ELLIOTT 900 SERIES SIMULATOR

ALGOL SYSTEM: SYSTEM FILES AND DEMONSTRATION PROGRAMS.

There is a directory for each version of ALGOL (903 ALGOL, MASD ALGOL, 920 ALGOL, HUNTER and AJHALGOL) containing copies of the system paper tapes suitable for loading under initial instructions. There is also a directory for ALMAT and ESP.

In addition, each directory contains a suite of demonstration programs for the system in question. The files have embedded commands for the benefit of the simulator. If you write a program you should adopt the directives from whatever example seems appropriate.

Most of the ALGOL versions follow the Issue 6 conventions for accessing interpreter state (i.e., FP=138), but the 903 and MASD load-and-go systems follow the Issue 5 conventions (i.e., FP=38, etc).

903, MASD, HUNTER and AJH ALGOL fail when run on the 920M processor due to the different side-effects of B-modification and Jumps on the Q-register between the two models.

Note that 920 ALGOL requires text input to be in 920 telecode and binary input, both in mode 1.

DIRECTORY: ALMAT

ALMAT(ISS2).900 - "900 ALMAT ISSUE 2"

ALGOL Matrix package source, issue 2.

DIRECTORY: ESP

ESP1(ISS2).900 - "ESP 900 TAPE 1 ISS 2 Copy 502"

ALGOL source code for the Elliott Simulation Package.

ESP1(AJH).903 is a tidied version of ESP1(ISS2).900 used in the ESP demonstrations.

ESP2(ISS2).RLB - "ESP TAPE 2, ISSUE 2 Copy 259"

ELLIOTT 900 SERIES SIMULATOR

ESP2(ISS2A).RLB – ESP TAPE 2, ISS 2A 16-3-72 Copy 502"

ESP2(ISS2).RLB and ESP2(ISS2A).RLB are the intermediate files for the random number generator used by the Elliot Simulation Package. ESP2(ISS2) is for versions of ALGOL up to and including issue 5, ESP(ISS2A) for later issues. These files are created by using ASSEMBLE.DAT to assemble ESP(ISS2).900 and ESP(ISS2A).900, the corresponding sources.

DIRECTORY: HUNTER

DEMO1 is just a table of squares.

DEMO2 is Knuth's Man or Boy test, running up to k=11. It takes about twenty-three minutes on a 903 (3 seconds on a 2.6GHz dual core Pentium PC) and uses about 40K or 50K words of store. The crux is at k=4. For each successive value of k the recursion depth and evaluation time double. The recursion depth reaches about 1024 in both the A and B real procedures. Notice the patch at the front of DEMO2 to inform the Hunter ALGOL system that 64K of store is available.

DEMO3 is Knuth and Merner's program that prints several primes in one statement. It has been arranged to print 10 primes to a line and ten primes in all, but could easily be edited to do more. If so, the first line takes a couple of minutes and a second line about 17 minutes on a 903. Remember that output occurs only when a complete line is ready. To see progress you can declare another integer, say t, set it to zero at the start, and increment and print it inside the integer procedure G.

DEMO4 is taken from an article by Bryan Higman but adapted to evaluate PI by integrating one eighth of the volume of a sphere. In this example the fact that local variables are in static storage is evaded by holding them in the procedure called LOCAL. There is a lot of use of parameter comments where this construct:

)a comment :(

boils down to a comma.

DEMO5 illustrates the use of lower case characters and reading input data. Note that data can follow immediately after the last end of a program.

DEMO6 shows special characters in strings. The same program is used to read in strings containing the special characters of 900 and 903 telecode respectively in two successive runs of the program. Note use of AT command options to switch from the default 900 telecode to 903 telecode.

DEMO7 shows library operations. It illustrates several features:

1. Loading of relocatable binary code containing additional procedures in addition to the standard library after the ALGOL text.

2. The declarations code ... algol that declare the headings assembly-coded procedures supplied as relocatable binary code.

3. The relocatable binary code itself. Each integer represents one row of tape. About ten blank rows (zeros) precede and follow each section.

4. The run-time loader halt code is three integers to make up one 903 word. Special things happen if the word is non-zero (if less than 8 then set the word into the loader options, or else treat the number as an address and start loading at that address).

Note that the procedure REVERT is declared but not called and therefore not loaded; FLOOR is declared and called in the outermost block and therefore it is loaded; ADVANCE, BUFFER and DECODE do not have to be declared as they are built in to the translator, but as they are called, they will be loaded.

DEMO8 is adapted from the Pascal program to calculate the rising and setting times of the Sun and Moon given in the book, Astronomy on the Personal Computer, by Montenbruck and Pfleger. It takes nearly two minutes to print each output line on a 903 and just 3 to 4 seconds on the simulator. It illustrates several features:

1. An initial patch to the translator to make the code compact, but to disallow call-by-name extension. The default is to have "5 86" instead of "8 86" at location 8273. (This facility is only provided by Hunter ALGOL).

2. Loading of relocatable binary code containing additional procedures in addition to the standard library after the ALGOL text.

6. It gives an idea of the maximum program size. Compare the address for NEXT with the bottom of the Name List, which starts at 7512, but remember that this grows downwards during translation when names are discarded at the end of a block. Each name needs four words.

DEMO9 is Quicksort, unchanged but for the declaration of local variables and even the layout of the original has been more or less followed. It just sorts an array of ten real numbers into order, but the data tape could be changed to alter the array size. It uses the Wichmann-Hill random number generator.

DEMO10 calculates the function mentioned in Knuth's Art of Computer Programming Vol. 2 Ed. 2, for estimating the size of a factor base in either the Continued Fraction or Multiple Polynomial Quadratic Sieve factorisation programs. Be careful when running it with an increased number of steps as the run time goes up as the square of this.

DEMO11 shows the use of an early 2-pass version of Hunter ALGOL. This version of Hunter ALGOL can be run in an 8K machine.

DEMO12 is just DEMO9 with an entry at 11 (translate in report mode and run) rather than 8 to produce a table of addresses for procedures and blocks in the program to help with decoding error messages. The use of this information is explained in the section on run-time errors.

DEMO13 illustrates the addition of an Elliott 4100 card reader and line printer as devices for use in ALGOL read and print statements. A set of code procedures is first defined in SIR to initialize the devices and set up the interrupt system. The devices are handled by the standard QCARDIN/QCINCH and QLPOUT routines (see 903SIR DEMO13 for more information about these routines). The code procedure lprint sets up the printer and cardin sets up the reader with an integer array for its buffer (the card reader is single buffered in this example). Note in addition to the use of read and print statements, the advance, decode, instring and outstring library routines are demonstrated with these devices. Note also that card input does not include newlines, reading the last column of the previous card is followed by the result of reading the first column of the next card.

DEMO14 is taken from the KDF9 ALGOL User Manual, Appendix 1, "A Specimen ALGOL Program". The program computes a numerical solution to an equation modelling the propagation of an

impulse in a viscous-locking medium taken from a paper of that title by J.W. Miles, A.S.M.E Trans. Series E, J. Appl. Mechs. March 1961, 21-24. Note that the program contains two means of computing of the complementary function. The procedure erfc is the code from the KDF9 manual that uses the trapezoidal rule for integration. The alternative procedure erfca uses a much faster approximation taken from Wikipedia.

To allow for comparison, the version of this demo in the HUNTER directory uses erfc and takes about 45 minutes of 903 time whereas the one in the HUNTERP director uses erfca and runs in 5 minutes, nine times faster!

Six further programs, CORNU.DAT, MOON.DAT, SUNDIAL.DAT NAVIG.DAT, LOOP.DAT and WALK.DAT demonstrate the use of the plotting library. Note that these all use the 900 plotting model as in Don Hunter's simulator, plotting to a VGA screen. (If plotting to the 903 model is required, the 903 ALGOL library tape (Tape 3) should be loaded in place of the HUNTER ALGOL library.)

CORNU.DAT plots a Cornu spiral.

MOON.DAT plots the trajectory of the Apollo 8 mission under simplifying assumptions. MOONQUIK plots the same trajectory using a table of pre-computed coordinated.

SUNDIAL.DAT produces a graph for correcting local time (as measured by a sundial) to astronomical time at different times of year. SUNDIAL is quicker to run than CORNU or MOON.

NAVIG.DAT plots the ellipses within which a ship is likely to lie with various probabilities, given four position lines from star sights.

LOOP.DAT shows the path of an aeroplane doing a loop the loop, under simplifying assumptions.

WALK.DAT just does a random walk of 1000 steps.

The ten programs COPY.DAT, DET.DAT, SIGMA.DAT, SIMPS.DAT, SOL.DAT, TAN.DAT, EXAM1, EXAM2 and EXAM3 demonstrate the use of libraries of pre-compiled procedures.

MODULE.DAT is a script for running the MODULE program described in the section on libraries).

TAN.DAT is a script to pre-compile a tan procedure that calls the standard sin(x) and cos(x). Following translation of the source code, the TAN.DAT script then runs MODULE.DAT on TAN.BIN to create an intermediate relocatable binary tape suitable in the file TAN.RLB for including in a library. The resulting intermediate code has been included in LIBRARY.BIN.

EXAM1.DAT uses the pre-compiled tan(x) procedure. Note the use of an inaccessible call to a procedure qatrig to force sin and cos to be loaded even though they are not explicitly mentioned in EXAM1.

SIMPS.DAT is a script to pre-compile the simps procedure from DEMO4. Note the tricks used to build up constants like 2 and 3 from 0, 1 and 3. The resulting intermediate code has been included in LIBRARY.BIN.

EXAM2.DAT is a version of DEMO4 that uses the pre-compiled simps procedure created by SIMPS.DAT.

DET.DAT and SOL.DAT are scripts to pre-compile two procedures for fitting some normal equations to the data using code written by T.J. Dekker at the Amsterdam Mathematisch Centrum in 1963. DET computes the determinant on the N-th order matrix given in the array a[1:n,1:n]. The first index of the array elements is the row, the second is the column index as usual. The method is triangular decomposition according to Crout with row interchanges. This process yields a lower triangle L and a unit upper triangle U such that L*U equals matrix A with interchanged rows. Together with each row interchange the sign of the non-pivotal row is reversed, so that the determinant equals the product of the diagonal elements of L. At each stage the pivot is chosen in a column of L such that its modulus divided by the Euclidean norm of the corresponding row of a is maximal. The integer array P[1:N] is an output vector in which the pivotal row indices are recorded. The elements of A are replaced by the corresponding calculated elements of L and U. So enough information is retained for subsequent solution of linear systems and for matrix inversion.

SOL is adapted from AP205 written by T.J. Dekker at the Matematisch Centrum in 1963. SOL replaces the vector given in array b[1:n] by the solution vector X of the linear system L*U*X = B with interchanged (and possibly sign reversed) elements. Here L is the lower triangle and U the unit upper triangle which are given in array lu [1:n,1:n] such that the elements with first index < second index are elements of U and

the other elements of lu are elements of L. The integer array p[1:n] defines the interchanges with sign reversals of the elements of B in correspondence with the pivoting administration in DET. Hence a call det(a,n,p), followed by a call sol(a,b,n,p), has the effect that b is replaced by the solution vector X of the linear system sigma (a[i,j]) * X[j]) = B[i]. The procedure SOL leaves the elements of lu and p unaltered;

COPY.DAT and SIGMA.DAT are scripts to pre-compile a couple of toy procedures copy and sigma required by EXAM3.DAT.

EXAM3 is a real program for finding out whether a billing program ran faster over night or during the day.

An input row to the program consists of the number of attempted invoices divided by 100, the number of generated ones divided by 100, the number of generated ones when run by day divided by 100, or zero when run by night, and finally the run time in seconds divided by 100, all for a particular day. The division by 100 keeps the problem from giving floating point overflow!

A pair of asterisks in the output signifies a daytime run. You can see that it did run faster by day.

Note that the calling program declares 8 - reals because DET has local variables. Additionally, EXAM3 pretends to call sqrt because sqrt is called within det, and would not be loaded otherwise.

In addition to the above demonstration programs, the directory contains a test programs TESTS.DAT that checks the implementation of various fixes to problems in 903 ALGOL.

The first test shows how HUNTER ALGOL can be patched to handle both 903 and 900 telecode correctly. (Similar patches were used to make MASD ALGOL from 903 ALGOL by T.J. Froggatt.) The test shows the use of different quote symbols and curly brackets as string quotes, and the correct treatment of two alternative positions for the sterling symbol.

The second test checks handling of overflow in integer comparisons. The result is a run time error.

The third tests checks resumption after a failed call to the sqrt function: each failed call results in a runtime error and the file contains directives to jump to entry 9 for

continuation. This test program also includes a check on the handling of integer procedure parameters.

Note all these tests succeed.

System Files.

M0.DAT and M1.DAT are memory images for module 0 and module 1 of store, in binary format copied from Don Hunter's simulator.

ALG16KLG(HUNTER).BIN is an image of a paper tape version of Don Hunter's ALGOL system which corresponds to the memory images in M0.DAT, M1.DAT.

LIBRARY.RLB: an extended version of the standard Elliott ALGOL library tape (tape 3), including plotting functions, the standard built-in functions and a number of pre-compiled ALGOL procedures. (In Don Hunter's ALGOL distribution, this file is called LIBRARY.DAT. It conforms to the coding conventions for ALGOL issue 6 and later. Note that it has an incorrect RLB checksum, which is only a problem if you attempt to use this file to re-establish the library).

To facilitate editing the library it is in binary format. Each procedure in LIBRARY.RLB starts and ends with a few blank lines, and the procedures are in alphabetical order. A comment in brackets names the relevant procedure.

The file ends with the sequence 112 0 0 representing the loader halt code. Nothing should appear after this because it would be treated as further input data to the ALGOL translator.

Although most of the library is the original code derived from assembling procedures written in SIR, there are about half a dozen compiled from ALGOL, namely: COPY, DET, SIGMA, SIMPS, SOL and TAN. To confirm the contents, the library loading map can be seen by executing:

```
ATTACH PTR FILE LIBRARY.RLB
JUMP 18
```

The code in the library can be explored using the PRINT command, which will produce a readable version of the relocatable binary format content, or RLBTOSIR to obtain a representation in the form of a SIR program.

MODULE.DAT is the script used in creating libraries from translated ALGOL procedures. It takes two parameters: the first is the name of the input file of intermediate code and the second is the name of the output file of intermediate code suitable for adding to a library.

System Files

ALG1(HUNTER).BIN is a version of the 903 ALGOL Issue 6 Translator (Tape 1) with the Hunter modifications, dated 30/12/76. (It does not include the removal of the requirement to declare labels).

ALG2(HUNTER).BIN is a version of the 903 ALGOL Issue 6 Interpreter (Tape 2) with the Hunter modifications, dated 30/12/76.

LIBRARY.RLB is a library tape assembled by Don Hunter comprising the standard Elliott issue 6 tape 3 and additional code procedures written by Don. Most are in fact separately compiled ALGOL procedures as per the examples explained above. There are three additional machine code procedures:

unsigned: do not output a space or - when printing a integer.
revert: restore normal integer printing.
floor: compute the floor of a real number.

DIRECTORY AJHALGOL

This directory is a copy of the demonstrations in the HUNTER directory but using AJH ALGOL.

The systems files are:

ALG1(AJH).BIN: ALGOL Tape 1.

ALG2(AJH).BIN: ALGOL Tape 2.

ALG16KLG(AJH).BIN: Load-and-Go system for a 16K machine.

ALG16kLP(AJH).BIN: Large Program system for a 16K machine.

ALG64KLP(AJH).BIN: Large Program system for a 64K machine.

LIBRARY.RLB: A modified version of the HUNTER file LIBRARY.RLB with the unsigned and revert procedures replaced by four new procedures:

unsigned: disable printing a space or - before an integer.
revrtu: restore normal sign printing.
leadzero: show leading zeros when printing an integer.
revrtl: restore normal printing of leading zeros as spaces.

DIRECTORIES 903ALGOL, MASDALGOL

These directories contain parallel versions of the same demonstrations for standard and MASD versions of 903 ALGOL respectively. The demonstrations are different to those used for HUNTER ALGOL because neither 903 ALGOL nor MASD ALGOL support the language extensions provided in HUNTER ALGOL. However, all the 903/MASD ALGOL demonstrations will run under HUNTER and AJH ALGOL.

There are twelve demo programs:

DEMO1.DAT: Calculating the roots of f(x) = cosh x + cos x - 3 = 0 using a Newton-Raphson iteration. Demonstrates the "load and go" system. A version of this program can also be found for FORTRAN II (903FORTRAN DEMO3.DAT) and FORTRAN IV (905FORTRAN DEMO5.DAT).

DEMO2.DAT: Compares calculating sines using Taylor's series with the built-in function sin. The Taylor series version cannot handle large angles and eventually generates a real-time overflow. Build using the 2-pass system. Note use of entry at 14 to combine translation with reports. A version of this program can also be found for FORTRAN II (903FORTRAN DEMO2.DAT) and FORTRAN IV (905FORTRAN DEMO6.DAT).

DEMO3.DAT: A derivative of DEMO5 for Hunter ALGOL that prints a simple table of sines compiled using the two-pass system in "library mode". This version uses a for ... while loop rather labels. Note: since 903 ALGOL requires labels to be declared before first use, the Hunter DEMO5 program would need a switch directive to be added to run under 903 ALGOL.)

DEMO4.DAT: Computes the first 20 Fibonacci numbers. Demonstrates the use of the "large program" system, including the facility to dump out a loaded program as a binary tape.

DEMO5.DAT: Shows the use of the ALMAT matrix package and check functions and large program mode. After testing the various

members of the ALMAT library the largest size test matrix that can be made to fit in 32K is inverted, which takes about 20 minutes on a PC.

DEMO6.DAT: Shows use of the ESP (Elliott Simulation Package), including sampling distributions, manipulating histograms and a discrete event simulation of a simple queuing system. Note that it loads ESPB.RLB (the ESP random number code procedures) as a library because the code procedures reference interpreter state (see section on using the Large program system).

Note that only the 32K version of Pontefract ALGOL can run this very large program. MASD and Issue 6 ALGOL are only available as a 16K system and therefore run a cut down version of the demo.

DEMO7.DAT: Also shows ESP, using the example given in the Elliott 900 Programming Information Manual, Volume 2C.

DEMO8.DAT: This demonstration illustrates the code procedure facilities for accessing formal parameters and producing results. Examples are given of scalar and array parameters, call-by-name and call-by-value, label, switch and string parameter. There is also code to show how processor interrupts can be used in ALGOL programs.

DEMO9.DAT: This demonstrates building in code procedures to the 2-pass ALGOL system translator and interpreter. The translator is updated by compiling a code procedure declaration and then dumping out the updated system. The interpreter is updated by loading the assembled SIR code for the procedure as a library (by entering at 12) and then dumping out the updated system. A simple test program that uses the code procedure is then compiled and run using the modified translator and interpreter.

DEMO10.DAT: This program illustrates the use of the simulator facility for plotting using the Elliott 903 ALGOL plotter functions, rather than the modified versions due to Don Hunter. The program is a recreation of a demonstration program written by Eric Wright at the Computer Unit, Medway and Maidstone College of Technology in Kent that plots a number of interesting mathematical curves in a pleasing layout. Interesting aspects of the ALGOL code are the use of memorization to speed up plotting of the scalloped border and the careful calculation of angular increments when plotting in polar coordinates to minimize the number of polar to XY conversions calculations made. Note this program uses the

ALGOL Issue 7 library with Terry Frogatt's revised plotting routines.

DEMO11.DAT. This is the same as HUNTER ALGOL DEMO13 and shows how to set up a card reader and line printer as an additional device in the ALGOL Interpreter. The details are explained in HUNTER DEMO13. The demo has two versions. DEMO11A uses the two-pass ALGOL 6 and issue 6 library conventions, whereas DEMO11B uses the 16K load-and-go system that follows the issue 5 library conventions.

DEMO12.DAT this program shows the use of the MTALGOL library for magnetic tapes. It is equivalent to 903 FORTRAN DEMO12, itself based on 903 FORTRAN DEMO6 (Solving simultaneous equations using the Gauss-Siedel method). DEMO12 is intended to work on very large sets of equations with up to 400 unknowns and therefore the matrix of equation coefficients is too large for an 8K or 16K store). Instead each row of the matrix is written as a block on tape and read in when needed.

The script starts by initializing a tape using MTINIT, the runs the main program. The program first opens the tape and the coefficients of the equations to solve are input and copied to it using raput to store real numbers in the tape buffer, mtbuffer, which is an integer array. The input format is as for 903 FORTRAN DEMO6, but requires that the data be presented in row order (column order within rows can be arbitrary).

Once the data has been read, the program executes the Gauss-Siedel iteration using mtread to read in each block as required. Coefficients are read using raget to extract real numbers from mtbuffer. Since tape blocks are numbered, starting at block 1 for the header, data block (n+1) will correspond to row (n) of the matrix of coefficients. By asking mtread to find a numbered block, the tape will be automatically rewound after every iteration.

When the iteration converges, the results are output and the tape rewound.

Other files

TESTS.DAT: the same tests as in HUNTER ALGOL.

System Files

DIRECTORY 903ALGOL

This directory contains the 903 ALGOL tapes.

ALG1(ISS6).BIN: "903 ALGOL TAPE 1, SCB ISS 6 Copy 502"
ALG2(ISS6).BIN: "903 ALGOL TAPE 2, SCB ISS 6 Copy 502"
ALG3(ISS5).RLB: "903 ALGOL TAPE 3, (LIBRARY) ISSUE 5 Copy 259"
ALG3(ISS6).RLB: "903 ALGOL LIBRARY, (RLB) ISS 6 Copy 259"
ALG3(ISS7).RLB: "900 ALGOL III LIBRARY, RLB ISS 7 Copy 502"
ALG3(TJF).RLB: a version of ALG3(ISS7).RLB with plotting functions added by Terry Froggatt (see DEMO10 above) "ALGOL ISSUE 7 LIBRARY 4/11/12, RLB Mode 3".
ALG16KLG(ISS5).BIN: "903 ALGOL 16K, LG ISS 5 Copy 502"
ALG16KLP(ISS6).BIN: "900 ALGOL 16K (LP) LOADER/INTERPRETER, SCB ISS 6 Copy 502".
MTALGOL(ISS3).RLB: "MTALGOL ISS 3 Copy 502".
RAPUTGET.RLB: RLB version of an implementation of raput and raget written by the author and Terry Froggatt.

Note that the LG system is Issue 5 and adheres to the Issue 5 conventions for code procedures. The 2-Pass and LP systems all assume Issue 6 conventions.

Note that the Issue 7 version of the library lacks the plotter routines.

Note that MTALGOL requires an Issue 5 Interpreter.

DIRECTORY MASDALGOL

MASD ALGOL is a derivative of 903 ALGOL Issue 6. The only difference is the support for { and } as string quotes.

ALG1(MASD).BIN: "903 ALGOL TRANSLATOR 1/1/74".
ALG2(MASD).BIN: "903 ALGOL INTERPRETER 1/1/74".
ALG3(MASD).RLB: "903 ALGOL LIBRARY 1/1/74".
ALG3(ISS5).RLB: "903 ALGOL TAPE 3, (LIBRARY) ISSUE 5 Copy 259".
ALG16KLG(MASD).BIN: "903 ALGOL 16K LOAD-&-GO 1/1/74".
ALG16KLP(MASD).BIN: "903 ALGOL 16K LONG PROG 1/1/74".

Note that the load-and-go system is Issue 5 and adheres to the Issue 5 conventions for code procedures whereas the two pass and large program systems are based on 903 ALGOL Issue 6. A suitable library for use with the load and go system is ALG3(ISS5).RLB in the directory 903ALGOL.

ELLIOTT 900 SERIES SIMULATOR

DIRECTORY 920ALGOL

This directory contains a version of ALGOL that uses 920 telecode. It is only available as a two-pass system. It uses the issue 6 linkage conventions for code procedures.

There are three demo files:

DEMO1: A minimal "Hello World" program.

DEMO2: Computes exponentials using Taylor's series.

DEMO3: Essentially a duplicate HUNTER\DEMO6.DAT, demonstrating use of the 920 telecode character set.

System files

ISS6).BIN: the translator "920 ALGOL TAPE 1, ISS 6 (SCB)". Note this file must be reads in Mode 1.

ALG2(920).BIN: the interpreter "920 ALGOL TAPE 2, ISS 6 (SCB)". Note this file must be read in Mode 1.

ALG3(920).RLB: the library "920 ALGOL LIBRARY, TAPE 3 [900 ALGOL LIBRARY ISSUE 7]". Note this file must be read in Mode 1.