Elliott 903 Software

Andrew Herbert
2nd February 2013

Background

In September 2011, I became the owner of an Elliott 903 computer. Originally delivered by Elliotts to AERE around 1967, the machine was purchased by Don Hunter in 1978 and used by him as a home computer for many years. Now advanced in years, Don no longer wanted to the machine in his home, so aided by Terry Froggatt, I moved it to my house in Cambridge. As well as the computer Don handed over several boxes of paper tapes and manuals. After re-commissioning the machine with Terry's help I set myself the task of archiving and cataloguing the paper tapes and documentation, with the aim of establishing master copies of what a 903 user would expect to have in the early 1970s just before Elliotts ceased providing support and the machines became obsolescent.

The Elliott 903

The Elliott 903 computer is described in some detail on the "Our Computer Heritage" web site. In summary, it is an 18-bit minicomputer with a minimum of 8K of core store. The 903 was a civilian packaging of the military Elliott 920B computer. Designed in the early 1960s, it was a discrete transistor machine with a typical instruction time of around 25 microseconds. It was a popular teaching machine in universities and colleges where the 16K configuration was commonly deployed to allow load-and-go operation of ALGOL 60 and FORTRAN. Some industrial users further extended the memory to 24K and beyond to run programs on large data sets.

The order code is very simple with 16 basic instructions operating on fixed-point fractions: floating-point arithmetic has to be performed by software.

The primary means of input/output is 8-hole paper tape. While nominally an optional peripheral, most 903s came with an attached ASR33 teletype. Graph plotters, magnetic tape drives, line printers and card readers were also available (identical to those supplied with the Elliott 4100 series). Physically the basic machine resembled a modern chest freezer. Further cabinets could be added to hold additional core store, up to a maximum of 64K, and/or additional peripheral interfaces, along with additional power supplies. There was an operator's control panel that stood on top of the cabinet together with paper tape reader and punch. Many installations also had an engineer's diagnostic panel with neon displays of the principal registers etc.

While there were interrupts to allow multi-programming and asynchronous input-output, there was no virtual memory provision so most 903 computers were used as single user machines.

When new, in the late 1960s, a typical system would cost around £25,000.

Elliott 903 Software

Elliotts supplied software to users bundled in with the purchase and on-going maintenance support for the computer itself: in those days software was seen by manufacturers as a cost of doing business rather than a source of revenue. The standard suite consisted of:

1. Programming language systems (ALGOL 60, FORTRAN II and SIR assembler)
2. Subroutine libraries
3. Development utilities (e.g., editors, debugging aids)
4. Applications
5. Test programs.

Notably this list does not include an operating system. Most users ran programs on the bare machine and the programming language systems were self-contained. There was a 9 KHz magnetic tape based batch operating system called FAS which I encountered as a schoolboy, sending FORTRAN programs to be run on Elliott 903 at the now defunct Medway and Maidstone College of Technology in Rochester, Kent. There was also a disc operating system called RADOS for the Elliott 905 computer, a faster and upwards compatible successor to the 903, but that is outside the scope of this article.

Elliotts distributed software on paper tape. Programming systems, utilities and applications were shipped as "sum checked binary" tapes suitable for loading using the 903's initial instructions. Libraries were supplied either as source code or an intermediate "relocatable binary code (RLB)" suitable for linking with compiled programs.

Elliotts settled on using the emergent ASCII code for source tapes, for compatibility with the 4100 series, moving away from the earlier private code used on their 503 computers. Unfortunately ASCII changed somewhat during the life of the 903 with some symbols unhelpfully swapping positions in the code table. Most 903 software essentially ignored this – a later "900 telecode" was introduced in place of the original "903 telecode" but in reality all that happened was an update to the documentation to reflect the new graphics associated with the binary codes used in the software. This was particularly tiresome for ALGOL 60 users where string quotes changed from acute and grave accents ('a pretty string') to quote and at sign ('an ugly string@), and as with many other contemporary British machines, £, # and $ played musical chairs.

Programming languages

Elliotts provided three programming language systems for the 903: ALGOL 60, FORTRAN II and SIR (Symbolic Input Routine – i.e., an assembler).

ALGOL and FORTRAN were both provided as either a two-pass system for use on 8K machines, or as an integrated "load and go" system for 16K machines. In the two pass systems an intermediate tape was produced for communication between the separate passes.

In the case of ALGOL, the first pass was the compiler (called the "translator" by Elliotts) that produced an intermediate stack-based code rather than relocatable binary machine code (although the same tape format was used for both so that the standard 903 loader could be used to put the bits into store. The second pass run time system comprised the interpreter for the intermediate code and standard libraries.

In the case of FORTRAN, the compiler produced either SIR code or RLB code and therefore the runtime was basically just the standard loader and a set of libraries.

In both systems the user could reclaim the memory allocated to the standard library by scanning a "library tape" containing RLB and only those routines needed by the program would be loaded.

Both systems also provided a "large program" second pass, essentially the standard second pass modified to allow programs and data to extend beyond the 8K limit of the basic system.

The ALGOL system was developed by CAP and based on the Whetstone ALGOL system for the KDF9 developed by Randall and Russell, written up by them in their book "ALGOL 60 Implementation" published by Academic Press in 1964. Indeed having found some source tapes, I discovered that the 903 ALGOL system is essentially a SIR transcription of the flow charts in that book. 903 ALGOL has a few restrictions compared to its parent: recursion and own variables are disallowed and all identifiers have to be declared before they can be used. More usefully, the language supports "PRINT" and "READ" statements as found in 803 and 503 ALGOL which are convenient to use and allow good control of output formatting.

The restriction on recursion is easily circumvented with a patch to remove the check for it in the translator. Everything is fine at run time provided the recursive procedure does not have local variables – since these are statically allocated (i.e., making them own variables….). A programmer can achieve the effect of dynamic local variables by nesting a local procedure within the recursive one. Don Hunter developed such a patch and several others to add further extensions that are embodied in his ALGOL system for the Elliott 903 simulator to be found on the CCS simulator archive. I've subsequently reverse engineered a paper tape version from this, added further patches due to Terry Froggatt that deal with the character code issue, resulting in the first new release of Elliott ALGOL 60 in nearly 30 years!

The FORTRAN II system is little more than a macro-generator: FORTRAN statements are translated line-by-line into machine code with no optimization. While generating machine code might be thought to lead to faster programs when compared to ALGOL, the fact that both have to implement floating-point arithmetic using an interpreter erodes any difference. Moreover, when

debugging, ALGOL's stricter checking of integer arithmetic for overflow can be an advantage.

Both ALGOL and FORTRAN provided facilities for writing procedures in machine code. In the case of ALGOL, and optionally for FORTRAN, these have to be independently compiled and linked. FORTRAN also allows SIR assembly code to be included in-line. In both cases there are strict rules defining how machine code should be written to conform to the runtime conventions of the corresponding language system.

It was not possible to separately compile and link ALGOL procedures, using Elliott's software. In part this was because the output of the ALGOL translator was intermediate code for the runtime interpreter rather than machine code. However Don Hunter subsequently provided a means to compile ALGOL procedures independently for the Elliott 903 simulator and in principle his system could be run on a real 903.

The main assembler provided by Elliotts was called SIR (for Symbolic Input Routine), itself was the successor to an earlier more primitive assembler called T2 that had been developed for the first machines in the 900 series (i.e., the 920A).

SIR allows the use of integer, fractional, octal and alphanumeric constants, symbolic labels, literals, relative addressing and comments as its main features. It does not provide any form of macro generation facility and has limited ability to do arithmetic on labels as addresses. Instruction function codes are expressed numerically rather than mnemonically.

T2 by contrast only accepts instructions with simple relative addressing and integer constants. While essentially obsolete at the time of the 903, T2 was given to users as a number of utilities were written in T2 or required data laid out as if a T2 program block. Many library routines were also supplied in T2 notation. By design SIR was made upwards compatible with T2 so these could be assembled in both systems.

SIR could be operated in either load-and-go mode or made to produce RLB tapes preceded by standard RLB loader. Load-and-go mode was convenient for small programs, but if the space occupied by the assembler was needed, or a self-contained binary tape for loading under initial instructions was required the RLB option was preferable. There is no support in SIR for program code to be located outside the first 8K of memory, which was to become an issue later in the history of the 900 series as 16K and larger memories became prevalent. (The work around used to assemble the 16K load-and-go systems was to build them in the bottom 8K, then run a utility program to copy them to the upper 8K. As the machine code is based on 8K relative addressing there was no need to fix up addresses, except for references between the two 8K modules).

Library routines

The library routines provided by Elliotts were principally input-output and mathematical functions. A double precision interpreter (QDLA) and a floating-point interpreter (QF) were provided along with mathematical functions for these formats. In the case of ALGOL and FORTRAN versions of these packages were built into the language systems: the standalone library versions were intended for SIR programmers to use.

For ALGOL programmers there was a source code matrix library called ALMAT. This package originated on the 803/503 computers and provided routines for matrix operations and linear algebra functions.

For machines with graph plotters, line printers and card readers there were appropriate library routines (device drivers in modern terms) for the SIR programmer. ALGOL had its own plotter library (based on the library for the Elliott 4100 series). Both ALGOL and FORTRAN integrated card reading and line printing into their higher-level input-output facilities.

The library routines were all known by short identifiers beginning with the letter Q (e.g., QSQRT, and strictly "Q not followed by U"): those which had been inherited from the early 900 series software were often also known by shorter code names such as “B1” (QLN), reminiscent of the EDSAC convention for naming library tapes.

The only data processing oriented routine was a Shellsort package for in-memory “files”, and the interface to this was very basic – essentially a table defining record structure and the order in which fields were to be sorted. There was no explicit support for fields that spanned multiple words or had complex structure (e.g., floating point numbers) although a programmer who understood the machine representation of the data could work around this.

Utility Programs

The utility programs provided by Elliotts were mostly concerned with paper tape preparation. EDIT was a simple text editor, reading in a steering tape of editing commands to be applied to a source tape and an updated version punched. It has commands for copying or deleting to an identified string, making replacements and inserting new text. Two tape copy programs were provided: COPYTAPE for short tapes and QCOPY for long tapes. The former read the entire tape into store before punching a copy, which was kinder on the tape reader than QCOPY, which operated character by character, forcing the reader to continually brake the input tape as it waited for the punch. Given the potential unreliability of paper tape as a medium all these routines provided a means to rescan the input and read back the output to ensure misreads and/or mispunches were detected. (And as Terry Froggatt points out, the truly paranoid read back the input again after punching to confirm the absence of store corruption...)

Debugging aids comprised two utilities: MONITOR and QCHECK. Both had to be assembled as part of the system being debugged. As the name suggests, MONITOR provided facilities to monitor the execution of a program by printing

out registers and memory etc. whenever the program executed one of a specified set of program locations. QCHECK provided an interactive debugging interface enabling memory to be inspected and modified and for further program execution to be triggered. QCHECK itself could be triggered by an interrupt from the operator's console. Both operated at the machine code level and were of little use to ALGOL and FORTRAN programmers.

A separate group of utilities provided a set of tools to allow the programmer to build binary tapes that would load under initial instructions (i.e., not requiring the relocatable binary loader).

QBINOUT was used to punch out short binary programs to paper tape, in a form suitable for loading by the initial instructions. Typically the punched program would itself be a more powerful loader, of which there were several. T22/23 provided just a loader (the T22 part) and the means to dump regions of store out in T22 format (the T23 part). The paper tape format was more compact that that produced by QBINOUT and included checksums to guard against errors. A further utility C4 was provided to check the contents of a punched T22/23 a tape against store. (interestingly some internal documentation refers to T22 as QSCBIN and T23 as QSCBOUT but these names didn't make it to the 903 user documentation.) A more advanced loader was that used by the three language systems to link and load separately compiled libraries and programs help in RLB form.

Applications

There were just two applications provided by Elliotts – the Elliott Simulation Package (ESP) and a PERT project planning system.

ESP consisted of a machine code library and accompanying ALGOL source code routines for generating random numbers drawn from different distributions, histogramming and setting up event based simulations. The user was expected to write their simulation using the supplied source code as a foundation, compile the result and link with the small machine code library that contained a random number generator.

The PERT application was standalone, reading a steering tape to set the system up then a series of data tapes describing individual projects to be analysed. The application would then print tables of shortest paths, earliest and latest deadline and so forth.

Test programs

Elliotts offered a large collection of test programs for the 903. Most were intended for use for fault-finding by engineers, but a few were supplied to users for use at start of day to check the condition of the machine and paper tape system. All the test tapes were named Xn, as in ‘X3’ the processor function test.

Other Software

The advent of the Elliott 905 brought a new version of SIR called MASIR which included macro generation facilities and new symbolic address formats to better enable programming across 8K memory module boundaries as 16K and larger memory configurations became the norm. Instruction mnemonics were introduced to cover the fact that the additional instructions in the 905 repertoire were all coded as function code 15. MASIR came with a new linking loader (called "900 LOADER") with a new RLB format. Both MASIR and the 900 LOADER were issued to 903 users.

A new FORTRAN IV system was provided for the 905. This provided a full implementation of the ASA standard and generated RLB suitable for the new 900 LOADER. While 905 FORTRAN runs in the 903 it would appear it was not supplied to owners of the older machines.

With the advent of the 905 it became more common for there to be an operating system and the final releases of a number of the Elliott systems moved to a command line based user interface in place of using the machine control panel to jump to program entry points. (And while outside my own experience I would presume this facilitated integration with operating systems like RADOS).

There were other sources of software for the 903 beyond Elliotts: there was an active user group and members often made their software available to one another. Other languages such as BASIC and ML/1 became available through this route. Elliotts also had a CORAL system but it was not issued to 903 or 905 users.

Within Elliotts, other divisions at Borehamwood, Rochester and elsewhere developed their own variants of the standard software more appropriate to embedded use on 920B and 920M machines, and whose development systems often lacked an online teleprinter. Examples include more a powerful editor (BOWDLER) and a two-pass SIR system that directly punches sum checked binary tapes suitable for loading under initial instructions. Many of these utilities continued to support the older 920 telecode as well as the 903 and 900 telecodes, including provision inputting in one and outputting in another. But this is a story for Terry Froggatt to tell as an Elliott insider.

Software Issues and Quality

Elliotts labelled successive versions of each piece of software and documentation pages with "Issue" numbers. These were incremented with each release but in a few cases there were minor releases with numbers like "Issue 4C". Most tapes were punched with a legible header giving the name of the tape and its issue number, although this was not done consistently and so some tapes lack headers, some have dates rather than issue numbers and some just have written or stamped legends. Surprisingly, it was rare for the issue number to be included as a comment in source code or printed as a diagnostic when running binary tapes.

Users were expected to update the manuals with re-issued pages and to replace obsolete versions of software. Interestingly the software and manuals I obtained from Don Hunter seem to relate to a second 8K system he acquired from the British Ceramics Research Association and it would appear this organization did a reasonable job of updating, although tiresomely Elliotts did not provide an updated index and catalogue against which users could check their holdings. The only way this can be done is to track through the "release notes" and other bulletins sent out by Elliotts. These are missing from Don's collection, but fortunately Richard Burwood has an almost complete set and has summarized the history of all the Elliott issued software in a helpful dossier. Terry Froggatt also has many of these notes, so with both their help I've been able to identify the last known releases for every item and ensure I have a "current" copy of both software and documentation.

As I've worked through all the software and documentation checking one against the other, mostly using my own 903 simulator as this offers a faster development cycle than paper tape on the physical machine, I've uncovered a surprising number of quality issues beyond the cavalier approach to tape labelling. Many of the examples in the documentation either contain significant errors and don't run, or don't produce the stated results. In part this is inevitable when documentation was prepared from a handwritten script by a typist, but some errors indicate nobody had ever checked out the examples. A case in point is QOUT1, a library routine for printing numbers, where the sample code produces different results due to rounding errors than what is documented. Some of the later issues of the SIR mathematical routine sources contain simple syntactic errors, easily corrected, but again showing a lack of attention to detail. It's also evident from the sources that there was no common coding standard used by Elliott software developers, leading to irritating inconsistencies between how the same thing is done in different sub-systems.

Summary

The Elliott 903 came with three major programming systems: ALGOL, FORTRAN and SIR, together with associated utilities for software development using paper tape and mathematical libraries for engineering and scientific problems. There was little in the way of data processing facilities.

While one can quibble about some quality issues with how it was documented and distributed, the software generally did what was expected of it and made the machine a good system for teaching use and for use in scientific and engineering applications.

Postscript

As a side effect of exploring Elliott software the author has written his own simulator in F#, a recently launched Functional Programming language developed by Microsoft Research, Cambridge. This simulator is based on the Froggatt/Hunter simulator (which was written in Ada) extended with a full implementation of the "undefined" effects of each model in the 900 range,

interrupt handling, asynchronous input-output and an extensive range of commands for unpicking various paper tape formats, debugging and tracing etc. It has only been run on Windows 7 and is still a work in progress. Readers are welcome to request a copy. Included in the simulator are files containing all the Elliott issued software, demonstration scripts to illustrate their use and a manual documenting both the simulator and the Elliott software.