Issue 2 November 2004

Ferranti Mercury.
X4 Software and sample programmes (Ferranti preferred spelling).

D5. Software Technical Details.

Programme Libraries.

Mercury had a comprehensive set of library routines[1] held on sectors 0 to 63 of the main store (drum). Those on sectors 0 and 1 were regarded as important enough to be read only. Some subroutines (including sqrt, cos, log, tan, radius, sin, exp, arctan) were called **quickies** and had code called from the drum and held in computational store. This reduced operation time from 23 mS to 6 mS. Notionally there are 1000 library routines including I/O routines, printing numbers, reciprocal, reciprocal square root etc. They were grouped as follows.

0 - 49	Output of numbers.
50 - 99	Non-numerical output.
100 - 149	Input of numbers.
150 - 199	Non-numerical input.
200 - 219	Roots, powers.
220 - 239	Exp, log etc.
240 - 259	Hyperbolic functions and inverses.
260 - 279	Other functions of one variable.
280 - 299	Other functions of more than one variable.
300 - 319	Quadrature.
320 - 339	Interpolation and curve fitting.
340 - 359	Inverse interpolation; zeros of polynomials
360 - 379	Power series.
400 - 419	Ordinary first order differential equations.
420 - 439	Ordinary differential equations, not first order.
440 - 459	Other ordinary differential equations.
460 - 479	Partial differential equations.
500 - 509	General purpose linear algebra.
510 - 529	Linear equations.
530 - 549	Eigenvalues and vectors.
550 - 579	Other special purpose matrix operations
580 - 599	Linear programming.
600 - 629	General purpose aids to coding.
630 - 649	Complex numbers.
650 - 679	Multiple precision.
680- 699	Subroutines for programmed arithmetic routines.
850 - 899	Test routines.
900 - 999	Miscellaneous - Checking, organisation, non-numerical..

As examples, logarithm or exponential took 5 mS to execute.

Applications Packages.

In addition to the above, Ferranti held a library of programmes written by their customers and made available for others users[2]. The reference contains a brief description of 237 programs under 11 headings. By far the greater number (105) are in the Mathematics section. Other sections include:-

Aircraft Industry (2)	Atomic Physics (4)	Commercial and DP (3)
General Engineering (3)	I/O conversion(25)	Nuclear Engineering (29)
Operational Research (10)	Organisation and checking of programmes (22)	
Statistical (28)	Special subjects (6)	

The solution of 112 simultaneous equations is quoted as taking 16 minutes.

"Operating System".

Mercury had no operating system as such. Programmes were punched on paper tape and entered by the use of hand switches as detailed below. In Manchester a 'bootstrap' was written to reduce the use of hand switches, described by its author as 'a dreadful hack'. The Oxford machine (and others?) had some sort of operating system known as PIG (details please).

Operating Instructions.

Complete control**:-**

WRITE CURRENT switch to ON
Inhibit stop key is OFF unless STOP is required in the programme
Set bottom row of hand switches as required by input routine
Insert tape into tape reader
Press clear tape button
Set Auto/Manual and Single/Continuous to required mode
Press initial transfer

Modes:-

Auto, Single	Only 1 instruction
	Pressing the prepulse button reads the next instruction (i.e. single step the programme).
Auto, Continuous	Operate normally.
Manual, Single	Instruction on the top row of keys run once.
Manual, Continuous	Instruction on top row of keys run repeatedly.

Languages.

There was an assembler language which enabled programmes to be written in a more readable form. Examples are shown below.

There was also a Mercury Autocode, developed from an earlier Mark I Autocode. Later it was developed to become Atlas Autocode and Algol in its various versions. Examples below.

Programming.

Machine code[3].

Fixed Point: $-1 \leq x\ 1$ with the point after the first digit. The 'sign' is added to the fraction (2's complement).
Floating Point: $x2^y$; x is the argument; y the exponent. x is in standard form in the range ½ to 1. $-256 \leq y < 256$ with the two most significant bits the same (for overflow detection). Zero is expressed as 0.2^{-256}. The half words are H_0, the exponent, and H_1 - H_3, H_3 being the most significant.

Words are expressed as (e.g.) L28 = M28, M29 = H28, H28+, H29, H29+

There are a notional 100 functions (instructions), 00 to 99 - see function code in X3.

Floating point arithmetic is rounded by making the least significant bit of the argument 1, which is biased. Functions are provided to permit unrounded arithmetic - useful for multilength etc. (Note:- The modern floating point standard requires that in the default case 1 be *added to* a bit *one less significant* than the least significant of the argument *sometimes*. The standard has a total of four rounding modes).

Drum transfers - see example below. They take 128 B-instructions → 7¾ mS. There is a further 960 µS between sectors allowing two sectors to be read consecutively.
Switches allow half the drum to be write protected. Sectors 0 - 63 hold library routines (I/O etc). Routines on sectors 0 and 1 cannot be overwritten in any normal way.

Some routines called Quickies can be held in the computational store (see list below).

'Open' subroutines are held on master tapes and copied to programme tapes as required. 'Closed' subroutines are entered and exited by jumps with the return address being held in B_1.

Programmes are divided into chapters of up to 15 pages. A subroutine may not cross a chapter boundary.

An instruction may refer to a label by means of a v-number - e.g. 181 v1, and also to non-labelled instructions, e.g. 2v4 is register 2 beyond label 4 or -3v4 is register 3 before label 4.

Page 0 contains some useful fixed information as follows;
0 +0 2 -1 Useful constants, held as long numbers (so no 1 or 3)

Four instructions giving access to the chapter changing sequence

4	670	479
5	690	0
6	670	478
7	680	0
8	597	0 (return)

Four instructions giving entry to the error print.

9	670	479
10	690	0
11	670	5
12	680	0

In addition there are 25 preset parameters, x1 - x25, which can be set by (e.g.) x10 = 15. x10 is 15 until set again.

Directives.

C	Chapter	S	Sector	P	Page
T	Title	F	First sector	R	Routine
Q	Quickie	E	Enter followed by address of first instruction.		
W	Wait	L	Line - correct a programme without repunching.		

The console has a row of switches that allows manual programming. It also contains six lights: signs of B and sac (B_7) and the 99 instruction - stop. Two CRTs act as a display which can show B-lines 1 to 7, the exponent and argument of the accumulator, Present Function, Control register and the argument of the number being read from store to the arithmetic system.

Autocode[4].

Variables; maximum of 480 main variables in a maximum of 15 groups. v -> 479 are v_0 - v_{479}.
15 special variables; a - h, u - z, π. All these except π can have a primed version. i - t are indices in the range -512 to 511.
Numerical: 9 or 10 decimal digits (29 binary digits).

Example: 2mna(m + 1) + amn + man means $2 * m * n * a_{m+1} + a_m * n + m * a_n$

There are 10 functions of one variable and three functions of two variables, designated by ϕ, e.g. y = ϕsqrt(any expression). Others include integer part, polynomial, max(x,n,m) (being the maximum element of x_n - x_m) and min(x,n,m). The two variable functions are division (x/y), arctan(y/x) and sqrt($x^2 + y^2$).

Repeats: i = p(q)r means from p by q to r. q can be negative.

Large programmes are divided into a maximum of 832 chapters each with its own labelling. An ‘across’ instruction enables jumping to/from chapters; e.g. Across 2/3 means jump to instruction labelled 2 in chapter 3 (takes 160 mS).

Jumps: jump 1, $x \geq 0$; jump 2; 1 and 2 are labels.

Read number can take several forms. Printing uses a ? before or after an instruction to calculate a quantity.

Chapter 0 is placed at the end of the programme followed by close which starts the programme running.

```
Either  ch0                              Or      ch1
        f → 180                                  f → 180
        n = 4(1)20                         1)    prog
        a = 0.25n                                up              to repeat
        prog                                     close
        repeat                                   ch0
        end                                      n = 4(1)20
        close   transfers control to n =..       a = 0.25n
                                                 down 1/1        to label 1)
                                                 repeat
                                                 end
                                                 close           transfer as before
```

This covers basic facilities. Further facilities include:-

Quickies
Auxiliary variables (up to 10,752)
Complex numbers
Integration of differential equations
Pseudo random number generation
Programme library
Rounded/unrounded
Preserve/restore in use of subchapters
Double precision
Alpha-numeric input
Matrix operations

There are some special features on the Manchester and ICI machines only.

Short integers and long numbers can be listed at the head of a chapter with special preceding symbols.

Long numbers in fixed point style preceded by + or -.

Generation of a sequence of rectangularly distributed pseudo random numbers.
Generation of a sequence of normally distributed pseudo random numbers.

An additional three matrix operations.

Additional instructions 78, 90 - 97, 11, 31^5. Details not listed except for
11 B' = CA + 1 + n (internal code 003) and
31 B' = S = n (internal code 103).

Examples of machine code programmes.

1**. The reciprocal library routine. A is replaced by its reciprocal.**
June 1956 (before the machine was delivered); MSIM reference F2 series 6 Box 18/5 supplemented.

The following includes John Gosling's comments supplemented by Joan Travis and combined with those in the Ferranti document.

1. This routine uses a Newton-Raphson iterative process based on the iteration

$$y_{n+1} = y_n (2 - xy_n)$$

With a suitable starting value y_0 three iterations were needed. Note that this is one of the best algorithms for computation of reciprocal and has been used in many machines (e.g. IBM 360 model 91).

2. The argument, D, must be in standard (normalised) form, $a2^p$, and the approximate reciprocal is $y_0 = b2^q$.
As a c [1/2, 1), then 1/a ∈ b' c (1, 2]. Let a = 1 - z. Then 1/a = 1/(1 - z) which is approximately 1 + z. This is greater than 1 and must be shifted down one place and the exponent increased to compensate. Thus q = 1 - p. Since z = 1 - a, b' = 2 - a and b is 1 - a/2.

The value of y_0 is derived from a straight line approximation to the hyperbola y = 1/D. The line is ideally chosen to cut the curve in two places between D = ½ and 1 making the relative error between curve and line equal at the two ends of the range and the maximum between the two intercepts. This leads to an equation y = -mD + c where neither constant is an integer. m is approximately 2, so if it is made equal to 2, mD becomes a simple shift. c is then found by making the relative error the same at D = 1 and the maximum value between the two intercepts. The value of y_0 is 4(sqrt(3) - 1) - 2D = 2.9282 - 2D. Shifted down one place this becomes 1.4641 - D. If the integer 750 is represented in 10 bits and then interpreted as a number with one integer bit, the number '750' is 1.46484375, which is the nearest to 1.4641 in 10 bits (line 4 below).

If a is negative, the required value of y_0 can be obtained by subtracting 2.9282 from the value otherwise obtained (line7).

The expression is chosen to ensure y_0 is sufficiently close to 1/x to enable three iterations to suffice. The exact y_0 is not critical and b is found to 10 digits only. The number of accurate bits doubles in each iteration. This approximation is accurate to only about 4 bits.

3. The last iteration is different to reduce round off error. $xy_2 < 1$ so the exponent = 0. If $(2 - xy_2)$ is formed directly it has exponent of 1 and one bit of xy_2 is lost before the multiplication. $1 - xy_2$ is very small and is corrected after the final multiplication.

0	410	32	L32= A;	Store x in L32; four half words, M32, M33	
1	300	1	sac = 1		
2	230	32	sac = sac - p;	q = 1 - p; sac is 10 digits so this reads only the most significant half word of 32.	
3	210	34	L34 = q	Again, the most significant half word.	
4	300	750	sac = 750	Interpreted as 1.46484375.	
5	230	33+	sac = sac - 33+	MS 10 digits of argument mantissa subtracted.	
6	490	8x	if A ≥ 0 control to 8		
7	330	476	sac = sac - 476	subtract 2.9282 if a is negative.	
8	210	35+	sac to MS 10 digits of long word at 34; This is the most significant 10 bits of b, the rest being zero.		
9	107	-1	set count		
10	510	34	A = -A x L34	$-xy_0$	
11	430	20x	A = A - (-2)	$2 - x_n$; 20x is the 20th location beyond the beginning of the programme and must contain -2. This is more accurate than adding +2, since $+2 = 1.2^2$, whereas $-2 = -1.2^1$ and xy_0 is close to 1, so in floating point there is less rounding error.	
12	500	34	A x L34	$y_1 = y_0(2 - xy_0)$	$y_2 = y_1(2 - xy_1)$
13	410	34	L34 = A	y_1 to L34	y_2 to L34
14	510	32	A = -AL32	$A = -xy_1$	$A = - xy_2$
15	187	11x	If Bt ≠ 0 control = 11; $B_7 = B_7 + 1$ (= 0)		Bt = 0; go to 16
16	430	22x	A = A - L22	$1 - xy_2$; 22x must contain -1. See note 3	
17	500	34	AL34	$y_2(1 - xy_2)$	
18	420	34	A = A + L34	$y_2 + y_2(1 - xy_2) = y_2(2 - xy_2)$	
19	590	24x	control = 24	stop (beyond the end of the programme	
20	=1, =0			20x = -2	
21	=0, =512				
22	=0, =0			22x = -1	
23	=0, =512				
24				see 19**.**	

2. Write the contents of pages 4 - 8 into sectors 115 - 119 of the main store (drum) and replace the contents of pages 4 - 8 by the contents of sectors 120 - 124.
Programming Manual list CS 158 July 1957; MSIM reference F2 Series 6 Box 18/12.

3.56	107	-4	$B_7 = -4$ set count in B_7
3.57	677	119	sector register T = 119 modified by -4 = 115 (first time)
3.58	697	8	Sector 8 + B_7 = 4 (first time) to sector T
3.59	187	3.57	$B_7 \neq 0$ control = 3.57 $B_7 = B_7 + 1$ (else continue)
3.60	107	-4	
3.61	677	124	Sector register = 124 - 4 =120 (first time)
3.62	687	8	Sector T to page 4 (first time)
3.63	187	3.61	B_7 g 0 control = 3.61 $B_7 = B_7 + 1$ (else continue)

To input an Autocode library tape.

On main tape put title
Programme n

All block isolation switches DOWN
Tape in the reader
Key 2 of the bottom row of hand switches UP
ITB What does this mean?
Switch on continuous
Tape reads - continuous hoot
Switch off to single
Isolate switches 0 - 3 on Drum 0

To input an Autocode tape.

Reset all stores to standard state to Sectors 0 - 31 and 80 - 127 inclusive, then isolate
Tape in reader
Bottom row of hand switches all zero (normal input)
Or Key 4 UP if printing using the ? prefix required
ITB
Switch on continuous

Programme translated and entered on reading a starting chapter 0.

Autocode examples. List CS270 July 1960; MSIM reference F2 Series 6 Box 4/22

1. Calculate $\mu = \dfrac{td^2 (r - q)g}{18L (1 + \frac{2.4d}{D})(1 + \frac{5d}{3L})}$

Let h = t; d1 = d; d2 = D; u1 = r; u2 = q; x = L

Numerator a = hd1d1u1g - hd1d1u2g
(better:- p = u1 - u2; a = phd1d1g - JBG)
Denominator factors:- b = 1 + 2.4d1/d2 c = 3 + 5d1/x
Denominator:- d = 6xbc
μ= a/d

2. Load locations a_1 - a_{100} from tape

i = 1(1)100
Read(ai)
Repeat

3. Find the three largest values in a_0 - a_{100}

```
i = 0max(a0,1,100)          ai = greatest
b1 = ai                     store ai
ai = -999999999             make ai very large negative
j = ϕmax(a0,1,100)          aj is second greatest
b2 = aj                     store aj
aj = -999999999
k = ϕmax(a0,1,100)          ak is third largest
ai = b1
aj = b2                     Reset ai, aj
newline
print (ai, 3, 6)
print (aj, 3, 6)
print (ak, 3, 6)
```

4. Print the powers of 2.

Note: 1. Register 0 contains floating point +0 and register 2 holds -1.0.
2. Quickie 9 punches A fixed point as described below.

```
T                           Title
PRINT POWERS OF 2
C1                          Chapter 1
R6                          Routine 6
103     1                   B3 = 1; set count, n
400     v1      (5          A' = Long number in label 1 = +0 initially, 1 in second pass..
450     2                   A' - L unrounded. Reg 2 contains -1; this is, therefore
                                A' + 1 = 1 first time...
410     v1                  Store A; L' = n + 1; 1 first time, 2 second..
101     *                   B1 = address of this instruction
590     v3                  control = v3 (Q9 here; to punch A fixed point)
=2, =0                      parameters for Q9 = print 2 integer and 0 fractional digits
                                preceded by line feed etc
400     2v1                 A' = L , the second register after v1 = 1 initially, 2 second time
440     2v1                 A' = A + L; 2 first time, 4 second..
410     2v1                 store A; 2 first time, 4 second...
101     *                   B1 = address of this instruction
590     4v3                 control = 5th instruction in Q9; prints without line feed etc
=9, =0                      9 integer, 0 fractional digits
173     26                  Bt = B3 - 26; Repeat if B3 not equal to 26
183     v5                  If Control not 0, B3 = B3 + 1; 2 first time
990     0                   stop
+0      (1                  n
=1, =0, =0, =256            2^n; 1 initially
Q9      (3
591     3                   control = B1 + 3 lines on from where quickie code is entered
                                (101    *)
EV/6                        Enter v routine 6
```

List of Quickies.

1	A’ = 1/A	A must be standardised; error if $\|A\| < 2^{-253}$
2	A’ = 1/íA	A must be standardised; error if A Ÿ 0
4	A’ = e^A	error if $e^A > 2^{256}$
5	A’ = tan A	
6	A’ = sin A (cos A if entered at 2nd instruction)	
7	A’ = cos A	
*8	Punch sac	integer in range -512 to +511.
*9	Punch A fixed point	enter with 101 * 590 - =m, =n where m,n are number of decimal digits before and after the decimal point.
*10	Punch A fl pt	enter with S = number of decimal digits.
*11	Punch Sac +	unsigned integer in range 0 - 1023.
12	A’ = sqrtA	error if A < 0.
14	A’ = $\log_e$A	error if A Ÿ 0.
15	A’ = arctan y/x;	y = L32, x = L34 error if 0/0; range 0 - 2.
16	A’ = arcsin A	error if |A| > 1; range = /2, + /2.
18	read integer to S	integers beginning with +, - or decimal digit, terminating with CR or Sp; FS, LF, ER ignored and also CR and Sp between numbers.
19	read fixed or floating number to A	Form sign, int part, point, frac part, comma sign exponent CR LF or Sp Sp. FS ER Sp (single) ignored, also CR LF Sp between numbers.

- Each number is preceded by FS CR LF CR and terminated by Sp Sp.

Simulator.

No working simulator known at present

D3. References.

[1] *Library Index*. Ferranti List CS 93 June 1956; MSIM reference F2 Series 6 Box 18/9.

[2] *Programmes Available in the Interchange System*. Ferranti List CS 218B 1962; MSIM F2 Series 6 Box reference 4/16.

[3] *Programming Manual*. Ferranti List CS 158 July 1957; MSIM reference F2 Series 6 Box 18/12.

[4] RA Brooker, B Richards, E Berg: *Mercury Autocode Manual*. Ferranti List CS242A, July 1961; MSIM reference F2 Series 6 Box 4/21.

[5] *Internal Function Codes.* Ferranti List CS 188A, MSIM reference F2 Series 6 Box 4/11.

Reciprocal Library Routine. MSIM reference F2 Series 6 Box 18/5; June 1956 (before the machine was delivered).

Autocode Examples. Ferranti List CS 270 July 1960; MSIM reference F2 Series 6 Box 4/22.