y,filecase

p5feb1972,

```
file                blocks

punch
 parity
  routines          3
textprint
 iam-dam            1
 nam                1
decimal
 input
  fancy             1
  simple            1
newfloat
 pack               12
 sqrt               1
 log                2
 exp                2
 cossin             2
 elliptic           3
 arcsincos          2
 atan               2
 cbrt               1
 ran                1
 phi                1
 a↑b                1
~~readptape          2~~
 bessel             6
 gauss              4
 a↑i                1
 gamma              2
 descripti          10
~~lineplot           3~~
 phinvr             2
 shellsort          2
 displ2mar          3
 ktmdesc            4
~~charplot           7~~
~~write              1~~
 ktm                 6
plotter
 minimal            1
 text.plot          11
 line
  shorter           2
  basic             3
 dpy                3
ascii
 flexo
  program
   b                3
   a                11
  subroutin         4
prime
 tester             1
pen
 follow
  old               1
  new               2
random
 number
  tester
```

obsolete→ plotter

only because plotter is now on function bus.

```
     ms         2
   correlati    2
  simple        1
  more
   random       1
  gaussian      3
single
 precision
  sqrt          1
  sqrt.sum
   sqrs         1
  sinVcos       1
  sincos        1
  lognat        1
double
 precision
  sqrt          1
  decimal
   print
    fraction
     old        4
    integer     1
print
 octal          1
 decimal
  signed        1
scope
 decimal
  print         2
 line
  fast          1
  window
   fixed        2
   variable     3
 character
  display
   1            5
   faster       6
7-bit           2
et
 typewrite
  simulator     3
 read/writ      7

505 free tape blocks
0 free directory entries
```

note: scope
decimal
print
(twos) is on public tape 4
(twos mode version)

Numeric typeout routines

5 February 1972

Numeric typeout routines

```
/octal print subroutine
/leading zero suppressing
/call by jdp opt with number in AC

opt,        0
            lio opa+4
            rcl 3s          LRC 3
            sni
            jmp .-2
opa,        A→IAP           A → IA =
            sni
            law ~~[illegible]~~   "0
            ivk 100
opa+4,      law 0
            rcl 3s          LRC 3
            A→IAP           A → IA =
            jmp opa
            jmp i opt
```

pencilled changes for twov mode

Numeric typeout routines

ONES or TWOS mode

```
/ decimal print subroutine
/ called by jdp dpt with number in AC

dpt,        0
            ~~TA IA<M~~
            ~~jmp .+4~~
            ~~law charac r-~~
            ~~ivk 100~~
            ~~CIA~~
            dac dp1
            dzm dp2
dpp,        dac dp3
            mul (1
            div .+1
            12
            sas dp2
            jmp dpp
            TIAP|
            law ~~charac r0~~ "0
            ivk 100
            lac dp3
            dac dp2
            lac dp1
            sas dp2
            jmp dpp
            jmp i dpt

variables
constant
```

lio ("-
NAA7P
NAA1
ivk 300

Decimal input routines

5 February 1972

Decimal input routines

```
/unsigned decimal input from typewriter
/call with <jdp gnu>
/returns with number in A, non-numeric break character in I, X.
/rph 4aug71

gnu,        0
            ZIXP
gnv,        mul (5   make this 10. for twos mode    ones mode
            law i 20
            XΛAX          ↖
            X+IX            lan 21
            TAI
            ivk 200
            X→AXP
            X+I<=    ←  make this  X+I<    for twos
            TXXI|
            jmp gnv
            jmp i gnu
```

Decimal input routines

```
/signed decimal input from typewriter
/call with <jdp decin>
/returns with number in A, non-numeric break character in I, X.
/restarts on centerdot, accepts minus sign before number
/rph 4aug71

decin,          0    _
decin+1,        dzm gny
               ZIAXP
gnt,            lac gnz
               A~II_
                dio gnz
gnv,            mul (5
                law i 20
               X∧AX
               X+IX
               TAI
                ivk 200
                sas (40
                jmp gnw
                law 77
                ivk 100
                jmp decin+1
gnw,            sas (54
                jmp gnx
                lio gny
               CIIM|
                jmp gnt
gnx,            dio gny
               X→AXP
               X+I<=
               TXXI|
                jmp gnv
               xor gnz
                jmp i decin
```

one mode

```
/knob hysteresis
/iam or dam
/e1m or e2m
/call with
/          ckn n00  ←
/          jsp hys  ←
→/kbn,     0
/expects old value of knob in kbn,
/ returns new value there, and in AC

hys,       TAX
           lac i 0      / get old value
           AMIAI<       / sub new
           NAA
           add (2       /increase for more hysteresis
           SAA<
           ZA|
           ~~[illegible]~~ TIZ
           NAA
           adm i 0 <lia   /this instruction can be deleted if ~~an~~ value is
           aam                desired only in AC.
           jmp 1
```

Text printing routines

5 February 1972

Text printing routines

```
/text printing subroutine
/call by jsp txx followed by text
/text should end with •
/control returns to location after end of text
/iam or dam
/runs in core 0 only

txx,       dap txy
           aam
           lio txy
           idx txy
           lac (607600
           rcl 6s
           sad (~~lai~~ 760040
txy,       jmp .
           sad (swp
           jmp txx+1
           ivk 100
           jmp txy-3
```

Text printing routines

```
/text printing subroutine
/call by jsp txx followed by text
/text should end with •
/control returns to location after end of text
/nam
/runs in core 0 only

txx,        dap txy
            lio i txy
            idx txy
            lac (607600
            rcl 6s
            sad (lai   760040
txy,        jmp .
            sad (swp   760060
            jmp txx+1
            ivk 100
            jmp txy-3
```

7-bit character routines

5 February 1972

7-bit character routines

```
/7-bit character  put  and  get
/5 seven-bit characters packed in a word pair
/bit-0,word 0     spare
/chr 0 in word 0 (bits 1-7)
/chr 1 in word 0 (bits 8-14)
/chr 2 in word 0 (bits 15-17) and word 1 (bits 0-3)
/chr 3 in word 1 (bits 4-10)
/chr 4 in word 1 (bits 11-17)


/get
/character address in A, less than 400000 octal
/character returned in A
/23 cycles

get,        0
            clf 1
            jmp pg1

/put
/character address in A, less than 400000 octal
/character in t0
/uses t1,t2
/32 cycles

put,        0
            stf 1
pg1,        mul (146315
            dac t1              /word address
            AMII                /correction factor
            TAX                 /I[0-2 ] = 7,4,1,6,3
            rcl 3s
            sza i
            law 7
            lio i 1             /word 1 of pair
            lxr i 0             /word 0 of pair
            X→AX
            xct i sht           /shift into A[11-17]
            szf i 1
            jmp pg3             /get
pg2,        and (-177           /leave others
            ior t0              /put in the character
            CXX                 /read ust backwards
            xct i ust           /unshift it
            lxr t1
            dac i 0             /put pair back
            dio i 1
            jmp i put
pg3,        and (177            /mask it out
            jmp i get

            rar 8s              /chr 0, unshift
sht,        ral 7s              /chr 3, unshift
            rcl 4s              /chr 2, shift
            ral 3s              /chr 1, unshift
            nop                 /chr 4, shift and unshift
            rar 3s              /chr 1, shift
            rcr 4s              /chr 2, unshift
ust,        rar 7s              /chr 3, shift
```

7-bit character routines

```
          ral 8s          /chr 0, shift
```

Ascii to flexo conversion

5 February 1972

```
Ascii to flexo conversion

ascii-et
/ type n to not convert lower and upper case
/ type c to undo n
/ type r for ascii reader to flexo text
/ type p for flexo text to ascii punch

0/
          jmp 102

100/
          jmp 102
          nop
          law 77
          ivk 0
          jmp in

ftoa,     0
          sas (72
          sad (74
          jmp fcs
          ior cas
          TAX
          law 177
          and i tab
          sad (15
          jmp cr
cvc,      skp 600         /skip if converting to upper case
          jmp i ftoa
          sub (140
          spa
          add (40
          add (100
          jmp i ftoa
cr,       aam
          xct ftoa
          law 12          /generate line feed after carriage return
          jmp i ftoa
fcs,      sub (72         /case shift
          ral 5s
          dac cas
          idx ftoa
          jmp i ftoa
```

Ascii to flexo conversion

```
atof,     0
          sad (12
          jmp lf
          clf 6
          sad (15
          stf 6           /last character was carriage return
af3,      TAAX
          xct cvc         /skip if converting to lower case
          jmp af1
          sub (100
          TAA>
          jmp af1
          sub (32
          TA>
          law 40
          A+XX
af1,      lac i tab
          cli
          sar 9s
          dac ftemp
          and (100
          sad cas
          jmp af2
          dac cas
          rar 5s
          add (72
          aam
          xct atof
af2,      law 77
          and ftemp
          jmp i atof
lf,       szf i 6
          jmp af3
          clf 6
          idx atof
          jmp i atof
```

```
Ascii to flexo conversion

define z f,a
            f×1000 a
termin

tab,        z 76,40
            z 76,61
            z 76,62
            z 14,63
            z 76,64
            z 76,65
            z 76,66
            z 76,67
            z 75,70
            z 36,71
            z 77,0
            z 76,14
            z 13,3
            z 77,0
            z 34,0
            z 35,0
            z 76,60
            z 76,57
            z 76,163
            z 76,164
            z 76,165
            z 76,166
            z 76,167
            z 76,170
            z 76,171
            z 76,172
            z 76,0
            z 76,54
            z 76,17
            z 76,16
            z 76,11
            z 76,0
            z 0,72
            z 105,152
            z 101,153
            z 103,154
            z 103,155
            z 104,156
            z 106,157
            z 102,160
            z 57,161
            z 55,162
            z 173,0
            z 154,0
            z 33,55
            z 54,51
            z 73,73
            z 21,50
            z 20,0
            z 1,141
            z 2,142
            z 3,143
            z 4,144
            z 5,145
            z 6,146
            z 7,147
```

Ascii to flexo conversion

```
z 10,150
z 11,151
z 40,0
z 56,56
z 107,0
z 133,10
z 110,0
z 121,15
z 120,40        /100
z 161,42
z 162,47
z 163,44
z 164,45
z 165,41
z 166,46
z 167,74
z 170,76
z 171,136
z 141,0
z 142,0
z 143,0
z 144,0
z 145,0
z 146,0
z 147,100
z 150,77
z 151,123
z 122,124
z 123,125
z 124,126
z 125,127
z 126,130
z 127,131
z 130,132
z 131,0
z 157,75
z 156,17
z 155,16
z 111,11
z 140,0
z 102,137
z 61,112
z 62,113
z 63,114
z 64,115
z 65,116
z 66,117
z 67,120
z 70,121
z 71,122
z 41,0
z 42,0
z 43,53
z 44,135
z 45,134
z 46,133
z 47,0
z 50,101
z 51,102
z 22,103
```

Ascii to flexo conversion

```
z 23,104
z 24,105
z 25,106
z 26,107
z 27,110
z 30,111
z 31,0
z 157,52
z 156,0
z 155,10
z 103,0
z 76,15
```

Ascii to flexo conversion

```
/call with jdp rset, AC contains field number of text

/ac,io,flags,address mode saved.  Enter in iam,nam,dam
/rbl must be a submultiple of 400 and a multiple of 40
rbl=400
dimension rbf(rbl)

rset,      0
           dap rsr
           dac ac
           dio io
           rar 6s
           add (400-rbl
           dac ddp
           dzm rpc
           lac (isp rpc
           dac rfg         /flag for end of file
           law rbf+rbl-1   /last word of buffer
           dap inp
           law rbf
           mta
           lio (340
           law 40
rsr,       ivk .
           hlt
           lac ac
           rar 6s
           add rbf 32
           sub (20000
           dac dne
           lio io
           lac ac
           jmp i rset
```

Ascii to flexo conversion

```
readch,     0
rfg,        isp rpc          /becomes jmp rp2 after end of file
            jmp inp
            law i 3
            dac rpc
            idx inp
            dap inq
            sas (lio rbf+rbl
            jmp 9k
            law rbl
            adm ddp
            lia
            ral 6s
            dap rdr
            law rbf
            dap inp
            dap inq
            mta
            law rbl
rdr,        ivk
            hlt
            ZAP
9k,         sub (lio rbf
            add ddp
            sad dne
            jmp rp3
inp,        lio
            cla
            rcl 6s
inq,        dio
            sas (77
            sad (13
            dzm rpc
rp1,        lia
            jmp i readch

rp3,        lac (jmp rp2
            dac rfg
rp2,        law 14
            jmp rp1
```

Ascii to flexo conversion

```
/et buffer write

/ofl must be a submultiple of 400 and a multiple of 40
/call with jdp wset, ac contains field number of text to be written

ofl=400
dimension ofb(ofl)

wset,       0
            dac ac
            dap wrdr
            mta 300
            nop
            sni
            bpt
            rar 6s
            dac wfld
            add (400-ofl
            dac odp
            law i 3
            dac cp
            law ofb
            dap ofp
            dap ofq
            lac ac
            dzm clst          /last character
            jmp i wset
```

Ascii to flexo conversion

```
/call with jdp writec, io contains character in ivk format

w100,       0                   /write from ac
            dac ac
            dio 95p
            jdp writec
            lac ac
            lio 95p
            jmp i w100

w300,       0
            dac 95p
            dio ac
            lai
            jdp writec
            lac 95p
            lio ac
            jmp i w300

writec,     0
            and (77
            dac 96p
            sas (13
            jmp . 4
            law 77              /make sure this is preceded gy 77
            sas clst
            jmp force
            lac (lio ofb+ofl
            sas ofp
            jmp ofp-1
            law ofl             /write out buffer
            adm odp
            lia
            and (770000
            sas odp
            jmp wo5
            ral 6s
            mta 300
            nop
            sni
            bpt
            jmp . +2
wo5,        ral 6s
            dap wrd
            law ofb
            dap ofp
            dap ofq
            mta
            law ofl 20
wrd,        ivk
            hlt
ofp-1,      lac 96p
ofp,        lio
            rar 6s
            rcl 6s
ofq,        dio
            isp cp
            jmp ou3
            law i 3
```

Ascii to flexo conversion

```
          dac cp
          idx ofp
          dap ofq
wox,      lac 96p
          dac clst        /character just written
          jmp i writec

ou3,      lac 96p
          sas (77
          sad (13
          jmp ofp
          jmp wox

force,    lac writec
          dac acw
          law 77
          jdp writec
          lac acw
          dac writec
          lac ac
          jmp writec 1
```

Ascii to flexo conversion

```
/call with jdp clean, finishes up output, writes out bufer

clean,     0
           dac ac1
           dio io
           law 13
           sas clst          /force 13 to precede end of text
           jdp w100
           lac ofp
           add odp
           sub wfld
           add (20000+ofl-[lio ofb]
           dac b77
           law ofb+ofl
           dap ofp
           jdp writec        /write out last buffer
           law b77-32
           mta
           lio (340
           law 60
wrdr,      ivk .
           hlt
           lac ac1
           lio io
           jmp i clean

b77,       0
           20400
```

Ascii to flexo conversion

```
in,         iam
            clf 7
            ivk 200
            sad (char c
            jmp con
            sad (char n
            jmp con-1
            sad (char rr
            jmp red
            sad (char p
            jmp pun
            jsp txx
            text ..35.?.347740..
            jmp in

red,        lac (30017
            mta 306
            jmp bsy
            law 2
            jdp wset
il,         ivk 17
            jmp redun
            stf 1
            sza
            sad (377
            jmp il
            and (177
            jdp atof
            jdp w100
            jmp il
redun,      szf i 1
            jmp roff
            jdp clean
end,        law 17
            mta 204
            dsm
bsy,        jsp txx
            text ..3577.busy.347740..
            jmp in
roff,       jsp txx
            text ..77.turn on reader and try again.7740..
            law 17
            mta 204
            jmp in

con-1,      ZAP
con,        law 600
            dap cvc
            jmp in

pun,        lac (40017
            mta 306
            jmp bsy
            law 2
            jdp rset
            law i 20
            jdp fee
ol,         jdp readch
            sad (14
            jmp oe
```

Ascii to flexo conversion

```
              jdp ftoa
              jdp out
              jmp ol
oe,           law i 60
              jdp fee
              jmp end
fee,          0
              TAI
              ivk 17
              SIIP
              jmp .-2
              jmp i fee
out,          0
              TAI
              law 2325
              rcr 7s
              ral 7s
              dap .+1
              ril
              rar 7s
              rcl 7s
              law 200
              A~IA
              ivk 17
              jmp i out

txx,          dap txy
              aam
              lio txy
              idx txy
              lac (607600
              rcl 6s
              sad (lai
txy,          jmp .
              sad (swp
              jmp txx+1
              ivk 100
              jmp txy-3

constants
variables
start in
```

Ascii to flexo conversion

```
/ascii to flexo / flexo to ascii

/runs in iam or dam
/calling sequence for flexo to ascii
/(ascii to flexo is similar)

/flexo character in A
/          jdp ftoa
/          jdp put
/return here

/"put" is a user supplied routine which accepts the
/          ascii characters produced by ftoa in A.
/          It may be called 0, 1, or 2 times
/          during a single call to ftoa

/these routines do not save any registers
/"cas" will contain the flexo case (0 or 100)
/"cvc" should be a "skp 0" if conversion between upper
/          case ascii and lower case flexo is not desired.
/          It should be a "skp 600" if this conversion
/          is desired. (this is the default condition)

/atof uses flag 6 (it should be initially cleared)
```

Ascii to flexo conversion

```
ftoa,      0
           sas (72
           sad (74
           jmp fcs
           ior cas
           TAX
           law 177
           and i tab
           sad (15
           jmp cr
cvc,       skp 600        /skip if converting to upper case
           jmp i ftoa
           sub (140
           spa
           add (40
           add (100
           jmp i ftoa
cr,        aam
           xct ftoa
           law 12         /generate line feed after carriage return
           jmp i ftoa
fcs,       sub (72        /case shift
           ral 5s
           dac cas
           idx ftoa
           jmp i ftoa
```

Ascii to flexo conversion

```
atof,      0
           sad (12
           jmp lf
           clf 6
           sad (15
           stf 6            /last character was carriage return
af3,       TAAX
           xct cvc          /skip if converting to lower case
           jmp af1
           sub (100
           TAA>
           jmp af1
           sub (32
           TA>
           law 40
           A+XX
af1,       lac i tab
           cli
           sar 9s
           dac ftemp
           and (100
           sad cas
           jmp af2
           dac cas
           rar 5s
           add (72
           aam
           xct atof
af2,       law 77
           and ftemp
           jmp i atof
lf,        szf i 6
           jmp af3
           clf 6
           idx atof
           jmp i atof
```

```
Ascii to flexo conversion

define z f,a
           fx1000 a
termin

tab,       z 76,40
           z 76,61
           z 76,62
           z 14,63
           z 76,64
           z 76,65
           z 76,66
           z 76,67
           z 75,70
           z 36,71
           z 77,0
           z 76,14
           z 13,3
           z 77,0
           z 34,0
           z 35,0
           z 76,60
           z 76,57
           z 76,163
           z 76,164
           z 76,165
           z 76,166
           z 76,167
           z 76,170
           z 76,171
           z 76,172
           z 76,0
           z 76,54
           z 76,17
           z 76,16
           z 76,11
           z 76,0
           z 0,72
           z 105,152
           z 101,153
           z 103,154
           z 103,155
           z 104,156
           z 106,157
           z 102,160
           z 57,161
           z 55,162
           z 173,0
           z 154,0
           z 33,55
           z 54,51
           z 73,73
           z 21,50
           z 20,0
           z 1,141
           z 2,142
           z 3,143
           z 4,144
           z 5,145
           z 6,146
           z 7,147
```

Ascii to flexo conversion

```
        z 10,150
        z 11,151
        z 40,0
        z 56,56
        z 107,0
        z 133,10
        z 110,0
        z 121,15
        z 120,40          /100
        z 161,42
        z 162,47
        z 163,44
        z 164,45
        z 165,41
        z 166,46
        z 167,74
        z 170,76
        z 171,136
        z 141,0
        z 142,0
        z 143,0
        z 144,0
        z 145,0
        z 146,0
        z 147,100
        z 150,77
        z 151,123
        z 122,124
        z 123,125
        z 124,126
        z 125,127
        z 126,130
        z 127,131
        z 130,132
        z 131,0
        z 157,75
        z 156,17
        z 155,16
        z 111,11
        z 140,0
        z 102,137
        z 61,112
        z 62,113
        z 63,114
        z 64,115
        z 65,116
        z 66,117
        z 67,120
        z 70,121
        z 71,122
        z 41,0
        z 42,0
        z 43,53
        z 44,135
        z 45,134
        z 46,133
        z 47,0
        z 50,101
        z 51,102
        z 22,103
```

Ascii to flexo conversion

```
        z 23,104
        z 24,105
        z 25,106
        z 26,107
        z 27,110
        z 30,111
        z 31,0
        z 157,52
        z 156,0
        z 155,10
        z 103,0
        z 76,15
```

ET reading and writing routines

5 February 1972

ET reading and writing routines

typewriter to e.t. text

```
dimension ac(3)             io=ac+1
setup,      0
            lio (ent
            law 0
            mta 307
            bpt
            dap enn
            ral 6s
            lia
            mta 405
            bpt
            dap eno
            mta 401
            law 7
            jdp wset
            jmp i setup

finish,     0
eno,        law
            mta 401
            mta 204
enn,        law
            mta 204
            jdp clean
            jmp i finish

ent,        dap epc
            law 1700
            A∧IA
            rar 6s
            dap var
            lio (ac
            law 11
epc,        ivk
            law 1600
            and ac
            lia
var,        law
            TAXP|
            SIX
            iam
            xct i vars
            jdp writec
            law 71
            xct epc
            qit

vars=.-1
            lac ac          /100 - out from ac
            hlt
            lac io          /300 - out from io
            repeat 14,hlt
```

ET reading and writing routines

```
/et buffer write

/ofl must be a submultiple of 400 and a multiple of 40
/call with jdp wset, ac contains field number of text to be written

ofl=400
dimension ofb(ofl)

wset,       0
            dac ac
            dap wrdr
            mta 300
            bpt
            rar 6s
            dac wfld‾
            add (400-ofl
            dac odp
            law i 3
            dac cp
            law ofb
            dap ofp
            dap ofq
            lac ac
            dzm clst‾          /last character
            jmp i wset
```

ET reading and writing routines

```
writec,     0
            and (77
            dac 96p
            sas (13
            jmp . 4
            law 77          /make sure this is preceded by 77
            sas clst
            jmp force
            lac (lio ofb+ofl
            sas ofp
            jmp ofp-1
            law ofl         /write out buffer
            adm odp
            lia
            and (770000
            sas odp
            jmp wo5
            ral 6s
            mta 300
            bpt
            jmp . +2
wo5,        ral 6s
            dap wrd
            law ofb
            dap ofp
            dap ofq
            mta
            law ofl 20
wrd,        ivk
            hlt
ofp-1,      lac 96p
ofp,        lio
            rar 6s
            rcl 6s
ofq,        dio
            isp cp
            jmp ou3
            law i 3
            dac cp
            idx ofp
            dap ofq
wox,        lac 96p
            dac clst        /character just written
            jmp i writec

ou3,        lac 96p
            sas (77
            sad (13
            jmp ofp
            jmp wox

force,      lac writec
            dac acw
            law 77
            jdp writec
            lac acw
            dac writec
            lac ac
            jmp writec 1
```

ET reading and writing routines

```
/call with jdp clean, finishes up output, writes out bufer

clean,     0
           dac ac1
           dio io
           law 13
           sas clst         /force 13 to precede end of text
           jdp writec
           lac ofp
           add odp
           sub wfld
           add (20000+ofl-[lio ofb]
           dac b77
           law ofb+ofl
           dap ofp
           jdp writec       /write out last buffer
           law b77-32
           mta
           lio (340
           law 60
wrdr,      ivk .
           hlt
           lac ac1
           lio io
           jmp i clean

b77,       0
           20400
```

ET reading and writing routines

constants
variables
start

ET reading and writing routines

```
/e.t. text reader/writer

/this is a set of subroutines to allow one to read and write e.t. type
/text fields on drum using subroutine calls that work in the same way
/as typewriter ivk instructions, i.e., they take and return characters
/in the rightmost byte of the ac or io registers.  These routines
/preserve the contents of the io,ac,xr, and flag registers.

/for writing onto the drum:

/      jdp wset     -- this initializes everything for writing on the drum
/                      the ac must contain the field number on which text
/                      is to start.  This may be any field.
/      jdp w100     -- this takes one character from the ac and
/                      writes it on the drum
/      jdp w300     -- this takes one character from the io and writes
/                      it on the drum
/      jdp clean    -- this terminates the writing of text onto the drum,
/                      clears buffers, etc.
/      jdp aset     -- this initializes things for appending to an
/                      existing text file.  The ac contains the field
/                      number where the text starts.  Write using
/                      jdp writec and terminate with jdp clean.


/for reading from the drum:

/       jdp rset      -- this initializes things for reading from the drum.
/                        The field number to be read from is in the ac.
/       jdp r200      -- read one character from the drum into the ac
/       jdp r400      -- read one character from the drum into the io.
/                        A 14 indicates the end of text[redacted].

/All routines may be used in nam, iam, or dam, and they
/save the address mode.  See changes for 2-mode.

/for any reading or writing, a 13 is the end of page character.


/see D. Thiel for additional information, problems, bugs, etc.
                    17/10
/  D. Thiel      ~~10/17~~/70
/modified C. Landau 22/10/70
/modified by D. Thiel 1/4/71
/modified by D. Thiel 18/6/71
```

ET reading and writing routines

```
/call with jdp rset, AC contains field number of text

/rbl must be a submultiple of 400 and a multiple of 40
rbl=400
dimension rbf(rbl)
```

Penciled changes for 2's mode

```
rset,       0
            dap rsr
            dac ac
            dio io
            rar 6s
            add (400-rbl
            dac ddp
            dzm rpc
            lac (isp rpc
            dac rfg             /flag for end of file
            law rbf+rbl-1       /last word of buffer
            dap inp
            law rbf
            mta
            lio (340
            law 40
rsr,        ivk .
            hlt
            lac ac
            rar 6s
            add rbf 32
            sub (20000
            dac dne
            lio io
            lac ac
            jmp i rset
```

ET reading and writing routines

```
/call with jdp readch, returns character in io in ivk format

r200,       0                   /read into ac
            dac ac
            dio io
            lia
            jdp readch
            lio io
            jmp i r200

r400,       0                   /read into io
            dac ac
            dio io
            jdp readch
            lac ac
            jmp i r400


readch,     0
rfg,        isp rpc             /becomes jmp rp2 after end of file
            jmp inp
            law i 3
            dac rpc
            idx inp
            dap inq
            sas (lio rbf+rbl
            jmp 9k
            law rbl
            adm ddp
            lia
            ral 6s
            dap rdr
            law rbf
            dap inp
            dap inq
            mta
            law rbl
rdr,        ivk
            hlt
            ZAP
9k,         sub (lio rbf
            add ddp
            sad dne
            jmp rp3
inp,        lio
            cla
            rcl 6s
inq,        dio
            sas (77
            sad (13
            dzm rpc
rp1,        lia
            jmp i readch

rp3,        lac (jmp rp2
            dac rfg
rp2,        law 14
            jmp rp1
```

ET reading and writing routines

```
/et buffer write

/ofl must be a submultiple of 400 and a multiple of 40
/call with jdp wset, ac contains field number of text to be written

ofl=400
dimension ofb(ofl)

wset,       0
            dac ac
            dap wrdr
            mta 300
            bpt
            rar 6s
            ~~dac wfld~~
            add (400-ofl
            dac odp
            law i 3
            dac cp
            law ofb
            dap ofp
            dap ofq
law77       lac ac
   dac      ~~dzm~~ clst         /last character
            jmp i wset
```

Penciled changes are for 2's mode

ET reading and writing routines

```
w100,      0                 /write from ac
           dac ac
           dio 95p
           jdp writec
           lac ac
           lio 95p
           jmp i w100

w300,      0
           dac 95p
           dio ac
           lai
           jdp writec
           lac 95p
           lio ac
           jmp i w300

writec,    0
           and (77
           dac 96p
           sas (13
           jmp . 4
           law 77            /make sure this is preceded by 77
           sas clst
           jmp force
           lac (lio ofb+ofl
           sas ofp
           jmp ofp-1
           law ofl           /write out buffer
           adm odp
           lia
           and (770000
           sas odp
           jmp wo5
           ral 6s
           mta 300
           bpt
           jmp . +2
wo5,       ral 6s
           dap wrd
           law ofb
           dap ofp
           dap ofq
           mta
           law ofl 20
wrd,       ivk
           hlt
ofp-1,     lac 96p
ofp,       lio
           rar 6s
           rcl 6s
ofq,       dio
           isp cp
           jmp ou3
           law i 3
           dac cp
           idx ofp
           dap ofq
wax,       lac 96p
```

```
           dac clst        /character just written
           jmp i writec

ou3,       lac 96p
           sas (77
           sad (13
           jmp ofp
           jmp wcx

force,     lac writec
           dac acw
           law 77
           jdp writec
           lac acw
           dac writec
           ~~lac ac~~ law 13
           jmp writec 1
```

ET reading and writing routines

f

```
/call with jdp clean, finishes up output, writes out bufer

clean,     0
           dac ac1
           dio io
           law 13
           sas clst          /force 13 to precede end of text
           jdp ~~w100~~ [redacted] writec
           lac ofp
           add odp
           sub wfld
           add (20000+ofl-[lio ofb]
           dac b77
           law ofb+ofl
           dap ofp
           jdp writec        /write out last buffer
           law b77-32
           mta
           lio (340
           law 60
wrdr,      ivk .
           hlt
           lac ac1
           lio io
           jmp i clean

b77,       0
           20400
```

ET reading and writing routines

```
/call with jdp aset, ac contains field number of text to be appended to

aset,       0
            dac ac
            dio io
            dap ard
            dap wrdr
            rar 6s
            dac wfld
            law ofb
            mta
            lio (340
            law 40
ard,        ivk .
            hlt
            lac wfld
            sub (20000
            add ofb 32
            dac ac1
            and (~~-~~[of1-1] ~777777
            lia
            sub (of1
            dac odp
            lai
            ral 6s
            dap ard1
            law ofb
            mta
            add ac1
            AMIA
            dap ofp
            dap ofq
            law i 3
            dac cp
            law of1
ard1,       ivk
            hlt
            law 13
            dac clst          /end of page was last character
            lac ac
            lio io
            jmp i aset
```

ET reading and writing routines

constants
variables
start

Punch parity routines

5 February 1972

Punch parity routines

```
/parity generator
/uses all three registers
/high 12 bits of register must be clear
            TIAX            /or TAIX or TXIA
            sar 1s
            A~II
            sza
            jmp .-3
            SII
            sil 7s
            XVII            /or XVIIA etc.
/garbage left in high 10 bits
```

? TIXIA maybe 3

obsolete RRN
use twos mode

e.g.

```
RRN 7
cma
LRC 7    /result in AC
```

Punch parity routines

```
/parity generation subroutine for low order 6 io bits
/ac saved, only low 8 bits of io are significant rest is garbage
/entry  jda pty

pty,         0
             dap ytp                          /return address
             law 2325        /parity bits for io and io bit for ril inst
             rcr 277
             ral 7s                           /ac contains 0022xx, io 52gggg
             and .-2
             dap . 1                          /address patrt includes io bit
             ril .-.         /rotates io one plus parity of char
             rar 7s
             rcl 7s                           /rotate char with parity back
             lac pty                          /restore ac
ytp,         jmp .                            /return
start
```

Punch parity routines

```
/parity generation subroutine for low order 7 io bits
/ac saved, io will contain nothing but char with parity in low 8 bits
/entry  jda pty

pty,       0
           dap ytp                        /return address
           law 2325        /parity bits for io and io bit for ril inst
           rcr 7s
           ral 7s
           dap . 1
           ril .-.                        /rotates correct parity bit in
           rar 7s
           rcr 1                          /prepend parity to char
           cli                            /flush io garbage
           rcl 8s
           lac pty                        /restore ac
ytp,       jmp .                          /return
start
```

Punch parity routines

```
/super short parity generation subroutine for low 7 io bits
/ac not saved, io bits0-9 will be garbage
/entry    jsp pty

pty,        dap ytp                                   /return address
            law 2325           /parity for io and io bit for ril inst
            rcr 7s
            ral 7s
            dap . 1
            ril .-.                                   /rotates correct parity bit in
            rar 7s
            rcl 7s                                    /rotate char with parity back
ytp,        jmp .                                     /return
start
```

Light pen following routines

5 February 1972

Light pen following routines

```
/pen follow - rst - aug 66
/ts- only
/entry: jdp tpn, returns x and y in ac and io resp

tpn,      0
          lac x
          lio y
          stf 3
          jmp tp1
tpl,      add in1
          swp 3
          add in2
          swp
tp1,      dpy-i 300       /insert nop before here for non-ts
          szf 3
          jmp tpl
tpd,      sub x
          sar 1
          adm tx
          lai
          sub y
          sar 1
          adm ty
          lac in1
          lio in2
          cma 3
          dac in2
          dio in1
          spiVspa i
          jmp tpn 1
          lac tx
          lio ty
          dac x
          dio y
          dpy 300
          szf 3
          jmp i tpn
fpl,      dpy-i 300
          dac tx
          rar 1
          xor (110371
          add (110371
          lia
          rir 9s
          szf i 3
          jmp fpl
fpd,      lac tx
          lia
          rir 9s
          dac x
          dio y
          dio ty
          jmp tp1

in1,      -1000
in2,      -0

variables
constant
start
```

Light pen following routines

```
/pen follow
/jdp tpn
/returns here if can't find pen
/skips if pen found, coordinates in x,y and A,I

tpn,        0
            dzm tx
            dzm ty
            dzm tc2
            lac tsh
tp1,        and (777
            sal 1s
            cma
            dac tc1
tp2,        lac ran
            rar 7s
            xor (311071
            add (311071
            dac ran
            scr 9s
            sir 9s
tsh,        rcl 9s
            add y
            swp 3
            add x
            dpy 300
            szf i 3
            jmp tp3
            sar 4s              /pen seen
            adm tx
            lai
            sar 4s
            adm ty
            idx tc2
            law i 17
            ior tc1             /don't look at more than 17 more points
            dac tc1
tp3,        isp tc1
            jmp tp2
            lac ty
            mul (1
            div tc2             /no. of points under pen
            jmp tp4             /none
            sal 4s
            dac y
            idx tpn             /skip return
            law 3007
            dap tsh
            lac tx
            mul (1
            div tc2
            hlt
            sal 4s
            dac x
            lio y
            jmp i tpn

tp4,        lac tsh             /increase window size
            SAA
            and (777
          sza i
          jmp i tpn       /not found
          adm tsh
          jmp tp1         /try again

ran,      123456
tx,       0
ty,       0
x,        0
y,        0
tc1,      0
tc2,      0
```

Random number programs

5 February 1972

Random number programs

1's mode! (2's mode use of this routine tends to be nonrandom)

```
/simple random number generator
/good for most applications
ran,        0
            lac random
            rar 7s
            xor (311071
            add (311071     /use carry function
            dac random
            jmp i ran
```

Random number programs

```
/new improved random number generator

ran,      0
          lac r1
          xor (311071
          add (311071
          rar 9s
          dac r1
          lac r2
          xor (355671
          add (355671
          rar 7s
          dac r2
          xor r1
          jmp i ran        /return with number in ac
```

Random number programs

```
/gauss
/Routine to generate a random number with an approximate gaussian density.
/Entry is jdp ran.  The density has zero mean and unit variance, with
/the decimal point assumed between bits 3 and 4.

ran,      0
          law i 8         /Prepare to add 8 random numbers
          dac sct
          dzm ras
rn1,      law i 5         /Initialization to form 1 random 15 bit number.
          dac rct
          lac rda         /Generate a pseudo-random address for the first e
          rar 1s          /to the random number table.
          xor (311071
          add (311071
          dac rda
          and (377
          add (tbl
          dac ads
          lac rdm         /Generate a random increment between table entrie
          ral 3s
          lia
          and (177
          spi
          cma
          dac inc
          dzm rdm
1rn,      lac rdm         /Generate a 15 bit random number from 5-3 bit
          ral 3s          /table entries.
          add i ads
          dac rdm
          idx rct
          sma
          jmp rn2
          lac ads
          add inc
          and (377
          add (tbl
          dac ads
          jmp 1rn
rn2,      lio ras         /Complement the number at random.
          rir 2s
          lac rdm
          spi
          cma
          dac rdm
          adm ras         /Add to the sum of the previous random numbers.
          isp sct
          jmp rn1
          lac ras
          mul (122342     /unit variance
          jmp i ran
```

Random number programs

```
/Random Number Table

tbl,
5          3          4          7          1
1          1          5          0          3
6          1          2          2          1
7          5          5          2          6
4          0          2          3          4
0          5          7          7          3
3          5          1          4          3
2          1          1          6          2
5          5          7          3          4
4          7          0          3          2
5          1          1          6          1
6          4          2          6          2
4          0          4          4          6
4          3          3          2          7
7          3          1          2          2
3          7          7          5          7
6          6          0          2          3
3          2          7          7          7
4          5          2          3          7
1          1          1          4          2
1          3          5          6          2
3          1          5          6          5
7          3          5          6          4
0          7          7          6          7
5          7          2          3          0
3          1          7          2          4
3          1          1          2          3
0          6          1          2          1
4          5          3          6          2
1          6          2          2          4
2          0          5          5          1
3          0          1          7          6
4          7          2          1          5
3          4          1          6          4
2          4          3          6          2
6          6          3          6          3
0          3          4          4          7
2          2          6          2          1
0          6          1          1          7
1          7          4          3          3
3          1          4          3          7
1          3          2          1          3
3          0          1          7          7
4          7          6          7          3
6          6          3          5          4
1          3          0          6          3
0          7          1          6          3
5          0          5          3          5
5          5          3          2          6
0          0          4          1          0
5          5          3          4          2
4          4          7          6          3
7          0          3
```

Random number programs

```
/test random number generator by distribution of sum of successive
/random numbers.

go,         k=1000
            iam
            lxr (-k
            dzm i tab k 1
            771622
            jmp .-2

            law i 1
            add n
            lio (add
            sza i
            jmp . 4
            sil 1
            sar 1
            jmp .-4
            lai
            ior (sar
            dac sca
foo,        lac n
            cma
            dac cnt
            dzm tot
bar,        jdp ran
sca,        sar
            adm tot
            isp cnt
            jmp bar
            lac tot
            sar 9s
            and (777
            add (400
            and (777
            774020
            idx i tab
            szs i 10
            jmp foo
            lac (add
            dac x
            771020
dis,        lac i tab
            ral 9s              /determines vertical scale
            add (add
            lia
            lac x
            iot 307
            771620
            law 1000
            adm x
            sas (add 1
            jmp dis
            jmp foo

n,          1                   /number of terms in sum
tab,        0
tab k/

ran,        0
```

```
                           /insert random number generator here, return with
           jmp i ran

variab
consta
start go
```

Random number programs

```
/Test random number generator by correlation of a number with its
/n-th successor.  Put upper and lower limits in TW 0-8 and TW 9-17
/respectively with signs in TW 0 and 9.  Program plots distribution
/of those random numbers which were preceded on the n-th  call before
/by a random number in the set range.

go,         k=1000
            iam
            lxr (-k
            dzm i tab k 1
            771622
            jmp .-2
            latVcli
            rcr 9s
            ral 9s
            sar 1
            sir 1
            dac ulm
            dio llm
            771020
            jdp ran
            dac i buf
            771620
            lac n
            775402
            jmp .-5
            dzm bfp

foo,        jdp ran
            dac new
            lxr bfp
            lac i buf
            sar 1
            lio llm
            772613
            jmp bar
            lio ulm
            772607
            jmp bar
            lac new
            sar 9s
            and (777
            add (400
            and (777
            774020
            idx i tab
bar,        lac new
            lxr bfp
            dac i buf
            idx bfp
            lio n
            773415
            dzm bfp
            szs i 10
            jmp foo

            lac (add
            dac x
            771020
dis,        lac i tab
```

Random number programs

```
                ral 9s
                add (add
                lia
                lac x
                iot 307
                771620
                law 1000
                adm x
                sas (add 1
                jmp dis
                jmp foo
n,              1                  /correlation number

tab,            tab k/

ran,            0
                lac r1             /example of good random number generator
                xor (311071
                add (311071
                rar 9s
                dac r1
                lac r2
                xor (355671
                add (355671
                rar 7s
                dac r2
                xor r1
                jmp i ran

variab
consta
buf,
start go
```

Plotter routines

5 February 1972

Plotter routines

```
/move the plotter
/a and i have signed x and y increments
/jdp move  nam,iam,or dam  xr saved
/does all the 45 degree moves first, then the remaining 0 or 90 degree moves
/this is a minimum time path
/can be used to draw perfect 0, 45, or 90 deg lines
/others will have a bend
move,      0
           stf 5
           TAA≯
           cma 5
           dac xc
           TIIA<M
           cma
           dac yc
           lac xc
           TAAP|
           jmp .+4
           law 10
           szf 5
           law 4
           TII=
           SAA
           TI≤
           SAA
           cks
           ril 7s
           TI>P
           jmp .-3
           TAI
           iot 1111
           isp xc
           jmp .+3
           law i 14
           A∧II
           isp yc
           jmp .+3
           law i 3
           A∧IAP
           jmp .-14.
           jmp i move
xc,        0                /x count
yc,        0                /y count
```

OBSOLETE

and never very useful

Plotter routines

OBSOLETE

```
dpy plot
/ss2 up to plot,down to display
/jdp ini to initialize buffers
/jdp dsg to gronk dpy's
/jdp plt to plot a point not originally a dpy

6000/

ini,        0
            law tax
            dap ppx
            dap . 2
            dzm tae
            dzm
            idx .-1
            sas .-3
            jmp .-3
            law tay
            dap ppy
            dzm px
            dzm py
            jmp i ini

b=100
tax,        tax b/
tay,        tay b-1/
tae,        0

plt,        0
dsp,        iot 307
            skp i 20
            jmp i plt
            dac sac
            dio sio
siz,        sar 9s
            sir 9s
ppx,        dac
ppy,        dio
            lac (add-1
            dac ds
            law tax
            dap ppx
            law tay
            dap ppy
nex,        lac i ppx
            sub px
            dac dx
            spa
            cma
            lia
            lac i ppy
            sub py
            dac dy
            spa
            cma
            772610            /A-I,skp on >0
            swp
            sub ds
            sma
```

Plotter routines

```
          jmp plz
          adm ds
          lac dx
          dac fix
          lac dy
          dac fiy
          lac ppx
          dac spx
          lac ppy
          dac spy

plz,      idx ppx
          idx ppy
          sas (dio tae 1
          jmp nex
          lac spx
          dac ppx
          lac spy
          dac ppy
          lac i ppx
          dac px
          lac i ppy
          dac py
          law 4
          sub ds
          spa
          jdp lft
          jdp ln
          jdp drp
          dzm fiy
          dzm fix
          jdp ln
          lac sac
          lio sio
          jmp i plt

/dpy gronker

dsg,      0
          law
          dap . 1
di1,      lac
          and (760077
          sad (iot 7
          jmp di2
di3,      idx di1
          sas (lac i
          jmp di1
          jmp i dsg
di2,      lac di1
          sas (lac dsp
          sad (lac (760077
          jmp di3
          lac (jdp plt
          dac i di1
          jmp di3
```

← watch out! I wrote this before dpy had center control — RPH

Plotter routines

```
lft,       0
           lio (40
           cla
           dap plw
           jdp pl1
           jmp i lft

drp,       0
           law 600
           dap plw
           jmp i drp

pnd,       dio sv1
           lio (20
           iot 1111
           law i 1400
           771312          /A+1→A,skp on ≥0
           jmp .-1
           lio sv1
           cla
           dap plw
           jmp plw 2

pls,       0
           repeat 2,jdp pl1
           jmp i pls
```

Plotter routines

```
ln,         0
            lac fix
            ior fiy
            sza i
            jmp lm
            jdp lin
            jdp pls
            jdp txi
            jmp i ln

lin,        0
            lio fix
            law 1
            spi
            law 2
            dac lgd
            spi
            cmi
            dio fix
            lio fiy
            law 10
            spi
            law 4
            dac smd
            spi
            cmi
            lai
            dio fiy
            sub fix
            spa
            jmp li1
            lac fix
            dac fiy
            dio fix
            lac smd
            lio lgd
            dac lgd
            dio smd

li1,        lac fix
            cma
            dac npt
            cma
            sar 1
            dac nt
tis,        lac fiy
            adm nt
            sub fix
            cli 60
            spi
            jmp . 3
            dio nt
            lac smd
            ior lgd
            swp
            jmp i lin

txi,        0
            isp npt
```

Plotter routines

```
            jmp tis
            jmp i txi

lm,         lio (1
            jdp pl1
            lio (2
            jdp pl1
            jmp i ln
```

Plotter routines

```
pl1,       0
plw,       skp i
           jmp pnd
           iot 1111
           law i 240
           /771312         /A+1→A,skp on ≥0
           /jmp .-1
                           add (1
                           spa
                           jmp .-2
           jmp i pl1
constants
variables
start
```

Plotter routines

```
/lineplot, 11 sept 1969

/"getccm" will assign the plotter
/It also helps to turn on the Calcomp, etc.
/to start, line up your origin, then dzm px and dzm py

/the macros "drawto x,y" will move the pen from its
/present position to (x,y), where x and y are addresses of floating
/point numbers.  The x and y scales given here are for
/1.0 at seven inches.  You can change them during your
/run.  Watch out for going off the graph.

/"penup" and "pendwn" do the obvious thing
```

OBSOLETE

```
define     drawto x,y
           load y
           mulby scaley
           jdp fix
           0
           dac fb1
           load x
           mulby scalex
           jdp fix
           0
           lio fb1
           jdp pln
           terminate

scalex,    205657        0              /1400.0 scales 1.0 as seven inches
scaley,    205657        0              /other scales are possible

penup=jdp pup
pendwn=jsp pdn

define     getccm
           law flexo  q2
           arq
           bpt
           terminate
```

Plotter routines

```
/plot from (px,py) to (AC,IO), then update (px,py)
/everything saved
/runs in nam, iam, or dam
/coordinates are signed integer, step size is .005 inches

pln,        0                   /jdp
            dac tac
            dio tio
            sub px
            dac dx
            adm px
            lai
            sub py
            dac dy
            adm py
            lac dy
            lio dx
            sil 2s
            scl 2s
            law 12
            A~IA
            and (17
            dac dag
            lac dx
            lio dy
            spa
            cma
            spi
            cmi
            dac dx
            AMI<
            jmp .+4
            dio dx
            lio (-14
            A→IA|
            law i 3
            and dag
            dac str
            dio dy
            dzm sds
            dzm lds
plb,        lac lds
            sad dx
            jmp ret
            idx lds
            mul dy
            div dx
            jmp ret
            lio str
            sas sds
            lio dag
            dac sds
```

Plotter routines

```
                jdp plt
                jmp plb

ret,            lac tac
                lio tio
                jmp i pln
pup,            0                   /pen up jdp
                lio (40
pc2,            jdp plt
                jmp i pup

pdn,            dac pup             /pen up jsp
                lio (20
                jmp pc2

plt,            0
                lai
                cks
                ril 7s
                spi i
                jmp .-3
                lia
                iot 1111
                jmp i plt

tac,            0
tio,            0
dag,            0
str,            0
lds,            0
sds,            0
dx,             0
dy,             0

px,             0                   /where pen is now
py,             0
```

Plotter routines

OBSOLETE

```
/incremental line plotting routine
/call with jdp ln
/dx in A, dy in I

ln,         0
            A→IX            /CAX for Howell standard
            law 4
            X→IX>P
            CXX|
            law 10
            dac lna
            TII>P
            CII|
            SAA
            SAA
            dac lnb
            XMI<
            jmp lne
            X→IX
            xor lna
            dac lna
lne,        CXXA
            dac lnd
            dio lnc
            sar 1s
lnf,        SXX<=
            jmp i ln
            sub lnc
            sub lnd
            lio lnb
            spq
            jmp .+3
            add lnd
            lio lna
            dac lnfa
            lai
            jdp pl
            lac lnfa
            jmp lnf

pl,         0
            cks
            ril 7s
            jmp .-3
            lia
            iot 1111
            jmp i pl
```

needs to be fixed for function bus plotter

Plotter routines

OBSOLETE

See Fortran Library

```
/absolute line plotting routine
/call with jdp ln
/x in A, y in I
/absolute pen positions stored in px, py

ln,         0
            sub px
            swp
            sub py
            TAXA
            adm py
            TIIA
            adm px
            law 4
            X→IX>P
            CXX|
            law 10
            dac lna
            TII>P
            CII|
            SAA
            SAA
            dac lnb
            XMI<
            jmp lne
            X→IX
            xor lna
            dac lna
lne,        CXXA_
            dac lnd
            dio lnc
            sar 1s
lnf,        SXX<=
            jmp i ln
            sub lnc
            sub lnd
            lio lnb
            spq
            jmp .+3
            add lnd
            lio lna
            dac lnfa
            lai
            jdp pl
            lac lnfa
            jmp lnf

pl,         0
            cks
            ril 7s
            jmp .-3
            lia
            iot 1111

pl,         0
            cks
            ril 7s
            spi i
            jmp .-3
            lia
```

Plotter routines

```
            iot 1111
            jmp i pl
```

Plotter routines

```
/character plotting routines on eight pages
/txp
/text plotter
/define ori and siz, then bring pen to location
/jdp txp, followed by the text, ending with a •


txp,        .-.              /jdp
            rpf
            dio 47t
            nam

tpx,        lio i txp
            idx txp
            lac (607600
            rcl 6s
            sas (lai
            jmp . 6
            lio 47t
            lpf
            lac (nop
            dac ch1-1
            jmp i txp

            sad (swp
            jmp tpx
            dac 48t
            dio 48s
            jdp cpl
            lac (jmp ch2
            dac ch1-1        /so get right place for next letter
            lac 48t
            lio 48s
            jmp tpx 2
```

OBSOLETE

See FORTRAN library

Plotter routines

```
/character plotting subroutine

/before start, need to assign calcomp,  law flexo  q2, arq, hlt


cpl,        .-.             /use jdp cpl with the character in A (in concise
/code) and the orientation in 'ori' and the size code in 'siz'
/size code is -1 for eighth inch, -2 for quarter inch,
/up to -20 (-16.) for two inch (max)

            dac ch
            nop             /txp changes to jmp ch2
ch1,        rpf
            dio svf
/runs in nam, so if come in otherwise, need to restore
            nam
            lac px
            dac x0          /communicate with pln
            dac x2
            lac py
            dac y0
            dac y2

ch2,        law 77
            and ch
            add (dsp
            dac get
            lio i get


cas,        skp i 600
            ril 9s
            cla
            rcl 9s
            sub (6
            spa
            jmp g1
            add (tbl
            dac get
            lio i get
            cla
            rcl 4s
            add x2
            dac x1
            cla
            rcl 5s
            sad (37
            jmp spo         /special origin
            add y2
            dac y1
n0,         dzm t
            lac y1
            sub y0
            sza i
            jmp nx1
            and (400000
            sza
            law i 4
            add (10
            dac t
```

Plotter routines

```
nx1,        lac x1
            sub x0
            sza i
            jmp ny1
            and (400000
            ral 1s
            add (1
            adm t
ny1,        lac t
            sza i
            jmp go
            ior (40
            jda plt
            jmp n0
```

Plotter routines

```
spo,       idx get
           lac i get
           add x2
           dac x1
           idx get
           lac i get
           add y2
           dac y1
           idx get
           jmp n0

go,        lac (add
           adm get
           lio i get
           spa
           ril 9s
           cla
           rcl 9s
           sad (777
           jmp dun
           sad (776
           jmp i cpl
           dac t
           and (40
           rar 1s
           cma
           add (40
           dac plt              /pen up/down
           law 300
           and t
           sza i
           jmp nx2
           law 400
           and t
           rar 6s
           cma
           add (10
           adm plt
nx2,       law 200
           adm t
           and (300
           sza i
           jmp ny2
           law 400
           and t
           rar 8s
           add (1
           adm plt
ny2,       law 37
           and t
           cma
           dac t
           jsp plt+1
           isp t
           jmp .-2
           jmp go


plt,       0
```

Plotter routines

```
           dap plx
           lac siz          /size code, from main
           dac sz2
pl1,       lio plt
           law 1
           and ori          /orientation, from main prog
           sza
           jmp lr
lrx,       law 2
           and ori
           sza
           jmp ud
```

Plotter routines

```
udx,      iot 1111
/figure actual location for pln
          law i 1
          rir 1s              /test -y bit
          spi
          adm py
          law i 1
          rir 2s
          spi
          adm px
          ril 1s
          spi
          idx py
          rir 2s
          spi
          idx px

          lac plt
          xor pl2
          and (60
          sza
          law i 2000.
          sub (222.
          dac pl2
          isp pl2
          jmp .-1
          lio plt             /update position
          dio pl2
          isp sz2
          jmp pl1
          rir 1s
          spi
          idx x0
          rir 1s
          law i 1
          spi
          adm x0
          rir 1s
          law i 1
          spi
          adm y0
          rir 1s
          spi
          idx y0
plx,      jmp .
pl2,      0

lr,       rcr 2s
          ral 1s
          rcl 1s
          rar 2s
          rcr 3s
          ral 1s
          rcl 1s
          rar 2s
          rcl 1s
          ral 1s
          rcl 2s
          jmp lrx
```

Plotter routines

```
ud,        rcr 4s
           ral 2s
           rcl 2s
           rar 3s
           rcl 1s
           rar 2s
           rcl 1s
           jmp udx
```

Plotter routines

```
g1,         add (.+11
            dap .+1
            jmp .
            jmp dun         /space
            jmp tab         /tab
            jmp cr          /car. ret.
            jmp bks         /backspace
            ZAP             /upper case test
            law 600
            dap cas
            jmp dun 2

bks,        law i 22
            adm x2
            jmp dun 2

cr,         law i 40.
            adm y2
            dzm x2
            jmp dun 2

tab,        law 22.
            add x2
            mul (1
            div (220.
            hlt
            add (1
            mul (220.
            div (1
            hlt
            dac x2

dun,        law 22.
            adm x2          /advance to next character
            lio svf
            lpf
            jmp i cpl       /exit
```

Plotter routines

```
dsp,      0                    /space
          q1-tbl 6
          [qu2-tbl 6]x1000 q2-tbl 6
          q3-tbl 6
          q4-tbl 6
          q5-tbl 6
          q6-tbl 6
          q7-tbl 6
          q8-tbl 6
          q9-tbl 6
          0
          0
          0
          0
          0
          0
          q0-tbl 6
          [qqq-tbl 6]x1000 qsl-tbl 6
          qs-tbl 6
          qt-tbl 6
          qu-tbl 6
          qv-tbl 6
          qw-tbl 6
          qx-tbl 6
          qy-tbl 6
          qz-tbl 6
          0
          [qeq-tbl 6]x1000 qcm-tbl 6
          0
          0
          1001                 /tab
          0
          [qub-tbl 6]x1000 qcd-tbl 6
          qj-tbl 6
          qk-tbl 6
          ql-tbl 6
          qm-tbl 6
          qn-tbl 6
          qo-tbl 6
          qp-tbl 6
          qq-tbl 6
          qr-tbl 6
          0
          0
          [qpl-tbl 6]x1000 qmn-tbl 6
          [qrb-tbl 6]x1000 qrp-tbl 6
          [qvb-tbl 6]x1000 qob-tbl 6
          [qlb-tbl 6]x1000 qlp-tbl 6
          0
          qa-tbl 6
          qb-tbl 6
          qc-tbl 6
          qd-tbl 6
          qe-tbl 6
          qf-tbl 6
          qg-tbl 6
          qh-tbl 6
          qi-tbl 6
          5005                 /lower case
```

```
[qtm-tbl 6 ]x1000 qpe-tbl 6      /period,x
4004             /upper case
3003             /backspace
0
2002             /car. ret.
```

Plotter routines

```
tbl,

/numbers

q0,
103743     045143          262343          445543     662777
q1,
200046     403270          544777
q2,
740457     241157          244344          447544     642777
q3,
004744     047144          244344          144244     344447
544710     043777
q4,
540270     406642          541643          541642     543057
777000
q5,
004744     047144          244344          453254     057777
q6,
014053     744644          544447          344260     144047
744777
q7,
026242     057643          542643          542643     542643
542644     777000
q8,
214544     644744          047144          244344     144244
           344447          544644          744047     777000
q9,
004744     047144          260344          447544     644744
053777

/upper case numbers

qu2,       /single quote
351257     041657          777000
/punctuation

qsl,       /slash
000244     157245          777000
qqq,       /question mark
403441     541641          741041          141241     341203
246145
           244343          444543          644777
qpe,       /period
340041     141241          341441          541641     741777
qtm,       /times sign
106154     414754          777000
qcm,       /comma
440442     341241          141041          741642     544777
qeq,       /equals sign
111054     206454          777000
qcd,       /center dot
352041     141241          341441          541641     741776
qub,       /underbar
777000     -6              -4              000066     776000
qob,       /overbar
777000     -6              34              000066     776000
qvb,       /vertical bar
777000     7               -4              000277     776000
qlp,       /left paren
500346     254146          777000
```

```
grp,            /right paren
240146          254346          777000
qlb,            /left bracket
540447          270047          777000
```

```
qrb,        /right bracket
200047      270447          777000
qmn,        /minus sign
114054      777000
qpl,        /plus sign
114054      306654          777000
```

```
Plotter routines

/upper case letters

qa,
000263      145045          745663          317057      777000
qb,
000270      052744          644544          744644      544452
214052      777000
qc,
746642      544447          344260          144047      744642
777000
qd,
000270      053744          660544          453777
qe,
000270      057514          403054          514057      777000
qf,
000270      057514          403054          777000
qg,
510045      644544          447344          260144      047744
642777
qh,
000270      614057          214670          777000
qi,
200046      403270          403046          777000
qj,
006643      743044          143265          405052      777000
qk,
000270      616156          514754          777000
ql,
030670      057777
qm,
000270      747147          670777
qn,
000270      604757          605270          777000
qo,
004260      144047          744660          544447      344777
qp,
000270      053744          644544          453777
qq,
004260      144047          744660          544447      344102
004044      745777
qr,
000270      053744          644544          453003      754777
qs,
004744      047144          244344          447344      244144
047744      777000
qt,
030056      407670          777000
qu,
030664      744047          144264          777000
qv,
030661      747147          261777
qw,
030665      743144          744143          265777
qx,
000245      156245          416645          756645      777000
qy,
340256      347243          016643          547777
qz,
740457      245157          244457          777000

x0,         0               /where pen is now
```

```
y0,         0
x2,         0                /where next letter begins
```

Plotter routines

```
y2,        0

constants
variables
```

Scope routines

5 February 1972

```
/hack test and/or demo program for sdp
test,      e2m
           ckn
           lai
           ckn 100
           LZI 10.
           TIX
           ckn 200
           LZI 1
           XVII
           jdp sdp
           jmp test

/twos mode version of
/scope decimal print
  /jdp sdp
/IO contains scope coordinates
/          x, low 9 bits    y, high 9 bits
/AC number to be displayed

size=2     /power of 2

sdp,       0
           dio s̄dz
 sda,      cli∨swp
           div . 1
           12
           dac s̄dy
           swp
           LZA 1
           add (sdt
           dap sdc
           dzm s̄dx
           lan 21
sdb,       dac s̄dw
           dac s̄dv
 sdc,      lio .
           idx sdc
 sdd,      lac sdx
           sub (400001+6007xsize
           and (400000∨7007xsize
           spa
           add (400000+1001xsize
           dac sdx
           LRI 1
           spi i
           jmp sde
           dio s̄du
           add sdz
           lia
           RRA 9.
           iot 207
           lio sdu
 sde,      isp sdw
           jmp sdd
           lan 22
           sas sdv
           jmp sdb
           lan 6xsize
           add sdz
```

This twos mode version text is (or was) filed on public tape #4 ('pdp-4') under filename:

scope
decimal
print
(twos)   2

-CWR
730620

```
           and (776777
           dac sdz
           lac sdy
           sza
           jmp sda
           jmp i sdp

sdt,       175014           30137        /0
           27               360000       /1
           305215           31143        /2
           105014           31133        /3
           36100            217704       /4
           137114           631130       /5
           175114           231131       /6
           3610             620501       /7
           155114           231133       /8
           15114            231137       /9

variab
consta

start test
```

Scope routines

```
/scope decimal print
/jdp sdp
/IO contains scope coordinates
/          x, low 9 bits   y, high 9 bits
/AC number to be displayed

size=1         /power of 2

sdp,           0                  delete in 2twos
               dio sdz
sda,           cliVswp
               ~~scl 1~~
               div . 1            /10 if octal display wanted.
               12
               dac sdy
               swp
               ~~sal 1~~     LZA 1
               add (sdt
               dap sdc
               dzm sdx
         lan   ~~law i~~ 21
sdb,           dac sdw
               dac sdv
sdc,           lio .
               idx sdc                       400001     in twos
sdd,           lac sdx
               sub (400000+6007xsize
               and (400000+7007xsize      V in twos.
               spa
               add (400000+1001xsize-1    delete in twos
               dac sdx
               ril 1     LRI 1
               spi i
               jmp sde
               dio sdu
               add sdz
               lia
               ~~rar 9s~~     RRA 9.
               iot 207
               lio sdu
sde,           isp sdw
               jmp sdd
      lan?     ~~law i~~ 22
               sas sdv
               jmp sdb
      lan      ~~law i~~ 6xsize
               add sdz
               and (~~1000~~     776777
               dac sdz
               lac sdy
               sza
               jmp sda
               jmp i sdp

sdt,           175014            30137          /0
               27                360000         /1
               305215            31143          /2
               105014            31133          /3
               36100             217704         /4
```

```
Scope routines

            137114          631130          /5
            175114          231131          /6
            3610            620501          /7
            155114          231133          /8
            15114           231137          /9

variab
consta
start
```

Scope routines

```
/simple line display
/jdp dsp with coordinates of endpoints in dx1,dy1,dx2,dy2
/coordinates must be right justified, magnitude<1000
/dx1,dy1,dx2,dy2 are saved

dx1,         0                  /x coordinate of endpoint
dy1,         0                  /y coordinate
dx2,         0
dy2,         0

dx3,         0
dy3,         0

dsp,         0
             lac dx2            /could be modifed to enter with
             sub dx1            /x2 and x1 in AI
             TAX
             lac dy2
             sub dy1
             dac dy3
             spa
             cma
             TXIX>=
             cmi
             AMI>=
             swp
             sir 1
             A+II
             sir 2s             /change spacing here
             dio dn
             TXA
             CIX≠
             jmp dsl-3
             mul (200
             div dn
             hlt
             sal 1s
             dac dx3
             lac dy3        law 200
             mul (200       mul dy3
             div dn
             hlt
             sal 1
             dac dy3
dsl-3,  →    lac dx1
             lio dy1
             rcl 8s
dsl;         iot 207
             SXX            ≤
             jmp i dsp
             swp
             add dy3
             swp
             add dx3
             jmp dsl
dn,          0                  /no. of points -1
```

Scope routines

FOR TWOS MODE SEE CDHowe

```
/truncating line display with fixed window

/Coordinates in x1, y1, x2, y2, right justified, up to 17 bits
/Coordinates are truncated down to magnitude ≤ 777 (size of scope face)

dsp,        0
sqn,        iam                 /could be nam
            law tab
            dac x̄r1
sq0,        clf 6
            lxr xr1
            bam
            law i 777
            szf 6
            cma
            lia
            add i 3
            swp
            add i 4
            szf 6
            cmaVcmi
            TAAM|
            cla
            TIIM|
            cli
            AVI<M
            jmp lg6             /forget it, can't be seen
           A∧I>P
            jmp sq2             /no adjustment
            spi
            SXX|
            swp
            dac 11
           AMIA
            dac 12
            lac i 4
            sub i 3
            mul 11
            div 12
            hlt
           adm i 3
            lac i 1
            sub i 0
            mul 11
            div 12
            hlt
           adm i 0
sq2,        iam
            szf 6
            jmp .+3
            stf 6
            jmp sq0+1
            law 3
           adm xr1
            sas (tab+6
            jmp sq0
            lac x2
            sub x1
            sal 6s
```

Scope routines

```
            lia
            lac y2
            sub y1
            sal 6s
            dac lg3
            spa
            cma
            spi
            AMIAX |
            A+IAX
            dac lg9
            lai
            scr 7s
            div lg9
            jmp lg4
            dac lg7
            lac lg3
            scr 7s
            div lg9
lg3,        0
            dac lg3
            TXXA
            sar 8s
            CAX
lg4,        lio y1
            lac x1
            rcl 8s
lg5,        dpy-i 300
            add lg7
            swp
            add lg3
            swp
            SXX >P
            jmp lg5
lg6,        ~~nam~~ nam  → note, this should be deleted
            jmp i dsp

lg7,        0
lg9,        0
x1,         0
y1,         0
x2,         0
y2,         0

tab,        x1                x2                x1
            y1                y2                y1
            x1                x2                x1

.+6/

constants
variables
start
```

Scope routines

```
/line drawer with variable window
/takes beginning coordinates in (x1,y1), end in (x2,y2), these are
/destroyed. Coordinates are right justified, up to 17 bit magnitude.
/Coordinates are truncated according to the limits in upr, lwr, lft,
/and rgt, and the result is displayed. Goes to fgt in base mode if
/no part of line is visible. Displays line once and goes to lg6 if
/it is visible. Limits should not exceed ± 1777 (size of scope face)

sqn,        iam                 /could be nam
            law upr
            dac xr1
sq0,        claVclf 6
            dap sq1
sq5,        dap sq3
            lxr xr1
            bam
            lac i 10
sq1,        sub
            lia
            lac i 11
sq3,        sub
            szf 6
            cmaVcmi
            TAM|
            cla
            TIM|
            cli
            AVI<M
            jmp fgt             /forget it, can't be seen
            A∧I>P
            jmp sq2             /no adjustment
            spi
            SXX
            spi i
            swp
            dac 16
            AMIA
            dac 17
            lac i 11
            sub i 10
            mul 16
            div 17
            hlt
            adm i 10
            lac i 6
            sub i 5
            mul 16
            div 17
            hlt
            adm i 5
sq2,        iam
            stf 6
            idx sq1
            sas (sub 2
            jmp sq5
            law 3
            adm xr1
            sas (upr+6
            jmp sq0
```

Scope routines

```
          lio x1
          sil 8s
          dio x1
          sir 2s
          lac x2
          sal 6s
          AMII
          dio lg7
          lac y2
          sub y1
          sal 6s
          dac lg3
          spa
          cma
          spi
          cmi
          A+IA
          dac lg9
          lai
          scr 7s
          div lg9
          jmp lg4
          dac lg7
          lac lg3
          scr 7s
          div lg9
lg3,      0
          dac lg3
          lac lg9
          sar 8s
          cma
          dac lg9
lg4,      lio y1
          sil 8s
lg5,      lac x1
          dpy-i 300
          add lg7
          dac x1
          lac lg3
          A+II
          isp lg9
          jmp lg5
lg6,      jmp 4
fgt,      jmp 4

lg7,      0
lg9,      0
x1,       0
y1,       0
x2,       0
y2,       0

upr,      1777
lwr,      -1777
          0
rgt,      1777
lft,      -1777
          x1
          x2
          x1
```

Scope routines

```
           y1
           y2
           y1
           x1
           x2
           x1
.+6/
consta
variab
start
```

Scope routines

```
/character display
/upper and lower case
/jdp cd, character in ac[12-17]
/ac,io,xr lost
/nam or iam
/char 16 =>reset pointer to upper left corner of scope
/char 14 =>pointer =>
/code word format
/
/          L7  L14 |R3  R10 R17
/          L6  L13 |R2  R9  R16
/          L5  L12 |R1  R8  R15
/          L4  L11 |R0  R7  R14
/          L3  L10 L17 |R6  R13
/          L2  L9  L16 |R5  R12
/          L1  L8  L15 |R4  R11

cd,        0
           and (77
           sal 1
           add cds
           dap cd4-1
           cla
           dap cd6 2
           law cd5-1
           dap cd4
           xct cd4-1
           spa
           jmp c9d
cd1,       and (376000
           TAXA
           lio cdy
cd2,       A+XX<
           jmp cd3
           lac cdx
           dpy-i 200
cd3,       law 1000
           A+II
           lac (376000
           A∧XXA=
           jmp cd2
           law 1000
           adm cdx
           idx cd4
           lac .
cd4,       xct .
           jmp cd1
           idx cd4-1
           xct cd4-1
           rcr 4s
           jmp cd1

cd5,       ral 7s
           TAI|
           ral 3s
           rar 8s
           jmp cd6

cd6,       law 2000
```

Scope routines

```
             adm cdx
             law .
             adm cdy
             jmp i cd

cd7,         lac (77777        /tab
             ior cdx
             add (7001
             dac cdx
             jmp i cd
```

Scope routines

```
cdc,        ZAP                 /lower case
            law 200             /upper case
            add (cd9
            dac cds
            jmp i cd

cds,        cd9
cdx,        507000
cdy,        300000

c9d,        sas (add
            jmp c8d
            idx cd4-1
            dap . 1
            jmp .

c8d,        law 3000
            dap cd6 2
            cma
            adm cdy
            xct cd4-1
            jmp cd1

cd8,        lac (-14000         /c.r.
            adm cdy
            lac (507000
            dac cdx
            jmp i cd

ini,        lac (300000         /16 - initialize
            dac cdy
            jmp cd8+2
```

```
cd9,      0           0           /space
          27          740000      /1
          305214      462306      /2
          105014      462266      /3
          36100       437610      /4
          137114      462261      /5
          175114      462262      /6
          3610        441203      /7
          155114      462266      /8
          15114       452236      /9
          add         jmp i cd
          add         jmp i cd    /stop
          51253       305010      /pointer
          add         jmp i cd
          add         jmp ini     /initialize
          add         jmp i cd

          175014      60276       /0
          100200      401002      //
          221245      211000      /s
          10764       211000      /t
          171004      37000       /u
          70404       10034       /v
          171003      20074       /w
          210501      12104       /x
          417104      417600      /y
          211445      223104      /z
          add         jmp i cd
          500200      0           /,
          add         jmp i cd
          add         jmp i cd
          add         jmp cd7     /tab
          add         jmp i cd
```

Scope routines

```
        0               400000          /·
        501004          217200          /j
        376202          421000          /k
        1774            0               /l
        370047          401170          /m
        370040          236000          /n
        161044          216000          /o
        776110          441400          /p
        414110          477640          /q
        370040          202000          /r
        add             jmp i cd
        add             jmp i cd
        20100           402010          /-
        1012            107000          /)
        2010            40201           /‾
        342             120200          /(

        add             jmp i cd
        161044          237100          /a
        377104          414000          /b
        161044          212000          /c
        161044          237600          /d
        161245          226000          /e
        21760           440400          /f
        415114          457600          /g
        376100          434000          /h
        7               500000          /i
        add             jmp cdc
        4               0               /.
        add             jmp cdc 1
        add             jmp i cd
        add             jmp i cd
        add             jmp cd8
```

Scope routines

```
          0               0              /space
          30              600            /"
          0               140000         /'
          4010            40401          /~
          104422          110434         /⊃
          70402           10034          /∨
          70020           100434         /∧
          20242           120200         /<
          1012            105010         />
          10027           740404         /↑
          add             jmp i cd
          add             jmp i cd       /stop
          51253           305010         /pointer
          add             jmp i cd
          add             jmp ini        /initialize
          add             jmp i cd

          20102           507010         /→
          4015            42206          /?
          115114          462262         /S
          2017            740201         /T
          177004          20077          /U
          76404           10037          /V
          177003          20077          /W
          306240          405143         /X
          6047            401003         /Y
          303214          461303         /Z
          add             jmp i cd
          50241           205024         /=
          add             jmp i cd
          add             jmp i cd
          add             jmp cd7        /tab
          add             jmp i cd
```

Scope routines

```
601004          20100           /
101004          20077           /J
376101          210501          /K
377004          20100           /L
376020          200577          /M
376040          404177          /N
175014          60276           /O
376110          442206          /P
175015          50336           /Q
376111          452306          /R
add              jmp i cd
add              jmp i cd
20103           702010          /+
1014            77600           /]
7               740000          /|
1774            60200           /[

add              jmp i cd
370221          44574           /A
377114          462266          /B
175014          60242           /C
377014          60276           /D
377114          462301          /E
376110          442201          /F
175015          64262           /G
376100          402177          /H
1017            760200          /I
add              jmp cdc
104240          405042          /x
add              jmp cdc 1
add              jmp i cd
add              jmp i cd
add              jmp cd8
```

Scope routines

variab
consta
start

Scope routines

```
/character display
/upper and lower case
/jdp cd, character in ac[12-17]
/ac,io,xr lost
/nam or iam
/char 16 =>reset pointer to upper left corner of scope
/char 14 =>pointer =>
/code word format
/
/           L7  L14 |R3  R10 R17
/           L6  L13 |R2  R9  R16
/           L5  L12 |R1  R8  R15
/           L4  L11 |R0  R7  R14
/           L3  L10 L17 |R6  R13
/           L2  L9  L16 |R5  R12
/           L1  L8  L15 |R4  R11
.ds=1000                /character size control. If size>1111 octal, some law
/instructions become lac ( 's.

define amacro w
           repeat ifp 7777-w,law w
           repeat ifm 7777-w,lac (w
termin

cd,        0
           and (77
           sal 1
           add cds
           dap cd4-1
           law
           dap cd6 2
           law cd5-1
           dap cd4
           xct cd4-1
           spa
           jmp c9d
cd1,       and (376000
           TAXA
           lio cdy
cd2,       A+XX<
           jmp cd3
           lac cdx
           iot 207
cd3,       amacro .ds
           A+II
           lac (376000
           A∧XXA=
           jmp cd2
           amacro .ds
           adm cdx
           idx cd4
           lac .
cd4,       xct .
           jmp cd1
           idx cd4-1
           xct cd4-1
           rcr 4s
           jmp cd1
```

Scope routines

```
cd5,        ral 7s
            TAI|
            ral 3s
            rar 8s
            jmp cd6

cd6,        amacro .dsx2
            adm cdx
            law .
            adm cdy
            jmp i cd

cd7,        lac (77777      /tab
            ior cdx
            add (7001
            dac cdx
            jmp i cd
```

Scope routines

```
cdc,        ZAP                 /lower case
            law 200             /upper case
            add (cd9
            dac cds
            jmp i cd

cds,        cd9
cdx,        507000
cdy,        300000

c9d,        sas (add
            jmp c8d
            idx cd4-1
            dap . 1
            jmp .

c8d,        amacro .dsx3        /push bottom of 5x7 matrix down to
            dap cd6 2           /display symbols ,ypqgj
            cma
            adm cdy
            xct cd4-1
            jmp cd1                  — bug!

cd8,        lac (-.dsx14        /c.r.
            adm cdy
cd8 2,      lac (507000         /c.r. with no line feed (76)
            dac cdx
            jmp i cd

ini,        lac (300000         /16 - initialize
            dac cdy
            jmp cd8+2

c7d,        amacro .dsx7        /space
            adm cdx
            jmp i cd

c6d,        repeat ifp 7777-.dsx7,law i .dsx7         /backspace
            repeat ifm 7777-.dsx7,lac (-.dsx7
            adm cdx
            jmp i cd
```

Scope routines

```
cd9,    add         jmp c7d         /space
        27          740000          /1
        305214      462306          /2
        105014      462266          /3
        36100       437610          /4
        137114      462261          /5
        175114      462262          /6
        3610        441203          /7
        155114      462266          /8
        15114       452236          /9
        add         jmp i cd
        add         jmp i cd        /stop
        51253       305010          /pointer
        add         jmp i cd
        add         jmp ini         /initialize
        add         jmp i cd

        175014      60276           /0
        100200      401002          //
        221245      211000          /s
        10764       211000          /t
        171004      37000           /u
        70404       10034           /v
        171003      20074           /w
        210501      12104           /x
        417104      417600          /y
        211445      223104          /z
        add         jmp i cd
        500200      0               /,
        add         jmp i cd
        add         jmp i cd
        add         jmp cd7         /tab
        add         jmp i cd
```

Scope routines

```
0                400000          /.
501004           217200          /j
376202           421000          /k
1774             0               /l
370047           401170          /m
370040           236000          /n
161044           216000          /o
776110           441400          /p
414110           477640          /q
370040           202000          /r
add              jmp i cd
add              jmp i cd
20100            402010          /-
1012             107000          /)
2010             40201           /‾
342              120200          /(

add              jmp i cd
161044           237100          /a
377104           414000          /b
161044           212000          /c
161044           237600          /d
161245           226000          /e
21760            440400          /f
415114           457600          /g
376100           434000          /h
7                500000          /i
add              jmp cdc
4                0               /.
add              jmp cdc 1
add              jmp c6d         /backspace
add              jmp cd8 2       /c.r. with no line feed
add              jmp cd8
```

Scope routines

```
add             jmp c7d        /space
30              600            /"
0               140000         /'
4010            40401          /~
104422          110434         /⊃
70402           10034          /∨
70020           100434         /∧
20242           120200         /<
1012            105010         />
10027           740404         /↑
add             jmp i cd
add             jmp i cd       /stop
51253           305010         /pointer
add             jmp i cd
add             jmp ini        /initialize
add             jmp i cd

20102           507010         /→
4015            42206          /?
115114          462262         /S
2017            740201         /T
177004          20077          /U
76404           10037          /V
177003          20077          /W
306240          405143         /X
6047            401003         /Y
303214          461303         /Z
add             jmp i cd
50241           205024         /=
add             jmp i cd
add             jmp i cd
add             jmp cd7        /tab
add             jmp i cd
```

Scope routines

```
601004          20100           /-
101004          20077           /J̄
376101          210501          /K
377004          20100           /L
376020          200577          /M
376040          404177          /N
175014          60276           /O
376110          442206          /P
175015          50336           /Q
376111          452306          /R
add             jmp i cd
add             jmp i cd
20103           702010          /+
1014            77600           /]
7               740000          /|
1774            60200           /[

add             jmp i cd
370221          44574           /A
377114          462266          /B
175014          60242           /C
377014          60276           /D
377114          462301          /E
376110          442201          /F
175015          64262           /G
376100          402177          /H
1017            760200          /I
add             jmp cdc
104240          405042          /x
add             jmp cdc 1
add             jmp c6d         /backspace
add             jmp cd8 2       /c.r. with no line feed
add             jmp cd8
```

Prime tester

5 February 1972

Prime tester

```
/prime tester

/nam or iam
/preserves index register
/call with jdp prime, and an odd positiOe number in ac
/if number is prime, routine skips
/if number is composite, returns with smallest divisor in location d,
/zero in io, (number/d)-d in ac, and does not skip
/1 is prime

prime,0
dac n
law 1
dac d
a,
law 2
adm d
law 1
mul n
div d
hlt
sub d
sniVsma
jmp i prime                 /composite
szm
jmp a
idx prime
jmp i prime
```

Single precision arithmetic functions

5 February 1972

```
/square root
/jdp sqrt
/input in AI, binary point to right of A(0)
/output in A, binary point to right of A(0)
/(alternatively, input point to right of I(16),
/ output point to right of A(17))

sqrt,        0
             spa
             ZAI
             AVIP|
             jmp i sqrt
             dac t1
             dio t2
             ZXP
sq1,         SXX
             rcl 1s
             spa
             jmp sq2
             rcl 1s
             sma
             jmp sq1
             rcr 1s
sq2,         rcr 1s
             scr 1s         /initial guesss = x/2 + .47
             add (174000
             CXX |
             sar 1s
             SXX >
             jmp .-2
             CXX            /lxr (-1
srt,         sar 1s
             dac t3
             lac t1
             lio t2
             scr 2s
             div t3
             hlt
             add t3
             SXX >
             jmp srt
             jmp i sqrt
```

$$x_i = \frac{1}{2}\left(x_{i-1} + \frac{A}{x_{i-1}}\right)$$

Single precision arithmetic functions

```
/square root of sum of squares

rsq,        0
            sar 1
            sir 1
            dac rss
            lai
            spa
            cma
            dac rst
            mul rss
            sub rss
            spa
            cma
            add rst
            dac rsu                         /first guess - about 4 bits
            lac rss
            mul rss
            dac rss
            dio rsv
            lac rst
            mul rst
            adm rss
            swp
            adm rsv
            lia
            and (1
            adm rss
            div rsu
            nop
            add rsu
            cli
            rcr 1
            dac rsu                         /one iteration, Newton's method
            lac rss
            lio rsv
            div rsu
            nop
            add rsu
            cli
            rcr 1
            dac rsu                         /second iteration - scope accuracy
            lac rss
            lio rsv
            div rsu
            nop
            add rsu
            jmp i rsq

rss,        0
rst,        0
rsu,        0
rsv,        0
constant
start
```

Single precision arithmetic functions

```
/sin and cos
/Very accurate          runs in iam, dam
/On return, A = sin, I = cos

sin,        0
            add (100000
            cli
            rcl 2s
            TIX
            rcr 2s
            sub (100000
            dac x1
            mul x1
            dac x2
            scl 3s
            dac x3
            sar 4s
            sub (240573
            mul (373257
            mul x3
            add (377777
            dac x3
            lac x2
            sar 2s
            sub (205044
            mul (237010
            mul x2
            scl 2s
            add (311040
            mul x1
            scl 2s
            lio x3
            xct i blah
            jmp i sin

blah,       nop
            cmaVswp
            cmaVcmi
            cmiVswp

x1,         0
x2,         0
x3,         0

consta
start
```

Single precision arithmetic functions

```
/sine routine
/jdp sin with argument in AC
/V ery accurate
/fast - max 227 usec
/on input, pi=377777
/on output, 1=377777

sin,        0
            add (100000
            TAA IX
            and (177777
            sub (100000
            dac x1
            X+I<M               /testing XR bit 1
            jmp sc2
            mul x1
            scl 3s
            dac x2
            sar 4s
            sub (240573
            mul (373257
            mul x2
            add (377777
            jmp sc3

sc2,        mul x1
            dac x2
            sar 2s
            sub (205044
            mul (237010
            mul x2
            scl 2s
            add (311040
            mul x1
            scl 2s
sc3,        TX>P
            cma
            jmp i sin

x1,         0
x2,         0

cos,        0
            add (200000
            jdp sin
            jmp i cos
```

Single precision arithmetic functions

```
/natural log
/integer input, output point after bit 4

log,        0
            spq
            hlt
            ZXP             /or ZIXP
            SXX
            rcl 1s
            sma
            jmp .-3
            rcr 1s
            X→AX
            mul (-13056     /ln(2)
            scr 1s
            dio t2
            TXA
lob,        dac t1
            ral 6s
            and (16
            TAX
            lac i lot
            adm t2
            lac t1
            mul i lot+1
            rcl 3s
            sma
            jmp lob
            sub (400000
            sar 4s
            add t2
            add (274420     /17.x ln(2)
            jmp i log

lot,        -11624          73000
            -10045          65000
            -6372           60000
            -5061           54000
            -3444           50000
            -2600           46000
            -1336           43000
            -374            41000
```

Double precision routines

5 February 1972

Double precision routines

```
/double precis unsigned square root, +0 to +777777 777777
/binary point to right of io bit 17, jda sqt
/unsigned answer in ac with binary to right of bit 17
/if binary point shifted 2 bits in input, shift it 1 in ans.

sqt,        0
            dap sqx
            dio sq4
            lio sqt
            law i 22
            dac sq3
            dzm sqt
            dzm sq5

sq1,        sad (-11
            lio sq4
            lac sq5
            ral 1s
            dac sq5
            lac sqt
            rcl 2s
            sza i
            jmp sq2
            dac sqt
            sub sq5
            sub sq5
            sub (1
            spa
            jmp sq2
            dac sqt
            idx sq5
sq2,        isp sq3
            jmp sq1
            lac sq5
sqx,        jmp .

sq3,        -22
sq4,        0
sq5,        0
```

Double precision routines

constants

start

Double precision routines

```
/DECIMAL PRINT INTEGER.

/R. Alter, 6/9/64.

/Enter with number in AC, jda dpi.
/Assumes signed integer, binary point to right of bit 17.
/The + sign, and all leading zeroes except one just to the left of
/     the decimal point, print as space.
/Prints sign and 6 digits, no decimal point.
/Exits with AC and IO destroyed.


dpi,        0
            dap dix
            stf 1
            law ddv
            dap di4
            cli
            lac dpi
            sma
            jmp di1
            cma
            lio (char r-)
            dac dpi
di1,        tyo
            cla
            lio dpi
            ril 1s
di4,        div ddv
            hlt
            dio dpi
            lia
            idx di4
            sas (div ddv+6)
            jmp di2
            clf 1
            sni
            lio ddv+1
            tyo
dix,        jmp 0

di2,        sni
            jmp di3
            clf 1
            jmp di1
di3,        szf 1 i
            lio ddv+1
            jmp di1

decimal
ddv,        100000          10000           1000        100
100000      1
octal

constants

start
```

Double precision routines

```
/DECIMAL PRINT FRACTION.

/R. Alter, 6/9/64.

/Enter with number in AC, jda dpf, lac (n).
/Assumes signed fraction, binary point to right of bit 0.
/If n is positive, prints sign, decimal point, and |n| digits.
/     If n is negative, does the same except the sign is
/     completely suppressed.  |n| may not exceed 6.
/The + sign prints as space.
/Maximum error = 2 in the last digit position printed.
/Exits with AC and IO destroyed.


dpf,        0
            dap dfx
            xct i dfx
            stf 1
            spa
            jmp .+3
            clf 1
            cma
            sub (1)
            dac dfn
            add (7)
            sma
            jmp .+3
            law i 7
            dac dfn
            idx dfx
            cli
            lac dpf
            sma
            jmp df1
            lio (char r-)
            cma
            dac dpf
df1,        szf 1 i
            tyo
            clf 1
            lio (char r.)
df2,        tyo
            isp dfn
            jmp .+2
dfx,        jmp 0

            lac dpf
            mul (12)
            rir 1s
            dio dpf
            sza i
            law char r0
            lia
            jmp df2

dfn,        0
constants
start
```

```
/DECIMAL PRINT DOUBLE.

/Enter with number in AC and IO in mpy format, jda dpd, lac (n).
/     Bit 17 of IO is ignored.
/Assumes signed number, binary point to right of bit 17 of AC.
/Prints sign, 6 digits, decimal point, and |n| digits.  |n| may
/     not exceed 6.  Sign of n is ignored.
/The + sign, and all leading zeroes up to but not including the
/     one immediately to the left of the decimal point, print
/     as space.
/Maximum error = 2 in the last digit position printed.
/Exits with AC and IO destroyed.


dpd,         0
             dap ddx
             lac dpd
             ral 1s
             rcr 1s
             lac dpd
             dio dpd
             jda dpi
             xct i ddx
             sma
             cma
             dac dd1
             lac dpd
             jda dpf
             lac dd1
             idx ddx
ddx,         jmp 0

dd1,         0

start
```

Double precision routines

```
double decimal print. rwg

ddd,        repeat 10., 69

ddp,        0
            dap ddx
            law ddd-1
            dap dpp
            lac ddp
            swp
            spi
            cma
            dac dd1
            lio (charac r-
ddl,        idx dpp
            lac ddp
            spa
dpo,        tyo
            spa
            cma
            mul (1
            div (10.
dd1,        0
            dac ddp
            lac dd1
            rcr 9s
            rcr 8s
            div (20.
dd6,        dio ddd-1          /constant
            swp
            sub (10.
            xor (400000
            rcl 1
            sar 1
            spa
            add (10.
            dio dd1
            swp
            ior ddp
dpp,        dio .
            sza
            jmp ddl
ddo,        lio i dpp
            sni
            lio (charac r0
            xct dpo
            law i 1
            add dpp
            dap dpp
            sas dd6
            jmp ddo
ddx,        jmp .
```

Floating point package

23 January 1972

FLOATING POINT PACKAGE

Several programs are available to handle floating point data. All of the ones mentioned here are ~~in the file cabinet and~~ on ~~microtape~~ the filecase tape 4, under the name `newfloat`. That file has descriptive subfile names, so you can pull out the desired functions. The name `newfloat` is not meant to exclude other useful floating point data manipulation programs.

`Newfloat pack` includes the basic arithmetic functions and some macros. It must be edited into ET before the main program that you write to use it. Once you have done that, your program can get data by using `jsp fip    tyi`, except that e notation is not accepted. The datum is stored in two consecutive words. The first word consists of a sign bit, a nine-bit exponent, and the most significant 8 bits of the mantissa. The second word is the least significant 18 bits of the mantissa. The exponent field of the first word contains 400 more than the actual exponent. A negative number is represented by the 36 bit one´s complement of the corresponding positive number. To type the datum back out, use `jda fop    tyo`. These input-output instructions have been made into macros. Writing `getech x` will accept a value from the typewriter, echo it, carriage return, and leave the value in AI and in (x,x+1). Writing `type` will type out the current contents of AI, which are lost.

Once the data is in, arithmetic operations may proceed. To put the value of (x,x+1) into AI, write `load x`. To add the variable (x,x+1) to the contents of AI, write `jdp fad    x` or `fadd x`. To multiply by (x,x+1), write `jdp fmp    x` or `mulby x`. To divide, write `jdp fdv    x` or `divby x`. To subtract, `fsub x`. When done, ´store x´ puts the value into (x,x+1). [Don´t forget to dimension each variable with 2 spaces -- nor to double array dimensions.] To negate the contents of AI, write ´cma↓cmi´, or ´negate´. To make subscripted reference to arrays, the index register may be used to index `load` and `store`. That is, `load i x` will cause the address to be indexed. This will not work for the other macros. To index these, the effective address must be computed and placed under the call to the appropriate subroutine.

Floating point package

FUNCTIONS AVAILABLE

To fix a number, put it in the AI and write `jdp fix 0`. Change the latter constant if you want a scale factor. To float an integer in A, `jdp flo`.

There are some useful constants predefined, such as `zero, one,two,three,four,ten,pi,2pi`, etc.

To use the following functions, the appropriate newfloat file must be appended after the pack. The programs give further (self) description.

```
(exp)      `jdp exp` will replace the value of AI by e to that power.
(log)      `jdp ln`  will replace the value of AI by its natural log.
(ran)      `jdp ran` replaces by a random number in (-1.0,+1.0).
(cossin)   `jdp cos` replaces the (radian) value by its cosine.
(cossin)   `jdp sin`  replaces by the sine.
(sqrt)     `jdp fsr`  takes the square root
(bessel)   `jdp j0` replaces by Jo(AI), zeroth order Bessel fcn.
(bessel)   `jdp j1` Bessel of first order (and first kind)
(bessel)   `jdp j2` of second order
(gamma) `jdp gam` replaces by the gamma (factorial) fcn
(phi)    `jdp phi` replaces by std. normal distribution
(atan)  `jdp atn` replaces by the arctangent.
(shellsort) `jdp sortac` with (A) = base address, (I) = number
                   will sort a vector of floating numbers.
(a↑i)    `jdp ixp    arg` computes AI↑arg, arg integer.
(a↑b,exp,log)  `jdp fxp     arg` computes AI↑arg, arg floating.
     ´jdp fds     y1     x2     y2´ with x1 in the AI will draw a line
                         from (x1,y1) to (x2,y2), using + and - 1.0 as the
                         scope limits, and truncating everything outside.
```

~~´jdp txp´ followed by text will draw a title on the calcomp. There are macros ´drawto x,y´, ´penup´, and ´pendwn´ to control drawing on the Calcomp. They are described in ´newfloat plotpack´~~

To use the Calcomp, use FORTRAN.

There is also an ordinary differential equation integrator under the name ´newfloat ktm´. The program includes a description of how to call it. It features automatic step-size control, using a Kutta-Merson scheme.

Everything preserves your index register and flags.

****

The calling program must enter INDEX MODE before calling any of these subroutines.

****

Feel free to add functions, or to delete parts in your own copy, but please DO NOT MODIFY OR DELETE THE ONES MENTIONED HERE.

After all, you do not need to use ones you consider inappropriate.

An example program is appended.

2 Feb 1969
work done by William Ackerman, Rory Thompson, and Charles Landau.

Floating point package

```
/example program
/examine amplitude and phase shift of a simple difference scheme

dimension x(2),y(2),h(2),dlx(2),dly(2)

beg,    ~~carrig~~
        ~~write .give step size:.~~
        getech h

bg1,    load one
        store x
        load zer
        store y

bg2,    load x
        mulby h
        store dly
        load y
        mulby h
        negate
        store dlx

bg3,    load y
        floadd dly
        store y
        load x
        floadd dlx
        store x

bg4,    jdp fds
        y
        x
        y
        szs 10
        jmp beg
        jmp bg2

constants
variables
start beg
```

Floating point package

```
/floating pack 3 june 1969

 /normalize
/190+20N usec, N = number of steps

fnm, 0
     dac fmp
     X→AX
     dac fxr
     lac fnm
     dac fad
     TXXA<M
     cma
     sar 8s
     sub (1                   /-exp-1
     X→AX
     spa
     cma↓cmi
     scl 9s
     A↓IP|
     ƵXP
     SXX<M
     jmp .+5
     rcl 1s
     sma
     jmp .-4
     rcr 1s
     CXX
     scr 9s
     X→AX
     dac ft7                  /fdv will need this
     jmp fd8
```

Floating point package

```
/fix
/195 usec

fix, 0
     dio fmp
     TAAI>P
fx1, cma
     sar 8s
     aam
     sub fix
     sub (421
     szm
     cla
     add (43
     spa
     cla
     rar 3s
     add (add fx2+2
     dap fx2+1
     and fx1
     ral 3s
     add (fa4+2
     dap fx2
     idx fix
     lai
     lio fmp
     scl 9s
fx2, xct .
     jmp .
     repeat 4, scr 8s
     jmp i fix

/float
/540-20*[log2(AC)] usec

 flo,                    0
     lia
     sir 9s
     scr 9s
     xor (210400
     jdp fnm
     jmp i flo

fxr, 0
ft5, 0
ft6, 0
ft7, 0
ft8, 0
ft9, 0
```

```
/multiply
/665+20N usec
/N = number of steps to normalize result

fmp, 0
     dac ft8
     dio ft9
     X→AI<
     cmi
     sir 8s               /-exp(AC)
     dac fxr
     aam
     lxr fmp
     idx fmp
     dac fad
     lac i 0
     sma
     cma
     sar 8s               /-exp(mem)
     add (377
     A+IA
     dac ft7
     lac ft8
     lio ft9
     scl 9s
     scr 5s
     rir 1s
     dac ft8
     dio ft9
     lac i 0
     lio i 1
     scl 9s
     scr 5s
     rir 1s
     dac ft6
     dio ft5
     mul ft8
     scl 1s
     sal 8s
     dac fnm
     rir 2s
     dio fmp
     lac ft6
     mul ft9
     adm fmp
     rir 4s
     dio ft4
     lac ft5
     mul ft8
     adm fmp
     rir 4s
     swp
     adm ft4
     lac ft5
     mul ft9
     scr 3s
     add ft4
     TAAX                 /save sign of ac
     scr 6s
     scr 8s
     add fmp
     A$X<M
     jmp .+4
     TAAX>P
     CII|
     cma                 /change +0 to -0
     scr 8s
     add fnm
     A$X<M
     jmp .+4
     spa
     CII|
     cma
     scl 1s
     jmp fnr
```

```
/add
/665+20N usec
/N = number of steps to normalize result

fad, 0
     dio ft8+1
     dac ft8
     X→AI
     dac fxr
     aam
     lxr fad
     lac i 0
     sma
     cma
     sar 8s
     A→IA<M
     cma
     sar 8s
     dac ft7
     AMIA>
     jmp .+3
     dio ft7
     CAA|
     lxr (ft8            /must be 15 bit address
     add (47
     spa
     cla
     scr 3s
     add (fa5
     dap fa3+1
     cla
     rcl 3s
     add (fa4
     dap fa3
     law i 2
     adm ft7             /-exp-1
     lac i 0
     lio i 1
     scl 9s
     scr 3s
     and (377777
     scl 1s
     rir 1s              /each register begins with zero
     dio ft6
     dac fnm
     lac (ft8            /15 bits
     aam
     xor fad
     A$XX
     lac i 0
     lio i 1
     scl 9s
fa3, xct .
     jmp .

fa4, fan_=9s          r  epeat 12,scr fan          fan_=fan>2

fa5, repeat 4,scr 8s
     and (377777
     adm fnm
     idx fad
     cla↓swp
     rcr 1s
     lio fnm
     add ft6
fd3, TAAX               /right half
     and (377777
     A→IA>P             /check sign of left half
     SII                /carry into right
     I↓XX               /in case that turned on sign of right half
     and (377777
     TX<M
     jmp .+3
     SAA>P              /carry into left half
     SII                /wrap around to right half
     ril 1s
     rcl 1s
     scr 2s
     scl 2s

fnr, dac fmp            /save sign
     lxr ft7            /-exp-1 (must be _< -1)
     SX_<
     jmp fuv            /exponent underflow
     spa
     cma↓cmi
     A↓IP|
     ƵXP                /fraction is zero
     SXX<M
     jmp .+5
     rcl 1s
     sma
     jmp .-4
     rcr 1s
     CXX                /put exp in XR (_> -0)
     scr 8s
     scl 1s
     rir 1s
     SII>P
     SAA
     ril 1s
     scr 2s
     dac fdv
     ral 9s
     X→A<M
     jmp .+5
     SAX                /fraction overflow
     lac fdv
     scr 1s             /shift fraction down
     X→AX|
     lxr fdv
fd8, ral 8s             /exponent
     A+XA>P
     jmp fov            /overflow
     lxr fmp
     TX>P
     cma↓cmi
fnx, lxr fxr
     jmp i fad
```

```
fov, lac (377777
     cli↓cmi
     jmp fnx-3
fuv, cla↓cli
     jmp fnx-3
```

```
/divide
/835+20N usec
/N = number of steps to normalize result

fdv, 0
     dac ft5
     dio ft6
     TXA
     aam
     lxr fdv
     lio i 1
     lxr i 0
     X→AX
     jdp fnm
     scl 9s
     dac ft9
     idx fdv
     dac fad
     rir 1s
     dio ft8
     lac ft5
     sma
     cma
     sar 8s
     sub (402
     adm ft7
     lac ft5
     lio ft6
     scl 9s
     scr 1s
     div ft9
     jmp fov             /division by zero
     dac fnm
     swp
     mul fc1             /200000
     div ft9
ft4, 0
     dac ft5
     lac ft8
     mul (-200000
     div ft9
     jmp fov
     mul fnm
     add ft5
     lio fnm
     sir 1s
     lxr (377777
     I←XI
     spa
     I+XI
     dio ft5
     lio fnm
     sil 8s
     sil 8s
     A←XA
     I←XI
     A+IA
     lio ft5
     jmp fd3
```

Floating point package

```
/floating input

fip, dap fi1
     TXA
     dac flo
     law i 1
     TAX
     dap fim
     dzm fi8
     dzm fi9
     law i-Z
     dac fi2
fi1, xct .
     lac fi8
     A→IAP|
     jmp fi4
     sad (20
     cla                 /0
     sad (73
     jmp fi5             /.
     sas (54
     jmp .+4
     law 600             /-
     dap fim
     jmp fi1
     add (204000
     dac fi7
     and (-360
     sas fi7
     jmp fi4
     lac fi9
     jdp fmp
     ten               /should be 15-bit
     jdp fad
     fi7               / 15-bit
     dac fi9
     dio fi8
     law i 1
fi2, 0
     jmp fi1

fi4, idx fi1
     lac fi9
     SXX<M
     jmp fim
     jdp fdv
     ten              /15-bit
     jmp .-4
fim, skp i
     cma↓cmi
     lxr flo
     jmp i fi1
fi5, law i-A+XX
     jmp fi1-1
```

```
/floating output
/number of digits printed can be changed at fop+6 and fo8+4

fop, 0
     dap fo9
     SAA
     dap fox
     law i-Z
     dac flg
     law i 8.               /number of digits + 1
     dac fi7
     lac fop
     dzm flo
     A←IM|
     ZAI                    /minus zero
     dio fi8
     sma
     jmp fo1
     cma↓cmi
     dio fi8
     lio (charac r-
     xct fo9
     lio fi8
fo1, dac fi9
     jdp fix
     0
     lio (jdp fmp
     sza i
     jmp .+4                /number is < 1
     law i-CAA              /number is _> 1
     lio (jdp fdv
     dac flg
     dio fo2-1
     law fo5-2
     dac fo2          <---- should be dap
     law 100
fo2-6,                      dap fo3
     law 2
     adm fo2
     lac fi9
     lio fi8
     0                      /jdp fmp or fdv
fo2, 0                      /some power of ten
     dac ft9
     jdp fix                /this contains a dio fmp
     0
     xct flg
     TA<P
     jmp f33
     lac ft9
     lio fmp
     dac fi9
     dio fi8
     A↓IP                   /don't change exponent if exactly zero
fo3, law
flg, 0                      /Z, or CAA if num was _> 1
     adm flo
f33, xct fo3
     rar 1s
     sma
     jmp fo2-6
     xct flg
     spa
     jmp f00
     law i 1                  /number was _> 1
     adm flo
     idx fi7
     lac fi9
  lio fi8
     jmp fo7+6
f00, lio (charac r.
     xct fo9
     jmp fo8

/fi9, fi8 are now strictly < 1

fo7, lac fi9
     lio fi8
     jdp fmp
     ten
     dac fi9
     dio fi8
     jdp fix
     0
     TAI|=
     lio (charac r0
fo9, xct .
     rir 5s
     law 202
     rcl 9s
     cma↓cli↓cmi
     jdp fad
     fi9
     dac fi9
     law 7
     A↓IA                     /compensate for truncation
     dac fi8
     idx flo
     sza i
     jmp f00
fo8, isp fi7
     jmp fo7
     lac flo
     CAAI|<
     add (7                   /number of digits
     TAA=
     A$I>P
fox, jmp fox
     cli
     xct fo9
     lio (charac re
     xct fo9
     cli
     xct fo9
     lio (charac r-
     CAA>P
     CAA|
     xct fo9
     dac ft9
```

```
        dzm ft8
dpp,    dac ft7
        mul (1
        div .+1
        12
        sas ft8
        jmp dpp
        sni
        lio (charac r0
        xct fo9
        lac ft7
        dac ft8
        lac ft9
        sas ft8
        jmp dpp
        jmp fox

fo5,    352702                  360175     /10↑64
        265635                  613267     /10↑32
        233216                  067447     /10↑16
        215676             5     70200     /10↑8
        207234                  fc1,200000
hun,    203710                  0
ten,    202240                  0

fi9,    0
fi8,    0
fi7,    0                       0
```

Floating point package

```
/macros for new pack, 27 april 68

define                  getech a
     jsp fip
     tyi
     dac a
     dio a+1
     jda fop
     tyo
     cli↓cmi
     tyo
     lac a
     lio a+1
     terminate

define                  load a
     lac a
     lio a+1
     terminate

define                  fadd a
     jdp fad
     a
     terminate

equals floadd,fadd

define                  mulby a
     jdp fmp
     a
     terminate

define                  store a
     dac a
     dio a+1
     terminate

define                  fsub a
     negate
     jdp fad
     a
     negate
     terminate

define                  type
     jda fop
     tyo
     terminate

define                  divby a
     jdp fdv
     a
     terminate

negate=cma↓cmi
```

obsolete

```
define                  carrig
    cli↓cmi
    tyo
    terminate

define                  write a
    jsp txx
    text  a
    lio (236
    tyo
    terminate

txx, dap txy
    aam
    lio txy
    idx txy
    lac (607600
    rcl 6s
    sad (lai
txy, jmp
    sad . 2
    jmp txx 1
    swp
    tyo
    lia
    jmp txy-3
```

```
/constants

zero,                  0         0
zer=zero
one, 200600            0
two, 201200            0
three,                 201300    0
thr=three
four,                  201600    0
for=four
five,                  201640    0
fiv=five
/ten and hun (100.) exist elsewhere
pi,  201311            37553
pi2, 200711            37553
2pi, 201711            37553
m1,  577177            777777
m2,  576577          7   77777
haf, 200200            0
pt1, 176714            631463    /0.1
p01, 175243            656051    /0.01
rt2, 200665            11715     /sq root of 2.0

dimension tem(2),tmp(2) /so always available

dimension fb1(2),fb3(2),fb5(2)    /used by functions
```

```
/sqrt, 4 feb 1969
/Newton-Raphson method, 3 iterations
/call by jdp fsr
/sqrt(AI) → AI
/5.1 msec

fsr,  0
      spa
      negate
      A↓IP|
      jmp i fsr
      dac fi7
      dio flo
      scr 1s               /approximate sqrt(x) by
      rar 8s               / x/2+.4750 in [.5,1)
      spa                  /or x/2+.4645 in [1,2)
      add (202000          /relative error < .0355
      ral 8s
      add (100172
      dac fi9
      law i 3
      dac fix
fsa,  dio fi9+1
      lac fi7
      lio flo
      divby fi9
      floadd fi9
      sub (400
      dac fi9
      isp fix
      jmp fsa
      lac fi9
      jmp i fsr
```

```
/log, 18 dec 1968
/call by jdp fln
/ln(AI) → AI
/10 msec

ln2, 200261              3  44137     /.6931471 (ln(2))
fln, 0
     jdp lg2
     jdp fmp
     ln2
     jmp i fln

/log base 2
/Hastings, Approxs. for dig. comp., p166
/call by jdp lg2
/log2(AI) → AI
/10 msec

lg2, 0
     jdp fnm
     spq
     hlt                    /zero or negative
     and (377
     dio fb3 1
     lio (-401
     sub (265
     sma
     SII|                   />1/sqrt 2
     add (400               /<1/sqrt 2
     add (200265
     dac fb3
     lai
     add ft7
     jdp flo
     store fb1
     lac one
     cli
     floadd fb3
     store fb5
     cli↓cmi
     lac m1
     floadd fb3
     divby fb5
     store fb5
     mulby fb5
     store fb3
     mulby flh
     floadd flh+2
     mulby fb3
     floadd flh+4
     mulby fb3
     floadd flh+6
     mulby fb5
     floadd fb1
     jmp i lg2
flh, 177736                 256453    /.4342597
     200223                 460041    /.5765385
     200366                 161113    /.9618007
     201270              5   24353    /2.885390
```

```
/exp, 12 nov 1969
/call by jdp exp
/e↑(AI) → AI
/7 msec

exp, 0
     jdp fmp
     .+3
     jdp f2x
     jmp i exp

     200670                 524355     /1.4426950409 (log2(e))
```

```
/2↑x
/jdp f2x
/2↑(AI) → AI
/Hart et al, Computer Approx's, number 1063
/approximation is accurate to .18E-9 relative
/6.5 msec

f2x, 0
     store fb3
     div (204400
     hlt                 /| exp| _> 256.
     load fb3
     jdp fix
     -8.
     dac fb3
     and (777400
     spa
     sub (1
/separation of integer and fraction parts is bad by one bit
/here if arg is negative, maybe someone would like to fix it
     dac fb1
     law 377
     and fb3
     xor fc1             /200000
     jdp fnm
     floadd f2a+12
/if fraction part = 0.5, will get zero here, divide overflow
/will occur, seems to get correct result exactly
     store fb3
     load f2a
     divby fb3
     floadd fb3
     store fb5
     load f2a+2
     divby fb5
     floadd fb3
     floadd f2a+4
     store fb5
     load f2a+6
     divby fb5
     floadd f2a+10
     add fb1
     jmp i f2x

f2a, 202646              432240    / 20.818923793
     203720              400752    / 104.25093406
     575165            2  47724    /-17.334004949
     574473              706557    /-49.027969777
     577112            7  66062    /-1.4142135623 (-sq rt of 2)
     577577              777777    /-0.5
```

```
/cossin, 16 feb 1969
/jdp sin or jdp cos
/sin(AI) or cos(AI) → AI
/Hart et al, Computer Approx´s
/approx no. 3180 (7th degree poly) for sin
/approx no. 3700 (6th degree poly) for cos
/both are accurate to .9E-8 (relative)
/7.9 msec

cos, 0
     floadd pi2
     dac fb1
     lac cos
     dac sin
     lac fb1
     jmp sin+1

sin, 0
     divby pi2
     store fb3
     jdp fix
     0
     TXI
     dio fb1
     CAAI<M
     SAA|
     sub (1
     sar 1s
     scr 1s
     A$IX
     scl 2s
     jdp flo
     floadd fb3
/in first or fourth quadrant
/-1 _< AI _< 1, want sin (AI*pi/2)
/XR has sign
     A$XX
     spa
     negate
     dac fb5             /now in [0,1]
     sub (200222         /.5688
     spa
     jmp sn2
/use cos in [0,.4312*pi/2]
     lac fb5
     floadd m1
     dac fb5
     law sit+10
     jmp sn2+1
/use sin in [0,.5688*pi/2]
sn2, law sit
     dac sic
     add (2
     dac sia
     lac fb5
     dio fb5+1
     mulby fb5
     store fb3
     jdp fmp
```

```
sic, 0
     jmp sia-1
sib, lac cos
     mulby fb3
     jdp fad
sia, 0
     dac cos
     law 2
     adm sia
     sad (sit+20
     jmp scg              /end of cos
     sas (sit+10
     jmp sib
     lac cos              /end of sin
     mulby fb5
     jmp .+2
scg, lac cos
     TX>P
     negate
     lxr fb1
     jmp i sin

sit, 603151               767434     /-.00457813997
     176643               125500     / .07967150287
     577532               504277     /-.64596270916
     200711           0    37552     / 1.5707963124
     602127               721431     /-.02051888693
     177601               667174     / .25362870276
     577142               054232     /-1.2336989859
     200377               777777     / .99999999043
```

```
/elliptic functions, 18 feb 1969
/argument is parameter, or modulus↑2
/Hart et al, Computer Approx's, numbers 7303 and 7403
/both are form P(1-x)-log(1-x)Q(1-x), P and Q quintic
/accurate to .35E-9 absolute
/jdp elpe or jdp elpk
/jdp elpec or jdp elpkc for complementary functions
/20 msec

elpk,                   0
     dac el1
     lac elpk
     dac elpe
     law ept+24.
     jmp el0

elpe,                   0
     dac el1
     law ept
el0, dac ep5
     lac el1
     negate
     floadd one
el2, store el1
     jdp lg2
     store fb3
ep3, law 2
     add ep5
     dac ep6
     load el1
     jdp fmp
ep5, 0
     jmp ep6-1
ep4, lac elpk
     mulby el1
     jdp fad
ep6, 0
     dac elpk
     law 2
     adm ep6
     sas (ept+12.
     sad (ept+36.
     jmp ep7            /done with first polynomial
     sad (ept+24.
     jmp .+3
     sas (ept+48.
     jmp ep4
     lac elpk           /done with second polynomial
     floadd fb3
     jmp i elpe

ep7, dac ep5
     lac elpk
     mulby fb3
     negate
     store fb3
     jmp ep3
```

```
elpkc,                          0            /K(1-x)
     dac fb1
     lac elpkc
     dac elpe
     law ept+24.
     jmp ec0

elpec,                          0            /E(1-x)
     dac fb1
     lac elpec
     dac elpe
     law ept
ec0, dac ep5
     lac fb1
     jmp el2

ept, 173701                     025376       /.001472793465
     175367                     766326       /.015135576508
     176232                     032777       /.037610529537
     176604                     644471       /.064854252062
     177261                     343714       /.173286242518
     0                          0            /0.

     174772                     613277       /.007652960603
     175773                1     37152       /.030662347457
     176202                     057746       /.031761145525
     176353                6     26576       /.057566998484
     177742                     711674       /.443152874726
     200600                     0            /1.

     173647                     646422       /.001280405152
     175324                     750217       /.012997660452
     175777                     334213       /.031180446501
     176307                     122541       /.048623413677
     176661                     343532       /.086642909577
     177661                     344140       /.346573590280

     174731                     445301       /.006639801115
     175724                     540176       /.025962888453
     175702                     515632       /.023761224858
     176201                     210732       /.031559431628
     176705                     626014       /.096578619623
     200661                3     44140       /1.38629436112

el1, 0                          0
```

```
/arcsincos, 6 feb 1969
/Hart et al, Computer Approx's, number 4693
/call by jdp ars or jdp arc
/arcsin(AI) or arccos(AI) → AI
/approximation is accurate to 2.5E-9
/about 7 msec in [-0.5,0.5], 12 msec elsewhere

ars, 0
     dac fb1
     TXA
     dac xrs
     lac fb1
     sma
     SXX|
     negate
     sub haf
     spa
     jmp as6
     add (200200-400      /in [0.5,1.0]
     negate               / -x/2
     floadd haf           / (1-x)/2
     TAA_>
     hlt                  /arg was not in [-1,+1]
     jdp fsr
     CXX|
as6, add haf
     store fb5            /now in [0,0.5]
     mulby fb5            /uses x*P(x↑2)/Q(x↑2)
     store fb1            /P and Q are quadratic
     floadd as1
     store fb3
     load as1+2
     divby fb3
     floadd as1+4
     floadd fb1
     store fb3
     load as1+6
     divby fb3
     floadd as1+10
     mulby fb5
     TXX<M
     jmp .+5
     add (400             /if x was _> 0.5,
     negate               /use pi/2 - 2 arcsin(sqrt((1-x)/2))
     floadd pi2
     TX|=
     negate               /restore sign
     lxr xrs
     jmp i ars
```

```
arc, 0
     jdp ars
     negate
     floadd pi2
     jmp i arc

as1, 577075                 766502    /-1.5157679526
     577575                 543412    /-.50900634073
     576176                 750207    /-4.0326986467
     577021                 242072    /-1.8647138142
     177775                 576334    / .49559947479

xrs, 0
```

```
/atan, 27 mar 1969
/call by jdp atn
/arctan(AI) → AI
/Hart et al, Computer Approx´s, number 5051
/approximation is accurate to 1.05E-9 relative
/approx 11 msec

atn, 0
     dac fb1
     TA>P
     law cma↓cmi-opr
     dap svs             /save sign
     law at4+2
     dac at6
     lac fb1
     xct svs
     store fb1
     dzm at8
     sub (177611
     spa
     idx at8             /| x|  _< .26795 = tan(pi/12)
     sub (767
     spa
     idx at8             /| x|  _< 1 = tan(pi/4)
     sub (557
     spa
     idx at8             /| x|  _< 3.7321 = tan(5pi/12)
     law 3
     sub at8
     jdp flo
     mulby pi6
     store fb7           /amount to be added back later
     law 2
     sub at8
     SAP|
     jmp at3             /no correction
     ral 1s
     add (at1
     dac at5
     dac at8
     load fb1
     jdp fad
at8, 0
     store fb1
     TXA
     lxr at5
     lio i at9-at1+1
     lxr i at9-at1
     X→AX
     divby fb1
     negate
     jdp fad
at5, 0
at2, store fb1           /now | AI|  _< tan(pi/12)
     mulby fb1
     store fb3
     mulby at4
     jdp fad
at6, 0
```

```
     store fb5
     law 2
     adm at6
     sad (at4+10
     jmp at7
     load fb3
     divby fb5
     jmp at6-1
at7, load fb1
     divby fb5
     floadd fb7
svs, opr
     jmp i atn
at3, load fb1
     jmp at2+2

at1, 200735                 547535    /sqrt(3) = 1.73205080757
     200223            6     32351    /sqrt(3)/3 = .577350269190
     0                      0         /0

at9, 201600            0              /4
     200652                 525253    /4/3
     200600                 0         /1

at4, 177636                 714464  /.31035080523
     200640                 037746    /1.2504875062
     201300                 000111    /3.0000043545
     200600                 000000    /1.0000000011

pi6, 200206                 025107    /pi/6 = .523598775598
fb7, 0                      0
```

```
/ati, 20 dec 1968
/call by jdp ixp followed by exponent (integer)
/AI↑exponent → AI
/$ 1.1 log2(exponent) msec

ixp, 0
     store fb3
     lio one
     dio fb1
     dzm fb1+1
     TXA
     dac fb5
     lxr ixp
     idx ixp
     lxr i 0
     lac i 0
     lio (jdp fdv
     spa
     CAA|
     lio ixb+2          /jdp fmp
     dio ixo
ixc, scr 1s
     TAX
     spi i
     jmp ixb
     load fb1
ixo, 0                  /jdp fmp or fdv
     fb3
     store fb1
ixb, load fb3
ixb+2,                  mulby fb3
     store fb3
     TXAP
     jmp ixc
     load fb1
     lxr fb5
     jmp i ixp
```

```
/a↑b, 20 dec 1968
/call by jdp fxp followed by exponent
/AI↑exponent → AI
/base must be _> 0, 0↑0 = 0
/17.5 msec

fxp, 0
     dac fmp
     aam
     lac fxp
     dac .+7
     idx fxp
     lac fmp
     sza i
     jmp i fxp             /base is zero
     jdp lg2
     jdp fmp
     0
     jdp f2x
     jmp i fxp
```

```
/cube root, 21 mar 1969
/Hart et al, Computer Approx's, number 560
/followed by 3 Newton iterations
/call by jdp cbr
/cbrt(AI) → AI
/12 msec

cbr, 0
     store fb5
     X→AX
     dac cbx
     TXXA>P
     negate
     store fb3
     sar 8s
     SAA
     mul (1
     div .+1
     -3
     add (125
     sal 8s
     dac fb1
     law 376
     A+II
     sil 8s
     law 377
     and fb3
     A↓IA
     lio fb3+1
     mulby cba
     floadd cba+2
     sub fb1
     TX>P
     negate
     lxr (-3
cbb, store fb3
     mulby fb3
     store fb1
     load fb5
     divby fb1
     negate
     floadd fb3
     divby mth
     floadd fb3
     SXXP
     jmp cbb
     lxr cbx
     jmp i cbr

cba, 200232              534046      /.6042181
     177750          0    12057      /.4531635
cbx, 0
mth, -201300             -0
```

```
/ran, 26 mar 1969
/call by jdp ran
/random number in (0,1] → AI

ran, 0
     lac ;r1
     xor (311071
     add (311071
     rar 9s
     dac r1
     lac ;r2
     xor (355671
     add (355671
     rar 7s
     dac r2
     xor r1
     cli↓swp↓cma↓cml
     rcl 7s
     and m1              /(-200600
     floadd two
     jmp i ran
```

```
/phi
/2 april 68
/call by phi

phi=jdp ph2

ph2,  0
      dac fb1
      TXA
      dac fb3
      lac fb1
      sma
      CXXM                    /if positive argument
      cma↓cmi
      divby rt2
      store fb1

      load a6
      mulby fb1
      floadd a5
      mulby fb1
      floadd a4
      mulby fb1
      floadd a3
      mulby fb1
      floadd a2
      mulby fb1
      floadd a1
      mulby fb1
      floadd one
      store fb1
      mulby fb1
      store fb1
      mulby fb1               /fourth power
      store fb1
      mulby fb1
      store fb1
      mulby fb1
      store fb1

      load haf
      divby fb1
      TXP
      jmp ph1
      negate
      floadd one
ph1,  lxr fb3
      jmp i ph2


a1,   176620                  334631
a2,   176255                  137631
a3,   175227                  706637
a4,   172237                  314031
a5,   172621                  336
a6,   171264                  476631
```

```
/pull in tape to first carriage return
/i.e., expects a title to ignore

hdr, .-.                  /use jdp hdr
     law char rr
     arq
nrd, hlt                  /reader not available
     rpa
     lai
     and (77
     sas (77
     jmp nrd+1            /if not cr
     jmp i hdr
```

Floating point package

```
/read one value and return it in AI

/use so:
/first,    jdp hdr   to get in the beginning
/then,    jsp fip     jdp hnd

hnd, .-.
     dac tmp
     rpa
     lai
     and (77)             /so looks like tyi
     dac tmp+1            /so exit via hn4 when accept
     sad (20
     jmp hn4              /0, so ok
     sas (0
     jmp .+7
     lai
     and (777
     sas (200
     jmp hn3-1            /exit if blank tape
     lac tmp
     jmp i hnd            /assuming space is acceptable to fip
     sub (12
     spa
     jmp hn4              /an integer, so okay
     lac tmp+1
     sad (73)             /period
     jmp hn4
     sad (54)             /minus
     jmp hn4
     sad (77)             /carriage return
     jmp hn4
     sad (36)             /tab
     jmp hn4
     sad (13)             /stop code
     jmp hn3-1
     jmp hnd+2            /ignore any other symbols


hn4, load tmp
     jmp i hnd

     TIIX
hn3, hlt                  /change to a jump if want an exit
```

```
/j0,   Bessel function of zeroth order and first kind
dimension rj2(2)

dimension xv3(2),xsav(2)
3vx=xv3

j0,   .-.                   /jdp j0 with argument in AI
      spa
      negate                /even function
      store xsav            /for ap943
      divby thr
      store xv3
      fsub one
      sma
      jmp ap943j

ap941,                      load xv3  /x .le. 3 in this approx
      mulby xv3
      store xv3
      mulby p00021
      fadd m00394
      mulby xv3
      fadd p04444
      mulby xv3
      fadd m31638
      mulby xv3
      fadd 1p2656
      mulby xv3
      fadd 2m2499
      mulby xv3
      fadd one
      jmp i j0              /value returns in AI

dimension jamp(2),jth(2)

ap943j,                     jdp ap943 /x .g. 3 for this approx
      load xsav
      jdp fsr
      store tem
      load jth              /phase
      jdp cos
      mulby jamp            /amp
      divby tem             /sq rt of x
      jmp i j0

p00021,                     172334    146720
m00394,                     603176    577746
p04444,                     176266    36000
m31638,                     600136    5114
1p2656,                     200641    777565
2m2499,                     576560    1
```

```
/j0 cont.
ap943,                  .-.         /amp and phase for j0 and y0.
     load thr
     divby xsav
     store 3vx
     mulby p00014
     fadd m00072
     mulby 3vx
     fadd p00137
     mulby 3vx
     fadd m00009
     mulby 3vx
     fadd m00552
     mulby 3vx
     fadd m0077
     mulby 3vx
     fadd p79788
     store jamp
ap9432,                 load 3vx
     mulby p00013
     fadd m00029
     mulby 3vx
     fadd m00054
     mulby 3vx
     fadd p00262
     mulby 3vx
     fadd m00003
     mulby 3vx
     fadd m04166
     mulby 3vx
     fadd m78539
     fadd xsav
     store jth
     jmp i ap943

p00014,                 172227      625335
m00072,                 604501      112620
p00137,                 173663      702141
m00009,                 606070      411533
m00552,                 603112      701473
m0077,                  611461      234004
p79788,                 200314      204246
p00013,                 172216      124751
m00029,                 605146      153650
m00054,                 604562      72516
p00262,                 174254      50701
m00003,                 606532      120376
m04166,                 601525      260246
m78539,                 577466      740224
```

```
/j1
/j1, Bessel function of first order and first kind

/requires cos and fsr
/dimension xv3(2),xsav(2) if not using j0

j1,     .-.                     /jdp j1 with argument in AI
        store xsav
        spa
        negate
        divby thr
        store xv3
        fsub one
        sma
        jmp ap946               /out of range for ap944

ap944,                          load xv3
        mulby xv3
        store xv3
        mulby p00001
        fadd m00031
        mulby xv3
        fadd p00443
        mulby xv3
        fadd m03954
        mulby xv3
        fadd p21093
        mulby xv3
        fadd m56249
        mulby xv3
        fadd haf
        mulby xsav
        jmp i j1                /answer returns in AI

p00001,                         170272    36276
m00031,                         605131    366156
p00443,                         174621    210452
m03954,                         601536    20431
p21093,                         177327    777041
m56249,                         577560    6
```

```
/j1, cont

/this can also be used for Y1, simply by using sin in place of cos below

ap946,                  load thr
     divby xsav
     spa
     negate             /approx for 3 < x
     store 3vx
     jdp ap946b
     load xsav
     spa
     negate
     jdp fsr            /sqrt
     store tem
     load jth
ap946y,                 jdp cos    /change to jdp sin for y1
     mulby jamp
     divby tem
     dac tem
     lac xsav
     sma
     jmp ap946p
     lac tem
     negate
     jmp i j1
ap946p,                 lac tem
     jmp i j1
```

/j1, cont.

```
ap946b,                    .-.
      load 3vx
      mulby m00020
      fadd p00113
      mulby 3vx
      fadd m00249
      mulby 3vx
   fadd p00017
      mulby 3vx
      fadd p01659
      mulby 3vx
      fadd p0015
      mulby 3vx
      fadd p79788
      store jamp

      load mooo29
      mulby 3vx
      fadd p00079
      mulby 3vx
      fadd p00074
      mulby 3vx
      fadd m00637
      mulby 3vx
      fadd p00005
      mulby 3vx
      fadd p12499
      mulby 3vx
      fadd 2m3561
      store jth
      load xsav
      spa
      negate
      fadd jth
      store jth
      jmp i ap946b
m00020,                    605455     740517
p00113,                    173624     757171
m00249,                    603534     366001
p00017,                    172263     267612
p01659,                    175607     753364
p0015,                     166721     302301
mooo29,                    605147     54074
p00079,                    173321     201751
p00074,                    173302     714142
m00637,                    603056     765526
p00005,                    171354     764646
p12499,                    176777     773733
2m3561,                    576551     150150
```

```
/j2

dimension rj2(2)
/uses recurrence:  j2 = 2 j1 over r  - j0

j2,  .-.                     /use jdp j2 with arg in AI
     store rj2
     spa
     cma
     and (377
     sas (0
     jmp nzrj2
     load zer
     jmp i j2

 nzrj2,                      load rj2
     jdp j1
     divby rj2
     mulby two
     store tmp

     load rj2
     jdp j0
     negate
     fadd tmp
     jmp i j2
```

```
/Gaussian integration subroutines
/call either the seven pt or the nine pt
/by jdp, with the lower limit of integration
/in AI, and the address of the upper limit
/in the address following the jdp.  The
/function to be integrated must be reachable
/by  xct i [second address after the jdp]
/with the argument in AI.
/e.g.,   load a    jdp gauss7    b    jdp fcn
/two routines are provided for independent estimates
dimension midpt(2),hafwd(2)
dimension fcnadd(2)

/abscissa
xi7, 177717                625675
200275                     651763
200362                     761356

/weights
177725                     775373
wi7, 177703                376351
177617                     153055
177204                     457324

xi9, 177646             1   1052
200235                     13641
200326                     14210
200367                     662541

177651                     52213
wi9, 177637                727722
177605                     335417
177270                     767523
176646                     346305


gauss7,                    .-.        /use jdp
     store hafwd           /lower limit of integration
     rpf
     nam
     dio fcnadd 1
     lac gauss7
     dac midpt
     lac i midpt
     dac midpt
     SAA
     dac midpt+1           /load i midpt to get upper limit now

     idx gauss7
     dac fcnadd
     idx gauss7

     load i midpt
     fadd hafwd
     divby two
     store midpt
     fsub hafwd
     store hafwd
     load zer
     store sum

     load midpt
     xct i fcnadd          /get value at midpoint
     mulby wi7-2
     store sum

     load xi7
     negate
     mulby hafwd
     fadd midpt
     xct i fcnadd
     mulby wi7
     fadd sum
     store sum

     load xi7
     mulby hafwd
     fadd midpt
     xct i fcnadd
     mulby wi7
     fadd sum
     store sum

     load xi7+2
     negate
     mulby hafwd
     fadd midpt
     xct i fcnadd
     mulby wi7+2
     fadd sum
     store sum

     load xi7+2
     mulby hafwd
     fadd midpt
     xct i fcnadd
     mulby wi7+2
     fadd sum
     store sum

     load xi7+4
     negate
     mulby hafwd
     fadd midpt
     xct i fcnadd
     mulby wi7+4
     fadd sum
     store sum

     load xi7+4
     mulby hafwd
     fadd midpt
     xct i fcnadd
     mulby wi7+4
     fadd sum

     mulby hafwd           /for given length
     dio hafwd
     lio fcnadd+1
     lpf
     lio hafwd
     jmp i gauss7         /result in AI
```

```
/Gauss9

/call just as gauss7

gauss9,                  .-.        /use jdp
     store hafwd
     rpf
     dio fcnadd+1
     lac gauss9
     dac midpt
     lac i midpt
     dac midpt
     SAA
     dac midpt+1

     idx gauss9
     dac fcnadd
     idx gauss9

     load i midpt
     fadd hafwd
     divby two
     store midpt
     fsub hafwd
     store hafwd

     load zer
     store sum

     load midpt
     xct i fcnadd
     mulby wi9-2
     store sum

     dzm ;k
gs9l,                    law xi9
     add k
     add k
     dap gs9a
     dap gs9ap
     SAA
     dap gs9a+1
     dap gs9ap+1

     law wi9
     add k
     add k
     dap gs9b+1
     dap gs9bp+1

gs9a,                    load
     negate
     mulby hafwd
     fadd midpt
     xct i fcnadd
gs9b,                    mulby wi9
     fadd sum
     store sum

gs9ap,                   load
```

```
/gamma function, 24 jan 1968
/call by jdp gam
/G(AI) → AI
/C. Hastings, Approx's for Dig. Comp. p 158
/accurate to 2.5E7 in range [1,2]
/approx 13.8+1.5[| z-1.5| +0.5] ms

gam, 0
     store fb3
     load one
     store fb5
     load fb3
     floadd m1
gm1, TAA>=
     jmp gm5              /too small
     store fb3
     floadd m1
     TAA>=
     jmp gm7              /0_<fb3<1
     store fb1            /too big
     load fb3
     mulby fb5
gm2, store fb5
     load fb1
     jmp gm1
gm5, load fb3
     store fb1
     floadd one
     store fb3
     load fb5
     divby fb1
     jmp gm2
gm7, law ga0+2
     dac gm8+4
     load ga0
     jmp .+2
gm8, lac fad
     mulby fb3
     jdp fad
gm8+4,                    0
     dac fad
     law 2
     adm gm8+4
     sas (ga0+22
     jmp gm8
     lac fad
     mulby fb5
     jmp i gam

ga0, 176222          7   25274       / .035868343
     600471           6    47537      /-.193527818
     177766              705533      / .482199394
     577476              221220      /-.756704078
     200353              037153      / .918206857
     577432              264747      /-.897056937
     200374              766077      / .988205891
     577554              172253      /-.577191652
     200600              0           /1.0
     mulby hafwd
     fadd midpt
     xct i fcnadd
gs9bp,                     mulby wi9
     fadd sum
     store sum

     idx k
     sas (4
     jmp gs9l

     lio fcnadd+1
     lpf

     load sum
     mulby hafwd
     jmp i gauss9
```

Floating point package

```
/phinvr
/inverse of normal distribution, for statistical problems
/Hasting's approximation
/max error about 0.00004, which is rather gross, but
/I do not know of a better one at the moment.
/uses flag 2

/call by jdp phinvr with arg in AI. returns value in AI
/note needs fln and fsr

phinvr,      0
     stf 2
/check if < 0, out of range
     sma↓sza
     jmp phnr1
     lac (400000
     cli
     jmp i phinvr
/check AI < 0.5  if so, use symmetry
phnr1,       dac phnrc
     sub (200200)       /floating monotone in AI
     spa
     jmp phnrok
     clf 2
     lac phnrc
     negate
     fadd one
     dac phnrc

/test if > 1, out of range
     sma↓sza
     jmp phnrok
     lac (377777
     cli↓cmi
     jmp i phinvr

phnrok,      lac phnrc
     jdp fln
     mulby m2
     jdp fsr
     store phnrc

/now the approx
     mulby phnrc+14
     fadd phnrc+12
     mulby phnrc
     fadd phnrc+10
     mulby phnrc
     fadd one
     store phnrc+16
     load phnrc
     mulby phnrc+6
     fadd phnrc+4
     mulby phnrc
     fadd phnrc+2
     divby phnrc+16
     negate
     fadd phnrc

     szf 2
     negate
     jmp i phinvr
phnrc,       0          0
201240       774353
200315       417430
175251       155426
200667       313144
     177301  637356
173653       342312
0    0
```

```
/sort
/ascending Shell sort
/rory wrote and tested this on 1 feb 69
/use jdp sortac with (A) = base address of vector to be sorted,
/and (I) = number of them
/saves X-register and flags/runs in lam, of course

sortac,     0
     dio ;nsort
     dio ;msort
     sub (2
     dac ;sortad
     TXA
     dac ;savx

20sort,      lac msort
     sar 1s
     dac msort
     sza
     jmp 30sort
     lxr savx
     jmp i sortac

30sort,      lac nsort
     sub msort
     dac ;ksort
     law 1
     dac ;jsort
41sort,      lac jsort
     dac ;isort
49sort,      lac isort
     add msort
     dac ;lsort

/compare to see if need to swap
     lac sortad
     add lsort
     add lsort
     dac ;sortld
     lac sortad
     add isort
     add isort
     dac ;sortid
     TAX
/it would be faster to compare the upper and lower parts separately,
/but more painful
     load i 0
     store sortem
     lxr sortld
     load i 0
     negate
     fadd sortem
     spq
     jmp 60sort

/swap
50sort,      lxr sortid
     load i 0
dimension sortem(2)
     store sortem
     lxr sortld
     load i 0
     lxr sortld
     store i 0
     lxr sortld
     load sortem
     store i 0

     lac isort
     sub msort
     dac isort
     sma↓sza
     jmp 49sort
60sort,      idx jsort
     sub ksort
     spq
     jmp 41sort
     jmp 20sort
```

Floating point package

```
/floating line display
/property of rory thompson - written by charles landau
/display a line from x1,y1 to x2,y2
/limits of scope are _+1.0
/all coordinates will be truncated to 1.0
/call by
/jdp fds
/address of x1
/address of y1
/ditto x2
/ditto y2
/control returns here

fds, 0
     TXA
     dac ;savx
     law fdq
     dac fdp
     jmp fdp 1
fdq, dac dx1
     jdp fdp
     dac dy1
     jdp fdp
     dac dx2
     jdp fdp
     dac dy2
     jdp dsp
     lxr savx
     jmp i fds

fdp, 0
     lxr fds
     lxr i 0
     idx fds
fdr, lac i
     spa
f1x, cma
     sar 8s
     sub (400              /377 makes scope limits 1/2, 401 → 2, ...
     szm
     jmp f1z
     add (43
     spa
     cla
     rar 3s
     add (add f1y 2
     dap f1y 1
     and f1x
     ral 3s
     add (fa4 2
     dap f1y
     lac i
     SXX
     lio i
     scl 9s
f1y, xct .
     jmp .
     repeat 4, scr 8s
     jmp i fdp
f1z, lac (377777
    lio i
    spi
    cma
    jmp i fdp
```

```
/line display
/jdp dsp with coordinates of endpoints in dx1,dy1,dx2,dy2
/dx1,dy1,dx2,dy2 are saved
/xr is not saved

dx1, 0        /x coordinate of endpoint
dy1, 0        /y coordinate
dx2, 0
dy2, 0

dx3, 0
dsp, 0
     lac dx2 /could be modifed to enter  with
     lio dx1 /x2 and x1 in AI
     sar 1
     sir 1
     AMIX
     lac dy2
     lio dy1
     sar 1
     sir 1
     AMIAI
     dac dy3
     spa
     cma
     TXIX>
     AMII|    /| x2-x1| +| y2-y1|
     A+II
     cla
csp, scr 2s  /increase for wider spacing
     scr 6s
     dio dn
     TXA
     CIX|=
     jmp cin-2
     mul (1
     div dn
dy3, 0
     sal 1
     dac dx3
     lac dy3
     mul (1
     div dn
dn,  0
     sal 1
     dac dy3
     lac dx1
     lio dy1
cin, iot 207
     SXX<M
     jmp i dsp
     swp
     add dy3
     swp
     add dx3
     jmp cin
```

A program to integrate systems of differential equations is available in the FORTRAN library. It uses a Kutta-Merson scheme with automatic step size adjustment.

```
        call ktmbeg(array,deriv,n,err,lim,init,max)
```

sets up the arrays and integration parameters.

All are real except n.

Array is a real array containing the variables. Array(1) is the independent variable.

Deriv is a real array containing the derivatives of the above array. Deriv(1) must contain 1.0. The calling program is responsible for this.

N is the number of variables in array and deriv.

Err is the maximum allowable error in any step of the integration. If it is exceeded, the program will do the step over with a smaller step size.

Lim is the limit of integration for the independent variable. The program will stop when the variable has increased by this amount, regardless of its initial value.

Init is the initial step size. It may be adjusted later if necessary.

Max is the maximum step size. When adjusting the step size, it will never be permitted to exceed max.

```
        call ktm
```

This begins the integration. The integrator will periodically (about four times per step) ask for the derivatives of the variables in array. The first time it will do so by returning from ktm. The calling program must then compute the derivatives of the current values of array and store same in deriv. It then returns to the integrator by

```
        call ktmret(10s)
```

where 10s is a statement number followed by the letter s (an \`abnormal subroutine return\` argument). If the integrator wants more values of the derivatives, it will return via this argument. The calling program must compute the derivatives again and call ktmret again. When ktmret makes a normal exit, the integration is complete, and the results are in array. It is then possible to call ktm again, and integrate further by a distance of lim in the independent variable.

example

```
      dimension y(2),d(2)
      call ktmbeg(y,d,2,1e-7,.1,.01,1.)
c initial step size = .01, stop every 0.1
      d(1)=1.0
      y(1)=0.0
c to begin the integration at zero
      y(2)=0.0
c to give it the right constant of integration
10    call ktm
20    d(2)=1.0/(1.0+y(1)**2)
      call ktmret(20s)
c dy/dx = 1/(1+x↑2)
      call fop(y(1))
      call tab
      call fop(y(2))
      call ret
c this makes a table of arctangents in steps of 0.1
      go to 10
```

Floating point package

```
/Kutta-Merson

/before calling, define initial values y(0) to y(n)
/where y(0) is the independent variable,
/define fin, the end of the domain,
/define n, the number of equations,
/define err, the approx max allowable error,
/not to mention h, the initial step size
/(which will be adjusted)
/or hlm, the maximum step size

/fcn is a function that returns the derivatives
/of y(n) in kfn(n).  i.e., kfn(0)=1
/int, the address of a location to handle the intermediate
/results ( none, if zero).  then jdp ktm

siz=20.
nm=2*siz
dimension y0(nm),f0(nm),f1(nm)
dimension err(2),hc(2),h(2)
dimension hc6(2),hc2(2),epv(2)
dimension hc8(2),fin(2)
dimension hc3(2),hlm(2)

n,    0        /number of equations
int,  0        /address

ktm,  0        /use jdp

      iam      /now can run in index mode
```

```
/p2 of ktm

     lio n
     law siz
     AMI>=
     bpt     /redefine siz and assemble again
     sil 1s
     dio ;2n2

     load h
     store hc

     ƵX
km0, lac i  y
     dac i  y0
     SXAX
     sas 2n2
     jmp km0

     load err
     divby ten
     store epv

km1, load hc /calculate these factors but once
     divby thr
     store hc3
     sub (400
     store hc6
     load hc
     sub (400
     store hc2
     sub (1000
     store hc8

/test if done,  if  fin - y0 < err
kmc, load y0
     cma↓cmi↓clf 6
     fadd fin
     store tem
     negate
     fadd err
     sma
     jmp i ktm           /exit

/if too close to end ( fin - y0 < hc), change hc
     load tem
     fadd epv            /to watch out for equal giving minus zero
     negate
     fadd hc
     spa
     jmp kms /hc okay
     load tem
     store hc            /change hc
     jmp km1 /get other constants
```

```
/p 3 of ktm
/proceed with step
kms, jdp fcn

      ƵX
km2,  load i kfn              /[kfn] $ [y´(t)] → [f0]
      store i f0              /[kfn]h/3 + [y0]→[y]
      mulby hc3
      store tem
      load i y0
      fadd tem
      store i y
      SXX
      SXAX
      sas 2n2
      jmp km2
```

```
/ p 4 of ktm

     jdp fcn

     ƵX
km3, load i kfn            /([kfn] + [f0])h/6 + [y0] → [y]
     store tem             /[y] $ [y(t+h/3)]
     load i f0
     fadd tem
     mulby hc6
     store tem
     load i y0
     fadd tem
     store i y
     SXX
     SXAX
     sas 2n2
     jmp km3
     jdp fcn

     ƵX
km4, load i kfn            /[kfn] $ [y´(t+h/3)] → [f1]
     store i f1            /([kfn]3 + [f0])h/8 + [y0] → [y]
     mulby thr             /[y] $ [y(t+h/2)]
     store tem
     load i f0
     fadd tem
     mulby hc8
     store tem
     load i y0
     fadd tem
     store i y
     SXX
     SXAX
     sas 2n2
     jmp km4

     jdp fcn
     ƵX
km5, load i kfn            /[kfn] $ [y´(t+h/2)] → [f2]
/([kfn]4 - [f1]3 + [f0])h/2 + [y0] → [y]
     mulby for             /[y] $ [y(t+h)]
     store tem
     load i f0
     fadd tem
     store i f0            /save 4kfn + f0 in f0 for next step
     store tem
     load i f1
     mulby thr
     negate
     fadd tem
     mulby hc2
     store tem
     load i y0
     fadd tem
     store i y
     SXX
     SXAX
     sas 2n2
     jmp km5
```

```
/ page 5 of ktm

     jdp fcn

     ƵX
km6, load i kfn            /([kfn] + [f0] + [f2]4)h/6 + [y0] → [f1]
     store tem             /[f1] $ [y(t+h)]
     load i f0
     fadd tem
     mulby hc6
     store tem
     load i y0
     fadd tem
     store i f1            /test if really want this step
     store tem
     load i y              /test if should change step size
     negate
     fadd tem
     spa
     negate
     sub (1000             /divby 4
     store tem
     negate
     fadd err              /given error bound
     sma
     jmp km7
```

```
/ page 6 of ktm
     law i 400           /error too big
     adm hc

     ZX
km9, lac i y0            /reset for first call of fcn again
     dac i y
     SXAX
     sas 2n2
     jmp km9

     jmp km1

/ here if error acceptable
km7, load tem
     negate
     fadd epv            /check if small enough so can increase step size
     spa
     stf 6   /dont increase size

     SXX
     SXAX
     sas 2n2
     jmp km6

     ZX      /accept the step
km8, lac i f1
     dac i y
     dac i y0
     SXAX
     sas 2n2
     jmp km8


     lac int /vent intermediate to outside
     szm
     jdp i int

kmz, szf 6
     jmp kmc
     load hc
     add (400            /*2
     negate
     fadd hlm            /for guard against too large a time step
     spa
     jmp kmc
     law 400
     adm hc
     jmp km1

/end of ktm
/remember to define int if use it.  Alternatively, can set fin at
/various desired steps and output then.
```

```
10 tape

bs=10

100/        hlt
            hlt
            law 500
            lio (30001
            ivk 16
            hlt
rdd,        lio (dir
            law 100
            jdp rdb
            jmp cr-1

consta
```

```
cr-1,       iam
cr,         law 15
            jdp atof
            ivk 300
            jdp rd6
            TIXP
            jmp err
            dac lst
            sad i tb1
            xct i tb2
            SXX
            jmp .-3

tb1,        0               /null command
            420000          /b - return to id
            440000          /d - delete file
            450000          /e - edit file
            460000          /f - file file
            560000          /p - print directory
lst,        0               /error

tb2,        jmp cr
            jmp id
            jmp del
            jmp edi
            jmp fil
            jmp err         /print not implemented
            jmp err

err,        law 74          ivk 100
            law 21          ivk 100
            law 72          ivk 100
            jmp cr
```

```
rdb,        0
            jdp trb
            ivk 16
            hlt
            jmp i rdb
wtb,        0
            jdp trb
            ior (add
            ivk 16
            hlt
            jmp i wtb

trb,        0
            rar 1s
            sma
            cma
            and (777
            swp
            jmp i trb
```

```
/read sixbit name into AC-IO and nm1-nm2
/call with jdp rd6
rd6,        0
            clf 7
            dzm nm1
            dzm nm2
            ZX
rdl,        ivk 400
            jdp ftoa
            sub (40
            spq
            jmp dun
            and (77
            cli
            aam
            jmp .+1
repeat 5,rcl 6s
            A→IA|
            ZAI|
            SXX
            adm nm1           /yes, I know
            lai
            adm nm2
            jmp rdl
dun,        sad (10-40
            jmp cr
            lac nm1
            lio nm2
            jmp i rd6
```

```
gfn,        0
            szf 1
            jmp err
            jdp rd6
            sza
            szf 1
            jmp err
            dac n̅am1
            dio n̅am2
            jdp rd6
            sza∨szf i 1
            jmp err
            law
            dap fwd
            jmp i gfn

ffn,        0
            dzm f̅n1
            ZX
ffl,        idx fn1
            lac nam1
            sas i dir
            jmp gfi
            lac nam2
            sas i dir+1
            jmp gfi
            lac nm1
            sas i dir+2
            jmp gfi
            lac nm2
            sad i dir+3
            jmp fff
gfi,        law 4
            X+A XA
            sas (4×27
            jmp ffl
            jmp i ffn
fff,        idx ffn
            jmp i ffn
```

```
/return to id
id,
wd,         skp i 600
            jmp id2
            law 100
            lio (buf
            jdp rdb
            lac dir+377
            sas buf+377
            -i
            law 100
            lio (dir
            jdp wtb
id2,        law i 0
            ivk 16
            nop
            law 16
            mta 204
            dsm
```

```
fil,        jdp gfn
            law 2
            jdp rset
            jdp ffn
            jmp .+2
            jmp err
            dzm fn1
            szs 20
            stf 2
            claVclf 1
            dap wd
rce,        jdp rev
            ZX
            dzm f̄n2
rce+3,      idx fn2
            lac i dir
            ior i dir+1
            ior i dir+2
            ior i dir+3
            sza i
            jmp nmf
            law 4
            A+XXA
            sas (4×27
            jmp rce+3
            jmp err
nmf,        lac nam1
            dac i dir
            lac nam2
            dac i dir+1
            lac nm1
            dac i dir+2
            lac nm2
            dac i dir+3
            dzm nam1
            dzm nam2
            dzm nm1
            lac fn2
            dac nm2
            ral 1s
            TAX
            law i 1
            and i dir+2×56-1
            dac i dir+2×56-1
            law i 1
            and i dir+3×56-1
            dac i dir+3×56-1
            law 1
            ior i dir+4×56-1
            dac i dir+4×56-1
            jdp fnb
            jmp err
            jmp fil1-1
```

```
fnfr,           law i bs-1
                xct fwd
                adm b̅lk
                jdp fnb
                jmp rce
fil1-1,         dzm 7̅pt
fil1,           law 12
                szs i 30
                szf i 1
                jmp .+2
                jmp fil3
                jdp readch
                jdp ftoa
fil2,           jdp put
                idx 7pt
                sas (200×5
                jmp fil1
                lac blk
                lio (buf
                jdp wtb
                law 3
                sas t̅0
                jmp fnfr
                jmp cr

fil3,           clf 1
                jmp fil2
```

```
edi,        jdp gfn
            law 2
            jdp wset
te,         jdp ffn
            jmp err
            jdp rev
            dzm nam1
            dzm nam2
            dzm nm1
            lac fn1
            dac fn2
            dac nm2
edb,        jdp fnb
            jmp te
            lac blk
            lio (buf
            jdp rdb
            dzm 7pt
edi1,       jdp get
            jdp atof
            jdp writec
            idx 7pt
            sas (200x5
            jmp edi1
            jmp edb
```

```
del,        jdp gfn
            law 0
            dap wd
            dzm fn2
dee,        jdp ffn
            jmp cr
            dzm i dir
            dzm i dir+1
            dzm i dir+2
            dzm i dir+3
            dzm nam1
            dzm nam2
            dzm nm1
            lac fn1
            dac nm2
            jdp rev
            jdp fnb
            jmp dee
            jmp .-2

rev,        0
            law cma-opr
            xor fwd
            dac fwd
            law 400
            xct fwd
            add (400
            dac blk
            jmp i rev
```

```
fnb,          0
              law i 1
fwd,          opr
              adm blk
              lio (1000
              szm
              AMI<
              jmp i fnb
              csc
              mul (2
              div (7
              hlt
              dac t0
              TAX
              lac i dir+4x27
              lxr i dir+4x27+1
              X→IX
              xct i 5st
              dac t1
              and (37
              sas fn1
              jmp fnb+1
              law i 37
              and t1
              ior fn2
              xct i 5ut
              lxr t0
              dac i dir+4x27
              dio i dir+4x27+1
              idx fnb
              jmp i fnb
5st,          ral 5s
              ral 7s
              rar 8s
              rar 6s
              rar 3s
              rar 1s
              rcl 2s

5ut,          rar 5s
              rar 7s
              ral 8s
              ral 6s
              ral 3s
              ral 1s
              rcr 2s
```

```
fcs,        sub (72
            ral 5s
            dac cas
            law i 2
            adm .+2
            jmp i .+1

ftoa,       0
            laiVclf 1
            sas (72
            sad (74
            jmp fcs
            ior cas
            add (tab
            dap .+1
            lc
            and (177
            sad (15
            stf 1
            szf 2
            jmp i ftoa
            sub (140
            spa
            add (40
            add (100
            jmp i ftoa

atof,       0
            sad (12
            jmp lf
            sad (15
            jmp .cr
            clf 6
af,         TAAX
            sub (100
            szs 50
            TAA>
            jmp af1
            sub (32
            TA>
            law 40
            A+XX
af1,        lac i tab
            cli
            rcl 9s
            law 100
            A∧IA
            sad cas
            jmp i atof
            dac cas
            rar 5s
            add (72
            swp
            aam
            xct atof
            lia
            jmp i atof

lf,         szf i 6
            jmp af
            idx atof
            clf 6
            jmp i atof

.cr,        szs i 40
            stf 6
            jmp af
```

```
get,        0
            clf 4
            jmp pg1

put,        0
            dac t0
            stf 4
pg1,        lac 7pt
            mul (146315
            dac t1
            AMII
            TAX
            rcl 3s
            sza i
            law 7
            lio i buf+1
            lxr i buf
            X→AX
            xct i sht
            szf i 4
            jmp pg3
pg2,        and (-177
            ior t0
            xct i ust
            lxr t1
            dac i buf
            dio i buf+1
            jmp i put
pg3,        and (177
            jmp i get

sht,        hlt
            rcl 3s
            hlt
            rar 1s
            rar 4s
            hlt
            rar 8s
            ral 7s

ust,        hlt
            rcr 3s
            hlt
            ral 1s
            ral 4s
            hlt
            ral 8s
            rar 7s
```

```
define z f,a
                f×1000 a
termin

tab,            z 76,40
                z 76,61
                z 76,62
                z 14,63
                z 76,64
                z 76,65
                z 76,66
                z 76,67
                z 75,70
                z 36,71
                z 77,0
                z 76,14
                z 13,3
                z 77,0
                z 34,0
                z 35,0
                z 76,60
                z 76,57
                z 76,163
                z 76,164
                z 76,165
                z 76,166
                z 76,167
                z 76,170
                z 76,171
                z 76,172
                z 76,0
                z 76,54
                z 76,17
                z 76,16
                z 76,11
                z 76,0
                z 0,72
                z 105,152
                z 101,153
                z 103,154
                z 103,155
                z 104,156
                z 106,157
                z 102,160
                z 57,161
                z 55,162
                z 173,0
                z 154,0
                z 33,55
                z 54,51
                z 73,73
                z 21,50
                z 20,0
                z 1,141
                z 2,142
                z 3,143
                z 4,144
                z 5,145
                z 6,146
                z 7,147
```

```
z 10,150
z 11,151
z 40,0
z 56,56
z 107,0
z 133,10
z 110,0
z 121,15
z 120,40            /100
z 161,42
z 162,47
z 163,44
z 164,45
z 165,41
z 166,46
z 167,74
z 170,76
z 171,136
z 141,0
z 142,012
z 143,0 3
z 144,0
z 145,0
z 146,0
z 147,100
z 150,77
z 151,123
z 122,124
z 123,125
z 124,126
z 125,127
z 126,130
z 127,131
z 130,132
z 131,0
z 157,75
z 156,17
z 155,16
z 111,11
z 140,0
z 102,137
z 61,112
z 62,113
z 63,114
z 64,115
z 65,116
z 66,117
z 67,120
z 70,121
z 71,122
z 41,0
z 42,0
z 43,53
z 44,135
z 45,134
z 46,133
z 47,0
z 50,101
z 51,102
z 22,103
```

```
   z 23,104
   z 24,105
   z 25,106
   z 26,107
   z 27,110
   z 30,111
   z 31,0
→ z 157,52
   z 156,0
→ z 155,10
→ z 103,0
   z 76,15
```

```
rbl=400
dimension rbf(rbl)

rset,       0
            dap rsr
            dac ac
            dio io
            rar 6s
            add (400-rbl
            dac ddp
            dzm rpc
            lac (isp rpc
            dac rfg         /flag for end of file
            law rbf+rbl-1   /last word of buffer
            dap inp
            law rbf
            mta
            lio (340
            law 40
rsr,        ivk .
            hlt
            lac ac
            rar 6s
            add rbf 32
            sub (20000
            dac dne
            lio io
            lac ac
            jmp i rset
```

```
readch,     0
            dac ac¯
rfg,        isp rpc          /becomes jmp rp2 after end of file
            jmp inp
            law i 3
            dac rpc
            idx inp
            dap inq
            sas (lio rbf+rbl
            jmp 9k
            lw rbl
            adm ddp
            lia
            ral 6s
            dap rdr
            law rbf
            dap inp
            dap inq
            mta
            law rbl
rdr,        ivk
            hlt
            ZAP
9k,         sub (lio rbf
            add ddp
            sad dne
            jmp rp3
inp,        lio
            cla
            rcl 6s
inq,        dio
            sas (77
            sad (13
            dzm rpc
rp1,        lia
            lac ac
            jmp i readch

rp3,        lac (jmp rp2
            dac rfg
rp2,        lw 14
            jmp rp1
```

```
ofl=400
dimension ofb(ofl)

wset,       0
            dac ac
            dap wrdr
            mta 300
            nop
            sni
            bpt
            rar 6s
            dac wfld
            add (400-ofl
            dac odp
            law i 3
            dac cp
            law ofb
            dap ofp
            dap ofq
            lac ac
            jmp i wset
```

```
writec,     0
            dio 95p
            dac ac
            law 77
            A∧IA
            sad (14
            jmp clean
            dac 96p
            lac (lio ofb+ofl
            sas ofp
            jmp ofp-1
            law ofl          /write out buffer
            adm odp
            lia
            and (770000
            sas odp
            jmp wo5
            ral 6s
            mta 300
            nop
            sni
            bpt
            jmp .+2
wo5,        ral 6s
            dap wrd
            law ofb
            dap ofp
            dap ofq
            mta
            law ofl 20
wrd,        ivk
            hlt
ofp-1,      lac 96p
ofp,        lio
            rar 6s
            rcl 6s
ofq,        dio
            isp cp
            jmp ou3
            law i 3
            dac cp
            idx ofp
            dap ofq
wox,        lac ac
            lio 95p
            jmp i writec

ou3,        lac 96p
            sas (77
            sad (13
            jmp ofp
            jmp wox
```

```
clean,      law 13
            lia
            sas 96p
            jdp writec
            lac ofp
            add odp
            sub wfld
            add (20000+ofl-[lio ofb]
            dac b77
            lw ofb+ofl
            dap ofp
            jdp writec        /write out last buffer
            law b77-32
            mta
            lio (340
            law 60
wrdr,       ivk .
            hlt
            jmp cr
b77,        0
            20400

constants
epeat ifm 2100-.,printx /lose/77//
variables
```

```
5000/
ini,        ZAP
in,         lw 600
            dap inn
            lac (add 100
            lio (40000
            ivk 16
            hlt
inn,        law
            dap wd
            sza
            jmp rdd
            jmp cr-1
constant
```

```
define w a,b,c,d,e,f,g
a×20000 b×400 c×10 [d∧34]>4
[d∧3]×200000 e×4000 f×100 g×2
termin

6000/
dir,        repeat 134,0
            w 36,36,36,36,36,36,36
            repeat 2×10,0
            w 33,0,0,0,0,0,0
            repeat 2×77,0
            repeat 10,w 36,36,36,36,36,36,36
            -0
            675756

repeat ifn .-6400,printx /loses/77//

buf=7000

start ini
```