# Mecury Autocode, Atlas Autocode and some Associated Matters.

**Vic Forrington**
***January 2014***

My first experience of the Manchester contributions to computing development was at the Royal Aircraft Establishment, Farnborough in 1959, where a Ferranti Mercury had just been installed. I had joined Mathematical Services Department from the Mathematical Laboratory at Cambridge where I had completed the Diploma in Numerical Analysis and Automatic Computing under Maurice Wilkes and the Edsac team. Incidentally Geoff Toothill, one of the original Manchester Baby team was at RAE at the time and I did a little bit of work with him on the concept of ternary arithmetic rather than binary, using the magnetic states of north, south and not magnetised.

Digital computing at RAE (yes, there were such things as analogue computers, they were very good at integration but not too good at arithmetic; indeed to seek the best of both worlds I worked for a time on the development of a digital differential analyser designed to simulate an analogue computer within a digital environment), was based on two English Electric DEUCE machines called Gert and Daisy, with memories based on drum storage. To get reasonable performance from a program one needed to know the position of the drum relative to the read/write heads so that an operand was passing just at the right time without the instruction having to wait another drum rotation. Thus programming was restricted to particularly masochistic individuals who numbered about twenty out of a scientific and engineering establishment of some 6000 souls.

Mercury, like Edsac 2, had a random access store and unlike Edsac had a drum backing store. It came with an assembler language, elegantly named Pig 2, with functionality similar to what I was used to with Edsac. It was assumed that Mercury and Pig 2 would deliver the Great Leap Forward in computing at RAE. Accordingly I was sent on a week-long Pig 2 course at Ferranti in Central London, meagre expenses paid, and instructed to initiate the computing revolution.

Mercury also came with a piece of software called Mercury Autocode (MA). At Cambridge high level languages, or autocodes as they were then called, were somewhat disparagingly described as playthings for amateurs with no real place in a professional world. I decided to play, and as a result soon abandoned Pig 2 as the basis for RAE computing and introduced MA as the standard throughout the establishment. Within a year there were some 250 active users of the Mercury and MA from all disciplines and departments. The age of the masochists was over!

I myself used MA for my 'real work', the numerical solution of large scale systems of ordinary differential equations involved in rocket guidance systems and never had to resort to a lower level language on either functional or efficiency grounds. It is my view that Tony Brooker and MA represented a paradigm shift in scientific and engineering computing. End users now had the power to chart their own destinies without the need for the intermediate programmers.

In 1961 I was appointed as a Research Assistant in the Computing Machine Laboratory,

then part of Electrical Engineering Department of Manchester University, where Tom Kilburn and his team were developing Atlas and Tony Brooker and his team were continuing their work on the Compiler Compiler. Tom had read mathematics at Cambridge before his war time work with FC Williams on radar who he subsequently joined at Manchester to design and built the world's first stored program electronic computer (the Baby). Tony had read mathematics at Imperial College, London and gone on to develop a relay- based arithmetic unit before moving to develop software at the Mathematical Laboratory at Cambridge where Maurice Wilkes had designed and built the world's second stored program electronic computer (the EDSAC). He then moved on to Manchester taking over from Alan Turing the development of programming methodologies and going on to his world-leading development of a high-level language, the Mark 1 Autocode which subsequently evolved into Mercury Autocode.. To be appointed to be part of a team, albeit a lowly part, led by these two giants of computing history was indeed a great privilege. However I was soon to be brought down to earth by Tony's opening remark when I first arrived. 'Welcome to the dustbin for failed mathematicians'!

Subsequently with both my Cambridge and Manchester backgrounds I was to learn that there was no love lost between the Mathematical Laboratory's 'let's get the theory right' approach and the Computing Machine Laboratory's 'let's get on and build something' approach.

I had been interviewed by Tony Brooker, who seemed somewhat surprised when I subsequently turned up. Nominally I reported to Tony although in those days there seemed no such thing as organisation structure or specific responsibilities; people got on with what they thought needed doing. In particular I helped out with the [University's] Mercury computing service, assisting users with the use of MA and often with the mathematical basis of tackling their problems, or numerical analysis as it was known. Many of the methods had been re-invented on the advent of computers without reference or even knowledge that Isaac Newton had originated them some 250 years earlier. However Newton could not summon 100 men for 100 years to undertake the rather tedious arithmetic which became easy meat for the 1960's electronic computer so the methods had soon been forgotten.

Users of all academic rank would stand in a queue awaiting their turn [for Mercury], to insert their paper tape into the reader and take their output tape to the printer, find that they had a syntax error, repair their input tape and rejoin the queue. Once when I was supervising the queue Professor FC Williams, co-inventor with Tom Kilburn of the stored program computer but small in physical stature, was involved in a minor altercation about queue conduct. He approached me afterwards and suggested that a large photograph of himself be affixed to the computer, pointing out his status in the computer world. He was a man of great humour as well as inventiveness who had ceased to be involved in computer design and going on to make fundamental developments in other areas of electrical engineering. Reputedly he claimed that the advent of the transistor had made computing move from invention to merely a branch of technology. I think that he may well have been right as far as basic hardware was concerned but that the Atlas Supervisor and the Compiler Compiler went on to demonstrate that fundamental software inventions would still take place.

I got involved with Atlas Autocode, working with Tony Brooker on the functional design of the language within Tony's very clear idea of the overall structure. He was keen, almost

too keen to incorporate any functionality I suggested might be useful, for example the extension of array definitions from conventional vectors and matrices to multi dimensional and sparse array definitions such as symmetric and tridiagonal matrices that were commonplace in linear algebra and partial differential equation computations. As well as functionality we looked at efficiency issues. For example in earlier work with Mercury Autocode I had reduced the running time of one program from eight hours to two hours merely by replacing divisions by four by multiplications by point two five (multiplication being hardware based and divisions by a software routine). So the AA compiler identified division by constants and replaced them with multiplications. However my main responsibility was to maintain a running document of the evolving functionality which served as both the specification of the language and eventually the first programming manual.

Tony went off to IBM in the USA for a mini sabbatical where as he told me later, nobody seemed interested in what he was doing, only in taking him away from his rival Manchester developments. So as usual he did his own thing, including the implementation of a multi-length arithmetic system. (He had after all pioneered software-based floating point arithmetic well before the Manchester implementation of hardware floating point.)

Even before AA became a working reality I was conducting programming courses to potential users, including a dozen or so heads of department and professors at Edinburgh University which had signed up to use the Atlas Computing Service when it became live.

One of the few times I had to resort to low-level coding, as opposed to autocodes, was to incorporate a standard routine (Runge-Kutta) for solving ordinary differential equations into the Atlas fixed store, which already had so many demands on it that it was more like getting a gallon into a pint pot rather than the traditional quart. I sidled up to Dave Howarth, the software guru of the fixed store who asked me how much space I would require. I told him 50 – 100 words; he sighed and said he could perhaps find 20 for me, to which I eventually agreed. He came back with 20 non -contiguous absolute addresses. He said I could have them until Monday, otherwise he would have to give them to someone else. It ate badly into my cricket and drinking weekend and caused me to get down to some shocking machine level coding but of course it had to be done. Not as an order but as an expectation based on pride rather than duty, as was much of the Atlas development work. Looking back on it however I suspect that my routine was never incorporated, nor should it have been as the efficiency gains would have been trivial as the great bulk of the processing would always have to be done by specific user-written code outside of the fixed store.

I rarely had need to venture into the computer room while Atlas was being built by the joint University and Ferranti team under the ever watchful eye of Tom Kilburn puffing on his pipe while sitting silently in his arm chair. An inspiring but somewhat fearful sight. I was however asked to help out by providing a program that would demonstrate the brute arithmetic power of the Atlas for the scheduled Grand Opening [in December 1962]. As AA was not yet available I wrote a simple Assembler program to solve Laplace's equation in a square, which is an iterative process, the number of iterations necessary being determined by the fineness of the mesh. By default this rather trivial demonstration became a major part of the presentation, the reason being that the Atlas arithmetic unit at that stage of its commissioning had a mean free error time of only a few seconds, causing many more

complex programs than mine to crash. However, because mine was an iterative process and only the arithmetic would fail rather than the logic, it meant that the program would eventually finish successfully. Hopefully nobody noticed that the number of iterations required for completion was highly variable!

One of my roles had evolved into writing programs which would demonstrate the sheer computing power of Atlas, and in some cases identify hardware and system software faults. In particular Frank Sumner made use of the variable number of iterations in my Laplace program to track down and rectify a problem with B-register operation under certain circumstances. As well as research and demonstration programs I continued to be involved in user problems, one particular one emanating from 'downstairs' in Electrical Engineering concerning the transient performance of polyphase induction motors. With the standard numerical method for solving the equations Atlas required a run time 50 minutes per case, illustrating that even with arguably the world's fastest arithmetic unit there were problems likely to be beyond its capacity, and warranting further development not just of computing power but of mathematical methods of solution. In this case a re-formulation of the problem reduced the run time to under five minutes and the development of an entirely new method of numerical solution reduced it to fifteen seconds.

Perhaps indicative of how the joint team operated was when at the end of the Grand Opening, Tom Kilburn, followed by Peter Hall, the Ferranti head of the project, led us all into the College Arms, not a particularly familiar territory to Tom but very well known to many of his team. However in his best Yorkshire diction Tom ordered 50 pints of best bitter and, turning to Peter said firmly 'and he's paying for it!' And as transpired as the evening wore on, he paid for many more. Years later, as the College Arms was demolished along with most of the Manchester we knew, it was pleasing to learn that stone engraving designating the pub was now incorporated into one of the walls of the brand new computer room [in the Kilburn Building].

By the time I left the department in 1964, Atlas and AA were fully operational and Mercury had been decommissioned. I had entered the more lucrative but not quite as exciting world of international management consultancy, although returning later on for a three year stint as a Visiting Lecturer when Tom had set up a full blown Department of Computer Science and introduced the first ever undergraduate degree course in computing. True to his background he insisted that this first course had a high engineering content, in addition to the more usual mathematical and software topics to be seen in the Computer Science courses that were to follow at many other universities.

The last time I saw Tom was [in 1998] at the 50$^{th}$ anniversary of the Baby, where of course he was the star of the show. He told me, without any false modesty that he could not understand why such a fuss was being made of this first ever stored program computer, which he said had only taken up six months of his time. He was much more proud of having set up the first computing department and the first degree course. That, however, will probably not be what he will be remembered for by computer historians.

Like other people at the time I felt that Tony had made a mistake in not including Algol as a subset within AA, irrespective of the sub-set's potential inefficiencies. Perhaps he thought this would have presented technical difficulties but more likely that it would have been inelegant by his exacting standards. However when it eventually went live on Atlas, AA

quickly became the de facto standard programming language at Manchester and those other bodies using the university computing service. I subsequently introduced it to clients of the consultancy company I had joined, based on the London University Atlas and also developed a high level language for inventory control applications that was implemented entirely in AA.

With the exception of Edinburgh University, where it flourished (as Edinburgh IMP) for many years, I believe that Atlas Autocode did not prosper outside the Atlas environment for two main reasons: a desire by Ferranti to retain Mercury Autocode and develop Extended Mercury Autocode for upwards compatibility within its product line, and a prejudice against Manchester University developments by a new breed of computer science academics who saw a universality in Algol which was never realised. However the Atlas Supervisor and the Compiler Compiler became the basis for major systems developments throughout the computing world.

*[My thanks to Simon Lavington for inviting me to make this contribution and for his encouragement throughout the process, not least in putting some of my recollections in more accurate historical perspective]*

Vic Forrington *(left)* and Jeff Rohl *(right)* at a graduation ceremony in 1963.