TO: Distribution List

FROM: J. Uskavitch

SUBJECT: MTC Subroutine Library

Group 22 Internal Memorandum No. 22-SA-56

Abstract: Five general-utility subroutines are presented in coded form for use on MTC. Information is also given about average operation time, accuracy, storage requirements and operating details pertinent to each subroutine. Explanatory notes are also included with each one for the aid of programmers.

The subroutines included are:

A diagram at the end of the memorandum shows one set of storage registers for eight subroutines for which tapes are available.

General Remarks:

All of the subroutines given in this memorandum have been tested on MTC and are believed to be free of errors. However, if mistakes are found in any cases, the author will sincerely appreciate having his attention called to them.

In developing these subroutines, the writer has attempted to make them as easy to use as possible. Consequently, time and space have sometimes been sacrificed to increase the ease of programming; moreover, whenever possible, space has been saved in preference to a few micro-seconds of operating time. However, the writer will sincerely appreciate receiving any information concerning excessive use of either space or time in these subroutines.

As will be noted in the following pages, all addresses have been written as octal numbers, thus minimizing the work needed for transcription into the normally used tapes. However, in all other respects, the notations used are identical with those of Memorandum 6M-3497 including the re-use of the letters a, b, c, d, and e to denote the individual subroutines. The writer has reused these denotations because he did not know what other additions were being made to the subroutine library and consequently did not know just where these subroutines would fit in the library. In any case, when the number of MTC subroutines become large enough, he hopes that Group 62 will re-issue all of them with identical formats in one memorandum.

The duration time given for each subroutine must be considered as only a rough approximation. All operations have been assumed to occur within core memory, and individual execution times were obtained from Drawing B-62366, which follows page 30 of 6M-2527-2. However, in all of the subroutines, the exact operating time depends on the input, and only an approximate averaging was performed over the possible inputs.

The accuracies associated with these subroutines are all derived from round-off errors and were calculated in the same manner as described in 6M-3497. The one exception to this statement is found in subroutine (e), the accuracy of that output being determined in part by that of SRL #7 of 6M-3497.

The general notation used in this memorandum may be summarized as follows:

pN ≡ relative address of a register in the operating part of a program, where p is a lower-case letter designating a particular subroutine and N is an octal integer.

KpN ≡ relative address of a register containing a constant used in the program or being used for temporary storage where p and N are as above.

k ≡ the field in which the particular subroutine operates.

m ≡ the field in which the main program operates.

Included at the end of this memorandum is a diagram which depicts the storage locations utilized by available tapes of several of the MTC subroutines. The writer gladly offers to lend either the flexo or the converted tape if a reproduction is desired by anyone.

(a) ADDITION, WITH SCALE FACTOR

Input : Store the number ($N \leq 2^{11}$) of the numbers (x_i) to be added in Ka3 and the extended address of the first number (x_1) in Ka4. Enter at a0 after ra a130 with m. Content of AC is immaterial. (All x_i must have been previously stored in consecutive registers in a single field.)

Output: $2^{-n} \sum_1^n x_i$ (the 15 most significant binary digits of $\sum_1^n x_i$) in AC, with the scale factor n left in Ka5. (The remaining n digits of the sum will be left in the first n digit positions of BR; i.e., positions 0 through n-1.) If the result is zero, it will be a negative zero, unless all $x_i = +0$.

Other Subroutines Needed: None.

Duration: Approximately 915(N+2) microseconds.

Accuracy: Maximum error in $\sum_1^n x_i$ is 2^n for $n > 0$. (For $n = 0$, no error exists.) The answer obtained is exactly that which one would get by just taking the 15 most significant binary digits of the sum without first rounding off. Therefore, the results in the AC is always smaller than the true value, within this tolerance. No error, if the n digits in the BR are also utilized.

Storage: 90 program registers plus 9 constants or temporary storage.

Notes:
(1) The original numbers (x_i) are not destroyed by the addition process.
(2) If x_1 is in register 376 of field 3, the extended address should be stored as 0.30376.
(3) The scale factor used is always the smallest one possible. For example, as long as the sum requires no more than 15 digits, no scale factor is used (n = 0).

(a) ADDITION , WITH SCALE FACTOR

```
Program:
(enter) a0   rf   a131    set up to leave subroutine
        1    ca   Ka0     +0                                  initial
        2    st   Ka5     n                                   conditions.
        3    st   Ka6     Σ n
        4    st   Ka7     ≤ r
        5    ca   a14     sr14
------------------------------------
(a53)-> 6    ra   a45
        7    ca   a14     sr14               prepare for ex-
        a10  su   a45     sr(14,11,6,3,0)    traction of prop-
        11   ra   a23                        er digits
        12   ca   Ka4     extended address of x_1
        13   ra   a21     ca(RCx_1)
        14   sr   14
        15   ra   a20     sof( )
        16   cs   Ka3     N
(a37)-> 17   st   Ka10    counter
------------------------------------
        a20  sof--
        21   ca --        x_i                 obtain x_i in AC.
        22   sof k
        23   sr --        (0,3,6,11,14)       extract the sign plus three
        24   et   Ka2     1.00007             digits, beginning with the
                                              3 most insignificant
        25   tn   a27                         is the sign negative?
        26   tr   a30                         if no, transfer to add. if
        27   su   Ka2     1.00007             yes, correct extracted number
                                              to provide true value of di-
                                              its.
        a30  ad   Ka6     ≤ n                 add to sum of same digits
        31   st   Ka6     ≤ n                 obtained from preceding x's
                                              and store.
        32   ca   a21                         prepare to extract
        33   ad   Ka1     +1                  from x_i+1.
        34   ra   a21
        35   ca   Ka1     +1                  add + 1 to counter.
        36   sm   Ka10
        37   tn   a17     have these digits been obtained from all x_i?
                          if no, return to pick up x_i+1
------------------------------------
        a40  ca   Ka6     Σ n                 if yes, shift 3 most insigni-
        41   sr   3                           ficent digits of the sum into
        42   st   Ka6     Σ n                 the BR. Store those left,
                                              for addition of next 3 ex-
                                              tracted digits.
```

(a) ADDITION, WITH SCALE FACTOR

Program:

a43	sr	15		obtain the remainder in the AC in correct position for addition to preceding remainders.
44	cr	403		
45	sr	--	(14,11,6,3,0)	
46	ad	Ka7	Σr	add to preceding remainders and store.
47	st	Ka7	Σr	
a50	ca	a45	sr(14,11,6,3,0)	prepare to change the position in the AC of the next remainder.
51	su	a41	sr 3	
52	tn	a54		have all digits been added? if yes, transfer to combine the two sums Σn and Σr. If no, return to reset initial conditions, and to add next 3 digits.
53	tr	a6		
54	ca	Ka0	+0	if yes, 15 most insignificant digits are now in Σr, with the larger ones being in Σn.
55	sm	Ka6	Σn	is $\lvert\Sigma n\rvert = 0$?
56	tn	a61		if no, transfer to continue.
57	ca	Ka7	Σr	if yes, put Σr in the AC, and transfer out.
a60	tr	a130		
61	ca	Ka0	+0	if $\lvert\Sigma n\rvert \neq 0$,
62	sm	Ka7	Σr	is $\lvert\Sigma r\rvert = 0$?
63	tn	a67		if no, transfer to continue.
64	ca	Ka6	Σn	if yes, make Σr the same sign as Σn. (this particular amount of of shifting was picked to also provide a needed constant.)
65	sr	21		
66	st	Ka7	Σr	
67	cs	Ka6	Σn	
a70	mh	Ka7	Σr	are the signs of Σn and Σr the same
71	tn	a106		if yes, transfer to find n, and to combine.
72	ca	Ka6	Σn	if no, proceed to prepare the two sums for proper combining.
73	tn	a76		
74	su	Ka1	+1	decrease the magnitude of Σn by 1.
75	tr	a77		
76	ad	Ka1	+1	
77	st	Ka6	Σn	
a100	ca	Ka7	Σr	
101	tn	a104		
102	su	Ka1	+1	decrease the magnitude of Σr by 1.
103	tr	a105		
104	ad	a1	+1	
105	st	Ka7	Σr	

(a) ADDITION, WITH SCALE FACTOR

Program:

```
(a71)→106   ca   Ka6     Σn      } to find n, first obtain |Σn| .
      107   sm   Ka0     +0      }
    →a110   st   Ka10    |Σn|    }
      111   su   Ka1     +1      } is |Σn| , or subsequent remainder
      112   tn    a121           } =0? if yes, transfer to combine to
                                 } Σn and Σr.
      113   ca   Ka5             }
      114   ad   Ka1     +1      } if no, add +1 to the scale factor
      115   st   Ka5             }  n.
      116   ca   Ka10    |Σn|    }
      117   sr    1              } shift 1 digit out of |Σn|
     a120   tr    a110   transfer back to store the remainder and to
                         ask if it is zero.
_____________________________________________________________
(a112)→121  ca   Ka5     n       } prepare to obtain the 15 most sig-
      122   ad    a65    sr21    } nificant digits in the AC.
      123   ra    a127           }
      124   ca   Ka7     Σr      } put Σr in digits 0-15 of BR.
      125   sr    17             }
      126   ca   Ka6     Σn        put Σ n in the AC.
      127   cr   --      (21+N)    obtain the 15 most significant
                                   digits in AC.
(a60)→a130  sof  --      } leave subroutine.
      131   tro  --      }
_____________________________________________________________

Ka0    +0
  1    +1
  2     1.00007      extractor
  3    --            N = no. of numbers to be added
  4    --            extended address of x1
  5    --            n = scale factor
  6    --            Σn           }
  7    --            Σr           } temporary storage
Ka10   --            counter; |Σn| →0 }
```

(b) DECIMAL PRINTOUT

Input: Store the position (n) of the binary point in Kb30: n = +0 for an integer, +0<n<+ 15 for a non-integer, n = +15 for a fraction less than one. After ra b53 with m, enter with x in AC either at b0 (if sign is wanted) or at b10 (if only the magnitude is desired).

Output: Prints + or - sign, if wanted. Then, if x is an integer, prints five decimal places, with initial zeros being replaced by spaces; if x is a fraction, then prints a decimal point followed by four decimal digits; if x is a non-integer >1, prints the integral part followed immediately by the decimal point and fractional part (each as described in preceding cases). Two spaces are printed after every number.

Other Subroutines Needed: None.

Duration: Approximately 0.125 second for the sign plus an average of 0.9 sec. for an integer or fraction less than one, and 1.8 seconds for a non-integer greater than one.

Accuracy: No error for an integer. A maximum error of 0.0001 (decimal) for both of the other cases. The fraction printed is exactly that which one would obtain by taking only four decimal places of the true value without rounding off; therefore the fraction printed is always less than its true value, within this tolerance.

Storage: 77 (decimal) program registers plus 27 constants or temporary storage.

Notes: 1. If the integral part of x is known to have less than five significant figures, time and space can be saved by ra Kb 24 with (or st Kb24) the address of the register containing +1000, +100, +10, or +1 depending on the number of significant figures.

2. For those more used to the conventional scale factor, it may be well to note that the position of the binary point is essentially the same as (+15 - n), when the scale factor is 2^{-n}.

(b) DECIMAL PRINTOUT

```
Program
(enter if -->  b0    rf    b54              set up to leave subroutine.
sign is         1    st    Kb31    x
wanted.)        2    tn    b5
                3    ca    Kb17    +13(+)
                4    tr    b6               find sign of x and print.
                5    ca    Kb20    +29(-)
                6    pr    400
                7    tr    b12
---------------------------------------------
(enter if --> b10    rf    b54
only mag.      11    st    Kb31    x
is wanted)     12    cs    Kb31    x        obtain and store magnitude
               13    tn    b15              of x.
               14    st    Kb31    |x|
               15    cs    Kb30    n
               16    ad    Kb7     +15      does  n = 15?
               17    tn    b37              if yes, transfer to print a
- - - - - - - - - - - - - - - - - - - - - - fraction.
              b20    su    Kb7     +15      if no, does n = 0?
               21    tn    b24              if n ≠ 0, transfer to print
- - - - - - - - - - - - - - - - - - - - - - non-integer.
               22    tr    b55              if n=0, transfer to print an
                                            integer.
(b102) -->     23    tr    b50              return, and transfer to leave
                                            subroutine.
---------------------------------------------
               24    cr    2000             clear BR
               25    ca    Kb30    n
               26    ra    b30     sr(n)    obtain and store for printing
               27    ca    Kb31    |x|      the integral part of x.
              b30    sr    (n)
               31    st    Kb31    x.->|x|
               32    cr    420              obtain and store temporarily
               33    st    Kb32    .x'      the fractional part of x.
               34    tr    b55              transfer to print an integer
- - - - - - - - - - - - - - - - - - - - - -
(b102) -->     35    ca    Kb32    .x       return, and prepare to print
               36    st    Kb31    .x->|x|  fraction.
(b17)  -->     37    ca    Kb21    +17(.)   print decimal point.
              b40    pr    400
               41    ca    Kb31    |x|      find 10,000 times the fraction
               42    mh    Kb0              and prepare to print as integer
               43    st    Kb31    |x|
               44    ca    Kb23    suKb1    prepare to start with a
               45    ra    b65              subtraction of +1000.
```

(b) DECIMAL PRINTOUT

```
                    46     ca    b111     tn b76         set up to print initial zeros.
                    47     tr    b60                     transfer to print.
------------------------------------------------
(b23, b102)-->     b50     ca    Kb22     +8(space)
                    51     pr     400                    when x has been printed,
                    52     pr     400                    print two spaces.
                    53     sof    --
                    54     tro    --                     leave subroutine
------------------------------------------------
(b22, b34)--->      55     ca    Kb24     su Kb0         when an integer, prepare to
                    56     ra    b65                     start with subtraction of +10000.
                    57     ca    Kb25     tn b107        prepare to replace initial zeros
(b47)   ------>    b60     ra    b66                     with spaces.
                    61     rf    b102                    set up to return.
(b106) ------>      62     ca    Kb26     ca Kb5         initially set flexo code
                    63     ra    b76                     to print a zero.
------------------------------------------------
(b75)  ------>      64     ca    Kb31     |x|            is |x| or later remainder ≥
                    65     su     --      (Kb0-Kb4)      10,000 (1000,100,10 or 1)?
                    66     tn     --      (b76 orb107)   if no, transfer to print the
                                                         digit or space if initial zero
                                                         not wanted.
                    67     st    Kb31     |x|            if yes, store the remainder.
                   b70     ca    b76
                    71     ad    Kb4      +1             add 1 to the Flexo-code
                    72     ra    b76                     storage, to be printed.
                    73     ca    b111     tn b76         since this digit is > 0, prepare
                    74     ra    b66                     to print subsequent zeros.
                    75     tr    b64                     return to subtract again.
------------------------------------------------
(b66, b111) -->     76     ca     --      (Kb5-Kb16)     obtain Flexo-code in AC
                    77     pr    400                     and print the digit
                  b100     ca    Kb27     su Kb3
                   101     su    b65      su Kb0-Kb4)    is this the last digit?
                   102     tn     --      (b23,b35,b50)  if yes, leave.
------------------------------------------------
(b114) ----->      103     ca    b65
                   104     ad    Kb4      +1             if no, prepare for subtraction
                   105     ra    b65                     of next smaller power of 10.
                   106     tr    b62                     return to find next digit.
------------------------------------------------
(b66) ------>      107     ca    Kb27     suKb3          if digit is an initial zero,
                  b110     su    b65      suKb0-Kb4      is it the last digit?
                   111     tn    b76                     if yes, transfer to print it.
                   112     ca    Kb22     +8(space)      if no, print a space.
                   113     pr    400
                   114     tr    b103                    prepare to find next digit.
------------------------------------------------
```

(b) DECIMAL PRINTOUT

Kb0	+10000		
1	+1000		
2	+100		
3	+10		
4	+1		
5	+62	(0)	Flexo codes
6	+21	(1)	
7	+15	(2)	
Kb10	+7	(3)	
11	+11	(4)	
12	+19	(5)	
13	+27	(6)	
14	+23	(7)	
15	+3	(8)	
16	+54	(9)	
17	+13	(+)	
Kb20	+29	(-)	Flexo codes
21	+17	(.)	
22	+8	(space)	
23	su Kb1		
24	su Kb0		
25	tn b107		
26	ca Kb5		
27	su Kb3		
Kb30	--	n (postion of binary point)	
31	--	x, |x|, remainder, etc.	temporary storage
32	--	.x(fractional part of |x|)	temporary storage

(c) POINT DISPLAY

Input: Having ra c36 with m, put abscissa (x) in Kc10 and ordinate (y) in Kc11 and enter at c0. Content of AC is immaterial. (The constants required in registers Kc1 - Kc7 and the two cr instructions in c21 and c25 must have previously been stored. See notes below.)

Output: If $x_{min} \leq x \leq x_{max}$ and $y_{min} \leq y \leq y_{max}$, where the minimum and maximum values are the desired limits for plotting, the point (x,y) will be displayed N times on the oscilloscope with whatever linear scale desired. If either coordinate is outside the set limits, no point is displayed and the program may be halted. (See notes below.)

Other Subroutines Needed: None.

Duration: (413 + 76N) microseconds.

Accuracy: Each coordinate is displayed with an accuracy of 1 part in 2^9.

Storage: 35 (decimal) program registers, 11 (decimal) constants or temporary storage.

Notes: 1. Since the intensity of the displayed point may want to be varied -- either to distinguish one set of points from another or to provide for easy distinction from a superimposed grid, the number of times (N) that the point is to be displayed must be stored in Kc1.

2. When the desired coordinate limits have been determined, the coordinates of the middle point (x_m, y_m) must be stored in Kc2 and Kc5, respectively. If this point is displayed, it appears at the center (+0,+0) of the display.

Examples:				
	x_{max}	+.9999	+200	+20000
	x_{min}	−.9999	+100	−20000
$\frac{1}{2}(y_{max} + y_{min})$:	Kc2	+0	+150	+0
	y_{max}	+100	−1	+.9999
	y_{min}	+0	−11	+.5000
$\frac{1}{2}(y_{max} + y_{min})$:	Kc5	+50	−6	+.7499

3. Since the limits of the display may be smaller than those of the available data, the desired ranges of the coordinates (with a factor of $\frac{1}{2}$) must be stored in Kc3 and Kc6. The factor of $\frac{1}{2}$ is included to provide for the times when the range is greater than 0.77777. A continuation of previous examples gives:

(c) POINT DISPLAY

$\frac{1}{2}(x_{max} - x_{min})$:	Kc3	+.9999	+50	+20000
$\frac{1}{2}(y_{max} - y_{min})$:	Kc6	+50	+5	+.2499

4. To provide for a display with the proper scales the two scale factors must be stored in Kc4 and Kc7. To be safe, each scale factor must be such that the product of it and the half-range $\leq$ +511; i.e., (Kc3) (Kc4) $\leq$ 511 and (Kc6) (Kc7) $\leq$ 511. continuing the previous examples:

x-scale factor:	Kc4	+511	+10	+.0255
y-scale factor:	Kc7	+10	+102	+2044

5. The cr instructions to be stored in c21 and c25 are determined by the ranges and scale factors used. If both are $\ngtr$ +1 and short-cycle is used, 12 (octal) places must be cycled; if either is $<$ +1, 31 (octal) places must be cycled. Concluding the previous examples:

program x:	c21	cr431	cr412	cr431
program y:	c25	cr412	cr412	cr431

6. If the point to be displayed lies outside the predetermined region, one of several actions may be wanted. As programmed here, ha 0 is given and pushing the restart button changes frames and again halts the program. However, if the point should merely be ignored, then the instruction tr c36 can be put in c41; if one wished to return to a different part of the main program when this happens, an sof and a tro instruction can be stored in c41 and c42 respectively.

(c) POINT DISPLAY

Program:

(enter)	c0	rf	c37	set up to leave subroutine.	
	1	ca	Kc10	x	obtain the difference between x and the median of the range. if this difference produces an overflow, x lies outside desired range; transfer to halt.
	2	su	Kc2	$x_m = \frac{1}{2}(x_{max}+x_{min})$	
	3	to	c40		
	4	st	Kc10	$x(= x-x_m)$	
	5	ca	Kc3	$x/2=\frac{1}{2}(x_{max}-x_{min})$	is $x_{min} \le x \le x_{max}$? if not, transfer to halt.
	6	sm	Kc10	$\lvert x-x_m \rvert$	
	7	tn	c40		
	c10	ca	Kc11	y	obtain the difference between y and the median of the range. if this difference produces an overflow, y lies outside desired range; transfer to halt.
	11	su	Kc5	y_m	
	12	to	c40		
	13	st	Kc11	$y(= y-y_m)$	is $y_{min} \le y \le y_{max}$? if not, transfer to halt.
	14	ca	Kc6	$\Delta y/2$	
	15	sm	Kc11	$\lvert y-y_m \rvert$	
	16	tn	c40		
	17	ca	Kc10	$x-x_m$	scale the distance from display center in x-direction.
	c20	mh	Kc4	x-scale factor	
	21	cr	( )	put in AC in position to display.(note5.)	
	22	st	Kc10	$x-x_m$	
	23	ca	Kc11	$y-y_m$	scale the distance from display center in y-direction.
	24	mh	Kc7	y-scale factor	
	25	cr	( )	put in AC in position to display. (Note 5.)	
	26	st	Kc11	$y-y_m$	
	27	cs	Kc1	N	set up counter.
	c30	st	Kc12	counter	
	31	ca	Kc11	$y-y_m$	display (x,y).
	32	ds	Kc10	$x-x_m$	
	33	ca	Kc0	$+1$	add $+1$ to counter.
	34	sm	Kc12	counter	
	35	tn	c30	if not displayed N times, go back to display again.	
	36	sof	--	leave subroutine.	
	37	tro	--		

(c) POINT DISPLAY

c40	(ha 0)	if x or y is not in desired range, halt.
41	(op 0)	upon RESTART, change frame and halt.
42	(ha 0)	(Note 6.)

Kc0	+1	
1	--	N: no. of times each point is to be displayed
2	--	$x_m = \frac{1}{2}(x_{max} + x_{min})$: median of desired range.
3	--	$\Delta x/2 = (x_{max} - x_{min})$:half the desired range.
4	--	x-scale factor
5	--	$y_m = \frac{1}{2}(y_{max} + y_{min})$: median of desired range.
6	--	$\Delta y/2 = \frac{1}{2}(y_{max} - y_{min})$: half the desired range.
7	--	y-scale factor
Kc10	--	x; $x-x_m$; } temporary storage
11	--	y; $y-y_m$; } temporary storage
12	--	counter

(d) AXES - GRID DISPLAY

Input: First ra c36 with m. Then enter at d0 if a new frame is wanted in the camera, otherwise at d1. Content of AC is immaterial. (The constants required in Kd1-Kd7 must have been previously stored. See notes below.)

Output: A set of orthogonal axes displayed N times on the oscilloscope with any predetermined origin. A set of grid lines, with any predetermined spacings, displayed once to provide linear scales measured from the origin. (See notes below.)

Other Subroutines Needed: (c) Point Display.

Duration: To display axes:
$[(134N)(\text{no. of points / axis}) + (134)(\text{no. of points/ grid line})]$ microsec. Maximum $\simeq 137(N+1)$ milliseconds.
For each grid line displayed, add:
$[109 + 67\,(\text{no. of points/grid line})]$ microseconds.
Maximum $\simeq 68$ milliseconds / line

Accuracy: No inaccuracy.

Storage: 63 (decimal) program register, 8 constants, and 6 registers of subroutine (c).

Notes:

1. Although this subroutine requires only 6 additional registers to make it independent of the preceding routine, the writer cannot foresee an independent use and consequently has saved 6 registers. The changes required to make this an independent subroutine are believed to be obvious.
2. This subroutine may be much longer than actually necessary because of the many variations permitted: not only may the no. of times (N) that the axes are displayed be changed independently of the no. of times that points are displayed in subroutine (c), but also the densities of the points in the axes and in the grid lines may be changed independently. This great variability has been included since large variations in the display equipment and/or photographic equipment and developing have in the past made it impossible to choose optimum values.
3. The densities of the axes and grid lines are determined by the values of Δ_a and Δ_g put in Kd2 and Kd3. These numbers are easiest stored as positive octal numbers, the first three digits giving the spacing between points and the last two being zeros. (See Note 6 for an example.)

4. The coordinates of the origin are also most easily stored as octal numbers, remembering that the <u>ds</u> instruction only uses the sign plus first three digits. If a coordinate is positive, the last two digits should be zeros, as is the case for the densities; if a coordinate is negative the last two octal digits should be made 77. (See Note 6 for example.)
5. Instead of storing the distance between adjacent grid lines, the half-distances are to be put in Kd6 and Kd7. Kd6 provides the spacing of the vertical lines, while Kd7 gives that of the horizontal ones. The half-distance is used to provide for the case when the desired spacing is greater than 0.77700. Each half-distance can be stored as a positive octal constant with a zero in the last digit, the first four digits producing the spacing. (See Note 6 for an example.) In case no grid lines are wanted, merely store 0.77740 in both registers. However, the axes are always further intensified by one pair of superimposed grid lines.
6. Example: Origin to be located in center of left half-plane. The axes are to be displayed 5 times with only 50% of the points being displayed. Only one out of every eight points is to be displayed in each grid line. The vertical grid lines are to be 100 points apart, whereas the horizontal spacing is to be 175 points.

Kd1	+5	Kd5	0.00000
2	0.00200	6	0.06200
3	0.01000	7	0.12740
4	1.37777 (or 1.40077)		

(d) AXES - GRID DISPLAY

```
Program:
(enter, if
frame to be
changed)    d0   op   0       change frame.
if not,      1   rf   c37     set up to return to main program.
enter)       2   cs   Kd1     N         } set up counter.
(d25) -->    3   st   Kc12    counter   }
             4   ca   Kd4     x_o                      }
             5   st   Kc10    x                        }
             6   ca   Kd0     1.00077                  } display
             7   ds   Kc10    x                        } y-axis
           d10   ad   Kd2     Δ_A ; producing y+Δ_A.   }
            11   to   d13                              }
            12   tr   d7                               }
------------------------------------------------
            13   ca   Kd0     1.00077                  }
            14   st   Kc10    x->0.77700               }
            15   ca   Kd5     y_o                      } display
            16   ds   Kc10    x->0.77700               } x-axis
            17   ca   Kc10    x->0.77700               }
           d20   ad   Kd2     Δ_A                      }
            21   to   d23                              }
            22   tr   d14                              }
------------------------------------------------
            23   ca   Kc0     +1        } add +1 to counter.
            24   sm   Kc12    counter   }
            25   tn   d3      if not displayed N time go back to display
                              again.
________________________________________________
            26   ca   Kd4     x_o                   } find coordinate of
(d34)-->    27   st   Kc10    x->x-position of      } first vertical grid line.
                              first grid line.      } position has been
           d30   su   Kd6     Δx/2                  } found when overflow
            31   to   d35                           } occurs.
            32   su   Kd6     Δx/2                  }
            33   to   d35                           }
            34   tr   d27                           }
------------------------------------------------
(d50)-->    35   ca   Kd0     1.00077               }
                              x = coordinate        } display vertical
            36   ds   Kc10    of grid line          } grid line.
            37   ad   Kd3     Δ_g; producing        }
                                y+Δ_g               }
           d40   to   d42                           }
            41   tr   d36                           }
------------------------------------------------
```

(d) AXES - GRID DISPLAY

Program:

(d40)→	42	ca	Kc10	x	find coordinate of next vertical grid line.
	43	ad	Kd6	$\Delta x/2$	
	44	to	d51		all vertical lines have been displayed when overflow occurs.
	45	ad	Kd6	$\Delta x/2$	
	46	to	d51		
	47	st	Kc10	$x+\Delta x$	
	d50	tr	d35	if vertical grid incomplete, return to display next line.	
	51	ca	Kd5	y_o	find coordinate of first horizontal grid line.
	52	st	Kc11	y → y-position of first grid line	
	53	su	Kd7	$\Delta y/2$	position has been found when overflow occurs.
	54	to	d60		
	55	su	Kd7	$\Delta y/2$	
	56	to	d60		
	57	tr	d52		
(d54,d56,d76)→	d60	ca	Kd0	1.00077	display horizontal grid line.
	61	st	Kc10	x → 0.77700	
	62	ca	Kc11	y = coordinate of grid line.	
	63	ds	Kc10	x → 0.77700	
	64	ca	Kc10	x → 0.77700	
	65	ad	Kd3	Δg	
	66	to	d70		
	67	tr	d61		
	d70	ca	Kc11	y	find coordinate of next horizontal grid line.
	71	ad	Kd7	$\Delta y/2$	
	72	to	c36		
	73	ad	Kd7	$\Delta y/2$	all horizontal lines have been displayed when overflow occurs; at that time, transfer to leave.
	74	to	c36		
	75	st	Kc11	$y+\Delta y$	
	76	tr	d60	if horizontal grid incomplete, return to display next line.	

Kd0	1.00077		
1	--	N; no. of times to display axes.	
2	0.---00	Δ_A: distance between points in an axis.	
3	0.---00	Δ_g: distance between points in a grid line.	
4	--	x_o	coordinates of origin.
5	--	y_o	
6	0.----0	$\Delta x/2$	half spacings between grid lines.
7	0.----0	$\Delta y/2$	

(e) SPECIAL SQUARE ROOT

Input: Enter at e0 with the output of the SQUARE ROOT subroutine (SRL #7 of memorandum 6M-3497) in AC, having previously stored the scale factor (n) of x in Ke0. (See Notes.)

Output: $+\sqrt{x}\ .2^{-n}$ in AC, when x is stored as $x\ .2^{-n}$. (See Notes)

Other Subroutines Needed: SRL#7, assumed to be stored in field k.

Duration: 159 microseconds, including the time spent after re-entering SRL #7.

Accuracy: Maximum error in $\sqrt{x}$ is $0.00090 \cdot 2^{n/2}$ (decimal), including that possible in the original output of SRL #7 ---- this error requiring a scale factor of 2^{-n}, the same as the output.

Storage: 4 program registers, 1 storage, plus 14 (decimal) registers of SRL #7.

Notes:

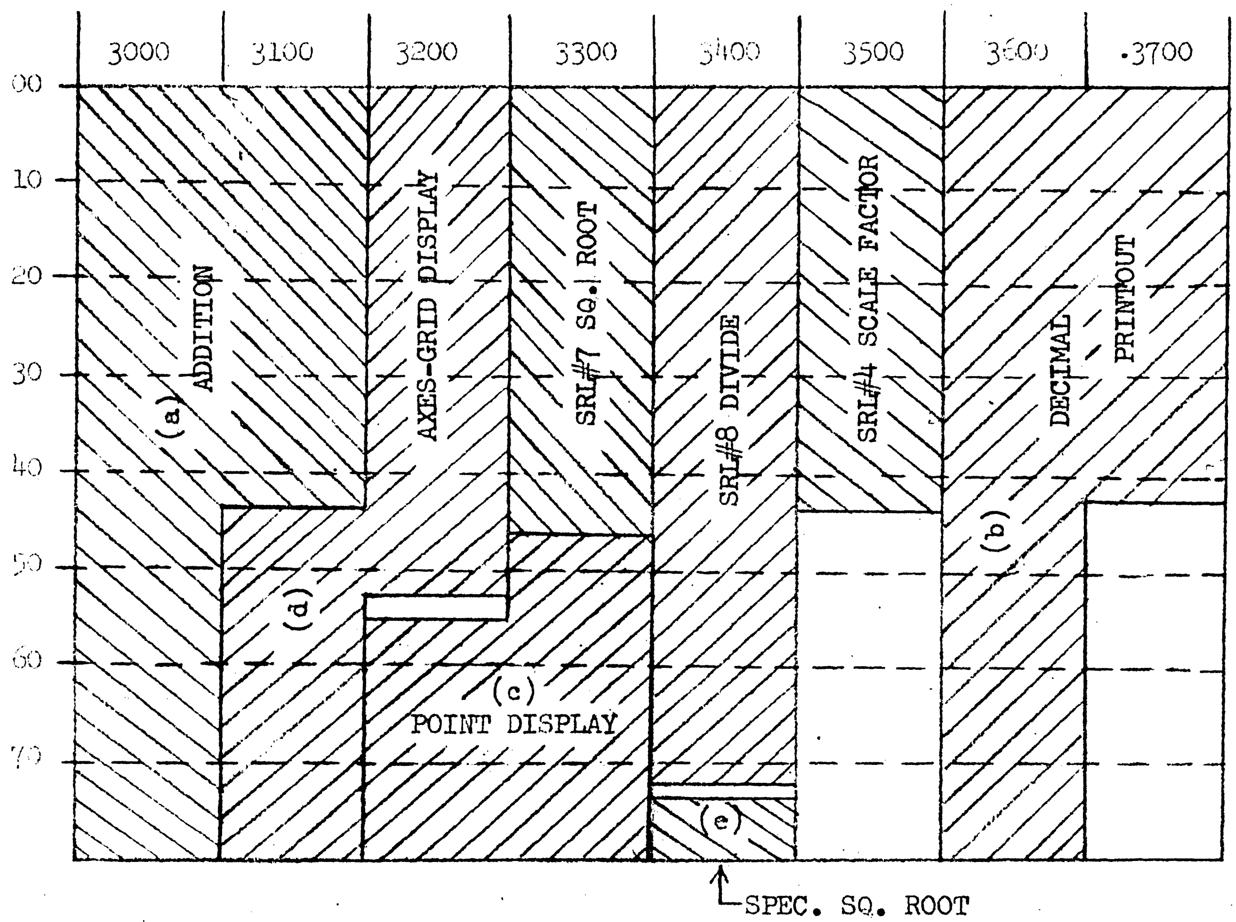
1. If $n \neq 0$, the result obtained from use of the SCALE FACTOR subroutine (SRL #7 of memorandum 6M-3497) possesses a different scale factor than that of x. The subroutine described here takes the results of SRL #7 and produces $\sqrt{x}$ with the original scale factor of x. For example: $\sqrt{+4}$ may be desired to appear as 0.00002; whereas SRL #7 gives 0.00552, which is $\sqrt{2^{-13}}$.
2. As used here, $n = +15$ indicates a scale factor of 2^{-15}, or, in other words, that the binary point is considered to be at the right-hand end of the register
3. Each relative address denoted with a g in the following program refers to the decimal address as given in SRL #7 of memorandum 6M-3497.

Program:

e0	rf	g28	prepare to return to main program.
1	st	Kg1	$\sqrt{x}$, as obtained from SRL #7.
2	ca	Ke0	n
3	tr	g17	transfer to last part of SRL #7.
Ke0	--		n: original scale factor of x.

J. Uskavitch

J. USKAVITCH



LOCATIONS OF TAPES AVAILABLE IN FIELD 2

DISTRIBUTION LIST   ATTACHMENT

Anderson, H.E.	B-209
Bagley, P.R.	C-147
Bailey, D.L.	C147
Benington, H.D.	C-170B
Bragar, P.	C-163
Burrows, J.H.	D-113
Buzzard, R.D.	B-121
Carmichael, R.L.	D-209
Chandler, A.R.	C-165
Clement, G.F.	D-209
Corderman, C.L.	B-129C
Dumanian, J.	B-32
Durgin, F.R.	B-155
Farley, B.G.	B-149
Feldstein, M.D.	C-170E
Festa, C.M.	B-271
Fleming, A.M.	B-275
Friedman, C.	C-169
Gates, E.	B-151
Gaudette, C.H.	C-170D
Gerhardt, R.H.	B-129A
Grennell, A.J.	FW
Harris, G.B.	B-267
Harris, W.F.	C-170C
Heart, F.E.	C-153
Hirshberg, L.H.	B-164
Holland, R.	B-271
Holst, W.F.	MIT 10-358
Hosier, W.A.	B-155
Houser, H.D.	C-169
Hughes, R.A.	B-191
Jensen, B.A.	B-320
Jones, N.T.	B-121
Lebow, I.L.	B-320
Luscher, R.W.	D-209
Marston, W.J.	D-209
Mathiasen, A.A.	C-171
Mayer, R.P.	B-107
Neumann, H.D.	C-169
Newitt, J.H.	B-151
Nolan, J.F.	C-184D
Olsen, S.C.	B-164
Parker, R.M.	Barta 110
Parrott, D.J.	B-164
Platt, H.J.	B-125
Reed, R.R.	C-155
Ross, D.T.	MIT Bldg. 32
Rundquist, H.I.	B-107
Salvato, J.	B-155
Schindler, B.W.	D-113
Sherrerd, C.S.	D-209
Smulowicz, B.	C-163
Stahl, B.	B-267
Storm, M.	B-155
Taylor, N.H.	B-209
Uskavitch, C.W.	B-275
Vanderburgh, A.	B-149
Wakstein, C.	
Woolf, J.	B-132