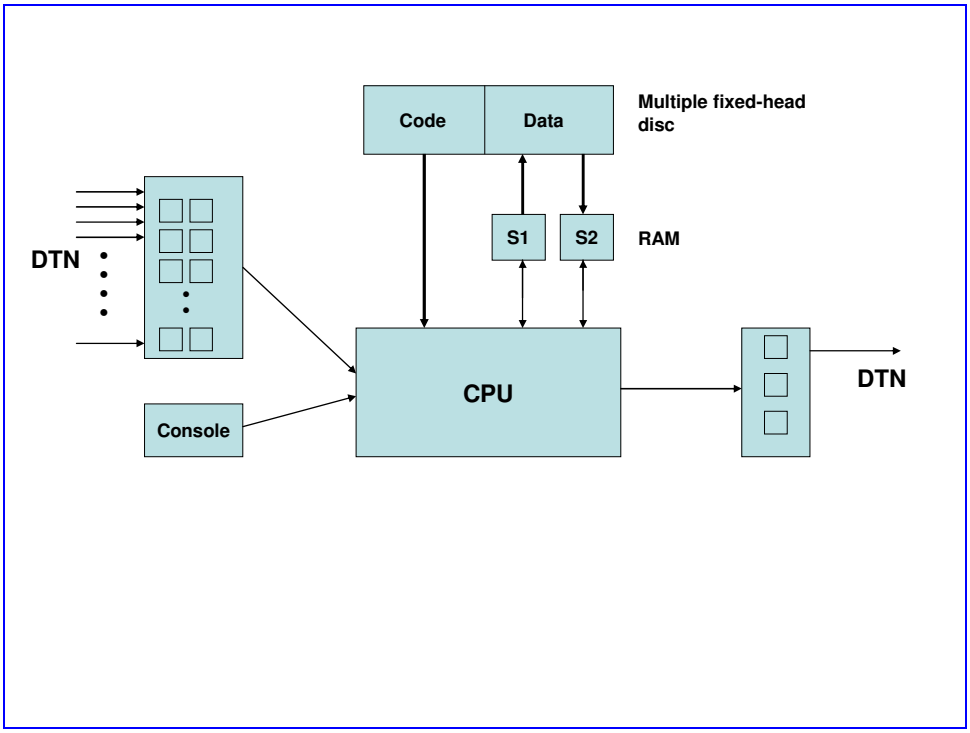
Software and sample programs for the Elliott 153 (DF computer).

The information in this section comes from reference 1 (see E1X5).

Input and output.

Input comes from any ten from a maximum of 20 teleprinter tapes – (the desired ten being connected manually via a jackfield). Each input is more precisely described as a 'teleprinter reperforator and tape reader pair', and each pair forms a direct connection with

a remote Direction-Finding (DF) station. The paper tape acts as an input buffer store, smoothing out asynchronies between the real-time, unpredictable, arrival of messages and the 153's internal operation. The 5-track paper tape readers operate at 10 characters per second. Output is to three teleprinters.



The Elliott 153 computer system, showing its connection to the Defence Teleprinter network (DTN).

Operating sequence.
A typical operational sequence for the 153 computer is as follows:

1. Read incoming DF reports from up to ten stations and assemble all reports relating to the same radio transmission (from a potentially hostile submarine or tank, for example). Store this set of DF data on disc, and give the set a *task serial number*. Indicate to the operator that a new set of task-data is available. Note that the disc may already contain sets of data from previous tasks. DF data for up to 255 tasks may be held at any one time. (By convention, 'task 0' is defined to signify 'no task').

2. The operator selects an appropriate set of DF data, by setting the task serial number up on a row of switches on the 153's control desk. In this context, the operator sees a 'task' as the calculation of a Best Point fix, etc. from one radio transmission from one target. If, from independent Intelligence reports, the DF data from one or more stations is currently suspect, the operator can set a switch which causes the data from one or more stations to be ignored during the task calculation. The data from rejected stations remains on disc, so that it may be included in subsequent calculations if required. The Best Point Fix calculations proceed under program control, as soon as the program has read the switch settings provided by the operator.

3. The computer performs the indicated task on the relevant set of DF data and produces the following output:

(a) the latitude and longitude of points defining both a 'Best Point' fix and a Probability Rectangle';
(b) a printed record of the relevant DF reports;
(c) the deviation of each station's bearing from the Best Point found by the machine.

This output is sent to a tape reperforator (Teleprinter 1) for possible onwards transmission to GCHQ, and also to a page teleprinter (Teleprinter 2) for a local record. Finally, a third page teleprinter (Teleprinter 3) produces information of use to the control desk operator, including an indication, by serial number, of the total number of DF reports so far received in respect of each radio transmission from a submarine.

4. Every time a new DF report arrives, the sequence repeats from step (1). At step (2), the operator can choose to re-calculate a result for one target incident or calculate a result for a new (target) incident.

Data structures.

When sending reports to the 153 computer at Scarborough, each remote DF station necessarily has to include information that identifies the station (as three 5-bit characters) and the date/time of the report. In modern terminology, each report is a fixed-format DF record. Each DF record occupies three words. If there are up to ten remote DF stations on-line to the Elliott 153, then each (submarine) transmission incident results in a set of up to ten 3-word DF records. Up to 255 sets of DF records can be held on side B of the 153's disc. Besides these incoming DF records, side B of the disc also holds:

(a) Fixed data (eg geographical position, etc.) for each of up to 20 DF receiving stations located around the world. All the station data is contained on one disc track, occupying a total of 240 words.
(b) Directories, or look-up tables, to aid the selection of DF data for a given task.

Operational performance.

No details are available – but see the example code in section E1X3.

Software and sample programs for the Elliott Nicholas.

Nicholas Assembler and library of subroutines.

The fully-developed system software for Nicholas was largely written by George and Ruth Felton. For general comments on Nicholas software in relation to other contemporary British systems, see references 11 and 12. Following the example of EDSAC at the University of Cambridge, users prepared programs for Nicholas on punched paper tape in terms of alphabetic mnemonics and decimal numbers. A user's program was then read in and converted into binary by the previously-loaded Nicholas Assembler. This Assembler, called the *Translation Input Routine*, occupies locations 0 – 213 of memory and includes subroutines for multiplication and printing. Locations from 214 upwards then contain various useful addresses and constants. User programs normally start at address 265.

A user might include in his or her program one or more well-tested sections of code, chosen from a library which in due course included the following groups of subroutines (references 5, 6):

Division,
Square root and other fractional powers,
Trigonometrical functions,
Exponential functions,
Hyperbolic functions,
Complex numbers,
Logarithms,
Quadrature,
Solution of differential equations,
Floating-point arithmetic,
Matrix manipulation,
Printing and layout.

The floating-point interpreter took about 0.5 seconds per floating-point instruction, according to its author, George Felton. The Matrix Interpretive Scheme was written by Felton to calculate flutter vibrations in aircraft. The de Havilland factory at Hatfield, where the Comet airliner was being designed, was quite close to Borehamwood and Elliotts sold a large amount of Nicholas computer-time to Peter Hunt who worked at de Havilland in the aerodynamics department. The general principles of the Nicholas subroutine library and interpretive software were later carried over by Felton into the Ferranti Pegasus Library – (Pegasus is described in sections F3/X1 – F3/X5).

It is instructive to give the Nicholas Print and Multiplication subroutines in full, since these illustrate the more esoteric uses of the TX and TN instructions. The descriptions which follow are taken directly from George Felton's explanation on pages 29 and 30 of reference 5.

The Print subroutine.
This closed subroutine, which is stored in locations 138 to 149 of the *Translation Input Routine,* is called by a *P* pseudo-instruction. P is translated into the binary equivalent of TX 138 during the Assembly phase of a user's program. To quote reference 5: "When the subroutine is entered the accumulator contains the binary equivalent of the teleprinter code for the character it is desired to print. This binary number is tested by the IX orders and doubled between each test until seven mark or space signals have been sent to the teleprinter. These signals have to be sent at intervals of 202 word-times [about 2 milliseconds] to operate the teleprinter correctly; this consideration determines the dummy addresses of the IX and D orders and the sequence of orders. The orders of the subroutine are as follows [as they would appear in written (source) form]":

Address	***lower (first) instruction***		***upper (second) instruction***	
138	IX	10	TX	144
139	IX	30	TX	143
140	IX	84	TX	142
141	D	15	TX	145
142	D	16	TX	139
143	D	17	TX	147
144	D	18	TX	140
145	IX	124	TX	149
146	IX	50	TX	141

147	IX	104	TX	148	
149	D	22	TX	146	
149	D	19	IX	70	*(absence of TX permits return to the main program).*

The appearance of the IX order in the print routine is not fully explained by the surviving source documentation. Commenting many years later, George Felton was still unsure how to explain the action. He recalled that: "The IX-order has function bits 100101 and is not precisely defined in the original manual. It appears in the Print Subroutine (see Appendix D, page 29, of reference 5) but nowhere else, as far as I know. I feel sure that this Print subroutine was devised by Charles Owen. I can't recall using the IX-order but I may have included it in my 'uniselector' program for the Initial Orders. My memory is weak on this point and much of the design documentation has now been lost... Appendix A on page 17 of reference 5 supports a *possible* meaning of the IX-order, namely: 'Read the tape and add it to the least significant end of the Accumulator' + 'special action'. This apparently clashes with its use in the Print routine, since this refers to *output*, not input. Maybe the 'special action' has two meanings – but I feel there is little chance of hitting the right one in view of the loss of so much information".

The multiplication subroutine.

This closed subroutine occupies storage locations 0, 2, 4, 5, 7, 9, 82, 84 and 86 of the *Translation Input Routine*. To quote reference 5, "When the order M n is obeyed control is transferred to location 0 and the accumulator goes double-length; the number originally in the accumulator occupies one half and the number from location n occupies the other half of this double length accumulator. Orders with an even address will affect the multiplier part of the accumulator and orders with an odd address will affect the multiplicand part. The orders of the subroutine are given below [as they would appear in written (source) form]."

"The orders in locations 0, 2, 4 and 5 change the signs of the numbers to make both positive and the multiplication proper is initiated by one of the orders YC in locations 7 and 9 (depending upon the sign of the product). YC clears one half of the double length accumulator leaving the multiplier in the other half and transfers the multiplicand to a multiplicand register. It also causes the accumulator to increase its length to 2 words plus 1 digit. In the ensuing word times, until coincidence occurs at either 82 or 84, the multiplier is shifted one digit every other word time and simultaneously the multiplicand is added to the other half of the accumulator if there is a digit in the multiplier at the most significant end. Just before coincidence occurs the multiplier has been shifted out and the double length product occupies the whole of the accumulator. On coincidence the accumulator returns to single length but the least significant digit is lost in the process. Order 82 or 84 makes the least-significant digit unity and since there is no TX (or TN) in either order, control is returned to the order immediately after the original multiplication order, M. In this process the least significant digit is always unity but there is, in general, no bias in the rounding-off error. However, if the double length product ends in 32 or more zeros, this extra digit may cause trouble".

The multiplication subroutine is given on the next page.

Address	***lower (first) instruction***		***upper (second) instruction***		
0	ZN	2	TN	4	*(Test sign of multiplicand)*
2	ZN	5	TN	7	*(Test sign of multiplier (negative multiplicand))*
4	ZN	7	TN	9	*(Test sign of multiplier (positive*
5	0	7	TX	9	*multiplicand)).*
7	YC	14	TX	82	
9	YC	16	TX	84	
82	A	86	N	88	*(Negative product)) Absence of TX or) TN* permits return
84	A	86	0	88	*(Positive product)) to main program.*
86		1			*(Round-off constant = 1 x 2 $^{-30}$)*