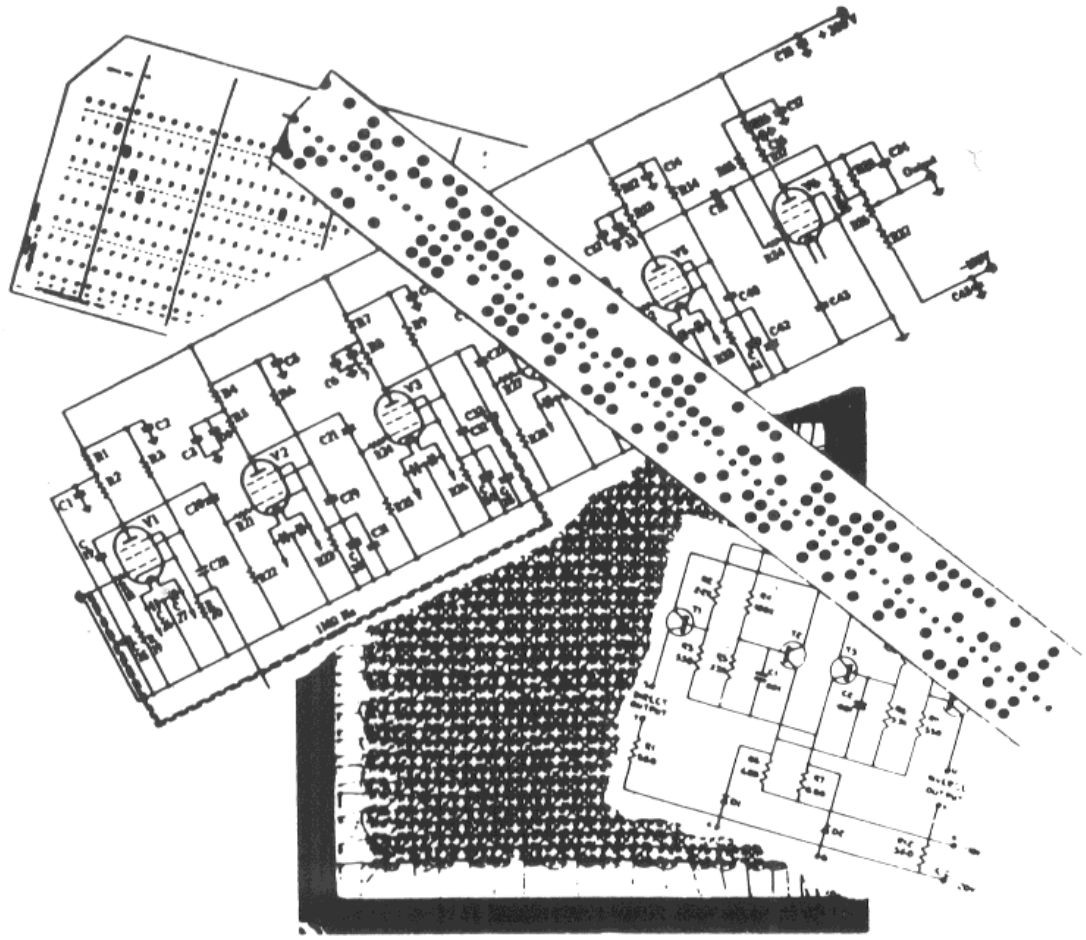
Issue Number 41 **Autumn 2007**

Computer

RESURRECTION

The Bulletin of the Computer Conservation Society



science museum

BCS THE BRITISH COMPUTER SOCIETY

THE MUSEUM OF SCIENCE & INDUSTRY IN MANCHESTER

Computer Conservation Society

Aims and objectives

The Computer Conservation Society (CCS) is a co-operative venture between the British Computer Society, the Science Museum of London and the Museum of Science and Industry in Manchester.

The CCS was constituted in September 1989 as a Specialist Group of the British Computer Society (BCS). It is thus covered by the Royal Charter and charitable status of the BCS.

The aims of the CCS are to

◊ Promote the conservation of historic computers and to identify existing computers which may need to be archived in the future

◊ Develop awareness of the importance of historic computers

◊ Encourage research on historic computers and their impact on society

Membership is open to anyone interested in computer conservation and the history of computing.

The CCS is funded and supported by voluntary subscriptions from members, a grant from the BCS, fees from corporate membership, donations, and by the free use of Science Museum facilities. Some charges may be made for publications and attendance at seminars and conferences.

There are a number of active Working Parties on specific computer restorations and early computer technologies and software. Younger people are especially encouraged to take part in order to achieve skills transfer.

Resurrection

The Bulletin of the Computer Conservation Society

ISSN 0958-7403

Number 41

Autumn 2007

Contents

News Round-Up **2**

Society Activity **4**

Computer Development in Cambridge 1937-80 **8**
Maurice Wilkes

Early Computer Production at Elliotts **15**
Laurence Clarke

The Origins of Fortran **23**
Peter Crouch

CCS Web Site Information **33**

Obituary: Dina St Johnston **34**
Simon Lavington

Letters to the Editor **35**

Forthcoming Events **36**

News Round-Up

The BCS@50 event was a great success. Outgoing CCS Chairman Roger Johnson has enthusiastically reported that "we achieved everything we set out to do". We trust members who attended any or all of the three days were equally satisfied.

101010101

Roger Johnson stepped down as Chairman of the CCS following the BCS@50 event, and has been succeeded by David Hartley. Roger, a Committee member since the Society's formation, will continue to serve the Society as London Meetings Secretary.

101010101

This year is full of landmark anniversaries! Following the golden jubilee celebrations for Fortran and the BCS it is now time for the diamond jubilee of J Lyons' decision to fund the building of the Leo computer. That decision was taken in October 1947, and the 60th anniversary celebration is scheduled for 19 October this year.

101010101

Dina St Johnston has died aged 76. Dina was one of the first programmers; she worked on Edsac and the Elliott 400 and 800 series, and set up the UK's first software house, Vaughan Programming Services, in 1959. Dina was married to Andrew St Johnston; both were active supporters of the CCS, and it was their 803 which is now maintained by the Society at Bletchley Park. See Obituary on page 34.

101010101

Professor Donald Michie died in July aged 83. Donald was a celebrated artificial intelligence pioneer, and had earlier been one of the team that developed Colossus at Bletchley Park.

101010101

Former BCS President John Ivinson died in July at the age of 63. John played a major role in the professionalisation of the BCS over the years, and recounted his memories of the time leading up to the achievement of the Royal Charter in *Resurrection* issue 38.

101010101

Ken Johnson died in June aged 79. Ken was a prominent electronic engineer in first Ferranti and then ICL from the time of the 401 up to 1980, and in recent years was a regular attender at North West Group meetings.

HRH Duke of Kent officially switched on the reconstructed Bombe at Bletchley Park on 17 July, the culmination of a decade of work by the Society's Bombe Rebuild Project team led by John Harper (right, standing next to the Duke). The three ladies in the group are all former WRNS who operated the Bombes during the War.

101010101

Brian Wichmann is leading a group that is compiling information about the English Electric KDF9 for the 'Our Computer Heritage' pilot study Web site. Brian would like to hear from anyone who used to program or maintain a KDF9; he can be contacted at *Brian.Wichmann@bcs.org.uk.*

101010101

The replica Small-Scale Experimental Machine ("Baby") has been disconnected during a refurbishment programme at the Manchester Museum of Science and Technology. Consequently the regular Tuesday demonstrations have been suspended temporarily.

North West Group contact details

Chairman Tom Hinchliffe: Email: *tom.h@dsl.pipex.com*
Secretary William Gunn Email: *william.gunn@ntlworld.com*

Society Activity

Pegasus Working Party

Len Hewitt and Peter Holland

Pegasus has been running at fortnightly intervals with the day changed to Wednesday. The machine has worked extremely well and we have been rewarded with lots of interested visitors from all over the world. We recently recovered a number of 42- and 35-bit delay lines from Blythe Road, some of which have been tested and pronounced usable as spares. This is a great relief as our stock of spare working lines was low.

We have also recovered an old Model 54 Creed Teleprinter which we hope to use as part of a rebuilt editing set. The Creed 75 was beyond repair as we could not obtain a clutch for our particular model. We still have a blind punch and also the PC editing set. We also recovered from Blythe Road some 40 or so paper tape reels which should last us for many years.

Unfortunately on our last visit on 12 August the compressor that provides the cooling agent decided to retire, though it is only seven years old, and we had to switch Pegasus off because it was getting hot. We are hoping the compressor is repaired for the next Pegasus day on 26 September.

Contact Len Hewitt at leonard.hewitt@ntlworld.com.

Bombe Rebuild Project

John Harper

We have now moved the machine into a much larger area. This is mainly to allow more people to gather around the machine at the same time but it also gives us more space for other equipment and displays. A new ‘fence’ has been constructed together with fence top displays and large display panels around the walls.

The photo opposite shows our Rebuild in its new area. The ‘fence’, fence top displays and larger floor mounted display panels can all be seen.

The move went more or less to plan and the machine did not suffer in any way. There was a short interim stage before the shop was moved but we took over the new area with a few days to spare before the BCS@50 event.

From the team's point of view the BCS@50 event went very well but this is perhaps for others to judge. The machine worked correctly when both groups visited on the Thursday afternoon and stopped with correct indications. Many of the delegates showed genuine interest and there were also follow ups in London on the following Friday and Saturday.

A few days after the BCS@50 event we had the Official 'Switch On' by His Royal Highness the Duke of Kent. Most of our larger Bombe Rebuild team of 50 or so were able to attend and greatly enjoyed the day.

There were many dignitaries and Bletchley Park Trustees in attendance. A few showed genuine interest in what we had achieved. HRH was very attentive and easy to talk to and gladly switched on our machine as requested. Various photos and videos were taken and some of these appear on the BCS Web site. There is also a reference in the September 2007 issue of *IT NOW* page 4.

Since the ‘Switch On’ the machine has been run regularly. We nearly always get a good stop using a variety of menus but I have to report that we have encountered a few ambiguities. These have been traced to sense relays occasionally malfunctioning. These relays are really the heart of the machine so everything depends on them functioning faultlessly. Initially we set these up using a specification found in a WWII document. This differed in some respects from a procedure published for the same relays in a 1950s BTM Engineers Manual. We are now part way through readjusting all 104 relays and the anomalies are reducing all the time. None of this precludes us from using the machine to the full.

We now have many WWII jobs being rerun. Also, we are finding more existing WWII decrypts in the Bletchley Park Archive. All of these will be rerun in due course.

A recent success has been the running of what is called a Double Input Menu. The term may not be familiar to many readers: what it does is to bring extra circuitry in play. Two sets of sense relays are connected in series: to obtain a correct stop it is therefore necessary for *two* banks of 26 Sense Relays to function correctly, and they do.

One of our next tests will be to run a menu with a different reflector panel. This became necessary during WWII when the Germans introduced a re-pluggable ‘D’ reflector wheel into their Enigma machines. Fortunately they did not fully introduce this feature for if they had Bletchley would have had great difficulty continuing to decode messages.

Our fully operational Checking Machine is part of our display but at the present time this can only be demonstrated when members of our team are around. When we have a lockable display case the Checking Machine will be on show at all times.

Similarly we only demonstrate a Typex machine that we modified to work as a German Enigma machine from time to time.

As reported above we have considerably improved the displays around the machine and now have in addition a major new ‘how we made it’ exhibition. With so much tooling becoming necessary as parts were built we ended up with a considerable collection. Some of this is now on display. In fact we could do with more display case space.

Our latest ideas include making a video to be run ‘on demand’ next to the Bombe. This will attempt to capture the same commentary that we give to visitors when they visit the Bombe, Checking Machine and

Typex. There is no way that our team can have a regular presence at BP so this will have to cover the rest of the time. It is early days: progress is being made.

One thing I do have to admit is that our Web site update is still not complete. I have put a lot of effort into this and much is done but new input still arrives.

Our Web site is at *www.jharper.demon.co.uk/bombe1.htm*.

CCS Archives

Hamish Carmichael

The ICL Archive, which I have been cataloguing for the past six years, is shortly to be transferred from Blythe House in West Kensington to the Science Museum's other site at Wroughton near Swindon. When it is properly settled into its new home it will have more space and will become more easily accessible. Unfortunately until the move is complete it is impossible to service requests for research access or to incorporate additional material. Members who want to unload surplus historical documentation please continue to contact me and I will maintain a list of pending donations.

The listing of that archive which is maintained on a Web site run from Leeds needs to be updated to reflect the way the collection has grown over the past two years. When that work has been done the appropriate reference to it will be included in the Society's own Web site.

Staff at the Science Museum Library tell me that the availability of that listing has made the ICL Archive by far the most active of all the Library's reserve collections.

The library of the National Museum of Computing at Bletchley Park now contains, under the heading "A Life in Computing", a very varied collection of papers put together by the late Fraser Duncan. He worked at various times at the NPL, at Kidsgrove and at several universities, was deeply involved in the genesis of Algol, contributed significantly to the development of standards, and kept extensive records of all his activities. Potential biographers please note.

Contact Hamish Carmichael at hamishc@globalnet.co.uk.

Computer Development in Cambridge 1937-80

Maurice Wilkes

This article gives the author's personal recollections of the events which led to the founding of the Computer Laboratory, under the name Mathematical Laboratory, at Cambridge in 1937, and its opening after the war with the author as Director. A brief account is then given, again from a personal point of view, of the Edsac project, and also of the Edsac 2 project that followed it. Attempts were made to establish further projects intended to form worthy successors to these pioneering efforts. The author makes a few remarks about these projects and what they achieved before he turns his attention to the Cambridge Ring, which was the final project in which he played a major personal part before he retired from the laboratory in September 1980.

Before the war there were no digital computers, but an increasing amount of digital computing with desk machines was being done. The power of numerical methods was becoming widely appreciated, and it was recognised that in the case of certain problems this was the only way in which progress could be made.

Though many of the individual workers taking part in this movement focussed closely on their own applications, others had a wider interest. In Cambridge one man had a very clear view of the importance of the trend. He was Professor JE Lennard-Jones, a theoretical physicist who had found a niche for himself in the Chemistry Department as a pioneer in structural chemistry using the method of molecular orbitals, and who ran an active research school. It had become clear that, with sufficient determination and persistence, significant progress could be made with real problems. There was some discussion of mechanical or electrical computing devices of an analogue type; I recollect no proposal for any kind of automatic digital computer.

Lennard-Jones evidently felt that the time was ripe for a major initiative, and he secured the acceptance by the University of proposals for the establishment of the Mathematical Laboratory. Prominent among these proposals was one for the purchase of a full-scale differential

analyser. This was a popular proposal at the time, differential analysers having caught the imagination of scientists and many others.

However, Lennard-Jones's vision went further. He took care that the new laboratory, when established, should take its place as a fully fledged Department of the University, with terms of reference that could not have been wider; they were the study and development of computing in all its aspects, analogue and digital.

It is hard to know who else would have been capable of such vision or possessed the political skills necessary to secure their adoption. Lennard-Jones clinched the whole deal by snapping up the old Anatomy School which the anatomists had just vacated for a new building. In the event this was very important since, after the war, accommodation was at a premium.

Lennard-Jones himself became the (part-time) Director of the Mathematical Laboratory and initially there was to be one staff member, myself. At the outbreak of war, the Laboratory existed on paper but had not yet opened.

When war was imminent, I received, through my Cavendish connections, an offer that, in view of my background, I could hardly refuse, namely to join the national radar effort. That was the right thing for me, and I accepted.

Birth of the Edsac

When I returned to Cambridge after the war at the beginning of September 1945, Lennard-Jones was still on war service. I was put in temporary charge of the Laboratory and soon afterwards, when it became apparent that Lennard-Jones would not be returning to his position as Director, I was confirmed in that office.

I had to get up to date and learn about the success of the Eniac and plans for the Edvac. Attending the later part of the famous Moore School lectures in Philadelphia completed my education and the future programme of the Mathematical Laboratory began to emerge in my mind while I was on my way back from Philadelphia. Because of the tight security, I did not hear anything about the Colossus until much later.

There was much talk at that time of very large scale and heavily funded projects for the construction of digital computers. I was not very seriously worried. My experience had shown me that in any large project, in industry or in government, the vital technical work was done by a

small group composed of a very few individuals. Most of the vast sums spent on such projects went into supporting staff concerned with corporate management, public relations, and so on. In the University such services were provided by the central organisation and not charged to individual projects. In any case the cost was, in that age of innocence and extreme academic decentralisation, very much smaller than would have been the case elsewhere. I felt that, with the resources available to the Mathematical Laboratory, we would be able to build a minimal, but useful, machine, with a well thought-out user interface.

I was myself fully informed about digital computers. As far as the Laboratory was concerned I had recruited the staff and taught them about computers, but they had not yet reached the stage at which they felt that they knew as much as I did, so I could take all the needful decisions myself. This situation changed as they came up to speed but, in the early days, it was a great help to me in getting the project off to a fast start and giving it the enduring stamp of a crash programme. In particular I had a free hand in the matter of architecture and the choice of circuits. On other occasions I have enlarged at some length on these decisions, as the pages of *Resurrection* show, and I will not now repeat what I said then.

I was proud of the input/output architecture. This was based on two instructions. One was an input instruction which transferred a single character from a five-hole paper tape to the memory and the other performed the corresponding action for output. These instructions enabled a bootstrapping routine – known as the *initial orders* – to be written and wired on to a rotating telephone switch. The initial orders were transferred to the main memory when the start button was pressed. The first sign that new members of the team were getting up to speed came with David Wheeler's work on the Initial Orders, for which he became famous.

From Building to Using

I have described the scattered community of leaders and students who were making progress in various areas of physical science in Cambridge by the use of numerical methods. I saw that the Edsac would be just what these people needed to turn progress at a snail's pace into something that would make the world sit up and take notice. It was plain to me that we could not do better than make the Edsac available to these people as soon as it became possible to have users from outside the laboratory.

In other words, we did not need to worry about where our first external users were to come from; they were already in place and waiting for us. I believe this would have been a sound plan anywhere, but it was particularly so in Cambridge where much more computing was being done than in most academic centres.

As time went on, some organisation became necessary, and we set up a Priorities Committee to select problems that were suitable for the machine and advise on how they should be tackled. As well as favouring humble research students, the plan was flexible enough to enable leaders of major research groups in radio astronomy and in molecular biology to get work done that helped them to win Nobel Prizes.

Not only did we make the Edsac available to research students from other departments in the University, but we welcomed them to the Laboratory and shared our accommodation with them. This created a real community in which experience could be shared and we were able to do it because, in consequence of Lennard-Jones's foresight, we had a capacious building. Naturally as the Laboratory itself grew, much of the space had to be reclaimed, but the sense of community and the habit of sharing experience happily survived.

Later Projects

The Mathematical Laboratory had created the Edsac and I had presided over its early success. Our next constructional project was Edsac 2 whose importance was that it demonstrated the viability of microprogramming as a design basis for a large computer. However, microprogramming did not come into widespread use until vacuum tubes had given place to transistors and read-only memory more suited to microprogramming had become available.

With these achievements behind us, we continued to look out for other projects of similar potential or projects which would, when they came to fruition, at least attract widespread attention in the industry. I can mention only one or two of these.

The Titan project provided the opportunity to introduce timesharing to Cambridge. In doing this we felt that we were working at the cutting edge of progress, although the basic ideas had come from outside. It was in this context that Roger Needham invented his system of scrambling passwords. The Titan (Atlas 2) computer was the first computer to have a cache, specifically an instruction cache. This was the joint invention of the project team, which included engineers from Ferranti Ltd. There is a

direct link between this work and the general purpose caches which later came into general use in the industry.

The CAP project also attracted great interest. One of its features was the design by David Wheeler, and the construction by laboratory staff, of a minicomputer in which capabilities were supported as a hardware type. However, the philosophy on which computers were designed began to change so rapidly and fundamentally that the less I say now about capability-based memory protection the better.

The final major project in which the Laboratory became involved in my time as head was the Cambridge Ring. In retrospect, I feel that, in importance and also in scale, this project came as close to the original Edsac project as one could hope for. To those who remembered the Edsac, it brought back some of the spirit and excitement of the early pioneering days.

Cambridge Ring

In January 1974 I visited the Hasler company in Berne, Switzerland. They showed me a ring based on the transmission of a megacycle bit stream along a twisted pair. They used this technology in a digital telephone application, but I immediately saw that the same technology had great possibilities for providing interconnection between computers and their peripherals and also interconnection between separate computers.

It is hard now to realise that such interconnection was then usually provided by equipment based on telephone technology and rented from the telephone company. I saw that it would be a major innovation, especially from the point of view of attainable bandwidth, to use equipment based on computer technology. It was obvious from work done with Ethernet at Xerox Palo Alto Research Centre that, as the move to desktop computers gained momentum, the need for wider bandwidths would be increasingly felt.

Much later the term *local area network (LAN)* came into use and the field became one of great activity. When the Cambridge Ring project was being launched there was little interest in local communications, and the term LAN had not been invented.

Before I could propose a major project, I needed to give some thought to implementation problems. Fortunately I had funds available that I could spend at my own discretion so that, if I could convince my

colleagues that the idea was a sound one, I would be able to go ahead without involving external funding agencies.

I needed first to verify that a station unit comprising a repeater and registers could be implemented without great cost. I consulted Chris Cheney, who worked in the Laboratory under David Hartley in the Computing Service, and who had experience with the then widely used 7000 series of MSI integrated circuits. His estimate of the number of parts needed was well within practical feasibility and quite close to the number used in the final product.

Next, I asked Norman Unwin to do some experiments to verify the essential viability of the proposed transmission system, with special reference to the repeaters. The results were reassuring, but inevitably, some anxiety would remain on the question of reliability until a working ring had been established and tested.

At this point, I felt sufficiently confident to define an architecture which, with some modifications, was finally adopted for the Cambridge Ring. The project then got under way.

I remained responsible for system architecture, with increasing participation by David Wheeler as time went on. Wheeler himself was responsible for the detailed design of the hardware. In the later stages, Roger Needham took an active lead in defining and developing the software which would turn a bare communication facility into a fully developed local area network. Soon a major part of the Laboratory became involved in one way or another.

In particular, David Hartley provided full support from the engineering organisation under his control. In the original Edsac project, fewer people were involved, and I was able to act personally as coordinator. For the ring project, I set up a coordinating committee with myself as chairman. All the senior hardware and software engineers in the Laboratory were members of this committee. Decisions were reached by consensus and progress was rapid.

As I have said, the project resembled that for the Edsac in that it had a hardware phase followed by a software phase. Since the coordinating committee included software as well as hardware engineers, there was no abrupt change in management during the transition from building to using. The same had, of course, been true of the Edsac project, and had been one of its main strengths. By the time I left the Laboratory towards the end of September 1980, there was in existence a working ring with some tens of computers and other devices attached.

Development of the Cambridge Ring continued after I left the Laboratory. This project demonstrated that advances in integrated circuits had made rings sufficiently reliable and low enough in cost to form the basis for a Local Area Network. From the operational point of view the Ring was a great success, and versions were manufactured and sold by various companies.

One of the devices on the ring was not a computer peripheral in the ordinary sense. It was a *pointing machine,* designed to assist an operator to put the wiring on a blank wire-wrapped circuit board of a kind then in widespread use for prototypes and low volume production. It would do this by pointing to the next pin needing the operator's attention and displaying instructions about what he was to do. The pointing machine was connected to the ring via a low level interface. The software needed to do the routing, to drive the pointing machine, and provide an interface to the operator was non-trivial. It ran in distributed fashion on several computers selected from those available on the ring. This made a very effective demonstration of a distributed application.

The pointing machine also proved so useful for its original purpose that it remained in use as a laboratory facility. Operating the pointing machine provided paid work for impecunious students who were willing to take on a tedious job in order to earn money.

This system was essentially Roger Needham's creation, with help from Andrew Herbert. When Needham took over from me as head of the Laboratory, he continued to develop the system. In due course, he and Herbert jointly published a book entitled *The Cambridge Distributed Computing System*, which described it in detail.

Editor's note: this article is a written version of an informal talk given by the author to the BCS@50 conference held at the BCS HQ in London on 13 July 2007. The author wishes to thank Professor Peter Robinson and Dr David Hartley for reading and commenting on a draft of this paper, and to acknowledge much appreciated help from Dr Hartley with the preparation of the final text.

Early Computer Production at Elliotts

Laurence Clarke

Rapid development of the base technologies and an emerging appreciation of the computer's potential made the nascent computer industry an exciting and demanding place to be in the early 1950s. The author recollects the frantic pace of life at the UK's first commercial computer manufacturer, Elliott Brothers.

In the early fifties people working in the computer field were limited in what they could achieve, yet inspired with the seemingly limitless possibilities which the digital computer might make available. It was an exciting and challenging environment for a new graduate such as me to find himself in, in 1950. It was *fun*.

At that time Elliott Brothers (London) Research Labs, led by John Coales, was the only non-academic, non-Government team developing its own range of computers. These machines, coming from a commercial stable, had by definition to be meeting a specific customer need or targeted at a commercial market. The direction of Elliott development was always heading towards computers being embedded in the application and away from concentration on stand-alone mainframes. Simon Lavington's research has shown that this stream of development, which spawned that of Ferranti in London, provided nearly as many UK computers in that period as all the rest put together.

This role of Elliotts (and of Andrew St Johnston in particular) has never received the recognition it deserved. I think this was caused in the first instance by the classified nature of the applications being addressed, and perhaps also by the public (and arguably BCS) concentration on "business data processing".

Before going into details of hardware or architecture it is worth spending a moment or two establishing the immense gulf between the capabilities of computers today and the tight constraints under which designers and users alike laboured in the 1950s.

The basic circuit frequency, and hence processing speed, has increased by a factor of at least 6000. The size of main memory has increased by 25×10^6. Programmers today could not imagine being restricted to 4000 bytes, which had to include what there was of an operating system. Main memory had (and still has) comparatively slow access speeds, so some type of random access memory was called for.

Here the transformation has been the most dramatic, with capacity increasing by 1.7×10^8.

With size and power consumption the comparison is the other way round, as of course it is with the question of cost. A straightforward cost/performance improvement ratio of 50 is transformed to a staggering level when taking into account inflation, giving a factor of about 5000 when comparing a 402 with a much more powerful laptop.

However, from the point of view of the user the most dramatic change has come in the user interface. In the early days the output was a display of either lights or blips of one sort or another on a cathode ray tube to signify binary digits. Numbers had to be inserted in binary – a far cry from the keyboard or touch screen of today.

With these immense disparities it is hardly surprising that many look on those machines as primitive and find it unbelievable that we should have dared to foresee computers being used for controlling all of life's activities as they do today. Nonetheless this was foreseen in the Reith Lectures given as far back as 1964, and in Leon Bagrit's book *The Age of Automation.*

Reliability was a key driver of development. In 1993 Ian Merry wrote in an article for the Computer Conservation Society:

"…most of those involved in the wartime radar development had been graduates in physics without academic engineering backgrounds since, in Britain at least, electronic engineering was widely regarded as a dilettante not to say insecure profession until well into the 1950s. This circumstance together with wartime pressures had confused the concept of design with the narrower field of circuit design, and established a widespread design tradition of 'suck it and see' whenever a problem arose outside the immediate experience of the designer."[1]

With the development of a limited set of digital packaged circuits, Charles Owen, and later I, began to use techniques introduced by Dummer at Malvern for 'tolerancing' components such as resistors to allow for maximum variation in performance without failure.

Thermionic valves were the usual binary element, but they had an expected life of only 1500 hours. This generated a demand to minimise the valve count in any machine (it could potentially mean one failure an hour in a machine using 1500 valves). The problem of valve failure was, to a considerable extent, ultimately helped by daily maintenance sessions

[1] *Resurrection* issue 7, page 29.

during which the heater voltage (which controlled cathode emissions) was reduced, causing premature failure during testing rather than during operational use.

Even in the 1950s the disparities in technology were beginning to be apparent. A decimal gas discharge tube called the decatron, which had a much longer life expectancy, became available, and the Atomic Energy Establishment made a machine using these devices. At what was probably the first national computer conference Cooke-Yarborough announced that he had just received the news, from Harwell, that his machine had run without failure from Friday night until Monday morning.

That seemed to be a significant feat of reliability. However FC Williams soon worked out that the Manchester machine of the day would have achieved the same amount of work in a minute or so, and it often worked for a few minutes without failure. So a balance between operational speed and reliability was a major consideration.

The need to locate failing valves and replace them led Elliotts, driven as they were by the requirements of both military and commercial users, to develop the system of standard plug-in modules which could be replaced easily and repaired offline.

These first, in 1948-49, used a glass substrate with silver deposited connections, plated through holes to get crossovers and deposited resistors. The active elements were germanium diodes and sub-miniature pentodes. The resultant computer was to be shipborne. All the techniques involved in making these ‘glass plates’, although commonplace today, were well ahead of their time, and in combination were not able to meet the rugged environment for which they were intended. They did, however, show the way to mount the small set of logic circuits developed by Charles Owen, using the same diodes and valves, for a second defence computer.

More importantly these logic elements formed the basis, when adapted to use the much cheaper twin triodes, on which the Elliott 400 range and the Ferranti Pegasus and its derivatives were all based. The logic design of these systems was greatly simplified by using a standard set rather than building up individual circuits.

For instance the logic system for the ‘Interference Test Set’ made to check the susceptibility of the 401 to environmental interference was designed by Andrew St Johnston in one evening: the design is recorded in pencil in his red book. The whole of the 401 was recorded in the same

way in his green book (now sadly missing!). Even the mechanical units were made in the local workshop from drawings taken from the green and red books.

The principal reasons for using plug-in units were the ease of maintenance and of replication in manufacturing, but a further benefit soon became apparent. In many ways they became analogous to a programmer's subroutines. The ease of designing new systems with different architectures allowed small teams to meet specific application requirements.

All of the subsequent 'Owen unit' derivatives were constrained to have a serial architecture and, as Ian Merry wrote in his 1993 article on Pegasus development:

"The thinking required in the logical design, particularly of the control functions, therefore required the designer to envisage successive machine states represented by circuit states changing autonomously and quickly under the inexorable flow of serial data. This is a duality which is difficult to represent conspicuously in any diagram form, and was quite beyond the descriptive mathematical techniques of the time."

Therefore designs were made to be as simple as possible. The Elliott Theory Group developed Nicholas with a main store using nickel delay lines in a similar way to Edsac's use of mercury tank delay lines. Nicholas was built to carry out a specific computation contract and, since the Theory Group did not have a local workshop, the team actually assembled the logic units themselves at home in their spare time. A serial architecture was used to minimise the number of components with a single address code, using a comparatively short store cycle.

The 401, which was being developed in parallel in the Elliott Computing Division with support from the NRDC, had a fixed head magnetic disc as main store with a cycle time of 13ms. This made it desirable to have a two-address instruction, where the specified address of the next instruction could be 'optimum programmed' so that rapid access was achieved.

This led to complex planning being necessary for the location of instructions and operands, but could achieve a 200 microsecond add time and 6.5 millisecond for multiplication. As an exercise for the crossword enthusiast this was exciting, but hardly productive when compared with the way that applications on today's systems are driven.

After the first demonstration of the 401 at the Physical Society exhibition in early 1953, Leon Bagrit made clear that the direction he

wanted Elliotts to take with digital computers was towards process control. When the exhibition closed the directors met the team at the company's London office and Dr Ross, the Hungarian Technical Director, pinned me in a corner and told me of the coming world when machines like the 401 would control "ze cat crackers". Tired out after the week's exertions (and loaded with gin) I was only dimly aware of the significance of what he was trying to get over, and its prophetic nature only became clear over time.

After the public exhibition the simplicity of design of the 401 was stretched with help from Christopher Strachey to make the programmers' job easier, in particular by introducing address modification (B-lines) using a few single word registers, and improving the double-length multiplier which originally catered for positive numbers only. However, even here Ian Merry's far-seeing thoughts on serial architecture complexity proved to be apposite.

In 1955 Colin Reeves was using at ICI a 402 with the same multiplier architecture, and came across a very infrequent error. None of the maintenance staff could understand what was going wrong and I, the designer, was in Australia with the 403 for a period of nine weeks. There were no fax facilities in those days and Colin's problem had to wait for my return. I found that, when 'optimally programmed' (with the next instruction using the result of the multiplication), the 64^{th} bit of the result was not shifted correctly, so an incorrect rounding error occurred – a perfect example of Ian's point.

The limitations on speed and storage were soon evident and larger drums were an obvious solution to the latter. A 330 kHz limit seemed to be placed on the speed of circuits. I do not know whether this was a real or imaginary limit at that time, or whether it was caused by the established investment in designs and hardware.

There was an established set of logic modules from which it was easy to generate new designs as potential customers came to us with their problems. The first, from overseas, was Professor Louis Couffignal from the Institut Blaise Pascal in Paris. His special requirement was easily met.

It was not acceptable in those times to have the first digital computer in France with an Anglo-Saxon name. A new term had to be invented and the machine is labelled a *Calculatrice Electronique*!

The next was Herr Doktor Marx from Ernst Leitz in Wetzlar, who required faster times for lens ray tracing. This could be achieved by adding a hardware floating point facility. This was done, though it

required a considerable number of extra logic units with cost and reliability implications. However, the effect on the delivery schedule, typically nine months, was small.

During this period two other much larger machines were developed and delivered. The first was the 403, created for the Long Range Weapons Establishment in Australia for analysis of missile range data. This machine had to be as fast as we could make it, so used 4-word nickel lines for its working store. In order to gain more speed it used what later (much later) became known as ‘pipelining’ to acquire and decode future instructions in parallel with implementation of the present orders. However in the case of conditional transfers the machine was not able to look ahead to the alternative address as well.

A much larger memory was provided on bigger magnetic disc and magnetic tape units. These were manufactured by Pye to the Cambridge University design by Donald Willis. An offline unit, the Output Converter, took the data from magnetic tape and controlled a Bull line printer and a series of plotters. These plotters were converted Mufax weather report receivers and were possibly the first plotters on a digital computer.

The machine dissipated about 15kw (bad news in the South Australian summer) and my first task on arriving in Australia to finish the re-commissioning was to switch it off, and, with great trepidation, tell the Superintendent that it was staying off until the air conditioning was in proper working order.

Borrowing many of the ideas of the 403, the 405 was designed to meet the requirements of the commercial market, which had been explored experimentally using a 402. It continued the concept of Autonomous Data Transfer from peripherals (although that term had not been coined). The magnetic disc, magnetic film, line printer and other input/output units each had assigned blocks of store with dual inputs. Access to the store from the computer was free unless a transfer had been initiated and was not yet completed *at the specified address*. Each device had a completely independent contro1ler.

The first machine was sold in 1958 to Norwich City Council for dealing with the rates, a very brave step by the City Treasurer, Mr Barnard. I should mention that this system was the first project for which Dina Vaughan (later to form Vaughan Programming Services, arguably the first software house) produced the programs (sadly Dina died in July). In all 30 machines were sold including one to the National Gas Turbine

Establishment for trials recording and analysis. This was the first of many Elliott online process systems.

These were challenging times. A team of six to 10 people had developed three very different machines in just 12 months. This was made possible by the ease of use of the Owen circuits, including rubber stamps for representing the logic elements.

At the time it seemed that a legion of opportunities was arising for using computers in online applications. One example investigated was in collaboration with The Central Fighter Establishment of the Royal Air Force. The use of a 402 to manage the recovery and marshalling of aircraft was demonstrated. Although this did not lead to an immediate installation it did pave the way to a later series of systems for controlling aircraft interceptions, itself an interesting example of enthusiastic systems engineers seeing unbounded possibilities in computer use coming against the tight restraints caused by the size/speed limitations of available technology.

A small team of analysts and programmers joined us with their salesman from Decca where they had been trying, unsuccessfully, to develop a large computer to control a hundred or so fighter interceptions. Meanwhile the RAF had no support for its Lightnings. We asked whether they thought that an 803, which was all we had, could control one interception. The answer was “yes”. Could it do two? Again “yes”. Three? This was doubtful. They were told to sell the idea of a number of systems to the RAF, which they did.

We were, therefore, all very busy and this had a fortunate by-product. We had no time to consider the use of point-contact transistors, and were getting acutely worried that our technology would become obsolete. However, resources were being stretched to breaking point and to keep the Elliott emphasis on track, an agreement was reached with the National Cash Register company for them to undertake sales and systems in the business data processing field. This deal gave us breathing space allowing us to begin considering a new technology – junction transistors – with the result that John Bunt developed the core-transistor logic circuits on which the 800 series were based, leapfrogging the point contact stage.

The 800 range still used serial logic to minimise the hardware and even, in the 802 and 803A, moved to a serial-serial arrangement with the accumulator and sequence control register being in the same circulation and using the same arithmetic unit.

The 803, highly successful as it was with over 200 deliveries, virtually marked the end of the discrete component era. The restrictions this article has addressed were removed by integrated circuits, which gave reliability and thus allowed complexity. They themselves became the logic elements in packaged form, until today we see them replacing whole units and even computers. The arrival of integrated circuits put an end to the intense excitement of exploring new ways to use the now limitless power of computers.

The 900 series machines varied both in mechanical form and in word length. These computers were used for applications in the process control field (communications, military, aviation, road and rail signalling) as well as stand alone computation, and were highly successful in their various fields.

In 1963 Leon Bagrit in preparing the Reith Lectures wrote "Perhaps the most far-reaching use of the new generation of computers will be in the retention and communication of information of all sorts within a national, possibly a worldwide information system". He was only enabled to make this prediction by *his* initial inspiration and our excited exploration of new fields and developing systems to verify his thoughts.

Postscript

Plus ça change, plus c'est la même chose!

Chris Edwards wrote in *Engineering & Technology* July 2007 that "The reliability time bomb has yet to detonate. No one has made a commercial chip that uses the 45nm process… It may take until the 32nm generation before big changes are needed to ensure that they work for more than a few months".

The article describes the way that the atoms in these small structures migrate in a random fashion which is reminiscent of the way that the silver connections in 152 plates migrated at high temperatures causing short-circuits and thus a complete rethink on packaging. So perhaps professional applications will have again to be constrained in their complexity and demand professional components rather than rely on the cheapest, consumer product components which are designed for the throw away society.

Editor's note: This article is an edited version of a talk given by the author to the BCS@50 conference on 13 July 2007.

The Origins of Fortran

Peter Crouch

This year marks the 50th anniversary of the public announcement of Fortran by IBM in February 1957 and the delivery of the first compiler, for the IBM 704 computer, in April of that year. On a more sombre note, 2007 also saw the death on 17 March of John Backus, who built and led the IBM team that created Fortran[2]. This article describes the thinking behind the development of Fortran and charts its progress over the early years.

The first Fortran compiler was a milestone in the history of computing. At that time computers had very small memories of a few thousand or even a few hundred bytes (it was common then to count memory capacities in bits); they were slow and had very primitive operating systems if they had them at all.

In the beginning the only practical way to program was in machine code. This was extremely tedious. Programmers required a detailed knowledge of the instructions, registers, and other aspects of the central processing unit (CPU) of the computer for which they were writing code. The source code itself was written in a numerical notation called octal.

In the course of time mnemonic codes were introduced, a form of coding known as assembly code. These codes were translated into the instruction words by programs known as assemblers. In the 1950s it became increasingly apparent that this form of programming was highly inconvenient, although it did enable the CPU to be used in a very efficient way.

These difficulties spurred a team led by John Backus of IBM to develop one of the first high level languages, Fortran. Their aim was to produce a language which would be simple to understand but almost as efficient in execution as assembly language. In this they succeeded beyond their wildest dreams. The language was indeed simple to learn, as it was almost possible to write mathematical formulae as they are written in mathematical texts (the name Fortran is a contraction of Formula Translation). This enabled working programs to be written faster than

[2] The home page of the BCS Fortran Specialist Group Web site has a number of links to obituaries on John Backus (www.fortran.bcs.org/index.php#jwb).

before, for only a small loss in efficiency, as a great deal of care was devoted to the construction of the compiler.

The pioneers of Fortran didn't invent the idea of writing programs in a high level language and compiling the source code to object code with an optimising compiler, but they produced the first successful high level language. They designed a language that is still widely used, and a compiler that produced very efficient code: the Fortran I compiler held the record for optimising code for 20 years!

Fortran pioneers gathered for a 25th anniversary banquet in 1982 include, from left, Richard Goldberg, Robert Nelson, Lois Habt, Roy Nutt, Irving Ziller, Sheldon Best, Harlan Herrick, John Backus and Peter Sheridan.

This first Fortran compiler was designed and written from scratch in 1954-57 by an IBM team lead by John Backus and staffed with innovators like Sheldon Best, Harlan Herrick, Peter Sheridan, Roy Nutt, Robert Nelson, Irving Ziller, Richard Goldberg, Lois Habt and David Sayre. Besides heading the group that designed the Fortran language and produced the first Fortran I and II optimising compilers for the IBM 704, Backus was system co-designer (with Gene Amdahl) of the 704, the first commercial computer with built-in floating point operations.

He was also a member of the design groups for the Algol 58 and Algol 60 programming languages. He proposed a language, based on the work of Emil Post, that Naur adapted to describe the syntax of Algol 60; it is now know as "Backus-Naur Form". He went on to develop a new class of "function-level" languages and the mathematics that applies to the programs of these languages.

Fortran was revolutionary as well as innovatory. Programmers were relieved of the tedious burden of coding in assembler language and were able to concentrate more on the problem in hand. Perhaps more important, however, was the fact that computers became accessible to any scientist or engineer willing to devote a little effort to acquiring a working knowledge of Fortran; no longer was it necessary to be an expert on computers to be able to write application programs.

The new invention caught on quickly. Programs computing nuclear power reactor parameters took now hours instead of weeks to write, and required much less programming skill. Another great advantage was that programs now became portable. Fortran was adopted by the scientific and military communities and used extensively in the space programme and military projects.

Fortran brought with it several other advances. It was, for instance, a language which remained close to and exploited the available hardware, rather than being an abstract concept. It also allowed programmers to control storage allocation in a simple way, a feature which was very necessary in those early days of small memories, though it is now regarded as being potentially dangerous.

Design considerations

The following excerpts are taken from the transcript of the film which IBM produced for the 25th anniversary of Fortran in 1982.

John Backus: *It was very informal so that, for example, when I thought that we should try to make this Fortran system, I simply wrote a letter to my boss at the time, Cuthbert Hurd, and suggested that we do that, and he said "Fine. Do it!".*

Cuthbert Hurd: *In August of 1952, the engineering model of the Defense Calculator, later named the 701, was running. And I think we had six orders. We invited a representative from each of the companies who had ordered the machine to come to Poughkeepsie for a week, and we each got a shot at this computer. You write this program and, BAM!, you get the result. And we all sat there and said, "How in the world are we going to keep these machines busy? They're so tremendously fast. How can we do that?".*

Irving Ziller: *And in the background was the scepticism, the entrenchment of many of the people who did programming in this way at that time; what was called "hand-to-hand combat" with the machine.*

By the early 1950s a number of approaches had been taken to 'automatic programming' as it was known. Some were what are now

known as assemblers while others simply provided a ‘synthetic’ computer which was much easier to program than its real counterpart but were in effect interpreters. All of the early ‘automatic programming’ systems were costly to use, since they slowed the machine down by a factor of five or 10. The most common reason for the slowdown was that these systems were spending most of their time in floating point subroutines.

This behaviour led to a great scepticism about ‘automatic programming’ in general as it existed in 1954, and about its ability to produce efficient programs in particular. Backus and Heising in 1964 emphasised how much the fear of not being able to compete with hand coded assembler code influenced the design of the first Fortran compiler. One effect of this was long compilation times, while the compiler tried to produce the most optimum code. This was particularly a problem as in the original Fortran system for the 704 (Fortran I), an entire application was a single Fortran program which was compiled in a single, often lengthy, run. This had to be repeated whenever there was a change to any part of the program, as frequently happened during debugging.

John Backus wrote: *Another factor which influenced the development of Fortran was the economics of programming in 1954. The cost of programmers associated with a computer centre was usually at least as great as the cost of the computer itself. (This fact follows from the average salary-plus-overhead and number of programmers at each centre and from the computer rental figures.) In addition, from one quarter to one half of the computer's time was spent in debugging. Thus programming and debugging accounted for as much as three quarters of the cost of operating a computer; and obviously, as computers got cheaper, this situation would get worse.*

This economic factor was one of the prime motivations which led me to propose the Fortran project in a letter to my boss, Cuthbert Hurd, in late 1953 (the exact date is not known but other facts suggest December 1953 as a likely date). I believe that the economic need for a system like Fortran was one reason why IBM and my successive bosses, Hurd, Charles DeCarlo and John McPherson, provided for our constantly expanding needs over the next five years without ever asking us to project or justify those needs in a formal budget.

The first version of the Fortran language, Fortran I, was not really designed. John Backus wrote that: *we simply made up the language as we went along. We did not regard language design as a difficult problem, merely a simple prelude to the real problem: designing a compiler which could produce efficient programs.*

Occasionally people did ask us how long it would take: "When are you guys gonna be done?". We would always say, "Six months. Come back in six months." We honestly felt all the time that we'd be done six months from now, but it turned out to be three years.

Birth of Fortran

The following two paragraphs are taken from the 1957 paper describing the first Fortran compiler.

The Fortran project was begun in the summer of 1954. Its purpose was to reduce by a large factor the task of preparing scientific problems for IBM's next large computer, the 704. If it were possible for the 704 to code problems for itself and produce as good programs as human coders (but without the errors), it was clear that large benefits could be achieved. For it was known that about two-thirds of the cost of solving most scientific and engineering problems on large computers was that of problem preparation. Furthermore, more than 90% of the elapsed time for a problem was usually devoted to planning, writing and debugging the program. In many cases the development of a general plan for solving a problem was a small job in comparison to the task of devising and coding machine procedures to carry out the plan. The goal of the Fortran project was to enable the programmer to specify a numerical procedure using a concise language like that of mathematics and obtain automatically from this specification an efficient 704 program to carry out the procedure. It was expected that such a system would reduce the coding and debugging task to less than one-fifth of the job it had been.

Two and one-half years and 18 man-years have elapsed since the beginning of the project. The Fortran system is now complete. It has two components: the Fortran language, in which programs are written, and the translator or executive routine for the 704 which effects the translation of Fortran language programs into 704 programs.

The following outline is based on the 1957 paper describing the first FORTRAN compiler.

In the paper the uses of the following types of FORTRAN statement were illustrated: Arithmetic; Function; DO; IF; GO TO; READ; PRINT; STOP; DIMENSION; FORMAT.

These examples were intended to give a representative picture of the Fortran language. However, many features had to be omitted. There were 23 other types of statement in the language, many of them analogous to some of those illustrated. They provided facilities for referring to other input-output and auxiliary storage devices (tapes, drums and card punch),

for specifying pre-set and computed branching of control, for detecting various conditions which might arise such as an attempt to divide by zero, and for providing various pieces of information about a program to the translator. A complete description of the language was given in the *Programmer's Reference Manual, the Fortran Automatic Coding System for the IBM 704*.

The Fortran translator consisted of six successive sections, as follows:

Section 1 read in and classified statements. For arithmetic statements it compiled the object (output) instructions. For non-arithmetic statements including input-output, it did a partial compilation and recorded the remaining information in tables. All instructions compiled in this section were put in the COMPAIL file.

Section 2 compiled the instructions associated with indexing, which resulted from DO statements and the occurrence of subscripted variables. These instructions were placed in the COMPDO file.

Section 3 merged the COMPAIL and COMPDO files into a single file, meanwhile completing the compilation of non-arithmetic statements begun in Section 1. The object program was now complete; it assumed an object machine with a large number of index registers.

Section 4 carried out an analysis of the flow of a program produced by sections 1 and 2, dividing it into ‘basic blocks’, which contained no branching, doing a Monte Carlo (statistical) analysis of the expected frequency of execution of basic blocks – by simulating the behaviour of the program and keeping counts of the use of each block – using information from DO and FREQUENCY statements, and collecting information about index register usage. This information was used by Section 5.

Section 5 converted the object program to one which involved only the three index registers of the 704.

Section 6 assembled the object program, producing a relocatable binary program ready for running. It produced a storage map of the program and data that was a compact summary of the Fortran output. It also obtained the necessary library programs for inclusion in the object program, including those required to interpret FORMAT statements and perform input-output operations. Also on demand it produced the object program in Share symbolic language.

Applications and user experience

The following two paragraphs are taken from the 1957 paper describing the first Fortran compiler.

The experience of the Fortran group in using the system has confirmed the original expectations concerning reduction of the task of problem preparation and the efficiency of output programs. A brief case history of one job done with a system seldom gives a good measure of its usefulness, particularly when the selection is made by the authors of the system. Nevertheless here are the facts about a rather simple but sizable job. The programmer attended a one-day course on Fortran and spent some more time referring to the manual. He then programmed the job in four hours, using 47 Fortran statements. These were compiled by the 704 in six minutes, producing about 1000 instructions. He ran the program and found the output incorrect. He studied the output (no tracing or memory dumps were used) and was able to localise his error in a Fortran statement he had written. He rewrote the offending statement, recompiled and found that the resulting program was correct. He estimated that it might have taken three days to code this job by hand, plus an unknown time to debug it, and that no appreciable increase in speed of execution would have been achieved thereby.

The language of the system is intended to be capable of expressing virtually any numerical procedure. Some problems programmed in Fortran language to date include: reactor shielding, matrix inversion, numerical integration, tray-to-tray distillation, microwave propagation, radome design, numerical weather prediction, plotting and root location of a quartic, a procedure for playing the game "nim", helicopter design, and a number of others. The sizes of these first programs range from about 10 Fortran statements to well over 1000, or in terms of machine instructions from about 100 to 7500.

John Backus wrote: *A report on Fortran usage in November 1958 said that "a survey in April (1958) of twenty-six 704 installations indicates that over half of them use Fortran (I) for more than half of their problems. Many use it for 80% or more of their work... and almost all use it for some of their work." By the autumn of 1958 there were some 60 installations with about 66 704s, and "... more than half the machine instructions for these machines are being produced by Fortran. Share recently (1958) designated Fortran as the second official medium for transmittal of programs [SAP (the Share Assembly Program for the 704) was the first] ..."*

From the advent of Fortran in early 1957, an extended Fortran called LRLTRAN became the most used programming language at Lawrence Livermore National Laboratory. It was typically used to compile compilers (Fortran-Fortran) and to maintain up-to-date software for succeeding generations of the Laboratory's large mainframes.

Westinghouse-Bettis was the first customer to receive the Fortran compiler in late April 1957. It arrived unannounced as an unmarked box of cards with no documentation late on a Friday afternoon, the Friday before a Share meeting. Westinghouse had a test program which had been written in an afternoon with the aid of the first *Fortran Programmer's Manual*. It was later estimated that the same program would have taken about two weeks to write in assembly language and another week or two to debug.

It was suspected that the anonymous box of cards was the overdue Fortran compiler. It was treated as such and loaded onto the 704 together with the source cards of the test program. After a few minutes of machine activity a single English language diagnostic of incredible specificity was printed, “SOURCE PROGRAM ERROR, THIS IS A TYPE-GO TO( ), BUT THE RIGHT PARENTHESIS IS NOT FOLLOWED BY A COMMA”. The compiler was right! After reproducing the card with a comma in the right place the program was recompiled. After some more computing around 28 pages of output were produced. Spot checking about 15 values suggested that the output was good to the six decimal digits printed. The fact that the newborn Fortran arrived on the last working day before a Share meeting – and that there was a workable test program that was ready to try the compiler – represented incredible, blind good luck.

Another tale of the early days of the Fortran compiler also involves Westinghouse. Frank Engel of Westinghouse, Pittsburgh noticed that none of the tapes on the 704 moved at the same time as another. So he decided he could improve the throughput if only he could overlap the tape operations. He requested a copy of the source code from IBM only to be told “IBM does not supply source code”. So Frank dumped the compiler in octal notation and patched it in octal, having decompiled it in his mind, and improved the throughput by a factor of three. An IBM representative on a visit noticed that the compiler was working very quickly and asked how it was done. Frank explained and a more senior IBM person came to see for himself. He was so impressed that he asked if IBM could have a copy, to which Frank replied “Westinghouse does not supply object code!”. (I am pleased to say that Frank Engel is alive and well, an

honorary Fellow of the BCS and a member of the Fortran Specialist Group.)

In the late 1950s Harvard Lomax, a pioneer in the field of computational fluid dynamics at NASA's Ames Research Centre, was modelling the behaviour of transonic airflows over the wings of jet aircraft. One afternoon in 1959 he was complaining to the other members of his carpool that his staff would have to redo a hand calculation which had taken days to complete because he had given them the wrong integral in the equation to be solved. One of the carpool was Marcie Chartz Smith, who was a machine-programming mathematician in Ames' electronic computing division and was responsible for operating an IBM 704. The next day she wrote the Fortran code for the equation that Lomax was interested in, ran the program on the 704 and delivered the results to him 15 minutes later. Lomax took a Fortran manual away with him and mastered the language in short order. This experience led him to design his theoretical investigations over the next 30 years so that they might benefit from the exploitation of the increasing computing capacity of digital computers.

Development

Fortran II, which was released in the spring of 1958, aimed to address a number of the shortcomings which had been revealed by field experience with Fortran I, such as improved error messages, separate compilation of subprograms, which could be in assembly language as well as Fortran, separate SUBROUTINEs and FUNCTIONs and the first appearance of the COMMON block.

At the same time Fortran III was being developed. It allowed one to write intermixed symbolic instructions and Fortran statements. In addition to this machine-dependent feature, which assured the demise of Fortran III along with that of the 704, it contained early versions of a number of improvements that were later to appear in Fortran IV. It had ‘Boolean’ expressions, function and subroutine names could be passed as arguments, and it had facilities for handling alphanumeric data, including a new FORMAT code ‘A’ similar to codes ‘I’ and ‘E’. It was made available to about 20 (mostly IBM) installations but was never distributed generally.

Fortran IV, which appeared in 1961, introduced named COMMON blocks, double precision and complex variables, logical variables and operators such as .AND., .OR. and .NOT., DATA statements and the ability to explicitly define the type of variables. Also many 704 dependencies were removed. The ‘Share Internal Fortran Translator’

(Sift) was written to enable users to convert their existing Fortran II programs to Fortran IV.

Fortran spread rapidly as it fulfilled a real need. Inevitably dialects of the language developed, which led to problems in exchanging programs between computers, and so, in 1962, a Fortran standards committee, operating under the procedures of the then American Standards Association, later the American National Standards Institute, and under the sponsorship of the Business Equipment Manufacturers Association, was formed to work on Fortran standardisation. This committee brought out the first ever standard for a programming language, now known as Fortran 66 after its year of publication.

The author (left) with then CCS chairman Roger Johnson at the Fifty Years of Fortran *seminar in January.*

Final thoughts

John Backus: *The really remarkable achievement that my friends made in producing the compiler is not well understood by the computer world in general. They set out to produce a compiler that would produce the most optimised programs possible. And they did this in such a superlative way by a remarkable group effort that this compiler produced*

the most optimised programs for the next 20 years. And when you compare that with other technological efforts, what other computer has ever survived for more than five years? What program stands as the best program for more than two or three years? One can hardly think of any technological effort that stands as the best work in its field for more than a few years. And this one stood for 20.

Frances Allen wrote: *The Fortran compiler established the standard for object-code efficiency. Perhaps more important, it established the feasibility of using high level languages. When I taught the Fortran class in 1957, there was tremendous resistance to using Fortran. That resistance was quickly eroded by the kind of code the compiler produced. I would like to conclude by quoting from a recent (1981) paper by Jean Sammet, in which she says, "Fortran has probably had more impact on the computer field than any other single software development." I believe that it established the feasibility of using high level languages.*

Editor's note: This article was prepared following a one day seminar entitled "Fifty Years of Fortran" presented by The Computer Conservation Society and the BCS Fortran Specialist Group in London on 25 January 2007[3]*. At this event an audience of some 60 delegates heard presentations from 11 speakers and a further six people made written contributions before and after the meeting (see www.fortran.bcs.org/2007/jubileeprog.php). Peter Crouch is Chairman of the BCS Fortran Specialist Group and may be contacted at* pccrouch@bcs.org.uk.

CCS Web Site Information

The Society has its own Web site, located at *www.bcs.org.uk/sg/ccs*. It contains electronic copies of all past issues of *Resurrection*, in both HTML and PDF formats, which can be downloaded for printing. We also have an FTP site at *ftp.cs.man.ac.uk/pub/CCS-Archive*, where there is other material for downloading including simulators for historic machines. Please note that this latter URL is case-sensitive.

[3] There will be a panel session with the same title at the SC07 supercomputing conference in Reno, Nevada, USA on the morning of 15 November 2007.

Obituary: Dina St Johnston

Simon Lavington

Aldrina Nia "Dina" St Johnston, pioneer programmer, died suddenly in the summer.

Dina joined the Borehamwood Laboratories of Elliott Brothers (London) Ltd. in 1953. There, she became responsible for program development for the 153 DF (Direction Finding) digital computer for the Admiralty. Later she was entrusted with writing the payroll program for the Elliott company. Dina was also responsible for the initial applications software for the Elliott 405 computer that was delivered to Norwich City Council in 1956, the first 405 to go to an external customer and the first electronic computer to be used by a local authority.

Dina has been described by a former colleague as "a formidable lady", in temperament more of an engineer than a mathematician. Another colleague has written: "As a programmer, Dina was unique. Not only was she inventive and structured, but also she was very accurate. She wrote with a Parker 51 fountain pen with permanent black ink and if there ever was a mistake it had to be corrected with a razor blade. Whereas the rest of us tested programs to find the faults, she tested them to demonstrate that they worked."

In the summer of 1958 Dina married Andrew St Johnston. She then left the Borehamwood Laboratories to start her own company, Vaughan Programming Services (VPS). Starting "with a dining room table and some paper" at her home, a converted pub called *Five Horseshoes*, VPS was considered to be the UK's first example of what later became known as a software house.

By 1966 VPS was employing 30 people, most of whom were programmers. She was ahead of her time in believing that computing could be for everyone – even in the world of intricate machine code and assembly programming.

The Vaughan company was sold in 1996 to Harman and traded as Vaughan Harman Systems, who sold out to GE Transportation Ltd, part of the General Electric empire, some years later.

Dina retired from active involvement with Vaughan Harman in 1999. She died suddenly at home at Hedgegrove Farm on the weekend of 30 June/1 July 2007.

Editor's note: This is a condensed version of an article scheduled to appear in the Computer Journal.

Letters to the Editor

Dear Mr Enticknap,

Reading the article by Graham Phillips gives me a suitable opportunity to recall that, together with the 'Computer Workshop' facility, the introduction of the 803B marked an occasion when likely users were able to develop various 'hands on' routines without direct assistance being immediately available!

This period cannot be acknowledged without referring to the provision of autocode program LC7B for plane frame analysis, as desribed in the report issued by the Institution of Civil Engineers on pages 201-214 in June 1966, written by Litton and Roper and made available by Elliotts at Borehamwood and elsewhere.

It is of some interest also to recall the use of a Zuse plotting table which was programmed to accept the 5-hole output tape to enable drawing of the frame outline with associated deflections and forces.

Yours sincerely,

Doug Brewster

Beckenham, Kent

1 July 2007

Contact details

Readers wishing to contact the Editor may do so by email to *wk@nenticknap.fsnet.co.uk*, or by post to 4 Thornton Court, Grand Drive, Raynes Park, London SW20 9HJ. Queries about all other CCS matters should be addressed to the Secretary, Kevin Murrell, at *kevin@ps8.co.uk*, or by post to 25 Comet Close, Ash Vale, Aldershot, Hants GU12 5SG.

Forthcoming Events

Every day Guided tours and exhibitions at Bletchley Park, price £10.00, or £8.00 for children and concessions. Exhibition of wartime code-breaking equipment and procedures, including the replica Bombe and replica Colossus, plus 60 minute tours of the wartime buildings. See *www.bletchleypark.org.uk* for details of times and special events.

16 October 2007 NWG seminar on the English Electric Deuce. Speaker Mike Wetherfield.

18 October 2007 London seminar on Cafs applications. Speakers Hamish Carmichael and Martin Wright.

20 November 2007 NWG seminar on artificial intelligence. Speaker Professor Max Brammer.

17 January 2008 London seminar on Electronic Publishing. Speakers David Penfold, David Brailsford and Conrad Taylor.

20 March 2008 London seminar on The BBC Micro and its Legacy. Speaker David Allen.

15 May 2008 CCS AGM, followed by London seminar on Programming and Software at Elliotts. Speakers Sir Tony Hoare, Jill Hoare and Jeff Hillmore.

Details are subject to change. Members wishing to attend any meeting are advised to check the Society Web site or in the Events Diary columns of *Computing* and *Computer Weekly*, where accurate final details will be published nearer the time. London meetings take place in the Director's Suite of the Science Museum, starting at 1430. North West Group meetings take place in the Conference room at the Manchester Museum of Science and Industry, starting usually at 1730; tea is served from 1700.

Queries about London meetings should be addressed to Roger Johnson at *r.johnson@bcs.org.uk*, or by post to Roger at Birkbeck College, Malet Street, London WC1E 7HX. Queries about Manchester meetings should go to William Gunn at *william.gunn@ntlworld.com.*

Committee of the Society

Chairman **Dr David Hartley FBCS CEng** Email: david.hartley@clare.cam.ac.ak
Vice-Chairman **Tony Sale Hon FBCS** Email: tsale@qufaro.demon.co.uk
Secretary, Chairman DEC Working Party and WebMaster **Kevin Murrell** Email: kevin@ps8.co.uk
Treasurer **Dan Hayton** Email: Daniel@newcomen.demon.co.uk
Science Museum representative **Tilly Blyth** Email: tilly.blyth@nmsi.ac.uk
TNA representative **David Glover** Email: david.glover@nationalarchives.gov.uk
Bletchley Park volunteers representative **Pete Chilvers** Email: petechilvers@beeb.net
Chairman, Elliott 803 Working Party **John Sinclair**
Email: john@eurocom-solutions.co.uk
Chairman, Elliott 401 Working Party **Arthur Rowles**
Email: rowles01@globalnet.co.uk
Chairman, Pegasus Working Party **Len Hewitt MBCS**
Email: leonard.hewitt@ntlworld.com
Chairman, Bombe Rebuild Project **John Harper CEng MIEE MBCS**
Email: bombe@jharper.demon.co.uk
Chairman, Software Conservation Working Party **Dr Dave Holdsworth CEng Hon FBCS** Email: ecldh@leeds.ac.uk
Digital Archivist & Chairman, Our Computer Heritage Working Party **Professor Simon Lavington FBCS FIEE CEng** Email:lavis@essex.ac.uk
Editor, Resurrection **Nicholas Enticknap** Email: wk@nenticknap.fsnet.co.uk
Web Site Editor **Alan Thomson** Email alan.thomson@iclway.co.uk
Archivist: **Hamish Carmichael FBCS** Email: hamishc@globalnet.co.uk
London Meetings Secretary **Dr Roger Johnson FBCS**
Email: r.johnson@bcs.org.uk
Chairman, North West Group **Tom Hinchliffe** Email: tom.h@dsl.pipex.com.
Dr David Anderson Email: cdpa@btinternet.com
Peter Barnes FBCS Email: panda.barnes@btinternet.com
Chris Burton CEng FIEE FBCS Email: cpb@envex.demon.co.uk
Dr Martin Campbell-Kelly Email: M.Campbell-Kelly@warwick.ac.uk
George Davis CEng Hon FBCS Email: georgedavis@bcs.org.uk
Peter Holland Email: peterholland@care4free.net
Dr Doron Swade CEng FBCS Email: doron.swade@blueyonder.co.uk

Point of Contact

Readers who have general queries to put to the Society should address them to the Secretary (see page 35 for postal address). Members who move house should notify Kevin Murrell of their new address to ensure that they continue to receive copies of *Resurrection*. Those who are also members of the BCS should note that the CCS membership is different from the BCS list and is therefore maintained separately.

Resurrection is the bulletin of the Computer Conservation Society. Copies of the current issue are available from the Secretary for £5.00 each.

Editor -- Nicholas Enticknap Typesetting -- Nicholas Enticknap
Cover design -- Tony Sale
Printed by the British Computer Society