903/920 SIR
SUBROUTINES

Book No. 108
Copy No. 93
Amendment No. 5

AIRBORNE COMPUTING DIVISION
ELLIOTT FLIGHT AUTOMATION LIMITED

PREFACE.

This book describes the following tapes:—

QS I/O	15/12/69	900-Series	Telecode
QSMATH	15/12/69	"	"
QDLA	15/12/69	"	"
QDMATH	15/12/69	"	"
QF	17/6/71	"	"
QF I/O	17/6/71	"	"
QF MATH	20/12/69	"	"
CH I/O + LEGTAPE S/R	31/3/70	"	"
CH I/O S/R	31/3/70	"	"
OPTIMISED MATHEMATICAL SUBROUTINES	1/12/71	"	"
SHELLSORT	17/8/71	"	"

The first seven of the above tapes enable single-length, double-length, and floating-point calculations to be performed on any 900-series 18-bit computer except, possibly, a 920A. (The suitability of these tapes for a 920A is uncertain). Their functions are summarised in the following table :-

	Single-length	Double-length	Floating-point
Arithmetic operations (+, −, ×, ÷)	Provided by computer	QDLA	QF
Input & output of data	QS I/O	QDLA	QF I/O
Mathematical functions	QSMATH Sine Cosine Square Rt. Arctan Log Exp	QDMATH Sine Cosine Square Rt. Arctan	QFMATH Sine Cosine Square Rt. Arctan Log. Exp.

These seven tapes all use a subroutine called ERROR to give error indications; QSI/O QDIA and QFI/O also use character input and output subroutines called CHIP and CHOP. Since these three subroutines are NOT contained within the seven tapes, the seven tapes may input and output data (and error indications) in ANY telecode via ANY peripheral; by providing suitable subroutines.

The next two of the tapes described in this book contain ERROR, CHIP and CHOP subroutines (for use with the above tapes or in any other program requiring a flexible input & output system) for input and output of data (and error indications) in 900-series or 920 telecode, (and, in the case of CH I/O + LEGTAPE S/R only, output in legible tape), via a punch and reader. These two tapes ARE suitable for use on a 920A, and use reader mode 3.

The tape "OPTIMISED MATHEMATICAL SUBROUTINES" duplicates some of the functions of QSMATH; but uses less store, less time, and is more accurate. (QSMATH has been retained for its other functions and for compatability with earlier programs).

"SHELLSORT" is a general sorting program for fixed-length records held in store.

The above eleven tapes are normally issued PUNCHED in 900-Series Telecode and are suitable for assembly by 1-PASS SIR 24/3/71 or 2-PASS SIR 7/1/71 as described in Book 103, '903/920 SIR'. These assemblers both accept tapes punched in either 900-Series or 920 Telecode; thus there is no need to convert the above nine tapes to the same code as the user's program; which itself may be in either code, (or a mixture of both, on separate tapes).

Tape reader modes, Telecodes, and the internal code used by the CHIP and CHOP subroutines are described in Book 106, '903/905/920 USEFUL NOTES'.

Note that the following abbreviations have been used in certain sections of this book:-

(T) Horizontal Tab
(N) Newline, or C/R+L/F
(H) Halt or Stopcode
(S) Space

The following table summarises the store used by these tapes and gives the abbreviated name used for them in error indications:-

	Consecutive locations	Literals	
QS I/O	218	24	QS I/O
QSMATH	377	6	QSM
QDLA	837	33	QDLA
QDMATH	336	4	QDM
QF	672	27	QF
QF I/O	784	43	QF I/O
QFMATH	500	24	QFM
CH I/O + LEGTAPE	681	40	CH I/O
CH I/O	422	36	
OPTIMISED MATHEMATICAL	143	22	-
SHELLSORT	239	6	-

QS I/O 15/12/69 900-Series Telecode

NOTE.

The "INDIRECT" T5-CUM-QIN1 and T6-CUM-QOUT1 contained in this tape should not be confused with the "DIRECT" versions issued as 2 separate tapes in the Utility Software Packages.

The "DIRECT" versions input and output via the punch and reader directly, in a fixed Telecode. They are thus inflexible, but occupy little store. They are provided as a debugging aid for programs occupying most of the store; or for programs which are to be run a few times but not to be kept.

The "INDIRECT" versions input and output via character input and output subroutines. They should be used in any program requiring single-length number input and output which is to be kept. Whilst the indirect versions (plus a suitable character input and output subroutine) occupy more store in total than the direct versions, the resulting program can easily be modified at a later date to work in any Telecode via any peripheral.

1. INTRODUCTION.

1.1. FUNCTION.

QSI/O contains 2 subroutines :-

T5-CUM-QIN1
T6-CUM-QOUT1

for the input and output of single-length numbers. These routines are described individually in sections 2 & 3 below.

1.2. STORE USED.

218 consecutive locations
24 literals

1.3. FORM OF DISTRIBUTION.

QSI/O is a SIR tape in 900-Series Telecode, and should be assembled as a block of the user's SIR program.

CHIP, CHOP, and ERROR subroutines must also be assembled in the user's SIR program.

2. TS-CUM-QIN1.

2.1. FUNCTION.

TS-CUM-QIN1 is a SIR subroutine to read in one integer or fraction from a data tape via the character input subroutine 'CHIP'. It is suitable for use on any priority level.

2.2 ACCURACY

Integers are stored exactly.
Fractions are rounded towards zero with a maximum error of 2^{-17}.

2.3 METHOD OF USE

T5-CWI-QIN is entered by the following instructions:-

```
        11  QIN1
         8  QIN1+1
```

or :-

```
        11  T5
         8  T5+1
         :   :
```

or :-

```
        11  INPUTL
         8  INPUTE
```

(The first of these 3 forms is recommended, the others are only provided for compatibility with old programs)

2.4 DATA TAPE

Integers should be punched

+ or -
a sequence of 1 to 6 digits
any non-digit (e.g. (N), (T), or (S))

Note that -131072 cannot be input.

Fractions should be punched

+ or - followed by .
a sequence of 1 to 6 digits
any non-digit (e.g. (N), (T), or (S))

Blanks & erases will be ignored everywhere, as will the character preceding the + or -

2.5 EXAMPLE OF DATA TAPE

+3 SAMPLES, 12/10/63.

A	+678	B	+.05	C	−.999999
A	+713	B	+.40	C	−.995000
A	+82	B	+.10	C	+.000010

This example would be read by entering TS-CUM-QIN 10 times.

2.6 ERROR INDICATIONS

TS-CUM-QIN will punch the message "QS I/O ERROR 1" using the ERROR subroutine, if any of the following errors is found:

The first character after the + or − is not a digit or .

The first character after a . is not a digit

More than 6 digits

Integer outside ±131071.

3. T6-CUM-QOUT1.

3.1. FUNCTION.

T6-CUM-QOUT1 is a SIR subroutine to punch the contents of the accumulator as an integer, fraction, or octal word via the character output subroutine 'CHOP'. It is suitable for use on any priority level.

3.2. METHOD OF USE

The entry instructions are as follows:—
To print the contents of the accumulator as:—

An integer	A fraction	An Octal word
11 QOUT1 8 QOUT1+1	11 QOUT1 8 QOUT1+2	11 QOUT1 8 QOUT1+3
or	or	or
11 T6 8 T6+1	11 T6 8 T6+2	11 T6 8 T6+3
or	or	or
11 PRTL 8 PRTEI	11 PRTL 8 PRTEF	11 PRTL 8 PRTE□

(The upper of these 3 forms is recommended, the others are only provided for compatability with old programmes)

In all cases the 2 entry instructions must be followed by the parameter word (see below).

The subroutine exits to the location after the parameter word.

3.3. THE NUMBER -131072

The integer -131072, and FRACTION -1, will be punched in OCTAL, (irrespective of the entry instructions used) as &400000.

3.4. OUTPUT FORMAT & PARAMETER WORD

The parameter word is punched in the form of an absolute addressed instruction, e.g.

/0 0
or 2 3.

If the modifier bit '/' is present, the output will start with a (N). If this is not present, output will start with (S)(S).

Below, { F refers to the function digits of the parameter
{ N " " " address " " " "

3.4.1. OCTALS will be punched in the usual format:-

- (N) or (S)(S)
- &
- exactly 6 digits

E.G. (N)&123456.

F & N are ignored

3.4.2. FRACTIONS will effectively be MULTIPLIED by 10^N before printing, and the least significant F digits will be omitted, I.E. the following will be punched:-

- (N) or (S)(S)
- + or −
- N digits (with 'leading' zeros replaced by SPACES)
- Decimal point (unless followed by no digits)
- (6−N)−F digits

Thus for the usual fraction format, e.g. (N)+.123456, use F=N=0.

3.4.3. INTEGERS will effectively be DIVIDED by 10^N before printing, and the least significant F digits will be omitted, I.E. the following will be punched:-

- (N) or (S)(S)
- + or −
- (6−N) digits (with 'leading' zeros OMITTED)
- Decimal point (unless followed by no digits)
- N−F digits

Thus for the usual integer format, e.g. (N)+123456, use F=N=0.

3.5. EXAMPLE OF USE OF PARAMETER

Given that the accumulator contains an angle in degrees scaled by 180° (so that +·25 = +45°) to print the angle on a new line, in degrees, to one decimal place :-

```
12    +·18
11    QOUT
8     QOUT1+2
      /2 3
```

F=2
+·1800⊠⊠
N=3

or

```
12    +18000
11    QOUT
8     QOUT1+1
      /1 2
```

F=1
+1800⊠
N=2

It is recommended that the parameter word is punched in the "address" column with F & N separated by one (5); so that it is easily recognised as a parameter.

3.6. WARNING

Although this subroutine is particularly suited to PRINTING quantities scaled by powers of 10, it is usually more accurate to STORE and OPERATE upon quantities scaled by powers of 2 or scaled by their maximum values, (as in the above example).

3.7 ERROR INDICATIONS

T6-CUM-QOUT will punch the message "QS I/O ERROR 2" using the ERROR subroutine if entered at the integer or fraction entry points and a meaningless parameter is used, i.e. if

N > 6

N < F for integers

6-N < F for fractions

3.8. ACCURACY

Integers, printed with F=0, are printed exactly.
Fractions, " " F=0, " rounded towards zero & hence contain a maximum error of 10^{-6}.

When digits are omitted by setting F>0, the printed digits are not changed, i.e. rounding is towards zero. For example +49 would be printed as +4 if F=1.

4. EXAMPLE OF PROGRAM USING QS I/O.

The following program will read in a data tape of the form

POUNDS +1234 SHILLINGS +5 PENCE +6

and punch out the equivalent number of old pence in legible tape form, using the CHIP, CHOP, & ERROR subroutines contained in "CH I/O + LEGTAPE S/R, 31/3/70":-

```
[START  QIN1  QOUT1  CHIPF
 CHOPF  CHOPC CHOPL  CHOPE]

START   4     +0
        5     CHIPF       (Initialise input subroutine)
        5     CHOPF       (Initialise output subroutine)

        4     &400000
        5     CHOPC       (For output in legible tape)

        11    QIN1
        8     QIN1+1      (Read in pounds)
        12    +240
        14    17
        5     PENCE

        11    QIN1
        8     QIN1+1      (Read in shillings)
        12    +12
        14    17
        1     PENCE
        5     PENCE

        11    QIN1
        8     QIN1+1      (Read in pence)
        1     PENCE

        11    QOUT1
        8     QOUT1+1     (Output integer on new line)
              /0 0

        4     +20
        11    CHOPL
        8     CHOPE       (Punch (H) using CHOP subroutine)
        8     j+0

PENCE   >1
```

QSMATH 15/12/69 900-Series Telecode

1. INTRODUCTION.

1.1. FUNCTION.

QSMATH contains 5 subroutines for the calculation of mathematical functions of single-length numbers :-

QSIN	Sine & Cosine
QSQRT	Square Root
QATAN	Arctangent
QLN	Natural logarithm
QEXP	Exponential

These are described individually in sections 2 to 6 below.

1.2. STORE USED.

377 Consecutive locations
6 Literals

1.3. FORM OF DISTRIBUTION.

QSMATH is a SIR tape in 900-Series Telecode and should be assembled as a block of the user's SIR program.

An ERROR subroutine must also be assembled in the user's SIR program.

2. QSIN.

2.1. FUNCTION.

To calculate

$\frac{1}{2} \sin \pi x$

and $\frac{1}{2} \cos \pi x$

where x is the fraction in the accumulator.

It may be run at any program level and in any store module.

2.2. ACCURACY.

The maximum error is 2^{-15} ($\simeq .00003$)

2.3. METHOD OF USE & ENTRY INSTRUCTIONS.

The operand, x, and the result must be treated by the programmer as pure fractions.

To enable this to be done QSIN calculates

$\frac{1}{2} \sin \pi x$

and $\frac{1}{2} \cos \pi x$

Note, therefore that on entry, the accumulator holds the value of an angle as a fraction of π radians (180°).

Entry is made by

11 QSIN
8 QSIN+1

On exit

$\frac{1}{2} \sin \pi x$ is in the accumulator
and in QSIN + 67
$\frac{1}{2} \cos \pi x$ is in QSIN + 68

QSIN must be declared as a global identifier in all blocks of the users program which refer to it.

2.4. TIME TAKEN.

1.4 - 1.8 milliseconds, on 903 or 920B.

3. QSQRT.

3.1. FUNCTION

QSQRT(B6) is used to calculate the single-length square-root of a single-length or double-length fraction.

It may be run at any program-level and in any store-module.

Two entry points are provided for single-length and double-length working.

3.2. ACCURACY

The maximum error is $\pm 2^{-17}$.

3.3. METHOD OF USE & ENTRY INSTRUCTIONS.

The operand is denoted by a and, if the operand is double-length, the most significant half is denoted by a (m. s.), and the least significant half by a (l. s.).

Double-length Working.

Entry	Place a (l. s.)	in	QSQRT+3
	Place a (m. s.) in the accumulator		
	Place link	in	QSQRT
	Jump to		QSQRT+1
Exit	The result is held, single-length, in the accumulator		
	and also in		QSQRT+45
	a (m. s.) is in		QSQRT+4
	a (l. s.) is in		QSQRT+3

Single-length Working

Entry	Place a in the accumulator	
	Place link in	QSQRT
	Jump to	QSQRT+2
Exit	The result is held, single-length, in the accumulator	
	and also in	QSQRT+45
	a is in	QSQRT+4

In a SIR program, QSQRT must be declared as a global identifier in all blocks which refer to it.

3.4. ERROR INDICATION

If a<0 then QSQRT will punch the message "QSM ERROR 1" using the ERROR subroutine.

3.5. TIME TAKEN

(The time for the single-length entry is in brackets).
If the final approximation is X_n
then the time taken is 680(805)+375n microseconds
The maximum time is 5.3(5.5) milliseconds
If a=0 the time taken is 250(375) microseconds
If $a \geq 1-2^{-17}$ the time taken is 300(450) microseconds.

} on 903 or 920B

4. QATAN

4.1. FUNCTION

To calculate

$$t = (1/\pi) \tan^{-1}(x/y)$$

where $-1 \leq x < +1$

$-1 \leq y < +1$

It can be run at any program level and in any store module.

4.2. ACCURACY

The maximum error is 2^{-15} ($\simeq .00003$).

4.3. METHOD OF USE & ENTRY INSTRUCTIONS.

All numbers must be treated by the programmer as pure fractions. To enable this to be done QATAN calculates

$$(1/\pi) \tan^{-1}(x/y)$$

Note, therefore, that on exit the accumulator holds the value of an angle as a fraction of π radians (180°).

Entry: place x in QATAN+89
" y in QATAN+90
"link in QATAN
jump to QATAN+1

Exit: the result is in the accumulator
x and y are not affected.

QATAN must be declared as a global identifier in all blocks of a SIR program which refer to it.

4.4. ERROR INDICATIONS.

If x=y=0, then QATAN will punch the message "QSM ERROR 2" using the ERROR subroutine.

4.5. TIME TAKEN.

The time taken depends on the values of x/y and of y.

The maximum time is about 3.2 milliseconds, on 903 or 920B.

5. QLN.

5.1. FUNCTION.

To calculate

$$1/16 \log_e x$$

where x is the fraction in the accumulator. It may be run at any program level and in any store module.

5.2. ACCURACY.

The maximum error is 2^{-16}. ($\simeq .000015$)

5.3. ENTRY INSTRUCTIONS & METHOD OF USE.

The 900 series use fractional machines and all numbers in the accumulator, on entry and exit, must be treated as pure fractions by the programmer.

On entry the accumulator contains the number whose logarithm is to be calculated. Entry is made by

```
11  QLN
 8  QLN + 1
```

On exit

$\frac{1}{16} \log_e x$ is held in QLN + 52
and in the accumulator.

QLN must be declared as a global identifier in all blocks of the user's program which refer to it.

5.4. ERROR INDICATION

If the accumulator's contents on entry are not positive then QLN will punch the message "QSM ERROR 3" using the ERROR subroutine.

5.5. TIME TAKEN

Between 1.3 and 2.8 milliseconds (dependent on the number of shifts required to scale x), on 903 or 9206.

6. QEXP.

6.1. FUNCTION

To calculate $\exp(2^p x)$

where

$$-1 \leq x < 0$$

$p \geq 0$, and p is integral.

It may be run at any program level and in any storage module.

6.2. ACCURACY

The maximum error is 2^{-16} ($\triangleq .000015$).

6.3. METHOD OF USE & ENTRY INSTRUCTIONS.

x is treated as a pure fraction;
p is treated as an integer.

The result is a pure fraction.

On entry

x must be placed in the accumulator

and p must be placed in QEXP + 53
p is not preserved by QEXP

Entry is made by

```
11 QEXP
 8 QEXP + 1
```

On exit

the result is in the accumulator

and in QEXP + 54
x is in QEXP + 52

QEXP must be declared as a global identifier in all blocks of the user's program which refer to it.

6.4. ERROR INDICATIONS.

If $x \geq 0$ or $p < 0$ QEXP will punch the message "QSM ERROR 4" using the ERROR subroutine

6.5. TIME TAKEN.

$(3.7 + 0.26\ p)$ milliseconds approx, on 903 or 920B.

QDLA 15/12/69 900-Series Takeoffs

Chapter 1: INTRODUCTION

1.1 Purpose

To perform arithmetic operations upon double-length fixed-point numbers (x) in the range

$$-1 \leq x \leq 1 - 2^{-34}$$

1.2 Summary

The double-length number routines are interpretive, providing equivalents of the machine code instructions and allowing input and output of numbers in fraction and integer formats.

When entered, QDLA proceeds to interpret the instructions in the store locations immediately following the entry in the users area. Thus operations can be performed on double-length numbers by placing in store the corresponding single-length instruction.

1.3 Form of Distribution

QDLA is a SIR tape in 900-Series Telecode, and should be assembled as a block of the user's SIR program.

CHIP, CHOP, and ERROR subroutines must also be assembled in the user's SIR program.

1.5 Restrictions

Some instructions cannot be interpreted. See notes in 2.4.

1.6 Accuracy

With the following exceptions no error is introduced by the routines:

(a) Multiply: maximum error is $+2^{-34}$ (0.6×10^{-10})

(b) Divide: maximum error is $+2^{-32}$ (0.2×10^{-9})

(c) Input and output: maximum error is $\pm 2^{-34}$ (0.6×10^{-10})

Chapter 2: FUNCTIONS

2.1 Notation

x(m.s.) = most significant half of x

x(l.s.) = least significant half of x

x = double-length number held in locations X and X+1

p = double-length number held in the pseudo-accumulator, p

C(X) = single-length number held in location X

C(B) = single-length number held in pseudo B-register

C(S) = single-length number held in pseudo S.C.R.

:= means "becomes equal to".

2.2 Format

A double-length number, x, is held in two consecutive store locations, X and X+1.

Location	Bit 18	Bits 17-1
X	sign	most significant bits of x
X + 1	0	least significant bits of x

N. B. Workspaces must be declared as '>2' NOT '>1'

Negative number representation is by the usual 2's complement notation. (N. B. bit 18 of X+1 is always zero).

2.3 Entry and Exit

Entry is made by

```
11 QDLA+20
 8 QDLA+21
```

The routine proceeds to interpret the double-length instructions that follow the entry, using pseudo registers which are analogous to the registers in the computer hardware. These pseudo registers are represented in locations within QDLA, as follows:

a pseudo-accumulator(double-length) located in (QDLA+16 (QDLA+17

a pseudo B-register (single-length) located in QDLA+18

a pseudo S-register (single-length) located in QDLA+20

Exit is made by placing zero (+0) in the location after the last instruction to be interpreted. Control is transferred to the location following the zero location. The pseudo-accumulator and pseudo B-register are not affected by entry and exit.

Notes

1. 0 instruction

The instruction 0 0 is interpreted as a terminator for double-length working (see 2. 3). It does not affect the pseudo B-register. The hardware A-register is set equal to p(m. s.).

2. 15 6144 instruction (and 15 6148) .

This must follow by a parameter word to specify the format used. The next instruction interpreted is that following the parameter word. See 2. 5. 2 for format of parameter word.

3. Modified instructions may be used: the contents of the pseudo B-register are added to the address digits to find the address of the operand.

4. Literal instructions may not be used.

2. 5 Input and Output

The 15 instructions corresponding to input and output of paper tape and teleprinter characters in machine code are interpreted as input and output of complete numbers : via the character input and output subroutines CHIP and CHOP.

The double length arithmetic functions effectively operate on fractional numbers in the range

$$-1.0 \leqslant x < 1.0$$

However, the programmer may wish to operate on numbers in other ranges. As with single length working, it is frequently convenient to regard a double length value as an integer (in the range $-2^{35} \leqslant x < +2^{35}$).

Input and output of numbers in this range is allowed, also input and output of mixed numbers which are scaled so that their internal representation is in the correct range. The programmer specifies the scaling constant to be used.

When using input and output of numbers in the additional ranges, the programmer must always remember that the internal representation is in fraction form, particularly when multiplying and dividing.

2.4 Operations Available

See 2.1 for notation used.

Pseudo Instruction	Name	New contents of P	New contents of X	Remarks
0 X	Load pseudo B-register	p	x	C(B):=C(X) see notes 1 and 3
1 X	Add	p+x	x	
2 X	Negate and add	x-p	x	
3 X	Store l. s. only	p	C(X):= the 17 least significant bits of p	C(X+1) unchanged
4 X	Load P	x	x	
5 X	Store P	p	p	
6 X	Store scaling factor	p	x	see 2.5.3.
7 X	jump if zero	p	x	if p=0 then C(S):=X
8 X	jump	p	x	C(S):=X
9 X	jump if negative	p	x	if p<0 then C(S):=X
10 X	Count in store	p	$C(X):=C(X)+2^{-17}$	C(X+1) unchanged
11 X	store pseudo SCR	p	C(S)	
12 X	multiply	p*x	x	
13 X	divide	p/x	x	see 3.2
14 X	$(*2^N$	$p*2^N$		n<47
14 X	$(*2^{N-8192}$	$p*2^{N-8192}$	x	n>8158
15 2048) 15 2052)	input number	<input>	-	see 2.5
15 6144) 15 6148)	output number	unchanged	-	see note 2 and 2.5

2.5.1 Character Set for Input

All characters which have a representation in internal code are acceptable. Blank, erase and carriage return are ignored wherever they occur.

The following characters are significant in the formation of numbers:

digits 0 to 9

decimal point

+ and -

All other characters are treated as separators.

On commencing input, separators are ignored until one of the significant characters is read. Once a digit has been read the occurrence of a separator terminates the number. Thus separators may be used freely between numbers, including letters used for descriptive text.

2.5.2 Input

The format of an input number determines the way it is processed. A number not containing a decimal point is treated as an integer, N, and stored as the fraction $N \times 2^{-34}$

The range of N is:

$$-17179869184 < N < 17179869184$$

A number containing a decimal point is treated as a scaled fraction, F. If M is the current value of the scaling factor (see 2.5.3.) the number will be stored as the fraction:

$$F \times 10^{-M}$$

The range of F is:

$$-10^{M} \leqslant F < +10^{M}$$

F may not have more than 10 digits in all.

2.5.3 Scaling Factor

The scaling factor, M, used for input and output is set by an interpreted 6 instruction.

A positive scaling factor is set by instruction:

6 M

A negative scaling factor is set by instruction:

6 (8129-M)

Examples:

6 Instruction	Scaling Factor	Number Input	New Content of P
6 3	+3	12. -123.46 1.5	0.012 -0.012346 0.0015
6 8190	-2	-.001234 .0001	-0.1234 0.01
6 0	+0	0.999 -.05	0.999 -0.05

The scaling factor set on the tape of QDLA is +0. However, it is advisable to set the scaling factor explicitly in all programs, even if factor zero is to be used. The scaling factor is NOT affected by entry to & exit from QDLA.

2.5.4 Output Format

Numbers are output right justified, with non-significant zeros suppressed and the sign floated (i.e. immediately preceding the first significant character).

Output format is controlled by the parameter word which follows the 15 6144 and 15 6148. This is written as a pseudo instruction: /f k. If / is used (i.e. the parameter word is negative) the number output is preceded by newline. The number f controls the output format as described below, and the address part k denotes the type of number to be output.

If k = zero the number p is output as an integer:

$$p \times 2^{34}$$

This output is independent of the scaling factor. The number output occupies 12-f printing positions.

If k = 4096 a fraction will be printed:

$$p \times 10^{M}$$

f digits are printed after the decimal point, and the total number of printing positions occupied is f + M + 3.

Examples:

Parameter	p	Output
/0 0	$-17179869183 \times 2^{-34}$	-17179869183
/0 0	2^{-17}	131072
/0 0	-0.03125 (i.e. -2^{-5})	-536870912
/5 0	123456×2^{-34}	123456
/4 4096	0.12345678 (and scaling factor + 3)	123.4567
/0 4096	0.99999 (and scaling factor + 4)	10000.
/8 4096	0.03125 (and scaling factor + 0)	.03125000

CAUTION.

f and M must be such that no more than 10 digits are printed, e.g.

Parameter /8 4096
Scale factor +3

will cause 11 digits to be printed.
The digits printed will be RUBBISH but
NO error message will be given !

Chapter 3: ERROR INDICATIONS

If an error occurs a message is given by QDLA using the ERROR subroutine.

Message	Meaning
QDLA ERROR 0	Attempt to divide by zero
QDLA ERROR 1	Input Format Error e. g. two decimal points in number
QDLA ERROR 2	Input overflow i. e. number too large for input with the current scaling factor
QDLA ERROR 4	More than 10 digits in input fraction
QDLA ERROR 5	J5 instruction with illegal address
QDLA ERROR 6	Overflow during division

Chapter 4: METHOD USED

4.1 The following steps are carried out for each pseudo-instruction interpreted:-

1. The address (modified if required) is placed in 19; of QDLA.

2. The pseudo S. C. R. is incremented.

3. If the function is f control is transferred to location f; of QDLA.

This location contains a jump to the appropriate routine for the operation. The operation is carried out and then the next pseuod-instruction is interpreted.

4.2 QDLA uses the following locations for the purposes indicated:

16; and 17;	pseudo-accumulator
18;	pseudo B-register
19;	address (modified if necessary) of pseudo-instruction
20;	pseudo S. C. R. and link for exit

4.3 Conditional Jumps

7 examines both locations.
9 examines first location only.

Chapter 5: TIME TAKEN

The following times are approximate; and are for a 903 or 920B:

Function Number	Operation		Time in μs
0	Set pseudo B-register		492
1	Add		658
2	Negate and add		688
3	Store least significant half		492
4	Load		569
5	Store		525
6	Store scaling factor		555
7	Jump if zero	(p<0	440
		(p=0	581
		(p>0	482
8	Jump		509
9	Jump if negative	(p<0	535.5
		(p>0	457
10	Count in store		467
11	Store pseudo SCR		539
12	Multiply		919-1208
13	Divide		2788-5481
14	Shift		695+3N
15	Input or output		Acts at the speed of the appropriate peripheral

For B modification add 161 μs to the above times.

Chapter 6: STORE USED

837 Consecutive locations
33 Literals

Chapter 7: EXAMPLE OF PROGRAM USING QDLA.

The following will read in a data tape of positive fractions and punch their cubes. Any negative number may be presented to stop the program. The CHIP, CHOP, & ERROR subroutines contained in "CH I/O S/R, 31/3/70" are used :-

```
[START  QDLA  CHIPF  CHOPF  CHOPC  CHOPL  CHOPB]

START   4     +0
        5     CHIPF     (Initialise input subroutine)
        5     CHOPF     (Initialise output subroutine)
        5     CHOPC     (Select output in 920 Telecode)

        11    QDLA+20
        8     QDLA+21

        6     0
LOOP    15    2048
        9     STOP
        5     W
        12    W
        12    W
        15    6144
        /8    4096      (Parameter)
        8     LOOP

STOP    +0                (Exit from QDLA)
        4     +20
        11    CHOPL
        8     CHOPC
        8     J+0       (Punch (H)).

W       >2                (Double-length workspace)
```

QDMATH 15/12/69 900-Senior Telecode

1. INTRODUCTION.

1.1. FUNCTION.

QDMATH contains 3 subroutines for the calculation of mathematical functions of double-length numbers :-

QDASIN	Sine & Cosine
QDASQRT	Square Root
QDAATAN	Arctangent

These are described individually in sections 2, 3, & 4 below.

1.2. STORE USED.

336 Consecutive locations.
4 Literals.

1.3. FORM OF DISTRIBUTION.

QDMATH is a SIR tape in 900-Series Telecode and should be assembled as a block of the user's SIR program.

QDLA and an ERROR subroutine must also be assembled in the user's SIR program.

2. QDASIN (B.104A)

2.1. FUNCTION

To calculate, as double-length fractions,

$$\tfrac{1}{2} \sin \pi x$$

and

$$\tfrac{1}{2} \cos \pi x$$

where x is a double-length fraction.

It can be run at any program level and in any store-module.

2.2. ACCURACY

The maximum error is $2^{-31}(0.5 \times 10^{-9})$.

2.3. METHOD OF USE & ENTRY INSTRUCTIONS.

The operand, x, and the result must be treated by the programmer as pure fractions.

To enable this to be done, QDASIN calculates

$$\tfrac{1}{2} \sin \pi x \quad \text{and} \quad \tfrac{1}{2} \cos \pi x$$

Note: therefore, that x is the value of an angle as a fraction of π radians (180°).

A double-length number is held in two consecutive store-locations, the description below gives only the first of the two.

Entry (for assembly by SIR)

place x(l. s.) in QDASIN+99
and x(m. s.) in the accumulator & optionally in QDASIN+98
and enter 11 QDASIN
8 QDASIN+1

Exit $\tfrac{1}{2} \sin \pi x$ in QDASIN+102
and in QDLA+16
$\tfrac{1}{2} \cos \pi x$ in QDASIN+104

N.B. The instruction pair 11 QDASIN
8 QDASIN+1

must not be part of a pseudo-program interpreted by QDLA.

QDASIN must be declared as a global identifier in all blocks of a SIR program which refer to it.

2.4 TIME TAKEN

The time taken is approximately 50 milliseconds, on 903 or 9208.

3. QDASQRT (B. 106A)

3.1. FUNCTION.

To calculate, as a double-length fraction, the square root of a double-length fraction, a.

It can be run at any program level and in any store module.

3.2. ACCURACY

The maximum error is 3×2^{-34}. (0.2×10^{-9})

3.3. METHOD OF USE & ENTRY INSTRUCTIONS.

A double-length number is held in two consecutive locations: only the first location is given below.

Entry

place a in QDASQRT+44
and enter 11QDASQRT
8QDASQRT+1

Exit $\sqrt{a}$ in QDASQRT+46

N. B. The instruction pair

11 QDASQRT
8 QDASQRT+1

must not be part of a pseudo-program interpreted by QDLA.

QDASQRT must be declared as a global identifier in all blocks of a SIR program which refer to it.

3.4. ERROR INDICATION

If a<0 then QDASQRT will punch the message "QDM ERROR 1" using the ERROR subroutine.

3.5. TIME TAKEN

Special Cases

a = 0 570 microseconds.

a = $1-2^{-34}$ 1053 microseconds.

General Cases

Approximate time taken is

3.0 + 12.5 n milliseconds

where n is the number of iterations necessary.

These times are for a 903 or 9208.

4. QDAATAN (B. 105A)

4.1. FUNCTION

To calculate, as a double-length fraction

$$t = (1/\pi) \tan^{-1} (x/y)$$
and $$b = (1/2\pi) \text{ true bearing,}$$

where x, y are double-length fractions.

It can be run at any program level and in any store-module.

4.2. ACCURACY

The maximum error is 2^{-34} (0.6×10^{-10})

4.3. METHOD OF USE & ENTRY INSTRUCTIONS.

All numbers must be treated by the programmer as pure fractions.

To enable this to be done QDAATAN calculates

$t = (1/\pi) \tan^{-1} (x/y)$

Note, therefore, that t is the value of an angle as a fraction of π radians (180°).

A double-length number occupies two consecutive locations; only the first is given below.

Entry (for assembly by SIR)

Place x in QDAATAN+136
y in QDAATAN+138
and enter 11QDAATAN
8QDAATAN+1

Exit

t in QDAATAN+142
b in QDAATAN+146
b(m. s.) in the accumulator

Note. The true bearing is found by taking
x along the easterly axis
y along the northerly axis
and measuring the angle in a clockwise direction.

N. B. The instruction pair must not form part of a pseudo-program interpreted by QDLA.

QDAATAN must be declared as a global identifier in all blocks of a SIR program which refer to it.

4. 4. ERROR INDICATION

If x=y=0, then QDAATAN will punch the message "QDM ERROR 2" using the ERROR subroutine.

4. 5. TIME TAKEN

Approximately 42. 4 milliseconds,
on 903 or 920B.

QF 17/6/71 900-Series Telecode

QF (FLOATING POINT SUBROUTINES)

1 INTRODUCTION

1.1 Purpose.

QF is used to perform operations on floating-point numbers.

1.2 Summary.

QF contains routines for operations corresponding to all the fixed-point operations except the function 15 (Functions 3 and 6 have special meanings in QF).

When entered, QF proceeds to interpret the instructions in the locations immediately following the entry-point in the user's program. Thus, operations are performed on floating-point numbers by placing in store the corresponding fixed-point instructions.

Two formats are available for floating-point numbers (see Paragraph 2.2). QF may be run in any program level.

1.3 Form of Distribution.

QF is a SIR tape in 900-Series Telecode, and should be assembled as a block of the user's SIR program.

An ERROR subroutine must also be assembled in the user's SIR program.

If QF is used WITHOUT QFMATH then the following short tape should be loaded AFTER QF, but BEFORE any other tape (other than QF I/O or the short tape loaded instead of it described below) :-

```
((DUMMY  QF  MATH)

SQRT  LN  EXP  SIN  COS  ARCTAN
          +0
          8  ERRF

(H)
```

Similarly if QF is used WITHOUT QFI/O, then the

following short tape should be loaded AFTER QF, but BEFORE any other tape (other than QFMATH or the short tape loaded instead of it described above) :-

```
((DUMMY QF I/O)

QFID PSTAND IOINT IORL PSET
     8 ERRF

(H)
```

1.5 Restrictions.

See Paragraph 2.4.

1.6 Accuracy.

If the result of the operation is y, multiplication gives a maximum error of $2^{-34}y$, division gives a maximum error of $2^{-32}y$. All other operations give a maximum error of $2^{-35}y$.

2 FUNCTIONS

2.1 Notation.

x(man) = mantissa of floating-point number, x.

x(exp) = exponent of floating-point number, x.

x = a floating point number held in 2 or 3 words from location X.

f = the floating point number held in the floating point accumulator (FPA).

b = the contents of the pseudo B-register (FBREG)

s = the contents of the pseudo S-register.

C(x) = the contents of location X

:= means "becomes equal to"

2.2 Format.

One of two formats may be used to hold a floating-point number in store. Normally the packed format is used, but the unpacked format allows a wider range of numbers and slightly greater accuracy. The two formats are summarised in the table below. In the unpacked format, the mantissa is a double-length fraction held in two consecutive locations and the exponent is a single-length integer held in the next location. For the packed format, the mantissa is truncated and the exponent held in the seven least significant bits of the second store location. In this case the exponent must be in the range -64 to +63.

Format	Location	Bit 18	Bits 17-8	Bits 7 - 1
Packed	X	sign	most significant bits of mantissa	
	X+1	0	least sig. bits of mantissa	exponent
Unpacked	X	sign	most significant bits of mantissa	
	X+1	0	least significant bits of mantissa	
	X+2	← exponent →		

N. B. Workspaces must be declared as '>2' or '>3' NOT '>1'.

Negative number representation for exponent and mantissa is by the usual 2's complement notation.

All internal working of QF uses the unpacked format.

Examples of floating point numbers in the two formats:

Number	Locn	Packed	Unpacked
0.25=	X	010 000 000 000 000 000	010 000 000 000 000 000
0.5×2^{-1}	X+1	000 000 000 001 111 111	000 000 000 000 000 000
	X+2	Not Used	111 111 111 111 111 111
$1-2^{-27} \times 2^{63}$	X	011 111 111 111 111 111	011 111 111 111 111 111
$\doteq 9.2 \times 10^{18}$	X+1	011 111 111 110 111 111	011 111 111 110 000 000
	X+2	Not Used	000 000 000 000 111 111
-1.0×2^{-64}	X	100 000 000 000 000 000	100 000 000 000 000 000
$\doteq -0.5 \times 10^{-21}$	X+1	000 000 000 001 000 000	000 000 000 000 000 000
	X+2	Not Used	111 111 111 111 000 000

2.3 Entry and Exit.

Entry is made by

11 QF
8 QF + 1 } to use packed format

or

11 QF
8 QF + 2 } to use unpacked format.

QF proceeds to interpret and execute the pseudo-program using

a pseudo-accumulator (FPA)

a pseudo-B-register (FBREG)

and a pseudo-S-register (QF)

See Paragraph 2.4 for the effects of each function.

Exit is made by placing +0 in the location after the last instruction to be interpreted. Control is then transferred to the location following the zero location.

The Machine Accumulator and B register are not preserved. The pseudo-accumulator and B registers are not affected by entry and exit.

2.4 Available operations.

See paragraph 2.1. for notation used.

Table 1

Pseudo Instruction.	Name	New Contents of FPA	New Contents of X	Remarks
0 X	Load pseudo B-register	f	x	b:=C(X) See Notes 1 & 3
1 X	Add	f+x	x	
2 X	Negate & Add	x-f	x	
3 X	Exchange	x	f	Not a basic 903 operation
4 X	Load FPA	x	x	
5 X	Store FPA	f	f	See Note 2
6 N	Conversion Routines	f	x	See Table 2
7 X	Jump if f= zero	f	x	See Note 3
8 X	Jump	f	x	See Note 3
9 X	Jump if f<0	f	x	See Note 3
10 X	Count in store	f	"x+1"	See Note 3
11 X	Store pseudo SCR	f	s	See Note 3
12 X	Multiply	f*x	x	
13 X	Divide	f/x	x	See Note 4
14 N	$*2^N$	$f*2^N$	x	N<4096 See Note 5
14 N	$*2^{N-8192}$	$f*2^{N-8192}$	x	N≥4096
15 N	error	f	x	See Paragraph 3

Notes

(1) The instruction 0 0 is interpreted as a terminator for floating-point working (See Paragraph 2.3).

(2) If packed format is in force during interpretation of a 3 or 5 instruction, then a test is made whether

$$-64 \leq f(\text{exp}) < +64$$

If $f(\text{exp}) < -64$ then x:=0 and the next instruction is interpreted.

If $f(\text{exp}) \geq +64$ then an error indication is output (See Paragraph 3).

(3) These instructions operate on single word items. The instructions 7, 8 & 9 may jump to another interpreted instruction: they must not jump out of the interpreted program except via a terminator.

(4) If an attempt is made to divide by zero an error indication is output (See Paragraph 3).

(5) The results of the following instructions are always standardised:

1, 2, 12, 13, 14.

The instruction 14 0 may be used to standardise the contents of the FPA.

(6) Modified instructions may be used and, if they are, the contents of the pseudo B-register are added to the address digits before obeying an instruction.

(7) Literal addresses may not be used i. e. constants must be stored in the correct format by the user.

Table 2

The address of a 6 instruction determines its meaning.

Function	Effect
6 1	Instructions interpreted after this assume packed format.
6 2	" " " " " unpacked "
6 3	Convert a single-length integer to a floating-point number and place the result in the FPA. Location QF+3 contains the address where the integer is held.
6 4	Form the integral part of the number in the FPA. Location QF+4 contains the absolute address where the (single-length) integer is to be placed. This routine always rounds down.
6 5	Convert a fixed-point fraction to a floating-point number and place the result in the FPA. Location QF+5 contains the absolute address of the (single-length) fraction.
6 6	Convert the number in the FPA to a fixed-point fraction. Location QF+6 contains the absolute address where the (single-length) fraction is to be placed. This routine always rounds down.

The instructions 6 1 and 6 2 do not convert numbers; they define the action of following instructions:-

In 6 3 to 6 6 the arguments and the addresses in QF+3 to QF+6 are unaffected. Error indications are output if an impermissible address is used or if overflow occurs. (See Paragraph 3).

Example

An integer is held in INT1 and a real number in RL2

The following section of program places the floating-point form of the first in RL1 and the entier of the second in INT2.

```
(SET ADDRESSES IN QF WORKSPACE)
        4     PSI1          (PSI1 holds the address of INT1)
        5     QF+3
        4     PSI2          (PSI2 holds the address of INT2)
        5     QF+4

(NOW PERFORM CONVERSIONS)
        11    QF            (ENTER QF)
        8     QF+1
        6     3             (INT1 in the FPA)
        5     RL1
        4     RL2
        6     4             (RL2 stored as integer)
        +0                  (RETURN to normal working)
        8     ;+0

PSI1    0     INT1
PSI2    0     INT2
```

3 ERROR INDICATIONS

If an error occurs, a message is given by QF using the ERROR subroutine.

Message	Significance
QF ERROR 1	Impermissible instruction
QF ERROR 2	Floating-point over-flow (5 or 13 instruction)
QF ERROR 3	Integer overflow
QF ERROR 4	Fraction overflow (6 6 instruction)

4 METHOD USED

The following steps are carried out for each pseudo-instruction interpreted.

(1) The pseudo S-register is incremented.

(2) The function and the address (modified if required) bits of the interpreted instruction are stored.

(3) Control is transferred to the appropriate routine to execute the instruction.

(4) Control is returned to the interpreter (via a standardising routine for instructions 1, 2, 12, 13, 14).

Subroutines from 903/Algol have been used for all arithmetic operations.

5 STORE USED

The floating-point package occupies
672 consecutive locations & 27 literals.

6 TIME TAKEN

The following times are approximate:-

Function Number	Operation	Times in Microseconds
0	Set pseudo B-register	440
1	Add	2150 to 3850 (average 2500 - see Note 1)
2	Negate and Add	2480 to 4180 (average 2800 - see Note 1)
3	Exchange	1980 packed or 1810 unpacked
4	Load FPA	770
5	Store FPA	770
6	Specifies format for the following instructions	440
7	Jump if zero	440
8	Jump	440
9	Jump if negative	440
10	Count in store	400
11	Store pseudo SCR	440
12	Multiply	2140
13	Divide	4300 to 6000 (average 4700 - see Note 1)
14	Shift	810 to 2510 (average 1210 - see Note 1)
	ENTRY	50
0 0	EXIT	150

Notes

(1) The time depends on the number of places the mantissa is shifted to standardise the result of the operation. The average given assumes a shift of 4 places.

(2) For modified instructions add 125 μs to the time taken.

(3) The times quoted above are for a 903 or 920B.

QF I/O 17/6/71 900-Series Telecode

Chapter 1: INTRODUCTION

1.1 Purpose.

These programs provide routines for general number input and output. The format conventions are those of Elliott Algol.

1.2 Method of Use.

The routines are entered via QF.

1.3 Summary.

The number input or output may be stored as a floating-point number or as an integer, input and output being performed via character subroutines CHIP and CHOP.

1.4 Accuracy and Range.

The maximum error is of the order of 10^{-8}.

The range of a floating-point number, x, is given approximately by

$$-9.2 \times 10^{18} < x < 9.2 \times 10^{18}$$

The range of an integer, n, is given by

$$-131072 \leq n \leq +131071$$

1.5 Form of Distribution.

The programs are distributed as a SIR mnemonic tape. The tape contains both input and output routines: it must be assembled after QF but may be preceded by floating point mathematical routines: no other programs may be assembled between QF and the input/output routines.

CHIP, CHOP, and ERROR subroutines must also be assembled in the user's SIR program.

Chapter 2: FUNCTIONS

2.1 Entry and Exit.

The routines are entered by interpretation by QF of a pseudo-instruction. If the instruction is 6 8191 QF interprets the next but one instruction after execution; otherwise return is made to the next pseudo-instruction.

The functions available and the corresponding call are listed below. Input/output instructions are assumed to refer to real numbers until a 6 7 instruction is interpreted.

Function	Pseudo-Instruction	Remarks
Input a number	15 2048	input via the character input subroutine CHIP.
	15 2052	format as described in 2.2.
Output a number	15 6144	output via the character output subroutine CHOP.
	15 6148	format as currently set
Following input/output instructions refer to integers	6 7	see note 1; the effect of this instruction is not destroyed by exit from QF.
Following input/output instructions refer to real numbers	6 8	
Reset presumed settings	6 0	See 2.3 for description of presumed settings
Set new parameters	6 8191	This instruction must be followed by a parameter word; See 2.3.

Notes

1. An integer is input to the location whose address is in QF+4
   An integer is output from the location whose address is in QF+3.
   Real numbers are input to or output from the Floating Point Accumulator.
2. An impermissible address gives an error indication (See para. 3)
3. A number is treated as real or integer depending on the entry used, not on the format.

Example

The following section of program inputs an integer, -n, followed by n real numbers. Their sum is output in freepoint (8) format; the CHIP, CHOP & ERROR routines contained in "CH I/O S/R 31/3/70" have been used :-

```
[ENTRY  QF  CHIPF
 CHOPF  CHOPC  CHOPL  CHOPE]


SUM     >2                      (Packed workspace)

COUNT   >1
PSI1    0   COUNT

ENTRY   4   PSI1                (HOUSEKEEPING)
        5   QF+4
        4   +0
        5   SUM
        5   SUM+1
        5   CHIPF               (Initialise input subroutine)
        5   CHOPF               (Initialise output subroutine)
        5   CHOPC               (For output in 920 Telecode)
        11  QF
        8   QF+1
        6   0
        6   7
        15  2048                (COUNT:= -n)
        6   8

LOOP    15  2048
        1   SUM
        5   SUM
        10  COUNT               +0      (Exit from QF)
        4.  COUNT
        9   LOOP                4 +20
        4   SUM                 11 CHOPL
        15  6148                8 CHOPE (Punch (H))
                                8 ;+0
```

2.2 Input Formats.

The character set accepted is as follows

0 1 2 3 4 5 6 7 8 9 + - . $_{10}$

<null> <delete> <carriage return>
<halt>
<space> <newline> all other internal code characters.

Characters on the first line may appear in a number.

Characters on the second line are always ignored.

If <halt> is read during execution of an input instruction the program waits.

Characters on the fourth line are treated as separators between numbers and are otherwise ignored.

Numbers may be signed or unsigned and may be punched in any of the conventions of 903 Algol. Note that + or - cannot terminate a number.

The number input is treated as integer or floating-point according to the entry-point used and not according to the format input. An integer must not exceed 131071 in magnitude (fractional parts are rounded to the nearest integer).

Example of numbers that may be input by this program as real numbers

104	+500	-2	500000
1002	10234.56		200.0 .
$4_{10}1$	$5_{10}-5$	$+2.75_{10}+10$	
$_{10}2$	$-_{10}+05$		

2.3 Output Formats.

In the following description the presumed settings after initial assembly are given in square-brackets. These settings apply to all output unless changed by a 6 8191 instruction and are reset by a 6 0 instruction. The formats satisfy the conventions of Elliott Algol.

The parameter word after a 6 8191 instruction is a pseudo-instruction

B F N

2.3.1 Lay-Out. [newline]

This affects both real and integer format.

For newline B=1
For sameline B=0

2.3.2 Real Format [freepoint(8)]

For freepoint(n) format F=0 N=n
" aligned(m, n) " F=1 N=16m+n
" scaled(n) " F=2 N=n

For aligned format $m+n \leq 15$
For freepoint and scaled format $n \leq 8$

The integer format is not changed by change of the real format.

2.3.3 Integer Format [digits(6)]

For digits(n) format F=4 N=n ;$n \leq 6$

The real format is not changed by change of the integer format.

An impermissible parameter word causes an error indication to be output. (See para 3).

2.4 Accumulators for Input and Output.

Real numbers are input to and output from the Floating-Point Accumulator of QF.

Integers are input to and output from store locations: these operations are related to the floating-point operations

6 3 and 6 4.

Chapter 3: ERROR INDICATIONS

3.1 Standard Indication.

If an error occurs a message is given by QF I/O using the ERROR subroutine.

3.2 Errors Detected.

Message	Significance
QF ERROR 1	Impermissible instruction or parameter
QF I/O ERROR 2	FPA not standardised on output
QF I/O ERROR 3	Integer overflow on input
QF I/O ERROR 4	Contextual error on input
QF I/O ERROR 7	Floating-point over-flow on input (see Note 1)

Note 1. If the number being input has a value greater than $_{10}+100$ approximately, floating point overflow occurs. This error will most probably be caused by a wrongly punched data tape.

3.3 Alarm Printing.

If the number to be output is too large for the format specified, alarm printing occurs. This is in scaled format and uses the same number of characters as the format demanded. If this is impossible H is output.

e.g. 1. format demanded: aligned (4, 3)
numbers to be output: -35286.741
output obtained: $-3.53_{10}+04$

e.g. 2. format demanded: aligned (2, 1)
number to be output: 123.45
output obtained: H

Chapter 4. STORE USED.

QF I/O occupies 784 consecutive locations and uses 43 literals.

QFMATH 20/12/69 900-Series Telecode

1 INTRODUCTION

1.1 Purpose.

To compute certain mathematical functions of floating-point numbers held in the floating-point accumulator (FPA). The functions are performed by sub-routines entered via the floating-point package (QF).

1.2 Summary.

The functions provided are square-root, sine, cosine, arctangent, natural logarithm, exponential.

1.3 Accuracy.

The maximum error is 8×10^{-8}.

1.4 Form of Distribution.

The routines are distributed as a single SIR mnemonic tape. This must be assembled immediately after the floating-point package (QF), but may be preceded by QF 1/0.

The routines are assembled as part of the block QF and are entered as floating-point subroutines. They may be run at any program-level and in any store-module.

2 FUNCTIONS

2.1 Entry and Exit.

Entry to all functions is made by a standard sub-routine entry which is interpreted by QF. On entry the argument is in the FPA which also contains the result on exit. (This result may be the effect of recovery after an error).

The available functions and their entry-points are listed below:

Function	Entry	Comments
square-root	11 SQRT 8 SQRT+1	entry with negative argument is an error
sine	11 SIN 8 SIN+1	argument is in radians
cosine	11 CØS 8 CØS+1	argument is in radians
arctangent	11 ARCTAN 8 ARCTAN+1	result is in radians and lies in the range $-\frac{\Pi}{2}$ to $+\frac{\Pi}{2}$
natural logarithm	11 LN 8 LN+1	entry with zero or negative argument is an error
exponential	11 EXP 8 EXP+1	the user should note that this operation may give an answer which cannot be held in packed format. This will be detected by QF

2.2 Example.

To calculate the function

$$y = \exp(x^2)^{\frac{1}{2}}$$

using packed format for the users workspace.

```
.
.
.
11    QF       (ENTER QF IF NECESSARY)
8     QF+1     (ASSUME PACKED FORMAT ON ENTRY)
4     X
12    X        (FORM X↑2)
11    EXP
8     EXP+1    (FORM EXP [X↑2])
11    SQRT
8     SQRT+1   (FORM {EXP [X↑2]}↑½)
5     Y
.
.
```

2.3 Global Identifiers.

The following labels are declared as global identifiers on the library tape and must be declared at the head of all blocks of the user's program which refer to them.

QF
SQRT
SIN
CØS
ARCTAN
LN
EXP

3 ERROR INDICATIONS

If a routine is entered with an impossible operand then an error is displayed. by QFMATH using the ERROR subroutine. The detected errors are listed below.

Function	Error Message	Cause (x is the argument of the function)
logarithm	QFM ERROR 3	$x \leq 0$
square root	QFM ERROR 1	$x < 0$
exponential	QFM ERROR 4	$x > 2^{16}$

4 STORE USED

500 consecutive locations & 24 literals.

5 TIMES

Typical times are; on a 903 or 9208 :

SQRT	5.3 millisec.
SIN	15.5 millisec.
COS	15.5 millisec.
ARCTAN	24.0 millisec.
LN	22.0 millisec.
EXP	13.0 millisec.

CH I/O + LEGTAPE S/R 31/3/70 900-Series Telecode

1. INTRODUCTION.

The tape "CH I/O + LEGTAPE S/R" comprises 3 Subroutines, punched in 900-Series Telecode :-

(a) A subroutine for the input of individual characters in 903/900-Series Telecode or 920 Telecode from paper tape, called "CHIP".

(b) A subroutine for the output of individual characters in 903/900-Series Telecode or 920 Telecode, or in legible form, on paper tape, called "CHOP"

(c) A subroutine for giving error indications, called, "ERROR".

These three subroutines are described individually in the following sections.

Many programs contain "built-in" routines to perform the functions performed by these subroutines. By using the above routines instead of "built-in" ones all input & output instructions can be avoided in tape-handelling programs and two advantages are obtained :-

(1) Writers of new programs can lift the above subroutines "off the shelf" instead of writing a "built-in" routine.

(2) Users of existing programs can easily change them to operate in a different Telecode or via a different peripheral, by writing just one new character subroutine.

2. CHIP; CHARACTER INPUT SUBROUTINE.

2.1 Function & entry instructions.

When CHIP is entered, using the instructions:-

```
11  CHIPL
 8  CHIPE
```

it reads (or, more strictly, appears to read) one character from the paper tape reader. On exit the internal code number corresponding to the character (see the table in Section 5) is in the accumulator and also in the location "CHIP" (which is declared within the subroutine).

The user's program MUST NOT alter the contents of the location CHIP.

The tape being read may be in the following codes which are described elsewhere:-

"900-Series" Telecode
(or ISO or ASCII, with even parity)
903 Telecode
920 Telecode

To enable the subroutine to decide which Telecode is being read, all tapes must start with a newline, carriage return, or linefeed symbol & the appropriate Telecode.

To indicate that a "new" tape is about to be read, the user must set the location "CHIPF" (which is declared within the subroutine) to +0, before reading the first character of the tape. The subroutine will set CHIPF to a non-zero value when it is entered.

Although, to the user, the CHIP subroutine appears to read one character from the tape-reader whenever entered, it actually buffers the text line-by-line; i.e. when first entered it reads a whole line of tape into an array, up to the end of a line or a haltcode symbol, and exits with the first item of the array in the accumulator (and the location "CHIP".) On subsequent entries, subsequent items are read from the array until it is empty, whereupon another line is read in.

Thus this subroutine is suitable for use with non stop-on character readers of up to 250 c/s, irrespective of the speed of the user's program, provided that each newline or linefeed symbol is followed by some blanks.

All tapes should end with a haltcode symbol; if this is not done the last line of the tape will be "lost" in the buffer array. (The subroutine must NOT be entered AFTER finding a haltcode until CHIPF has been reset to zero.) It is a requirement of some subroutines which use this subroutine that the haltcode is preceded by a newline, linefeed, space, or tab symbol, to terminate the last item of data on the tape; otherwise this item is lost.

Blanks and erases are ignored everywhere, as is the carriage return symbol except in the determination of the Telecode of a new tape.

Tapes are read in Mode 3, with a parity-check.

2.2. Special Characters.

The CHIP subroutine performs no code-conversion on 900-Series Telecode or 903 Telecode tapes other than stripping back 8, to obtain the internal code value.

Thus, for tapes punched in 900-Series code;-

"£"	will be given value	35
"?"	" " " "	63
"@"	" " " "	64
"\"	" " " "	92
"`"	" " " "	96.

whereas, for tapes punched in 903 code;-

"½"	will be given value	35
"↑"	" " " "	63
"¬"	" " " "	64
"£"	" " " "	92
"@"	" " " "	96.

Since the symbols "\" and "½" are not used in existing software it is suggested that they are considered to be typographical variants of "£", and that programs searching for "£" search for both value 35 and 92.

Since the symbol "@" is not used in existing software it is suggested that it is considered to be a typographical variant of "`" and that programs searching for "`" search for both values 64 and 96.

Since the symbol "?" is not used in existing software it is suggested that it is considered to be a typographical variant of "↑".

The CHIP subroutine converts 920 Telecode to internal code by means of a look-up table, in which the following conversions are made :-

"~"	will be given value	34	i.e. " "
"£"	" " "	35	
"?"	" " "	63	(as will "10")
"10"	" " "	39,	i.e. "'"
"11"	" " "	96,	i.e. "`".

Also the following compound symbols using the non-escaping vertical bar, will be recognised.

"$"	will be given value	36,	i.e. "$"
"†"	" " "	39,	i.e. "'"
"*"	" " "	96,	i.e. "`".

Vertical bar followed by characters other than S, <, or > (e.g. 2); Horizontal bar, and binary values having no 920 Telecode significance, will give an error indication.

2.3. Possible variants to CHIP subroutine.

The CHIP subroutine could be modified in several ways whilst retaining an identical or near-identical interface with the user's program. Possibilities are:-

a) Versions operating in other Telecodes, e.g. Telex 5-hole code. Versions operating in several Telecodes may need a location "CHIPC" holding a number, set by the user, determining the input code currently in use (like "CHOPC" in the "CHOP" subroutine, in section 3.)

b) Versions operating via other peripherals, e.g. on-line teleprinter. Versions capable of operating via more than one peripheral would require a location "CHIPD" specifying the address of the peripheral.

c) A version NOT using a line-at-a-time buffer array, for programs capable of driving the reader at full speed, or for use with stop-on-character readers. This would save about 80 locations.

d) A version containing a facility for printing the current contents of the line buffer array via the "CHOP" subroutine, for use in error messages.

e) A version with limited line editing facilities, e.g. use of the symbol "←" to delete the current contents of the buffer array.

3. CHOP; CHARACTER OUTPUT SUBROUTINE.

3.1. Function & entry instructions.

When CHOP is entered, using the instructions:-

```
11   CHOPL
 8   CHOPE
```

it punches a character once or several times on the paper tape punch. To punch a character once, the accumulator should contain the internal code number corresponding to the character (see the table in Section 5), when the subroutine is entered.

To punch the same character several times the accumulator should contain $C - 128.N$ where C is the internal code number of the character and N is the number of characters required. (This method can in fact also be used to punch a character once).

The effect of entering the subroutine with the accumulator in the range +128 to +131071 is undefined.

The tape may be punched in one of the following codes which are described elsewhere:-

"900-Series" Telecode
(or ISO or ASCII, with even parity)
903 Telecode
920 Telecode
or in Legible Tape Form.

To indicate to the subroutine which Telecode is to be punched, the location "CHOPC" (which is declared within the subroutine) should be set by the user before the subroutine is entered to one of the following values:-

+1 for "900-Series" or 903 Telecode
+0 for 920 Telecode
&400000 for Legible Tape

and the effect of using other values is undefined.

The subroutine will punch 4 blanks after every newline or linefeed symbol.

The subroutine will punch 18" of blanks before the first character on a tape, and after the last character on a tape.

To indicate that a "new" tape is about to be punched, the user must set the location "CHOPF" (which is declared within the subroutine) to +0, before punching the first character of the tape. The subroutine will set CHOPF to a non-zero value when it is entered.

To indicate the end of a tape, all tapes punched should end with a halttode symbol, EVEN if the output is in legible tape form. If this is not done, the 18" of blanks will not be punched.

On exit from the subroutine the value of the accumulator is the internal code number of the character just punched. (Thus the most significant 10 bits of the accumulator will be zero.)

3.2. Special Characters.

The CHOP subroutine performs no code conversion on 900-Series Telecode or 903 Telecode tapes other than the insertion of even-parity in track 8.

Thus, for tapes punched in 900-Series code:-

Value	35	will	be	punched	"£"
"	63	"	"	"	"?"
"	64	"	"	"	"@"
"	92	"	"	"	"\"
"	96	"	"	"	"\"

whereas, for tapes punched in 903 code:-

Value	35	will	be	punched	"½"
"	63	"	"	"	"10"
"	64	"	"	"	"\"
"	92	"	"	"	"£"
"	96	"	"	"	"@"

Since the symbols "\" and "½" are not used in existing software it is suggested that they are considered to be typographical variants of "£" and that programs punching "£" use value 35.

Since the symbol "@" is not used in existing software it is suggested that it is considered to be a typographical variant of "\" and that programs punching "\" use value 96.

Since the symbol "?" is not used in existing software it is suggested that it is considered to be a typographical variant of "10".

The CHOP subroutine converts internal code to 920 Telecode by means of a look-up table, in which the following conversions are made :-

Value 34,	i.e " " "	will be punched	" ~ "
" 36,	i.e. " $ "	" "	"	" $ "
" 39,	i.e. " ' "	" "	"	" ↓ "
" 64 }	i.e. " ↑ "	" "	"	" ↑ "
" 96 }				

The internal codes of Ⓑ, !, and ←, also the internal codes having no allocated meaning, will give an error indication if used to output in 920 Telecode.

The CHOP subroutine uses a set of tables to convert internal code characters to legible tape patterns, in which the following conversions are made :-

Value 20, i.e Ⓗ, is ignored, (except that it denotes the end of the tape, and thus causes 18" of blank to be punched)

Values 0 to 31 (excluding 20), which includes Ⓝ and Ⓣ, will be punched as Value 32, i.e. Ⓢ.

Values 96 to 127 will be punched as values 64 to 95. Thus lower-case letters will be punched in the same form as upper-case letters.

3.3. Possible variants to CHOP subroutine.

The CHOP subroutine could be modified in several ways whilst retaining an identical or near-identical interface with the user's program. Possibilities are:-

a) Versions operating in other Telecodes, e.g. Telex 5-hole code. Versions operating in one Telecode would not need the location "CHOPC".

b) Versions operating via other peripherals, e.g. on-line teleprinter. (This version need not output blanks after each newline, or at the start and end of the tape; to same time). Versions capable of operating via more than one peripheral would require a location "CHOPD" specifying the address of the peripheral.

c) A version containing no legible tape output. This would save about 260 locations.

d) A version containing legible tape output in which the lower-case letters are punched as 5-track high versions of the upper-case letters.

4. ERROR; ERROR INDICATIONS SUBROUTINE.

4.1. Function & entry instructions.

The subroutine 'ERROR' punches an error message in legible tape form on the paper-tape punch, preceded and followed by 18" of blank tape.

The subroutine is entered by the instructions :-

```
11  ERRORL
 8  ERRORE
```

and the message to be punched should be held in the locations following the entry instructions, in alphanumeric group form.

There is no limit to the length of the message. The end of the message should be indicated by a full-stop within the alphanumeric groups. (The full-stop will not be punched on the legible tape).

Any character having internal code value 32, 35, 36, or 38 to 93 can be punched in the message (see the table in Section 5.)

After punching the error message, this version of the ERROR subroutine enters a DYNAMIC STOP. (To continue after error indications, see section 4.2.)

4.2. Continuation after error indications.

If the user wishes to continue after an error indication has been given, the user's program should contain an entry point which jumps to the instructions

```
0       ERRORL
/8      1.
```

After punching an error message, the ERROR subroutine will enter a dynamic stop. If the above entry point is then used, the error subroutine will perform a conventional subroutine exit to the location after the error message. The user should ensure that this location contains a suitable recovery routine.

(Note that the call of the ERROR subroutine made by CHIP and CHOP does NOT contain an error recovery routine.)

5. THE INTERNAL CODE.

INTERNAL CODE, 1/12/69

0		32	Ⓢ	64	\	96	\
1		33	!	65	A	97	a
2		34	"	66	B	98	b
3		35	£	67	C	99	c
4		36	$	68	D	100	d
5		37	%	69	E	101	e
6		38	&	70	F	102	f
7	Ⓑ	39	'	71	G	103	g
8		40	(	72	H	104	h
9	Ⓣ	41	)	73	I	105	i
10	Ⓝ	42	*	74	J	106	j
11	Ⓥ	43	+	75	K	107	k
12		44	,	76	L	108	l
13		45	-	77	M	109	m
14		46	.	78	N	110	n
15		47	/	79	O	111	o
16		48	0	80	P	112	p
17		49	1	81	Q	113	q
18		50	2	82	R	114	r
19		51	3	83	S	115	s
20	Ⓗ	52	4	84	T	116	t
21		53	5	85	U	117	u
22		54	6	86	V	118	v
23		55	7	87	W	119	w
24		56	8	88	X	120	x
25		57	9	89	Y	121	y
26		58	:	90	Z	122	z
27		59	;	91	[	123	
28		60	<	92	£	124	
29		61	=	93	]	125	
30		62	>	94	↑	126	
31		63	ю	95	←	127	

Ⓑ Bell
Ⓣ Horizontal Tab
Ⓝ Newline, or C/R+L/F
Ⓥ Vertical Tab (Throw)
Ⓗ Halt or Stopcode
Ⓢ Space

6. STORE USED.

"CH I/O + LEGTAPE S/R" uses :-

681	Consecutive locations,
40	Literals.

7. ERROR INDICATIONS.

If an error occurs a message is given using the ERROR subroutine.

Message	Meaning
CH I/O ERROR 1	Parity Error on tape being read.
CH I/O ERROR 2	1st character on tape being read is not a newline, linefeed, or carriage return symbol.
CH I/O ERROR 3	When reading or punching in 920 Telecode; the character cannot be converted to or from internal code.
CH I/O ERROR 4	Over 120 characters, (other than blanks, erases, or carriage return symbols,) on one line.

8. EXAMPLES

8.1. CHIP.

The following program will count how often the symbol "e" occurs on a tape :-

```
((EXAMPLE OF CHIP)
[START  CHIPF  CHIPL  CHIPE]

START   4   +0
        5   CHIPF
        5   COUNT

LOOP    11  CHIPL
        8   CHIPE
        1   -20        (Test for (H))
        7   j+0
        1   -26        (CHIP-46, Test for full stop)
        7   j+2
        8   LOOP
        10  COUNT
        8   LOOP

COUNT   >1
```

8.2. CHOP.

The following program will output the word "Programme" on a newline in 900-Series Telecode:—

```
((EXAMPLE OF CHOP)
[START  CHOPC  CHOPF  CHOPL  CHOPE]

START   4   +1
        5   CHOPC
        4   +0
        5   CHOPF
        5   COUNT

LOOP    0   COUNT
        4   MESSAGE
        7   J+0
        11  CHOPL
        8   CHOPE
        10  COUNT
        8   LOOP

MESSAGE +10     (Ⓝ)
        +80     (P)
        +114    (r)
        +111    (o)
        +103    (g)
        +114    (r)
        +97     (a)
        -147    (mm, 109-256)
        +101    (e)
        +20     (Ⓗ)
        +0      (End of message marker)

COUNT   >1
```

8.3. ERROR.

The following program will punch the error message "UNLOCATED IDENTIFIER".

```
((EXAMPLE OF ERROR)

[EU   ERRORL   ERRORE]


EU      11  ERRORL
         8  ERRORE
        'UNL
        'OCA
        'TED
        'ID
        'ENT
        'IFI
        'ER.
```

8.4. CHIP & CHOP

The following program may be used to join several tapes into one tape. If assembled from 8, the entry points are:

8	to read in first tape, for output in 900-Series or 903 Telecode.
9	to read in first tape, for output in 920 Telecode
10	to read in subsequent tapes
11	AFTER reading in last tape.

((EXAMPLE 8.4)

[CHIPF CHIPL CHIPE CHIP CHOPC CHOPF CHOPL CHOPE]

```
(8)      8      START
(9)      8      ST920
(10)     8      CONTIN
(11)     8      END

START    4      +1
         8      j+2
ST920    4      +0
         5      CHOPC

         4      +0
         5      CHOPF

CONTIN   4      +0
         5      CHIPF

LOOP     11     CHIPL
         8      CHIPE
         1      -20
         7      j+0
         4      CHIP
         11     CHOPL
         8      CHOPE
         8      LOOP

END      4      +20
         11     CHOPL
         8      CHOPE
         8      j+0
```

8.5. CHIP, CHOP, & ERROR.

The following program may be used to delete all the comments from a SIR program, and punch the result in 900-Series Telecode.

((EXAMPLE 8.5)

```
[START CHIPF CHIPL CHIPE CHIP CHOPF CHOPL CHOPE CHOPC]

START   4   +1
        5   CHOPC
        4   +0
        5   CHOPF
        5   CHIPF

TEXT    11  CHIPL
        8   CHIPE
        1   -40         (Test is opening bracket)
        7   COMMENT
        4   CHIP
        11  CHOPL
        8   CHOPE
        4   CHIP
        1   -20
        7   ;+0
        8   TEXT

COMMENT 11  CHIPL
        8   CHIPE
        1   -20
        7   ERROR
        1   -21         (Test is closing bracket)
        7   TEXT
        8   COMMENT

ERROR   11  ERRORL
        8   ERRORE
        "THESE ENDS IN COMMENT."
```

8.6. CHOP used to punch Alphanumeric Groups.

The following print-up is of the ERROR subroutine, which shows how CHOP may be used to print Alphanumeric Groups.

```
((ERROR ROUTINE)

[ERRORL ERRORE CHOPF CHOPL CHOPE CHOPC]

(Prints out the error message following the entry
instructions in alphanumeric group form, until
a full-stop is found, then stops. The message
is given in legible tape form)

ERRORL    >1

ERRORE    4          +0
          5          CHOPF
          4          &400000
          5          CHOPC

NEXTWD    4          -2
          5          COUNT
          10         ERRORL
          0          ERRORL
          /4         0

NEXTCH    5          WORD
          14         8180
          6          &77
          1          -14          (6-bit code test if .)
          7          STOP
          1          +46          (Restore 7-bit code)
          11         CHOPL
          8          CHOPE

          4          COUNT
          7          NEXTWD
          10         COUNT

          4          WORD
          14         6
          8          NEXTCH

STOP      4          +20
          11         CHOPL
          8          CHOPE
          8          ;+0

COUNT     >1
WORD      >1
```

CH I/O S/R 31/3/70 900-Series Telecode

1. INTRODUCTION.

The tape "CH I/O S/R" is the same in function as "CH I/O + LEGTAPE S/R", described elsewhere, with the following differences :-

① "CHOP" cannot be used to output in legible tape form; thus possible values of "CHOPC" are restricted to :-

+1 for 903/900-Series Telecode
+0 for 920 Telecode

and the effect of using other values, including &400000, is undefined.

② As a consequence of ①, "ERROR" cannot give error indications in legible tape form; instead they are given in the current output Telecode, as determined by "CHOPC", preceded by (N) and followed by (H).

(The purpose of this variant is to reduce the amount of store required, by omitting the legible tape patterns which occupy a considerable amount of store).

2. STORE USED.

"CH I/O S/R" uses :-

422 Consecutive locations,
36 Literals.

OPTIMISED MATHEMATICAL SUBROUTINES, 1/12/71, Telecards

1.1. INTRODUCTION.

"OPTIMISED MATHEMATICAL SUBROUTINES" is a SIR tape in 900-Series Telecode, containing 3 subroutines separated by halfrecords. The subroutines are :-

SINE & COSINE SUBROUTINE,	1/12/71,
SQUARE ROOT & PYTHAGORAS SUBROUTINE,	7/8/68,
ARCTAN SUBROUTINE,	1/12/71.

They are all suitable for use on any program level.

1.2. TOTAL STORE USED.

The total store used by the above 3 subroutines is :-

143 Consecutive locations,
22 Literals.

2. SINE & COSINE SUBROUTINE.

2.1. FUNCTION

A subroutine for finding the sine or cosine of an angle (or both)

2.2. METHOD OF USE & ENTRY INSTRUCTIONS.

(1) There are 3 sets of entry instructions, for calculating sine, cosine, or both. Whichever entry is used, the accumulator should contain the angle, Θ, scaled by 180° or π radians, on entering.

E.G. $+\cdot5$ represents the angle $+90^\circ$ or $+\pi/2$
$-\cdot25$ " " " -45° " $-\pi/4$.

(2) The instructions

11 SINL
8 SINE

place $\frac{1}{2}\sin\Theta$ in the ACCUMULATOR

(3) The instructions

11 COSL
8 COSE

place $\frac{1}{2}\cos\Theta$ in the accumulator

(4) The instructions

11 SICOL
8 SICOE

place $\frac{1}{2}\sin\Theta$ in the location SICOS
and $\frac{1}{2}\cos\Theta$ in the accumulator.

Note SICOS is declared within the subroutines tape.

2.3. STORE USED

46 consecutive locations
12 literals

2.4. TIME TAKEN

Maximum number of obeyed instructions :-
21 for Sine
22 for Cosine
50 for Sine & Cosine

2.5. ACCURACY.

Maximum error believed to be $\pm 1 \times 2^{-17}$

3. SQUARE ROOT & PYTHAGORAS.

3.1. FUNCTION

A subroutine for finding the single-length square root of a single or double-length number, and for finding the hypotenuse of a right-angled triangle.

3.2. METHOD OF USE & ENTRY INSTRUCTIONS.

(1) To find the single-length square-root of the single-length fraction in the accumulator :—

```
11   SSSRL
 8   SSSRE
```

On exit the result is in the accumulator.
(Also on exit the operand will be in DSSRA, & location DSSRA+1 will be zero).

(2) To find the single-length square-root of the double-length fraction held in the normal format with m/s part in DSSRA & with l/s part in DSSRA+1 = DSSRQ :-

```
11   DSSRL
 8   DSSRE
```

On exit the result is in the accumulator.
(Also on exit the operand will still be in DSSRA & DSSRA+1).

(3) To find the quantity

$$\sqrt{\text{Accumulator}^2 + \text{PYTHB}^2}$$

```
11   PYTHL
 8   PYTHE
```

On exit the result is in the accumulator.
The operands may be integers, fractions, or numbers with any other scaling (provided they are both the same); the result will have the same scaling.

(Also on exit, Accumulator^2 + PYTHB^2 is held as a double-length number in DSSRA & DSSRA+1. Note that the squares are added double-length to retain single-length accuracy.)

(4) Note that the location DSSRA+1 is also labelled DSSRQ; and that both of these and the label PYTHB are declared within the subroutine tape.

(5) A single-length integer can be square-rooted by the instructions:- or:-

14	8175	5	DSSRA+1
5	DSSRA	4	+0
3	DSSRA+1	5	DSSRA
11	DSSRL	11	DSSRL
8	DSSRE	8	DSSRE.

The right-hand method is faster, but if it is used, para. 3.3. (1) does not apply.

3.3. NEGATIVE NUMBERS & OTHER ERRORS.

(1) If the operand of the routine is negative, the result +0 is given. An error indication is NOT given, since, even in correctly written programs the square-root subroutine may be entered with a small negative number in the accumulator due to rounding errors; in these cases +0 is the correct answer.

(2) It is the user's responsibility to ensure that the addition of (Accumulator^2 + PYTHB^2) does not overflow.

(3) The sign bit of DSSRA+1 will be ignored. In a correct program it should be 0.

3.4. STORE USED

51 Consecutive locations
3 Literals.

3.5. TIME TAKEN μsec.

		Obeyed instrucs.	920B 6μs store	920M 5μs store	920M 2μs store
Minimum except for special cases....	PYTH	34	1010	740	300
	SSSR	21	620	450	260
	DSSR	19	550	390	240
... plus, per iteration.		11	330	230	140

The number of iterations required depends on the magnitude of the operand.

SSSR takes 11 iterations maximum;
142 obeyed instructions.

3.6. ACCURACY.

Max error $\pm 1 \times 2^{17}$.

4. ARCTAN SUBROUTINE.

4.1. FUNCTION.

A subroutine to calculate the arctangent (i.e. the principal value of "inverse tangent") of a quantity.

4.2. METHOD OF USE & ENTRY INSTRUCTIONS.

To calculate Arctan $\left(\frac{S}{C}\right)$ place :-

C in the location ATANC
S in the Accumulator.

The instructions:-

11 ATANL
8 ATANE

will place $\frac{1}{\pi}$ · Arctan $\left(\frac{S}{C}\right)$ in radians,
i.e. $\frac{1}{180}$ · Arctan $\left(\frac{S}{C}\right)$ in degrees,
in the accumulator.

In the indeterminate case, i.e. $S=C=0$,
a result of +0 will be given.

Note ATANC is declared within the subroutine tape,
(and that, on exit, its original value will have been lost).

4.3. STORE USED.

46 consecutive locations
12 literals.

4.4. TIME TAKEN.

36 obeyed instructions maximum.

4.5. ACCURACY.

Max. error believed to be $\pm 1 \times 2^{-17}$

SHELLSORT 17/8/71 Telecode.

Chapter 1: INTRODUCTION

1.1 Purpose

To sort a number of fixed length records (data items) held in core store into ascending or descending order.

1.2 Summary

A record must be a number of consecutive words. The file to be sorted must consist of consecutive records in core store. Records are split into two parts, the data-area and the key (the part on which the sorting is to be performed). The key itself is split into parts which are to be sorted on individually. Any part of the key may occupy between 1 and 18 consecutive bits in any one word of the record. Information about the file and the way in which it is to be sorted is given in a sort-table and a sort-list. The former specifies details of the file and the latter details on how the file is to be sorted.

1.3 Form of Distribution

Shellsort is distributed as a mnemonic tape for input under 900 SIR.

1.4 Method of Use

Shellsort is assembled as a block of the users program and entered in the standard manner, storing the link in SHELLSORT and transferring control to SHELLSORT+1. A parameter word immediately following the entry must contain the address of the sort-table. Exit is to the location following the parameter word.

Example

```
11 SHELLSORT
8  SHELLSORT+1
0  SORTTABLE
```

1.4.1 Entry and Exit Conditions

The content of the A, Q and B registers is irrelevent on entry and undefined on exit. On exit the file will have been sorted

'in situ' by exchanging complete records.

1.5 Restrictions

The following restrictions on the use of SHELLSORT should be noted:-

(i) The file must be of the specified format.

(ii) If a file is sorted 'n' deep, then the two arrays SHIFTS and COLLS in SHELLSORT must be declared as ≥ (n+1). On the standard tape SHIFTS and COLLS are declared as ≥ 10, i.e. sorted 9 deep is permitted.

(iii) All records must have identical format.

Chapter 2: FUNCTIONS

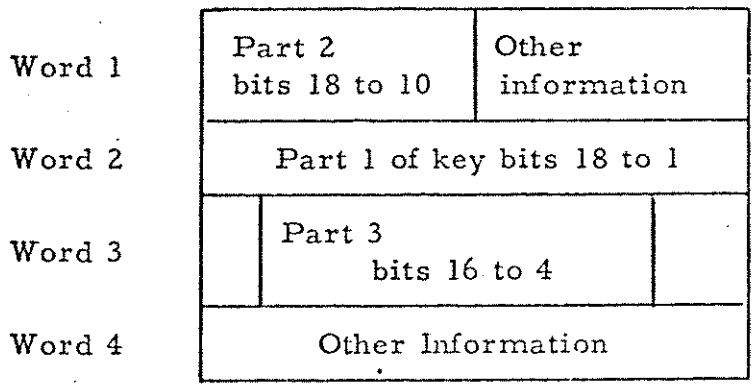
2.1 Format of File

The records to be sorted may be of one or more 18 bit words. They must reside in a continuous area of core store and must all be of the same length. The values on which the sort is to depend are held in a key, which must be of the same format in every record.

The key may form part or all of the record, it may consist of one or many parts, (the maximum number of parts is 9 in the standard program, but see Chapter 1.5).

Each part of the key may occupy the whole or part of an 18 bit word in the record. One part of a key cannot extend over more than one 18 bit word in core store, but see Chapter 2.5.

Example: 4 word Record, 3 part key



2.2 Sort-table

The sort-table is of the following form:-

```
SORTTABLE    +400          (number of records)
             +3            (number of words per record)
             Ø FILE        (the address of the first location
                            of the file)
             Ø SORTLIST    (the address of the sort-list)
```

+2	(the word of the record containing part 1 of the KEY (The first word of the record is word 1)
&ØØØ777	(The collating constant for part 1 of the KEY i. e. in this case part 1 of the KEY is the 9 least significant bits of word 2)
+3 &ØØ1777)	as above but for part 2 of KEY.
etc.	
+Ø	(end of sort-table).

2.3 Sorting on Negative parts of the Key.

Any part of the key consisting of n consecutive bits may be regarded as

i) An unsigned n bit number in the range 0 to $2^{(n-1)}$ i. e. always positive.

OR ii) An (n) bit number with bit (n) representing the sign bit, i. e. negative values are permitted, (stored in twos complement form, as for 18 bit machine words)

To specify the way in which any particular part of the key is to be interpreted (i. e. as above), the part numbers in the sorting priority table of the sort-list should contain +n (where 'n' is the relevant part of the key) if this is to be interpreted as (i) and -n if it is to be interpreted as (ii). The absolute value of the number entered will be taken as the relevant part of the key.

2.4 The sort-list

The sort-list is of the following form:-

SORTLIST	+1	(sort file in ascending order, would be -1)
		(if file to be sorted in descending order)
	+5	
	+3	
	+4	
	+6	(see below *)
	+1	(and 2.3)
	+2	
	+Ø	

*This table specified the priorities in which the parts of the KEY are to be sorted.

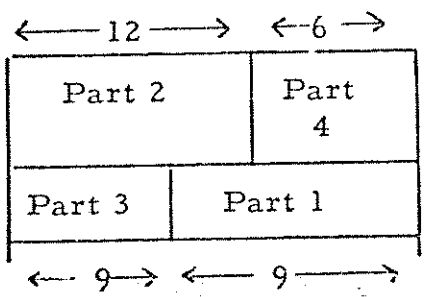
e.g. In this case the file is sorted on part 5 of the KEY. If two entries are found equal in part 5, then these two records are sorted on part 3 etc.

The word containing each part of the KEY and its collating constant are given in the SORT-TABLE.

The sort-table and the sort-list must be supplied by the user. The address of the sort-table must be placed in the location following the entry to SHELLSORT.

N.B. The order in which the parts of a KEY are numbered is arbitary. Having numbered the parts, however, the items in SORTTABLE +4 onwards must be set-up accordingly.

e.g. A KEY of the form

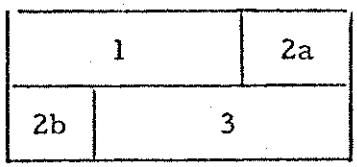


must be specified as

```
+2
&ØØØ777
+1
&7777ØØ
+2
&777ØØØ
+1
&ØØØØ77
+Ø
```

2.5 Sorting 'end-around'

It is possible that the key of a record may contain a part stored 'end-around' i. e.



Parts of PART 2 of the KEY are stored in both words 1 and 2 of the record. This may be sorted on by specifying the sort on part 2a, and, if these are found equal, then sort out part 2b. The parts must be numbered 1, 2, 3, 4 for 1, 2a, 2b, 3 respectively. This will achieve the same effect as sorting on part 2 stored as consecutive bits. Only positive parts of keys may be sorted on in this manner.

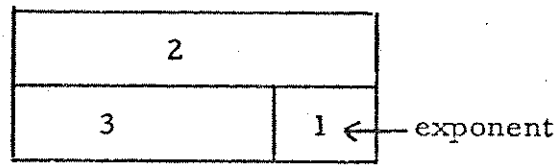
2.6 Character Sorts

Internal and/or tape code characters may be sorted using SHELLSORT by representing each character as a separate part of the key.

2.7 Sorting Floating Point Numbers

2.7.1. Packed Numbers

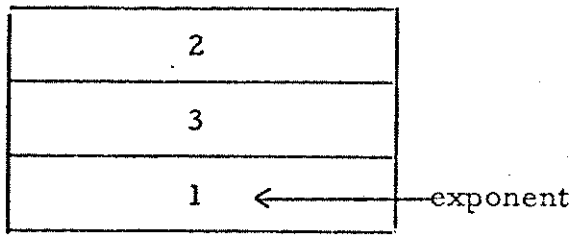
Packed numbers are regarded as two word records. Sorting is on firstly, the exponent, secondly the most significant bits of the mantissa and lastly on the least significant bits of the mantissa. i.e. the sorting sequence is:-



Negative numbers must be permitted in parts 1 and 2.

2.7.2 Unpacked Numbers

Specify 3 whole word keys, on the order : exponent, most significant mantissa, least significant mantissa.



2.8 Use of Extra Modules of Core Store

The main program, the sort-table, the sort-list and the file may be stored in different modules, providing that when the address of (a) the sort-table (b) the sort-list and (c) the file, is specified in (a) the parameter word (b) the sort-table and (c) the sort-table, the value specified is the address relative to the module containing SHELLSORT. e.g. If SHELLSORT is in module 1 and the file in module 2, then the sort-table should specify 1 FILE (i.e. 8192 + module 2 address of FILE), as the address of the file, regardless of the module containing the sort-table. Shellsort must be in the same module as the main program. The file may extend over

more than one module of core store, without restriction, except that it may only overwrite locations 8180 to 8191 of module 0 if the initial instructions are disabled.

2.9 Program Levels

Shellsort may be run at any program level.

Chapter 3: METHOD USED

The method used is a high speed sift sort technique with a varying interval of comparison and exchange. The method is described in "A HIGH SPEED SORTING PROCEDURE" by D. L. SHELL in "Communications of the A. C. M." Vol. 2 No. 7 of July 1959.

Chapter 4: SPEED AND STORE USED

4.1 Examples of Time Taken

On the 903, the time taken to sort 1000 seven word records, sorting 9 deep (i.e. each key consists of 9 parts) was approximately 2.5 minutes.40 three word records, sorted 6 deep, took approximately 5 seconds.

Equivalent times on 905, with one microsecond store, are 20 seconds and 0.6 seconds respectively.

It must of course be emphasised that times depend largely on the random nature or otherwise of the records, especially the number of records that can be sorted on the first part of the key.

4.2 Store Used

The store used is approximately 220 locations of code, and 17+2n (where n is the sort depth) locations of data; plus, of course, the users file, sort-table, and sort-list.

Actual store used by standard tape (i.e. n = 9):-

239 consecutive locations
6 literals