ORKUSTOFNUN
Jarðhitadeild

FORRIT TIL ÚRVINNSLU BORHOLUGAGNA

II

Listun forrita

Ingvar Þór Magnússon

OS 85032/JHD-08 B Maí 1985

ORKUSTOFNUN
Grensásvegi 9, 108 Reykjavík

Verknúmer : 330-315

FORRIT TIL ÚRVINNSLU BORHOLUGAGNA

II

Listun forrita

Ingvar Þór Magnússon

OS 85032/JHD-08 B Maí 1985

Fjölritunarstofa
Daníels Halldórssonar

EFNISYFIRLIT

TÖFLUR

MYNDIR

1 INNGANGUR

Í þessari skýrslu eru útprentanir eða listanir af forritum, sem voru skrifuð á borholujarðfræðideild Orkustofnunar árið 1984, einnig eru birtar DCL-skipanaskrár, sem eru notaðar til að vekja upp forritin. Forritin eru skrifuð á málinu FORTRAN-77 fyrir VAX 11/750 tölvu Orkustofnunar og er notkun þeirra lýst í fyrra bindi skýrslunnar.

2 HVAR ERU FORRITIN VARÐVEITT

Forritin eru varðveitt á disklingum á borholujarðfræðideild OS en höfundur geymir einnig samhljóða afrit af þeim (sjá töflu 1).

Tafla 1 Disklingar

Disklingur	Lýsing
BJ8401	DCL skipanaskrár
BJ8402	innsláttar- og almenn forrit
BJ8403	teikniforritið LOGPLOT
BJ8404	forritasafnið IMLIBRARY

Keyrslukódar forritanna (image files) eru á notendanúmeri osdisk1:<jd330314.exe>, en skipanaskrárnar eru á <jd330314.com> og koma nöfn þeirra fram í köflunum hér á eftir.

Forritasafnið IMLIBRARY geymir þýddan kóda (object code) SUBROUTINE, FUNCTION og ENTRY undirforrita, sem eru tengd við móðurforrit eða kallforrit með LINK skipuninni:

```
$ LINK forrit,.....,osdisk1:<jd330314>imlibrary/lib,....
```

LINK skipunin leitar í forritasafninu og sækir undirforritin, sem móðurforritin þurfa á að halda. Nánari lýsing á forritasafninu IMLIBRARY er í fyrri hluta skýrslunnar.

Vel á minnst; forritin LEGEND og MINERAL lesa samnefndar .DAT skrár á <jd330314.datafiles>, en þær eru lyklar að nöfnum jarðlaga og ummyndunarsteinda (sjá töflur 2 og 3). Ef þessar skrár uppfylla ekki kröfur notenda um nafngiftir geta þeir haft eigið kerfi í skrám LEGEND.DAT og MINERAL.DAT á eigin efnisskrá (directory).

Tafla 2 Legend.dat

no	jarðlagagerð
1	Fersklegt fín-meðalkorna basalt
2	Ummyndað fín-meðalkorna basalt
3	Fersklegt meðal-grófkorna basalt
4	Ummyndað meðal-grófkorna basalt
5	Dólerít innskot
6	Gabbró innskot
7	Fersklegt glerjað basalt
8	Ummyndað glerjað basalt
9	Basaltrík breksía
10	Túff
11	Súrt fínkornótt berg
12	Súrt grófkornótt berg
13	Ísúrt fínkornótt berg
14	Ísúrt grófkornótt berg
15	Fínkornótt set
16	Grófkornótt set
17	Sjávarset
18	Óákveðið
19	Svarf vantar
20	

Tafla 3 Mineral.dat

no	ummyndunarsteindir
1	Kalsít
2	Aragónít
3	Ópall
4	Kalsedón
5	Kvars
6	Kabasít
7	Tomsónít
8	Analsím
9	Skólesít/Mesólít
10	Stilbít
11	Heulandít
12	Mordenít
13	Laumontít
14	Wairakít
15	Prenít
16	Epidót
17	Wollastónít
18	Aktínólít
19	Hornblendi
20	Granat
21	Albít
22	K-feldspat
23	Anhýdrít
24	Sphen
25	Fe-oxíð
26	Pýrít
27	Pyrrhótít
28	Koparkís
29	Smektít
30	Blandlagsleir
31	Svellandi-klórít
32	Klórít
33	Illít
34	Önnur lagsílíköt

3 SKIPANASKRÁR

```
$ ! jens.com er skipanaskra borholujardfraedinga
$ ! im 1984
$
$ verification = f$verify()
$ set noverify
$
$ on  control_y   then   $ goto byrjun
$ on  warning     then   $ exit
$
$ byrjun:
$ Ðosdisk1:<jd330314.com>cleanup.com
$ type sys$input




      ADALVALMYND


                1  sla inn gogn
                2  nota almenn forrit
                3  nota teikni forrit
                4  gera annad


$ inquire val "       valnumer "
$
$ if  val .eqs. "1"  then  $ Ðosdisk1:<jd330314.com>innsla.com
$ if  val .eqs. "2"  then  $ Ðosdisk1:<jd330314.com>almenn.com
$ if  val .eqs. "3"  then  $ Ðosdisk1:<jd330314.com>teikna.com
$ if  val .eqs. "4"  then  goto  haetta
$ if  val .eqs. ""   then  goto  haetta
$
$ goto byrjun
$
$ haetta:
$ Ðosdisk1:<jd330314.com>cleanup.com
$ if  verification  then  $ set verify
$ exit
```

```
$ ! innsla.com er skipanaskra borholujardfraedinga
$ ! keyrir innslattar forrit
$ ! im 1984
$
$ Ðosdisk1:<jd330314.com>cleanup.com
$ type sys$input
```

INNSLATTUR GAGNA

1	jardlagasnid	LIT	4	borhradi	BOR
2	vatnsleidarar	AQU	5	log	LOG
3	steindir	MIN	6	merki os	NEA

```
$ inquire val "       valnumer "
$
$ Ðosdisk1:<jd330314.com>cleanup.com
$ assign/user_mode  sys$command sys$input
$
$ if  val .eqs. "1"  then  $ run osdisk1:<jd330314.exe>innlit
$ if  val .eqs. "2"  then  $ run osdisk1:<jd330314.exe>innaqu
$ if  val .eqs. "3"  then  $ run osdisk1:<jd330314.exe>innmin
$ if  val .eqs. "4"  then  $ run osdisk1:<jd330314.exe>innbor
$ if  val .eqs. "5"  then  $ run osdisk1:<jd330314.exe>innlog
$ if  val .eqs. "6"  then  $ run osdisk1:<jd330314.exe>innnea
$
$ exit
```

```
$ ! almenn.com er skipanaskra borholujardfraedinga
$ ! keyrir almenn forrit
$ ! im 1984
$
$ Ðosdisk1:<jd330314.com>cleanup.com
$ type sys$input
```

ALMENN FORRIT

1	finna mork fyrir dypi og maeligildi	BOUND
2	breyta gomlu jardlagasnidi i nytt	GAXIM
3	gera urdratt ur skra med steindanofnum	MINERAL

```
$ inquire val "       valnumer "
$
$ Ðosdisk1:<jd330314.com>cleanup.com
$ assign/user_mode  sys$command sys$input
$
$ if  val .eqs. "1"  then  $ run osdisk1:<jd330314.exe>bound
$ if  val .eqs. "2"  then  $ run osdisk1:<jd330314.exe>gaxim
$ if  val .eqs. "3"  then  $ run osdisk1:<jd330314.exe>mineral
$ if  val .lts. "1"  .or. val .gts. "3" then $  exit
$
$ type sys$input
0J
23;1H
$ ! <ESC>Þ0J     hreinsa skjainn   !
$ ! <ESC>Þ23;1H  lina 23 dalkur 1  !
$ inquire-
  val "                                          <ret> til ad halda afram  "
$ exit
```

```
$ ! teikna.com er skipanaskra borholujardfraedinga
$ ! keyrir teikniforrit
$ ! im 1984
$
$ Ðosdisk1:<jd330314.com>cleanup.com
$ type sys$input
```

TEIKNIFORRIT

1	borholumaelingar	LOGPLOT
2	snidskyringar	LEGEND
3	texti	TEXTPLOT
4	merki orkustofnunar	OSMERKI

```
$ inquire val "       valnumer "
$
$ if  val .ges. "1"  .and. val .les "4"  then  $ Ðosdisk1:<jd330314.com>pltaeki
$ dev = f$logical("PL_") - "PL_"
$
$ Ðosdisk1:<jd330314.com>cleanup.com
$ assign/user_mode  sys$command sys$input
$
$ if  val .eqs. "1"       then $ run osdisk1:<jd330314.exe>logplot
$ if  val .eqs. "2"       then $ run osdisk1:<jd330314.exe>legend
$ if  val .eqs. "3" .and. dev .eqs. "TEX"  then-
                               $ run osdisk1:<jd330314.exe>textplot1
$ if  val .eqs. "3" .and. dev .nes. "TEX"  then-
                               $ run osdisk1:<jd330314.exe>textplot2
$ if  val .eqs. "4"       then $ run osdisk1:<jd330314.exe>osmerki
$
$ exit
```

```
$ ! pltaeki.com er skipanaskra borholujardfraedinga
$ ! velur teiknitaeki
$ ! im 1984
$
$ Ðosdisk1:<jd330314.com>cleanup.com
$ type sys$input


$ plotter = "      TEIKNITAEKI  " + f$logical("PL_") - "PL_"
$ write  sys$output  plotter
$
$ type sys$input

                1  Hewlett Packard 7475     HP7475
                2  Hewlett Packard 7550     HP7550
                3  Hewlett Packard 7585     HP7585
                4  Houston Hiplot           HOU
                5  Tektronix 4663           TEX
                6  Visual 550 JHD           VIS
                7  Visual 550 VOD           VIS1

$ inquire val "       valnumer "
$
$ if  val .eqs. "1"  then  $  define  pl_  pl_hp7475
$ if  val .eqs. "2"  then  $  define  pl_  pl_hp7550
$ if  val .eqs. "3"  then  $  define  pl_  pl_hp7585
$ if  val .eqs. "4"  then  $  define  pl_  pl_hou
$ if  val .eqs. "5"  then  $  define  pl_  pl_tex
$ if  val .eqs. "6"  then  $  define  pl_  pl_vis
$ if  val .eqs. "7"  then  $  define  pl_  pl_vis1
$
$ exit
```

```
$ ! cleanup.com hreinsar skja
$
$ type sys$input
1;1H
2J
$ ! <ESC>Þ1;1H  heim    !
$ ! <ESC>Þ2J    hreinsa !
$ exit
```

4 INNSLÁTTARFORRIT

```
      program innlit

* specification and introduction * ..................................

*     undirforrit: peep

      character*1  answ
      character*72 file, line, text
      character*32 litform1 / '(t4,f9.2,t14,f9.2,t28,i2)' /
      character*32 litform2 / '(t4,f9.2,         t28,a)'  /
      character*32 litform3 / '(t4,f9.2,t14,f9.2,t28,a)'  /

      write(6,'(/,10(/,a),/)')
     £ '   program innlit audveldar innslatt jardlagasnids ',
     £ '                                                   ',
     £ '   Jardlagasnidsskra inniheldur upplysingar um :   ',
     £ '                                                   ',
     £ '       1  jardlagaskipan       4  kronugerd        ',
     £ '       2  athugasemdir         5  alag             ',
     £ '       3  fodringar                                ',
     £ '                                                   ',
     £ '     i fyrsta saeti inntakslinu taknar textalinu.  ',
     £ '   Til ad haetta er slegid a ..... CTRL-Z  '

* open outfile * ...................................................

      ios = 1
      do while ( ios.gt.0 )
        write(6,'(a,$)')            '   nafn a .LIT skra  '
        read(5,'(a)',iostat=ios)        file
        if (file.eq.' ' .or. ios.lt.0)  call exit
        open(unit=10,file=file,status='new',iostat=ios,
     £    defaultfile='.lit',carriagecontrol='list')
      enddo

* lithology * ......................................................

      write(10,'(a)')     '*  JARDLAGASKIPAN'
      write(6,'(/,a,$)')  '   viltu jardlagaskipan  <ret>=ja  '
      call echo ( answ, 1, iq, 0 )
      if ( ichar(answ) .eq. 26 )    goto 900   ! ctrl - z
      if ( ichar(answ) .ne. 13 )    goto 200   ! not <return>

      ios = 1
      do while (ios.ne.0)
        write(6,'(a,$)') '   byrjunardypi          '
        read(5,'(f9.0)',iostat=ios)    x1
      enddo

      write(6,'(a,/)')
     £ '   sladu inn lokadypi , jardlagagerd  CTRL-Z til ad haetta  '

      do while(.true.)

        write(6,'(f9.2,a,$)')    x1, '   '
        read(5,'(q,a)',end=200)  iq, line

        if ( line(1:1) .eq. '_' ) then
```

```
            write(10,'(a)') line(1:iq)
          else
            read(line(1:iq),'(f9.0,i2)',iostat=ios)        x2, k
            if ( line(1:1) .lt.'0' .or. line(1:1).gt.'9')  ios = 1
            if ( x2.gt.3000. .or. ( k.lt.1 .or. k.gt.20 )) ios = 1
            if ( ios .eq. 0 ) then
              write(10,litform1) x1, x2, k
*             x1 = x2
            else
              call peep (4)
              write(6,'(a)') '                villa '
            endif
          endif

        enddo

* explanations * ..................................................

200     write(10,'(a)')     '*  ATHUGASEMDIR'
        write(6,'(/,a,$)')  '   viltu athugasemdir  <ret>=ja  '
        call echo ( answ, 1, iq, 0 )
        if ( ichar(answ) .eq. 26 )    goto 900   ! ctrl - z
        if ( ichar(answ) .ne. 13 )    goto 300

        write(6,'(a,/)')
    £   '   sladu inn dypi , athugasemd  CTRL_Z til ad haetta  '

        do while(.true.)

          write(6,'(a,$)') '  > '
          read(5,'(q,a)',end=300)   iq, line

          if ( line(1:1) .eq. '_' ) then
            write(10,'(a)') line(1:iq)
          else
            read(line(1:iq),'(f9.0,q,a)',iostat=ios)  x1, iq, text
            if ( x1.gt.3000. )                        ios = 1
            if ( line(1:1) .lt.'0' .or. line(1:1).gt.'9') ios = 1
            if ( ios .eq. 0 ) then
              write(10,litform2) x1, text(1:iq)
            else
              call peep (4)
              write(6,'(a)') '                villa '
            endif
          endif

        enddo

* casings * ......................................................

300     write(10,'(a)')     '*  FODRINGAR'
        write(6,'(/,a,$)')  '   viltu fodringar  <ret>=ja  '
        call echo ( answ, 1, iq, 0 )
        if ( ichar(answ) .eq. 26 )    goto 900   ! ctrl - z
        if ( ichar(answ) .ne. 13 )    goto 400
```

```
      write(6,'(a,/)')
     £ '   sladu inn upphafsdypi, lokadypi , fodringar '//
     £ '   CTRL_Z til ad haetta '

      do while(.true.)

        write(6,'(a,$)') '  > '
        read(5,'(q,a)',end=400)   iq, line

        if ( line(1:1) .eq. '_' ) then
          write(10,'(a)') line(1:iq)
        else
          read(line(1:iq),'(2f9.0,q,a)',iostat=ios)  x1, x2, iq, text
          if ( line(1:1) .lt.'0' .or. line(1:1).gt.'9')    ios = 1
          if ( x1.gt.3000. .or. x2.gt.3000 .or. x1.ge.x2 ) ios = 1
          if ( ios .eq. 0 ) then
            write(10,litform3) x1, x2, text(1:iq)
          else
            call peep (4)
            write(6,'(a)') '                villa '
          endif
        endif

      enddo

* drill bit * ..........................................................

400   write(10,'(a)')     '*  KRONUGERD '
      write(6,'(/,a,$)')  '   viltu kronugerd  <ret>=ja  '
      call echo ( answ, 1, iq, 0 )
      if ( ichar(answ) .eq. 26 )    goto 900   ! ctrl - z
      if ( ichar(answ) .ne. 13 )    goto 500

      write(6,'(a,/)')
     £ '   sladu inn upphafsdypi, lokadypi , kronugerd '//
     £ '   CTRL_Z til ad haetta '

      do while(.true.)

        write(6,'(a,$)') '  > '
        read(5,'(q,a)',end=500)   iq, line

        if ( line(1:1) .eq. '_' ) then
          write(10,'(a)') line(1:iq)
        else
          read(line(1:iq),'(2f9.0,q,a)',iostat=ios)  x1, x2, iq, text
          if ( line(1:1) .lt.'0' .or. line(1:1).gt.'9')    ios = 1
          if ( x1.gt.3000. .or. x2.gt.3000 .or. x1.ge.x2 ) ios = 1
          if ( ios .eq. 0 ) then
            write(10,litform3) x1, x2, text(1:iq)
          else
            call peep (4)
            write(6,'(a)') '                villa '
          endif
        endif

      enddo
```

```
* drill weight * ..................................................

500     write(10,'(a)')     '*  ALAG '
        write(6,'(/,a,$)')  '   viltu alag  <ret>=ja  '
        call echo ( answ, 1, iq, 0 )
        if ( ichar(answ) .ne. 13 )    goto 900

        Write(6,'(a,/)') '   sladu inn upphafsdypi, lokadypi , alag '

        do while(.true.)

          write(6,'(a,$)') '  > '
          read(5,'(q,a)',end=900)   iq, line

          if ( line(1:1) .eq. '_' ) then
            write(10,'(a)') line(1:iq)
          else
            read(line(1:iq),'(2f9.0,q,a)',iostat=ios)  x1, x2, iq, text
            if ( line(1:1) .lt.'0' .or. line(1:1).gt.'9')    ios = 1
            if ( x1.gt.3000. .or. x2.gt.3000 .or. x1.ge.x2 ) ios = 1
            if ( ios .eq. 0 ) then
              write(10,litform3) x1, x2, text(1:iq)
            else
              call peep (4)
              write(6,'(a)') '                villa '
            endif
          endif

        enddo

* end section * ..................................................

900     call exit
        end
```

```
      program innaqu

* specification and introduction * ..................................

*     undirforrit: peep

      character*72 file, line, text
      character*32 aquform  /'(a,t4,f9.2,t28,a)'/
      character*1  nodd

      write(6,'(/,51(''-''))')
      write(6,'(a)') '   program innaqu audveldar innslatt vatnsleidara '
      write(6,'(51(''-''),/)')

* open infile * ....................................................

      ios = 1
      do while (ios.gt.0)
        write(6,'(a,$)')            '   nafn a .AQU skra  '
        read(5,'(a)',iostat=ios)        file
        if (file.eq.' ' .or. ios.lt.0)  call exit
        open(unit=10,file=file,status='unknown',access='append',
     £    iostat=ios,defaultfile='.aqu',carriagecontrol='list')
      enddo

* main section * ...................................................

      write(6,'(6(/,a))')
     £ '   sla inn:  dypi og texta  - textanum ma sleppa.  ',
     £ '     i fyrsta saeti inntakslinu taknar textalinu.  ',
     £ '   fyrst er spurt um fjolda orvarodda, sem settir  ',
     £ '   eru a orvarnar i teikningu sama spurning kemur  ',
     £ '   a ny ef slegid er a -1 eda adra negativa tolu.  ',
     £ '   til ad haetta er slegid a ..... CTRL-Z  '

100   write(6,'(/,a,$)')   '   fjoldi odda (<ret>=1)  '
      read(5,'(q,a)',end=100)  iq, nodd
      if ( iq.eq.0 )           nodd = '1'
      if ( nodd.lt.'0' .or. nodd.gt.'9' )  goto 100
      write(6,'(a)')

      do while(.true.)

        write(6,'(a,$)') '   dypi , texti  '
        read(5,'(q,a)',end=900)   iq, line

        if ( line(1:1) .eq. '_' ) then
          write(10,'(a)') line(1:iq)
        else
          read(line(1:iq),'(f9.0,q,a)',iostat=ios)  x, iq, text
          if ( ios.eq.0 .and. x .lt.0. )               goto 100
          if ( ios.eq.0 .and. line(1:1) .eq.'-' )      goto 100
          if ( line(1:1) .lt.'0' .or. line(1:1).gt.'9') ios = 1
          if ( ios .eq. 0 .and. x.lt. 5000. ) then
            write(10,aquform)  nodd, x, text(1:iq)
          else
            call peep (4)
            write(6,'(a)') '                villa '
```

```
            endif
          endif

        enddo

* end section * ..................................................

900     call exit
        end
```

```
      program innmin

* specification * ..................................................

*     undirforrit: peep
*                  num_string

      character*72 file, line
      character*32 minform /'(i1,t4,f9.2,t16,15i4)'/
      character*32 form1   /'(f9.0,15i4)'/
      character*32 form2   /'(15i4)'/
      integer      mineral(15)
      logical      userx

* open outfile * ...................................................

      write(6,'(/,51(''-''))')
      write(6,'(a)') '   program INNMIN audveldar innslatt steinda '
      write(6,'(51(''-''),/)')

      ios = 1
      do while ( ios .gt. 0 )
        write(6,'(a,$)')            '   nafn a .MIN skra  '
        read(5,'(a)',iostat=ios)        file
        if (file.eq.' ' .or. ios.lt.0)  call exit
        open(unit=10,file=file,status='unknown',access='append',
     £    iostat=ios,defaultfile='.min',carriagecontrol='list')
      enddo

* depth * ..........................................................

      dx_dumm = 2.00
      idummy  = 6

100   write(6,'(2(/,a),f9.2,4(/,a))')
     £  '   Sla inn :           upphafsdypi og billengd      ',
     £  '   billengd ma sleppa ma sleppa ef = ',    dx_dumm   ,
     £  '   Sama spurning byrtist a ny ef slegid er a -      ',
     £  '   Ef svarad er med  <return> verdur notandi        ',
     £  '   ad sla inn dypid asamt numerum steindanna        ',
     £  '   Til ad haetta er slegid a ........ CTRL-Z        '

      ios = 1
      do while ( ios.gt.0 )
        write(6,'(a,$)') '   Upphafsdypi og billengd ..................    '
        read(5,'(q,2f7.0)',iostat=ios)   iq, x, dx
        if ( ios.lt.0 )  call exit
      enddo

      userx = .false.
      if ( iq.eq.0 ) userx = .true.
      if ( dx  .eq. 0. )  dx = dx_dumm
      dx_dumm = dx
```

```
* method * ..........................................................

        ios    =  1
        method = -1
        write(6,'(a)')

        do while ( ios.gt.0 )
          write(6,'(a,i1,/,a,$)')
     £    '   Greiningar_adferd  <return> = ', idummy,
     £    '   annars:   3 = þunnsneid   4 = XRD   6 = svarf  '
          read(5,'(q,i1)',iostat=ios)   iq, method
          if ( ios.lt.0 )  call exit
        enddo

        if ( iq.eq.0 ) method = idummy
        idummy = method

* main section * ....................................................

        if ( userx ) then

          write(6,'(3(/,a),/)')
     £    '   Sla inn:  dypi og steind(ir) -  komma a milli   ',
     £    '     i fyrsta saeti inntakslinu taknar textalinu   ',
     £    '   Til ad haetta ..................... CTRL-Z   '

          do while ( .true. )

            write(6,'(a,$)')      '   Dypi , steind(ir)  '
            read(5,'(q,a)',end=900)   iq, line

            minerals = num_string ( line(1:iq), ',' )
            nblank   = num_string ( line(1:iq), ' ' )

            if ( line(1:1) .eq. '_' ) then
              write(10,'(a)') line(1:iq)
            else if ( line(1:1).eq.'-' .and. iq.eq.1 ) then
              goto 100
            else if ( nblank.eq.0.and.minerals.gt.0.and.minerals.le.15 ) then
              read(line(1:iq),form1,iostat=ios) x, (mineral(i),i=1,minerals)
              if ( ios.eq.0 ) then
                do i = 1, minerals
                  if ( mineral(i).eq.0 .or. abs(mineral(i)).ge.1000.) ios = 1
                enddo
                if ( ios.eq.0 ) then
                  write(10,minform)  method, x, (mineral(i),i=1,minerals)
                else
                  call peep (4)
                endif
              else
                  call peep (4)
              endif
            else
              call peep (4)
            endif
          enddo
        else
```

```
      write(6,'(4(/,a),/)')
     £    '   Sla inn:  steind(ir)     -      komma a milli  ',
     £    '   <return> ef engin steind er a vidkomandi dypi  ',
     £    '     i fyrsta saeti inntakslinu taknar textalinu  ',
     £    '   Til ad haetta ..... CTRL-Z  '

      do while ( .true. )

        iq = 0
        do while ( iq.eq.0 )
          write(6,'(f9.2, a,$)')    x,  '    '
          read(5,'(q,a)',end=900)   iq, line
          if ( iq .eq. 0 )          x = x + dx
        enddo

        minerals = num_string ( line(1:iq), ',' ) + 1
        nblank   = num_string ( line(1:iq), ' ' )

        if ( line(1:1) .eq. '_' ) then
          write(10,'(a)') line(1:iq)
        else if ( line(1:1).eq.'-' .and. iq.eq.1 ) then
          goto 100
        else if ( iq.gt.0 .and. nblank .eq. 0 .and.
     £              minerals .gt. 0 .and. minerals.le.15 ) then
          read(line(1:iq),form2,iostat=ios) (mineral(i),i=1,minerals)
          if ( ios.eq.0 ) then
            do i = 1, minerals
              if ( mineral(i).eq.0 .or. abs(mineral(i)).ge.1000.) ios = 1
            enddo
            if ( ios.eq.0 ) then
              write(10,minform)  method, x, (mineral(i),i=1,minerals)
              x = x + dx
            else
              call peep (4)
            endif
          else
            call peep (4)
          endif
        else
          call peep (4)
        endif
      enddo
    endif

* end section * ..................................................

900   call exit
      end
```

```
      program innbor

* specification and introduction * ..................................

*     subroutine: peep

      character*32 file
      character*80 line
      character*32 borform  / '(f9.2,'' , '',f9.2)' /

      write(6,'(/,47(''-''))')
      write(6,'(a)') '   program innbor audveldar innslatt borhrada '
      write(6,'(47(''-'')/)')

* open infile * ....................................................

      ios = 1
      do while (ios.gt.0)
        write(6,'(a,$)') '   nafn a BOR_hrada skra    '
        read(5,'(a)',iostat=ios)   file
        if (file.eq.' ' .or. ios.lt.0 )  call exit
        open(unit=10,file=file,status='unknown',access='append',
     £    iostat=ios,carriagecontrol='list',defaultfile='.bor')
      enddo

* main section * ...................................................

      dx_dumm  = 1.0            ! dummy billengd

100   ios = 1
      write(6,'(a)')
      do while ( ios .gt. 0 )
        write(6,'(a,$)')  '   upphafsdypi og billengd  '
        read(5,'(2f7.0)',iostat=ios)  x, dx
      enddo

      if ( ios .lt. 0 )   call exit
      if ( dx  .eq. 0. )  dx = dx_dumm
      dx_dumm = dx

      write(6,'(2(/,a),f9.2,4(/,a),/)')
     £  '   Sla inn :      maeligildi,lokadypi hvers bils.  ',
     £  '   Lokadypi ma sleppa ef billengd er    ',     dx   ,
     £  '     i fyrsta saeti inntakslinu taknar textalinu.  ',
     £  '   Spurningin um upphafsdypi og billengd birtist   ',
     £  '   a ny ef slegid er a -1 eda adra negativa tolu.  ',
     £  '   Til ad haetta er slegid a ..... CTRL-Z  '
```

```
      do while (.true.)

        write(6,'(2f9.2,''      '',$)')    x, x + dx
        read(5,'(q,a)',iostat=ios)   iq, line

        if ( ios.lt. 0 ) then

*         write(10,borform) x ,   vv
          write(10,borform) x , -999.00
          call exit

        else if ( line(1:1) .ne. '_' ) then

          read(line(1:iq),'(2f7.0)',iostat=ios)  v, xx
          if ( ( line(1:1) .lt. '0'  .or. line(1:1) .gt. '9' )
     £         .and. ( line(1:1) .ne. '-' ) )          ios = 1
          if ( ios.eq.0 .and. v.lt.0. )  then
            write(10,borform) x        ,  vv
            write(10,borform) x + dx   , -999.00
            goto 100
          else if ( ios .eq. 0 .or. iq .eq. 0 ) then
            if ( iq .eq. 0  ) v  = vv
            if ( xx .eq. 0. ) xx = x + dx
            write(10,borform,iostat=ios) x, v
            vv = v
            x  = xx
          else
            call peep (4)
            write(6,'(a)') '                villa '
          endif

        else

          write(10,'(a)') line(1:iq)

        endif

      enddo

      end
```

```
      program innlog

* specification and introduction * ..................................

*     undirforrit: peep

      character*32 file
      character*80 line
      character*32 logform / '(f9.2,'' , '',f9.2)' /

      write(6,'(/,47(''-''))')
      write(6,'(a)') '   program innlog audveldar innslatt maelinga '
      write(6,'(47(''-'')/)')

* open infile * ....................................................

      ios = 1
      do while (ios.gt.0)
        write(6,'(a,$)') '   nafn a LOG skra    '
        read(5,'(a)',iostat=ios)   file
        if (file.eq.' ' .or. ios.lt.0 ) goto 900
        open(unit=10,file=file,status='unknown',access='append',
     £    iostat=ios,carriagecontrol='list',defaultfile='.log')
      enddo

* main section * ...................................................

100   write(6,'(/,a,$)')  '   upphafsdypi og billengd  '
      read(5,'(2f16.0)',err=100,end=900)  x, dx
      if ( dx .eq. 0. )  dx = 1.0

      write(6,'(2(/,a),f9.2,5(/,a),/)')
     £ '   Sla inn :      maeligildi,lokadypi hvers bils.  ',
     £ '   Lokadypi ma sleppa ef billengd   = ',      dx    ,
     £ '    _ i fyrsta saeti inntakslinu taknar textalinu.  ',
     £ '    _ i fyrsta saeti taknar og eydu i maeligildum.  ',
     £ '   Spurningin um upphafsdypi og billengd birtist   ',
     £ '   a ny ef slegid er a -1 eda adra negativa tolu.  ',
     £ '   Til ad haetta er slegid a ..... CTRL-Z  '
```

```
      do while (.true.)

        write(6,'(f9.2,''    '',$)')       x
        read(5,'(q,a)',end=900)   iq, line

        if ( line(1:1) .ne. ' ' ) then
          read(line(1:iq),'(2f9.0)',iostat=ios)          y, xx
*         if ( ios.eq.0 .and. y.lt.0. )                  goto 100
          if ( line(1:1) .lt.'0' .or. line(1:1).gt.'9') ios = 1
          if ( ios .eq. 0 ) then
            if ( xx .eq. 0. ) xx = x + dx
            write(10,logform,iostat=ios2) x, y
            if ( ios2 .eq. 0 ) then
              x  = xx
            else
              call peep (4)
              write(6,'(a)') '                 villa '
            endif
          else
            call peep (4)
            write(6,'(a)') '                villa '
          endif
        else
          write(10,'(a)') line(1:iq)
        endif

      enddo

* end section * ..................................................

900   call exit
      end
```

```
      program innnea

*                   skrifar gogn i .nea skra

*     undirforrit: lib$put_screen
*                  lib$erase_line
*                  lib$erase_page
*                  nonotify
*                  noecho
*                  echo
*                  outf
*                  getf

      real*4         x_os        /  1.50 /
      real*4         y_os        /  1.50 /
      real*4         os_hgt      /  1.00 /
      real*4         os_rot      / 90.00 /
      character*72   text1       / 'JHD-BJ-9000 IM' /
      character*72   text2       / '84.09.1001   T' /

      character*1    answ
      character*32   neaform     /'(a,t9,f9.2,t22,f9.2,t43,a)'/
      character*64   file
      integer*4      iq1         / 14 /
      integer*4      iq2         / 14 /

100   call nonotify ( ' Merki Orkustofnunar ',8,1,0)

      call lib$erase_page (9,1)
      call lib$put_screen ('  nafn a .NEA skra  '    ,16,10,)

      read(5,'(a)',iostat=ios)  file
      if  ( ios.lt.0 .or. file.eq.' ')   call exit
      open(10,file=file,status='new',err=100,
     £ carriagecontrol='list',defaultfile='.nea')

      do while ( .true. )

        call lib$erase_page(9,1)
        call outf          (' 1) x_hnit          ',     x_os          ,13,1,)
        call outf          (' 2) y_hnit          ',     y_os          ,14,1,)
        call outf          (' 3) haed OS merkis  ',     os_hgt        ,15,1,)
        call outf          (' 4) snuningshorn    ',     os_rot        ,16,1,)
        call lib$put_screen(' 5) efri  lina i merki  '//text1(1:iq1) ,17,1,)
        call lib$put_screen(' 6) nedri lina i merki  '//text2(1:iq2) ,18,1,)
        call lib$put_screen('    <ret> = engin breyting  '           ,21,1,)
        call noecho (answ,1,iq,0)
```

```
  if ( answ .eq. '1' )  then
    call getf ('    x hnit  ',              x_os      ,21,1,1)
  else if ( answ .eq. '2' )  then
    call getf ('    y hnit  ',              y_os      ,21,1,1)
  else if ( answ .eq. '3' )  then
    call getf ('    haed OS merkis i cm  ', os_hgt    ,21,1,1)
  else if ( answ .eq. '4' )  then
    call lib$put_screen ('    snuningshorn positivt'  ,21,1,1)
    call getf (' fra x as teiknara  ',      os_rot    ,21,26,1)
  else if ( answ .eq. '5' )  then
    call lib$erase_line (21,1)
    call lib$put_screen ('    efri lina i merki  ',,,1)
    call echo(text1,72,iq1,0)
  else if ( answ .eq. '6' )  then
    call lib$erase_line (21,1)
    call lib$put_screen ('    nedri lina i merki  ',,,1)
    call echo(text2,72,iq2,0)
  else if ( ichar(answ) .eq. 13 )  then
    write(10,neaform) '_', x_os,   y_os,   text1(1:iq1)
    write(10,neaform) '_', os_hgt, os_rot, text2(1:iq2)
    close(unit=10)
    call lib$erase_page (1,1)
    call exit
  endif

enddo
end
```

5 ALMENN FORRIT

```
        program bound

* specification and introduction * ..................................

*       subroutine:  echo

        character*40 file, line
        character*1  answ
        logical      error
        real*8       x, y
        real*8       x_min, x_max
        real*8       y_min, y_max, y_sum, count

        write(6,'(/,62(''-''))')
        write(6,'(a)') '  program to find bounds for x and y      '
        write(6,'(a)') '  and calculate the mean value of y       '
        write(6,'(a)') '  y values less or equal to 0 are ignored '
        write(6,'(62(''-''),/)')

* open printfile * ..................................................

        open(unit=9,file='bound.lis',status='new',
     £  carriagecontrol='list',dispose='delete',iostat=ios)
        if ( ios.ne.0 ) call exit

* open infile * .....................................................

        write(9,'(62(''-''),/)')
100     ios = 1
        do while (ios.ne.0)
          write(6,'(a,$)')    '   infile          t> = stop   '
          read(5,'(a)',iostat=ios)  file
          if ( file.eq.' ' .or. ios.lt.0 ) goto 900
          open(unit=11,file=file,status='old',readonly,iostat=ios)
        enddo

* initalize * .......................................................

        error = .false.
        x_min =  1.E35
        y_min =  1.E35
        x_max = -1.E35
        y_max = -1.E35
        y_sum =  0.
        count =  0.

* main section * ....................................................

        do while ( .true. )

          read(11,'(a)',end=200) line
```

```
          if ( line(1:1).ne.' ' ) then
            read(line,'(2f16.0)',iostat=ios) x, y
            if ( ios.gt.0 ) then
              if ( .not.error ) write(6,'(a)')
              if ( .not.error ) write(9,'(a)')
              write(6,'(t4,a)') line
              write(9,'(t4,a)') line
              error = .true.
            else if ( y .gt. 0.0 ) then
              x_min  =  min(x,x_min)
              x_max  =  max(x,x_max)
              y_min  =  min(y,y_min)
              y_max  =  max(y,y_max)
              y_sum  =  y_sum + y
              count  =  count + 1.
            endif
          endif

        enddo

* output * ..................................................

200     if ( error .or. count.eq.0.) then
          write(6,'(/,a,//,62(''-''),/)') '   error in file '// file
          write(9,'(/,a,//,62(''-''),/)') '   error in file '// file
        else
          write(6,'(/,2(a,f9.2),/,3(a,f9.2),//,62(''-''),/)')
     £    '   x_min',x_min,'       y_min',y_min,
     £    '   x_max',x_max,'       y_max',y_max,
     £                    '       y_mean', y_sum/count
          write(9,'(t4,a)') file
          write(9,'(/,2(a,f9.2),/,3(a,f9.2),//,62(''-''),/)')
     £    '   x_min',x_min,'       y_min',y_min,
     £    '   x_max',x_max,'       y_max',y_max,
     £                    '       y_mean', y_sum/count
        endif

        close ( unit=11 )
        goto 100

* end section * .................................................

900     write(6,'(a,$)') '   want printfile  t> = no     '
        call echo (answ,1,iq,1)

        if ( ichar(answ) .ne. 13) then
          close(unit=9,dispose='print/delete')
          write(6,'(/,a,/)') '   printfile  now  in queue "TXA0:" '
        endif

        call exit
        end
```

```
      program gaxim

* specification and introduction * ..................................

*     subroutine: err

      character*72 file, line
      character*32 litform1  / '(3x,f9.2,x,f9.2,5x,i2)' /
      character*32 litform2  / '(3x,f9.2,15x,a)'        /
      character*32 litform3  / '(3x,f9.2,x,f9.2,5x,a)'  /
      logical      error     /.false./
      logical      err

      write(6,'(/,32(''-''),/,a)') '  program gaxim breytir           '
      write(6,'(a)')               '       .JSN skra (inntak HOLPLOT) '
      write(6,'(a,/,32(''-''),/)') '  i    .LIT skra (inntak LOGPLOT) '

* open infile * ....................................................

      ios = 1
      do while (ios.gt.0)
        write(6,'(a,$)')             '  .JSN innskra  '
        read(5,'(a)',iostat=ios)        file
        if (file.eq.' ' .or. ios.lt.0)  call exit
        open(unit=11,file=file,status='old',iostat=ios,
     £    defaultfile='.jsn',readonly)
      enddo

* open outfile * ...................................................

      ios = 1
      do while (ios.gt.0)
        write(6,'(a,$)')             '  .LIT utskra   '
        read(5,'(a)',iostat=ios)        file
        if (file.eq.' ' .or. ios.lt.0)  call exit
        open(unit=12,file=file,status='new',iostat=ios,
     £    defaultfile='.lit',carriagecontrol='list')
      enddo
      write(6,'(a)')

* read and write lithology * .......................................

      write(12,'(a)') '* JARDLAGASKIPAN'

      read(11,'(q,a)',end=900)   iq, line
      if(line(1:4).eq.'9999')    goto 200
      if(line(iq:iq).eq.' ')     error = err ( line )

      read(line(1:iq),'(f9.0,i)',iostat=ios)  x1, k1
      if (ios.gt.0)              error = err ( line )
```

```
        do while(.true.)

          read(11,'(q,a)',end=900)  iq, line
          if(line(1:4).eq.'9999')   goto 200
          if(line(iq:iq).eq.' ')    error = err ( line )

          read(line(1:iq),'(f9.0,i)',iostat=ios)  x2, k2
          if (ios.gt.0)              error = err ( line )

          if (.not.error) write(12,litform1,iostat=ios)  x1, x2, k1
          if (ios.gt.0)    error = err ( line )
          if (.not.error)  x1 = x2
          if (.not.error)  k1 = k2

        enddo

* read and write explanations * ....................................

200     write(12,'(a)') '* ATHUGASEMDIR'

        do while(.true.)

          read(11,'(q,a)',end=900)        iq, line
          if(line(1:4).eq.'9999')         goto 300

          read(line,'(f9.0)',iostat=ios)   x
          if (ios.gt.0)  error = err ( line )

          read(11,'(q,a)',end=900)  iq, line
          if (line(1:4).eq.'9999')  error = err ( line )

          if (.not.error) write(12,litform2,iostat=ios)  x, line(1:iq)
          if (ios.gt.0)   error = err ( line )

        enddo

* read and write casings * .........................................

300     write(12,'(a)') '* FODRINGAR'

        do while(.true.)

          read(11,'(q,a)',end=900)  iq, line
          if(line(1:4).eq.'9999')   goto 400

          read(line(1:iq),'(2f9.0)',iostat=ios)  x1, x2
          if (ios.gt.0)             error = err ( line )

          read(11,'(q,a)',end=900)  iq, line
          if(line(1:4).eq.'9999')   error = err ( line )

          if (.not.error) write(12,litform3,iostat=ios) x1, x2, line(1:iq)
          if (ios.gt.0)   error = err ( line )

        enddo

* read and write drill_bit * .......................................
```

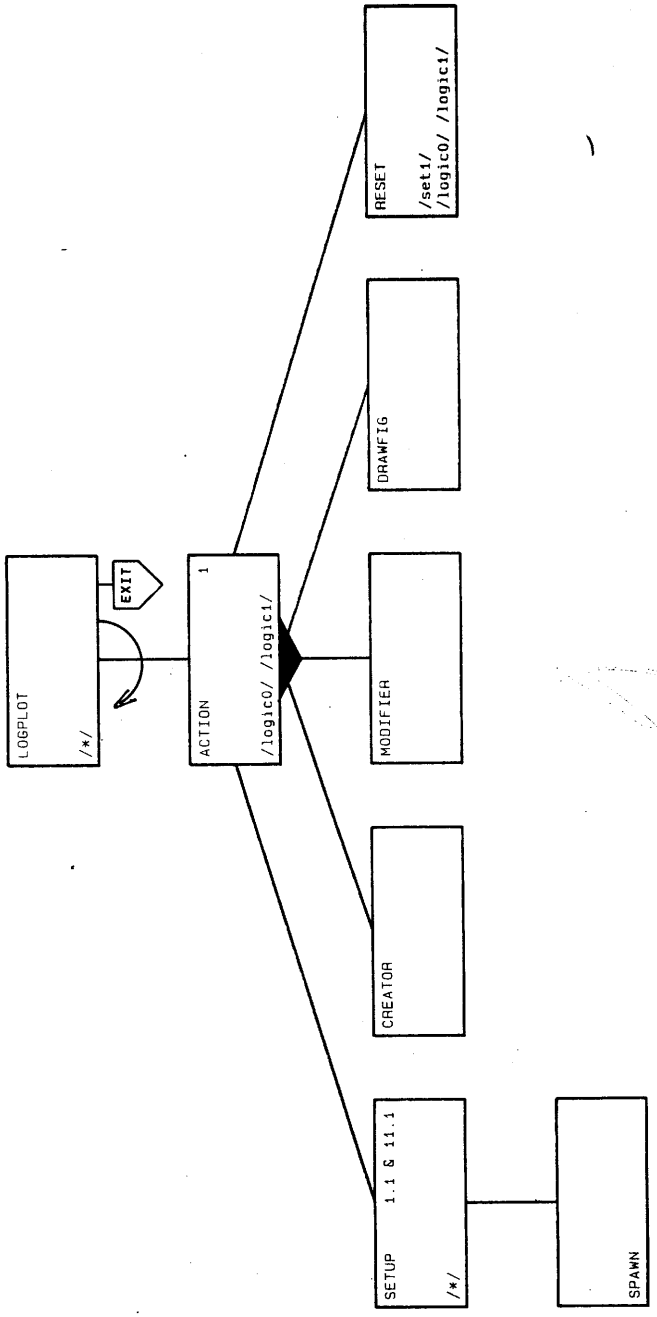
```
400     write(12,'(a)') '* KRONUGERD'

        do while(.true.)

          read(11,'(q,a)',end=900)  iq, line
          if(line(1:4).eq.'9999')   goto 500

          read(line(1:iq),'(2f9.0)',iostat=ios)  x1, x2
          if (ios.gt.0)             error = err ( line )

          read(11,'(q,a)',end=900)  iq, line
          if(line(1:4).eq.'9999')   error = err ( line )

          if (.not.error) write(12,litform3,iostat=ios)  x1, x2, line(1:iq)
          if (ios.gt.0)   error = err ( line )

        enddo

* read and write drill_weight * ....................................

500     write(12,'(a)') '* ALAG'

        do while(.true.)

          read(11,'(q,a)',end=900)  iq, line
          if(line(1:4).eq.'9999')   goto 900

          read(line(1:iq),'(2f9.0)',iostat=ios)  x1, x2
          if (ios.gt.0)             error = err ( line )

          read(11,'(q,a)',end=900)  iq, line
          if(line(1:4).eq.'9999')   error = err ( line )

          if (.not.error) write(12,litform3,iostat=ios)  x1, x2, line(1:iq)
          if (ios.gt.0)             error = err ( line )

        enddo

* end section * ...................................................

900     if ( error ) then
          write(6,'(/,a,/)') '  villa i inntaks_skra utskra eytt !! '
          close(unit=12,dispose='delete')
        endif

        call exit
        end
```

```
      program mineral

* specification and introduction * ..................................

*       subroutine: err

      integer*4      minios, min_dat(1000), iq_dat(1000)
      logical        error, err,  found(1000)
      character*80   file,  line, text_dat(1000)
      character*80   minform       / '(i1,t4,f9.0,t16,q,a)' /

      write(6,'(/,61(''-''),/,a,/,a,/,a,/,a,/,61(''-''),/ )')
     £ '   programid  MINERAL leitar að nofnum sem svara til numera  ',
     £ '   a steindum i .MIN skra.  Leitad er i skranni mineral.dat  ',
     £ '   finnst hun ekki er notud <jd330314.datafiles>mineral.dat  ',
     £ '   Utaksskrain er inntak i program LOGPLOT                   '

* open and read master file * ......................................

      open(10,file='mineral.dat',status='old',readonly,iostat=ios)

      if ( ios .gt. 0 ) then
          file = 'osdisk1:<jd330314.datafiles>mineral.dat'
          open(10,file=file,status='old',readonly,shared,iostat=ios)
          if (ios .gt. 0) stop ' villa:  mineral.dat finnst ekki '
          write(6,'(a,/,a,/)')
     £    '   skrain mineral.dat finnst ekki a efnisskranni, nu er ',
     £    '   lesin skrain osdisk1:<jd330314.datafiles>mineral.dat '
      else
          write(6,'(a,/)')
     £    '   skrain mineral.dat finnst a efnisskranni og er lesin '
      endif

      n_dat = 0
      ios   = 0

      do while ( ios .ge. 0 )
        read(10,'(q,a)',iostat=ios) iq, line
        if ( line(1:1) .ne. '_' .and. ios .eq. 0 )  then
          n_dat = n_dat + 1
          read(line(1:iq),'(i3,t6,q,a)',iostat=ios)
     £    min_dat(n_dat), iq_dat(n_dat), text_dat(n_dat)
          if ( ios .ne. 0 .or. n_dat .ge. 1000 ) stop '   villa i inntaki '
        endif
      enddo

* open .min file with mineral data * ...............................

      ios = 1
      do while ( ios .gt. 0 )
        write(6,'(a,$)')            '   .MIN innskra     '
        read(5,'(a)',iostat=ios)        file
        if (file.eq.' ' .or. ios.lt.0)  call exit
        open(11,file=file,status='old',iostat=ios,defaultfile='.min')
      enddo

* open outfile * ...................................................

      ios = 1
      do while ( ios .gt. 0 )
        write(6,'(a,$)')            '   .MIN uttaksskra  '
        read(5,'(a)',iostat=ios)        file
        if (file.eq.' ' .or. ios.lt.0)  call exit
        open(12,file=file,status='new',iostat=ios,
     £  defaultfile='.min',carriagecontrol='list')
      enddo
```

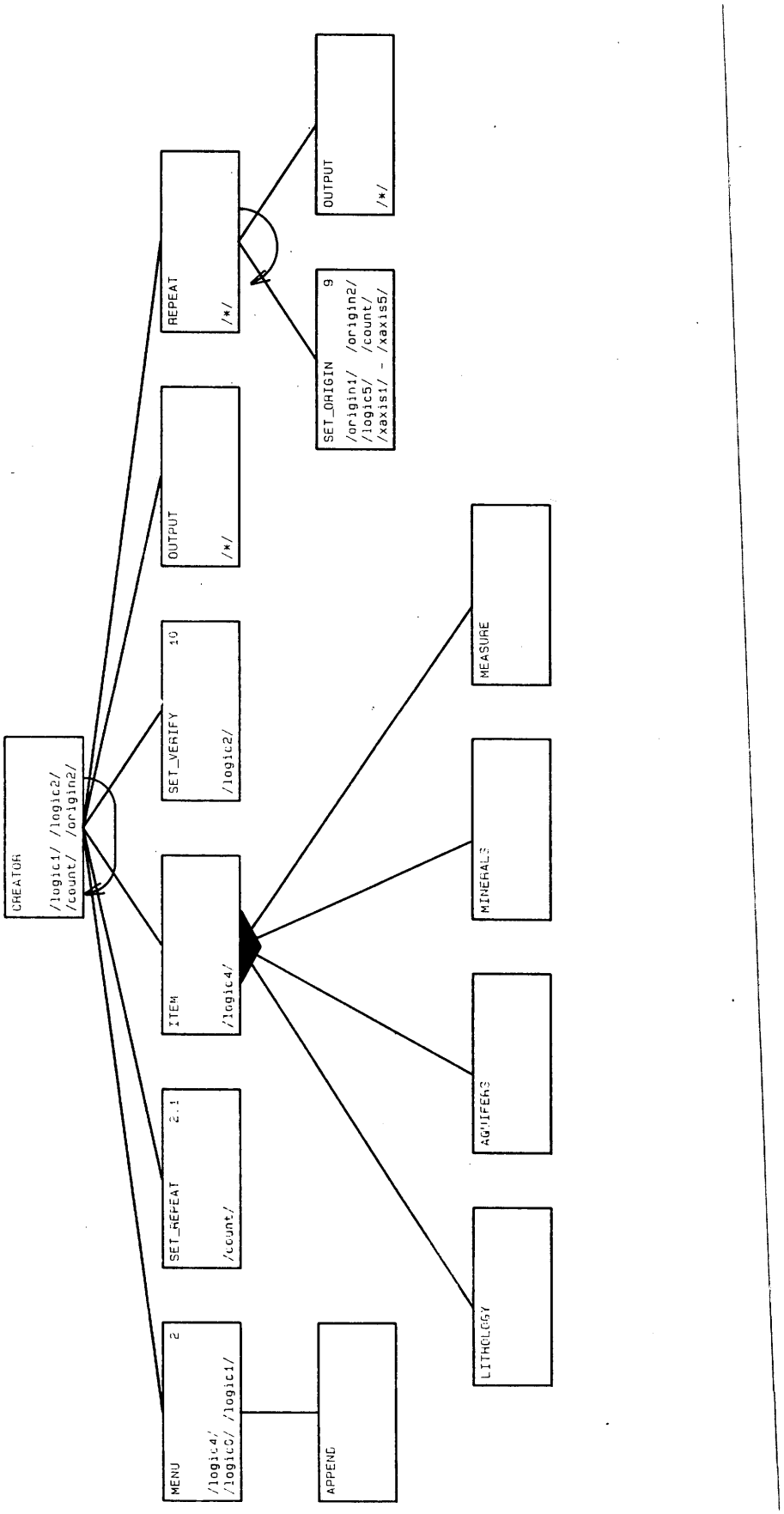
```
* main section * ..................................................

      do while ( minios .eq. 0 )

        read(11,'(q,a)',iostat=minios) iq, line

        if ( line(1:1) .ne. '_' .and. minios .eq. 0 )  then

          read(line(1:iq),minform,iostat=ios) method, x, iqq, line

          if ( mod(iqq,4) .ne. 0 )      ios = 1        ! error
          if ( line(iqq:iqq) .eq. ' ' ) ios = 1
          if ( iqq .gt. 60 )            ios = 1

          if ( ios .eq. 0 ) then                       ! search
            do i = 1, iqq/4
              read(line(4*i-3:4*i),'(i4)') minno
              minno = abs ( minno )
              do j = 1, n_dat
                if ( .not.found( minno ) .and. minno .eq. min_dat(j) ) then
                  found ( minno ) = .true.
                endif
              enddo
              if ( .not.found( minno ) ) then
                write(6,'(a,i3,a)') '   mineral no ', minno ,' finnst ekki '
              endif
            enddo
          else
            error = err ( line )
          endif

        endif

      enddo

* write outfile exit if error * ....................................

      if ( error ) then
        write(6,'(/,a,/)') '  villa i inntaks_skra utskra eytt !! '
        close (unit=12,dispose='delete')
      else
        do j = 1, n_dat
          if ( found (j) ) then
            write(12,'(i3,t6,a)') min_dat(j), text_dat(j)(1:iq_dat(j))
          endif
        enddo
        write(12,'(a)') '************'
        rewind ( unit=11 )
        do while ( ios .eq. 0 )
          read(11,'(q,a)',iostat=ios)        iq, line
          if( ios .eq. 0 )  write(12,'(a)')  line(1:iq)
        enddo
      endif

      call exit
      end
```

6 TEIKNIFORRIT

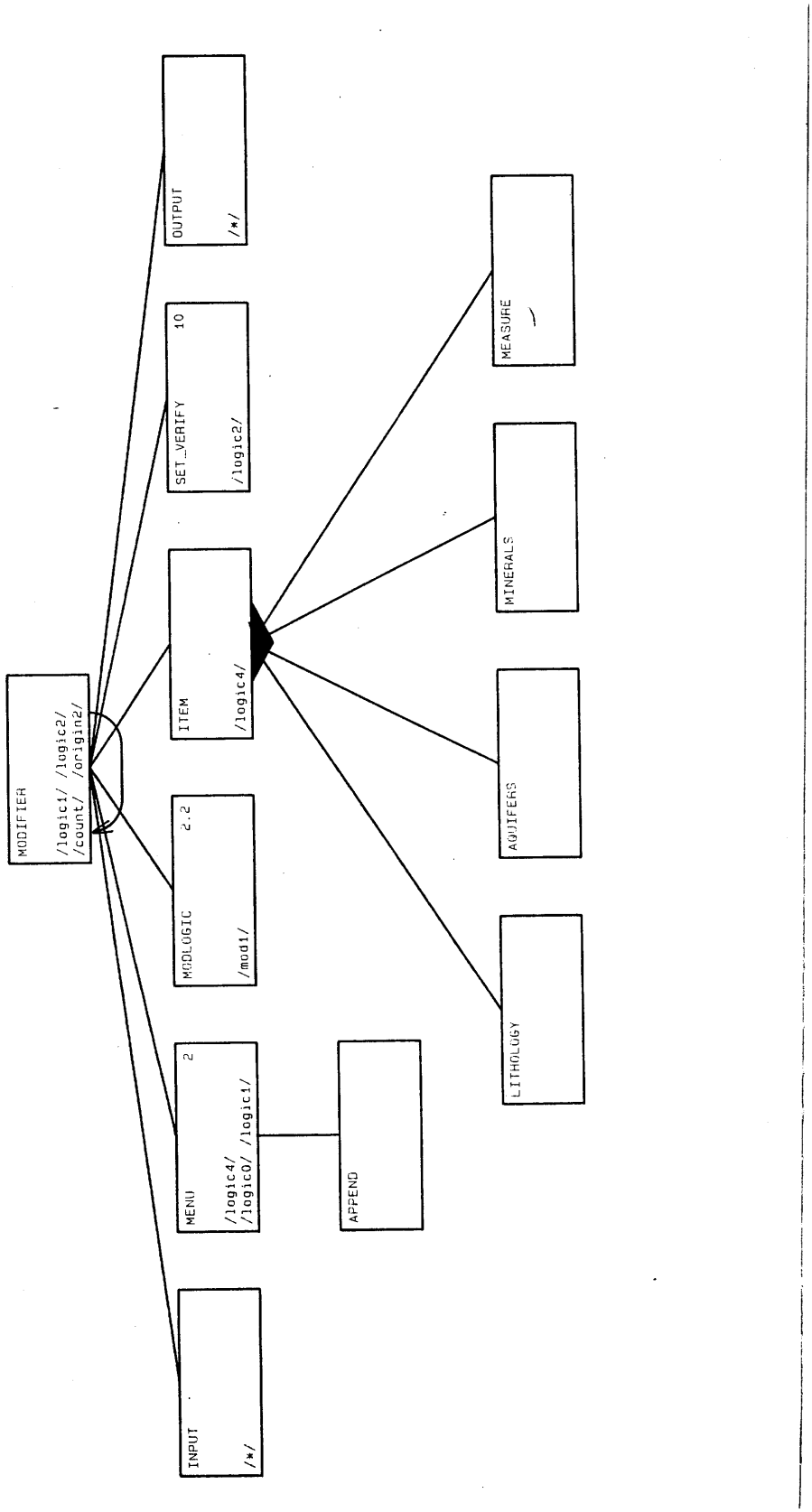
TEIKNIFORRITIÐ LOGPLOT



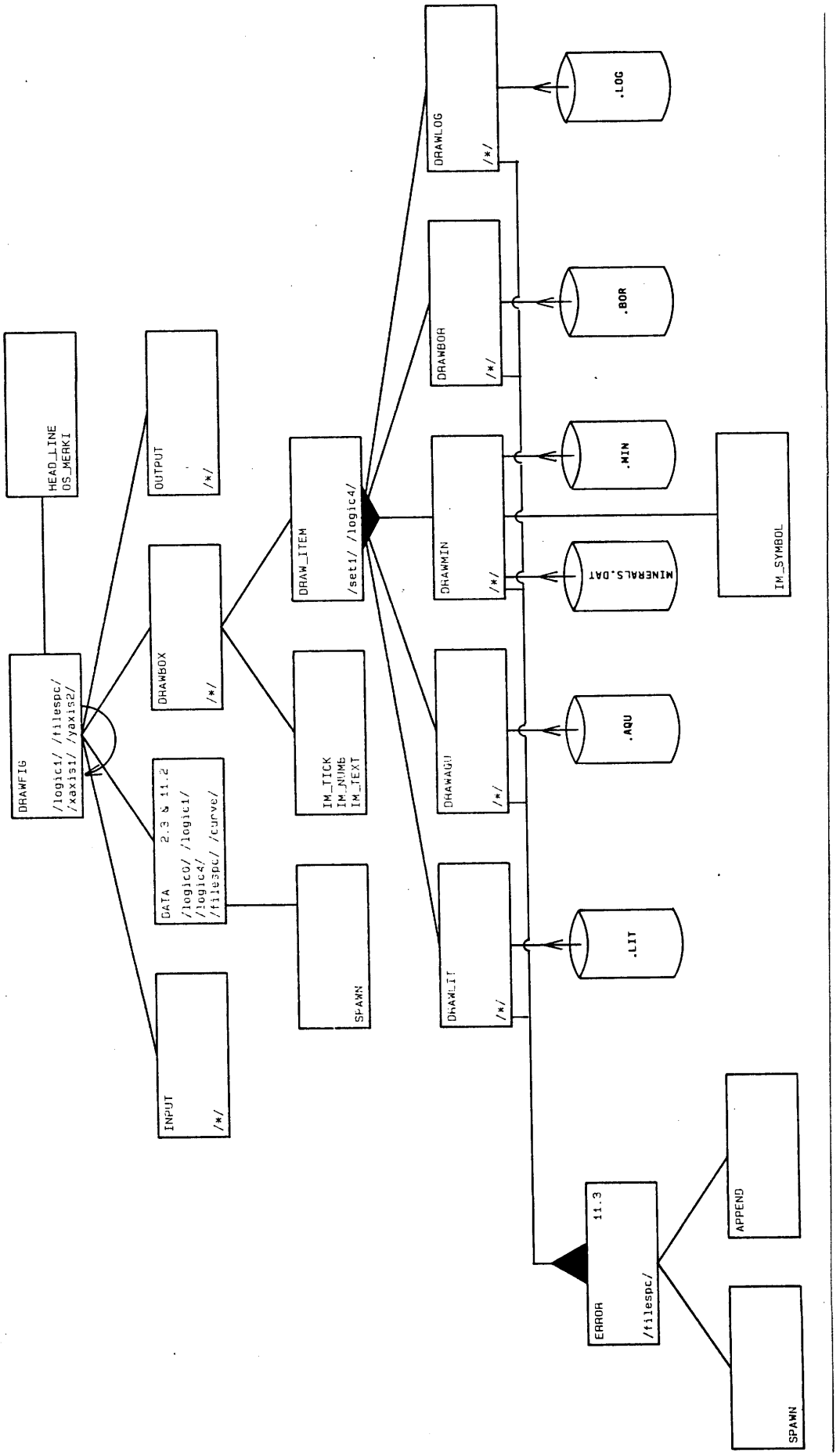
MYND 1. Trjámyndir forritsins LOGPLOT



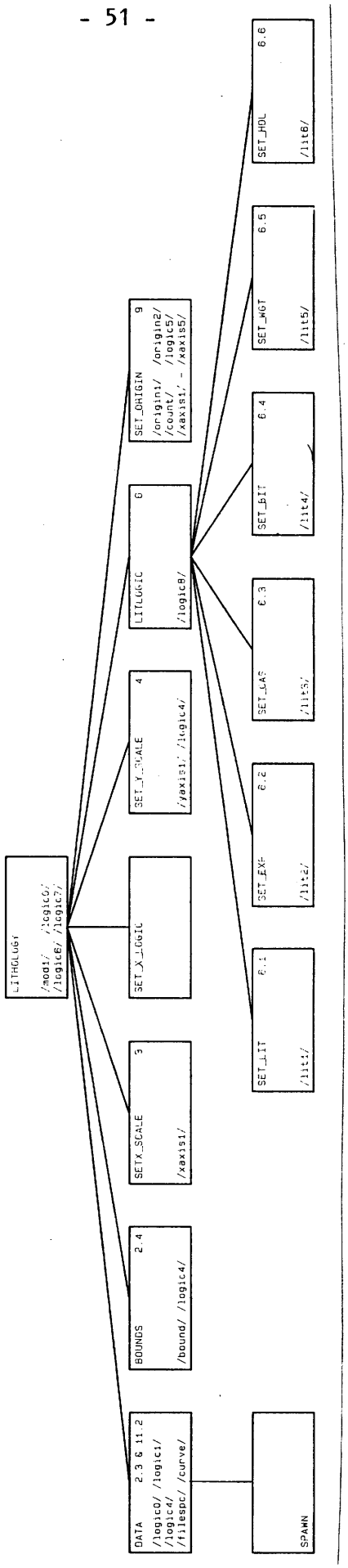
MYND 1. Trjámyndir forritsins LOGPLOT



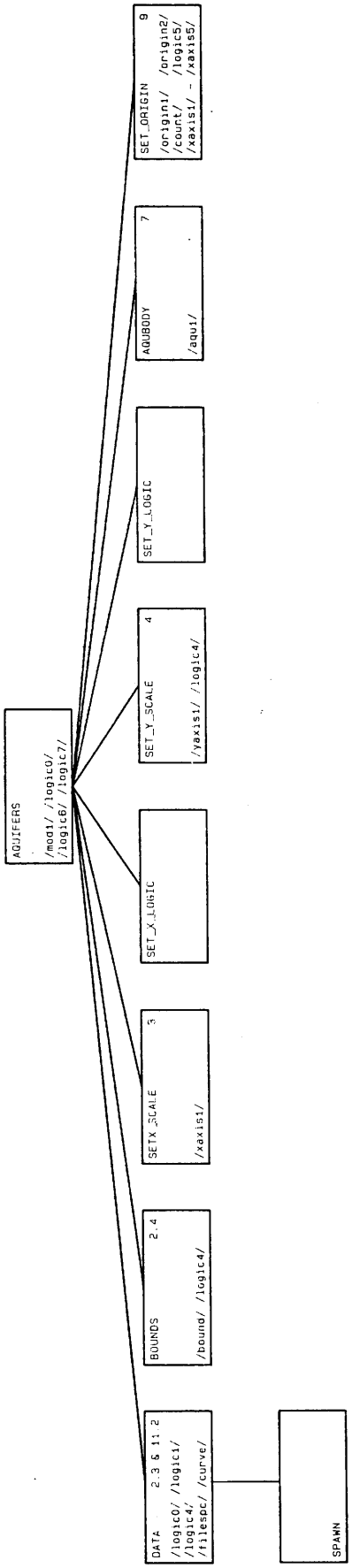
MYND 1. Trjámyndir forritsins LOGPLOT



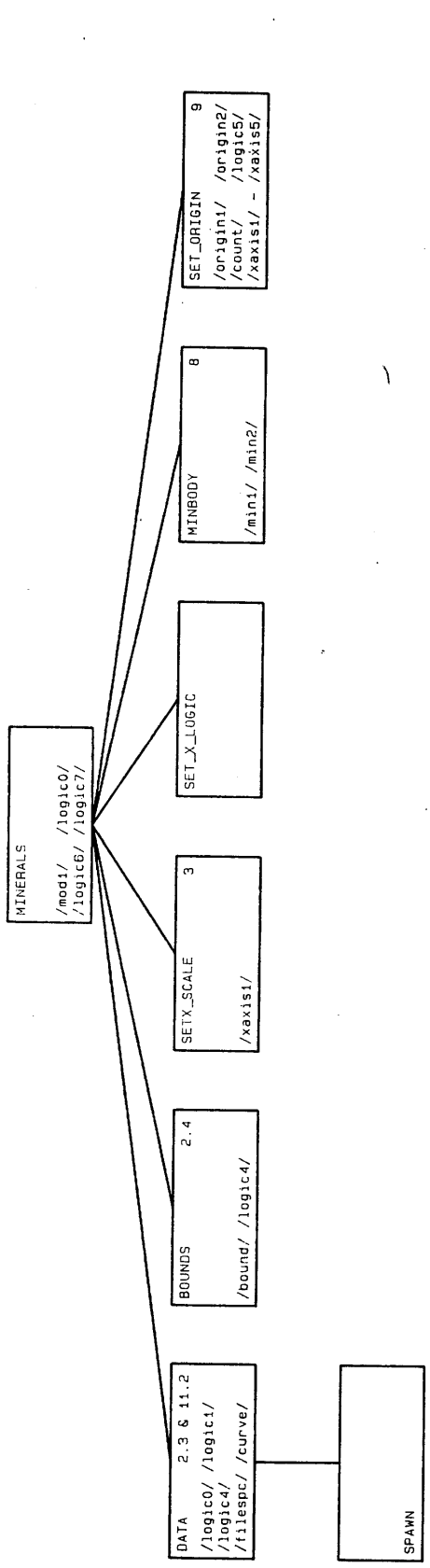
MYND 1. Trjámyndir forritsins LOGPLOT



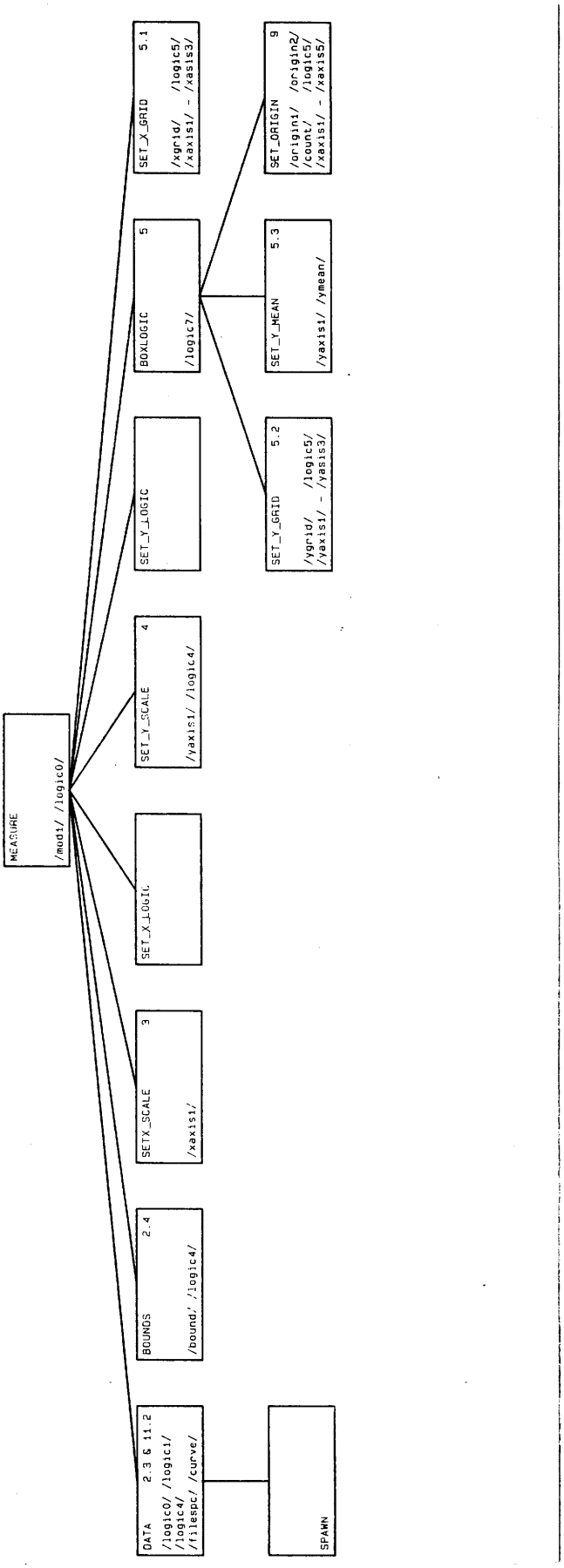
MYND 1. Trjámyndir forritsins LOGPLOT



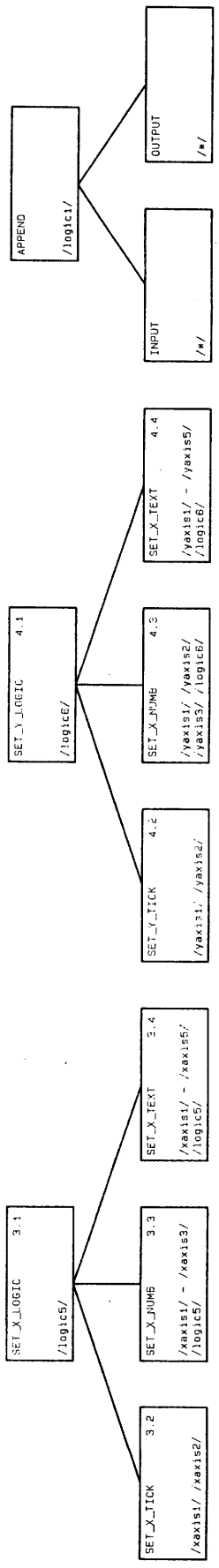
MYND 1. Trjámyndir forritsins LOGPLOT



MYND 1. Trjámyndir forritsins LOGPLOT



MYND 1. Trjámyndir forritsins LOGPLOT



MYND 1. Trjámyndir forritsins LOGPLOT

```
      program logplot

*     program to draw geological section, aquifers,
*     and distribution of secondary minerals,
*     drill rate and other borehole measurements.
*     figure is drawn via plotfile same plotfile
*     is created and can be modified by the program.
*     commonblk.for included in all main modules.

      include '<jd330314.logplot>commonblk.for/list'
      include '<jd330314.logplot>datvalues.for/list'

      call lib$erase_page (1,1)
      call esc6   ('  PROGRAM   LOGPLOT  ',11,15,)
      call esc6   ('  version.....1  ',12,15,)
      call esc6   ('  revision.....1  ',13,15,)
      call esc6   ('  Orkustofnun  1984  ',16,15,)
      call esc6   ('  - Jarðhitadeild -  ',17,15,)
c)    call esc6   ('  Ingvar  Magnusson  ',20,15,)
      call lib$set_cursor (24,1)
      call wait_s (0.75)

      do while ( .not.endofplot )
        call action
      enddo

      call lib$erase_page (1,1)
      call exit
      end
```

```
*       commonblk.for program logplot *.............................

        common /logic0/  create, modify, figure
        logical          create, modify, figure

        common /logic1/  endofplot, errorplot
        logical          endofplot, errorplot

        common /logic2/  store, verify
        logical          store, verify

        common /set1/    plotfile, plotter
        character*32     plotfile, plotter

        common /set2/    scale

        common /logic3/  icelandic
        logical          icelandic

        common /logic4/  draw_lit, draw_aqu, draw_min, draw_bor, draw_log
        logical          draw_lit, draw_aqu, draw_min, draw_bor, draw_log

        common /logic5/  draw_x_axis,               draw_x_line,
     £                   draw_x_tick,  draw_x_numb, draw_x_text
        logical          draw_x_axis,               draw_x_line,
     £                   draw_x_tick,  draw_x_numb, draw_x_text

        common /logic6/  draw_y_axis,  upside_down, draw_y_line,
     £                   draw_y_tick,  draw_y_numb, draw_y_text
        logical          draw_y_axis,  upside_down, draw_y_line,
     £                   draw_y_tick,  draw_y_numb, draw_y_text

        common /logic7/  draw_border,  draw_x_grid,
     £                   draw_y_grid,  draw_y_mean
        logical          draw_border,  draw_x_grid,
     £                   draw_y_grid,  draw_y_mean

        common /logic8/  draw_litlog, draw_explan, draw_casing,
     £                   draw_weight, draw_bittyp, draw_header
        logical          draw_litlog, draw_explan, draw_casing,
     £                   draw_weight, draw_bittyp, draw_header

        common /xaxis1/  box_length, x_first, x_last, x_scale
        real*4           box_length, x_first, x_last, x_scale

        common /xaxis2/  x_tickint, x_tackint, x_ticklen, x_tacklen
        real*4           x_tickint, x_tackint, x_ticklen, x_tacklen

        common /xaxis3/  x_numbint, x_hgtno, x_numbdist, ndec_x
        real*4           x_numbint, x_hgtno, x_numbdist
        integer*2        ndec_x

        common /xaxis4/  x_text
        character*40     x_text

        common /xaxis5/  x_textheight, x_textdist, x_nst
        real*4           x_textheight, x_textdist
        integer*2        x_nst
```

```
common /yaxis1/  box_width,  y_first, y_last, y_scale
real*4           box_width,  y_first, y_last, y_scale

common /yaxis2/  y_tickint, y_tackint, y_ticklen, y_tacklen
real*4           y_tickint, y_tackint, y_ticklen, y_tacklen

common /yaxis3/  y_numbint, y_hgtno, y_numbdist, ndec_y
real*4           y_numbint, y_hgtno, y_numbdist
integer*2        ndec_y

common /yaxis4/  y_text
character*40     y_text

common /yaxis5/  y_textheight, y_textdist, y_nst
real*4           y_textheight, y_textdist
integer*2        y_nst

common /xgrid/   x_gridint, x_gridlen
real*4           x_gridint, x_gridlen

common /ygrid/   y_gridint, y_gridlen
real*4           y_gridint, y_gridlen

common /ymean/   y_mean, mean_type
real*4           y_mean
integer*2        mean_type

common /origin1/ x0, y0
real*4           x0, y0

common /origin2/ y0_last
real*4           y0_last

common /filespc/ datafile, dataform
character*32     datafile, dataform

common /curve/   curve_type, ipen
integer*2        curve_type, ipen

common /bound/   x_min, x_max, x_cut, y_min, y_max
real*4           x_min, x_max, x_cut, y_min, y_max

common /lit1/    lit_mark, lit_raster, lit_omit
logical          lit_mark, lit_raster, lit_omit

common /lit2/    exp_height, exp_dist
real*4           exp_height, exp_dist

common /lit3/    cas_height, cas_dist, cas_space
real*4           cas_height, cas_dist, cas_space

common /lit4/    bit_height, bit_dist
real*4           bit_height, bit_dist

common /lit5/    wgt_height, wgt_dist
real*4           wgt_height, wgt_dist

common /lit6/    hdl_height, hdl_dist
real*4           hdl_height, hdl_dist
```

```
      common /aqu1/     aqu_height, aqu_dist
      real*4            aqu_height, aqu_dist

      common /min1/     min_int, min_tck, min_dis, min_hgt, min_rot
      real*4            min_int, min_tck, min_dis, min_hgt, min_rot

      common /min2/     min_symbhgt, min_brahgt, min_method
      real*4            min_symbhgt, min_brahgt
      integer*2         min_method

      common /mod1/     modify_data,    modify_bounds,
     £                  modify_x_scale, modify_x_logic,
     £                  modify_y_scale, modify_y_logic,
     £                  modify_body,    modify_origin
      logical           modify_data,    modify_bounds,
     £                  modify_x_scale, modify_x_logic,
     £                  modify_y_scale, modify_y_logic,
     £                  modify_body,    modify_origin

      common /count/    n_box, n_repeat
      integer*2         n_box, n_repeat

      character*1       answ
```

```
*       datvalues.for program logplot *.............................

        create        =  .true.    ! create plotfile .....  IF .TRUE.
        modify        =  .false.   ! modify plotfile
        figure        =  .false.   ! draw picture
        errorplot     =  .false.   ! if .true. error
        endofplot     =  .false.   ! if .true. call exit

        store         =  .true.    ! store setup in outfile
        verify        =  .false.   ! verify setup
        icelandic     =  .true.    ! character set icelandic  (else english)

        plotfile      = 'unknown'  ! plotfile
        plotter       = 'tex'      ! plotter   tex = tektronix 4663
        scale         =  1.0       ! scale for whole picture

        draw_lit      =  .true.    ! draw geological section
        draw_aqu      =  .false.   ! draw aquifers
        draw_min      =  .false.   ! draw minerals
        draw_bor      =  .false.   ! draw drill rate
        draw_log      =  .false.   ! draw log

        draw_x_axis   =  .true.    ! draw x_axis
        draw_x_line   =  .true.    ! draw line of x_axis
        draw_x_tick   =  .true.    ! controls whether x_tickmarks are drawn
        draw_x_numb   =  .true.    ! controls whether x_axis is numbered
        draw_x_text   =  .true.    ! controls whether x_text is drawn

        draw_y_axis   =  .true.    ! draw y_axis
        upside_down   =  .false.   ! direction of y_axis left to right
        draw_y_line   =  .true.    ! draw line of y_axis
        draw_y_tick   =  .true.    ! controls whether y_tickmarks are drawn
        draw_y_numb   =  .true.    ! controls whether y_axis is numbered
        draw_y_text   =  .true.    ! controls whether y_text is drawn

        draw_border   =  .false.   ! draw border of box
        draw_x_grid   =  .false.   ! draw horizontal lines through the box
        draw_y_grid   =  .false.   ! draw vertical lines through the box
        draw_y_mean   =  .false.   ! draw vertical line through box at fixed y

        draw_litlog   =  .true.    ! draw lithology
        draw_explan   =  .true.    ! draw explanations
        draw_casing   =  .true.    ! draw casing
        draw_bittyp   =  .true.    ! draw drill bit
        draw_weight   =  .true.    ! draw drill weight
        draw_header   =  .true.    ! draw headlines

        box_length    =  40.00     ! length in cm of the longer edge
        x_first       =   0.00     ! x value at origin of box
        x_last        = 200.00     ! x value at the other end of the box

        x_tickint     =   2.00     ! draw shorter tickmarks at this interval
        x_tackint     =  10.00     ! draw longer  tickmarks at this interval
        x_ticklen     =   0.15     ! length of shorter tickmarks on x_axis
        x_tacklen     =   0.30     ! length of longer  tickmarks on x_axis

        x_numbint     =  50.00     ! interval between numbers on x_axis
        x_hgtno       =   0.30     ! height of numbers on x_axis
        x_numbdist    =   0.50     ! distance between numbers and x_axis
```

```
ndec_x        =   -1      ! number of decimal digits in numbers

x_text        =  'Dýpi (m)'   ! text on x_axis
x_nst         =     9     ! number of characters in x_text
x_textdist    =  1.80     ! distance between text and axis
x_textheight  =  0.30     ! height of characters in x_text

box_width     =  6.00     ! length in cm of the shorter edge
y_first       =  0.00     ! y value at origin of box
y_last        = 30.00     ! y value at the other end of box

y_tickint     =  2.00     ! draw shorter tickmarks at this interval
y_tackint     = 10.00     ! draw longer  tickmarks at this interval
y_ticklen     =  0.15     ! length of shorter tickmarks on y_axis
y_tacklen     =  0.30     ! length of longer  tickmarks on y_axis

y_numbint     = 10.00     ! interval between numbers on y_axis
y_hgtno       =  0.30     ! height of numbers on y_axis
y_numbdist    =  0.50     ! distance between numbers and y_axis
ndec_y        =  -1       ! number of decimal digits in numbers

y_text        =  ' '      ! text on y_axis
y_nst         =   0       ! number of characters in x_text
y_textdist    =  1.20     ! distance between text and axis
y_textheight  =  0.30     ! height of characters in y_text

x_gridint     = 50.00     ! interval between x_grid_lines
x_gridlen     =  0.30     ! length of x_grid_tickmarks
y_gridint     = 10.00     ! interval between y_grid_lines
y_gridlen     =  0.30     ! length of y_grid_tickmarks

y_mean        =  0.00     ! mean value for y or other fixed y value
mean_type     =     0     ! type of line for y_mean  (0 = solid)

x0            =  8.00     ! origin of lower left corner of box in cm
y0            =  3.00     ! origin of lower left corner of box in cm
y0_last       =  0.00     ! right corner of previous box ( if any )

datafile   = 'unknown '   ! datafile
dataform   = '(2f16.0)'   ! default format for datafile
curve_type =        0     ! type of curve (0 = solid)
ipen       =        1     ! pen requested (1-9)      !  HP only

x_min        =     0.00   ! upper and lower bounds for x
x_max        = 99999.00
x_cut        = 99999.00   ! cut x axis at this value
y_min        =     0.01   ! upper and lower bounds for y
y_max        = 99999.00

lit_mark      =  .true.    ! mark upper and lower bounds of layers
lit_raster    =  .true.    ! generate rasters
lit_omit      = .false.    ! omit thin layers

exp_height    =  0.30      ! height of characters in explanations
exp_dist      =  0.15      ! distance between box and explanations
```

```
cas_height     =   0.30     ! height of characters in casings
cas_dist       =   0.50     ! distance between box and first casing
cas_space      =   0.12     ! distance between casings

bit_height     =   0.30     ! height of characters in drill bit
bit_dist       =   1.40     ! distance between box and drill bit

wgt_height     =   0.30     ! height of characters in drill weight
wgt_dist       =   2.20     ! distance between box and drill weight

hdl_height     =   0.30     ! height of characters in headlines
hdl_dist       =   1.20     ! dist. between top of box and headlines

aqu_height     =   0.30     ! height of characters in arrows
aqu_dist       =   0.20     ! distance between arrows and text

min_int        =   1.00     ! interval between minerals
min_tck        =   0.15     ! length of mineral tickmarks
min_dis        =   0.50     ! distance between mineral names and axis
min_hgt        =   0.30     ! height of characters in names
min_rot        =  45.00     ! rotation of mineral names
min_symbhgt    =   0.20     ! height of symbols
min_brahgt     =   0.25     ! height of brackets ( 0.0 = omit )
min_method     =     0      ! method  (0 = default)

n_box          =     1      ! number of current box
n_repeat       =     1      ! repeat current box n_repeat times

modify_data    = .false.    ! modification control
modify_bounds  = .false.
modify_x_scale = .false.
modify_x_logic = .false.
modify_y_scale = .false.
modify_y_logic = .false.
modify_body    = .false.
modify_origin  = .false.
```

```
      subroutine   action

*     purpose:     determine action ie create, modify, figure, endofplot

*     subroutines: lib$put_screen
*                  nonotify
*                  noecho
*                  setup
*                  reset
*                  creator
*                  modifier
*                  drawfig

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 1     ACTION  ' ,2,1,0)

      do while ( .true. )

        i1 = 0
        i2 = 0
        i3 = 0
        i4 = 0

        if ( create    )  i1 = 2
        if ( modify    )  i2 = 2
        if ( figure    )  i3 = 2
        if ( endofplot )  i4 = 2

        call lib$put_screen (' 1) create plotfile ',  15,40,i1)
        call lib$put_screen (' 2) modify plotfile ',  16,40,i2)
        call lib$put_screen (' 3) draw picture    ',  17,40,i3)
        call lib$put_screen (' 4) exit logplot    ',  18,40,i4)

        call lib$put_screen (' <ret> = no change  ',  21,43,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)          goto 900

        if ( answ .ge. '1' .and. answ .le. '4' ) then

          create    = .false.
          modify    = .false.
          figure    = .false.
          endofplot = .false.

          if ( answ .eq. '1' )   create    = .true.
          if ( answ .eq. '2' )   modify    = .true.
          if ( answ .eq. '3' )   figure    = .true.
          if ( answ .eq. '4' )   endofplot = .true.

        endif

      enddo

900   if ( .not.endofplot ) then
        call setup
        if ( create )  call creator
        if ( modify )  call modifier
        if ( figure )  call drawfig
        call reset
      endif

      return
      end
```

```
        subroutine   setup

*       purpose:     open old and new plotfiles
*                    if create .true. then reset all common variables
*                    else set common plotfile plotter scale language
*                    if figure .true. then call plots

*       subroutines: lib$sys_trnlog
*                    lib$set_logical
*                    lib$erase_page
*                    lib$erase_line
*                    lib$put_screen
*                    str$upcase
*                    lib$stop
*                    lib$get_lun
*                    lib$free_lun
*                    nonotify
*                    notify
*                    noecho
*                    outf
*                    getf
*                    spawn
*                    plots
*                    factor
*                    exit

        include '<jd330314.logplot>commonblk.for'
        character*32 name

* initialize *.....................................................

        if ( create ) then
          include '<jd330314.logplot>datvalues.for'
          goto 990
        endif

        n_box    = 1
        n_repeat = 1

        istat    = lib$sys_trnlog('PL_',,plotter,,,)
        plotter  = plotter(4:)

* main section *...................................................

100     call nonotify ('  FIG. 1.1   SETUP  ' ,2,1,0)

        do while ( .true. )

          call lib$erase_page   (3,1)
          call lib$put_screen   (' 1) (.plo) plotfile '//plotfile ,15,,)

          if ( figure ) then

            call lib$put_screen(' 2) plotter         '//plotter   ,16,,)
            call outf(          ' 3) scale      ', scale          ,17,,)
```

```
      if ( icelandic ) then
        call lib$put_screen(' 4) character set   icelandic ',18,,)
      else
        call lib$put_screen(' 4) character set   english   ',18,,)
      endif

    endif

    call lib$put_screen('    <ret> = no change  '           ,21,,)
    call noecho ( answ, 1, iq, 0 )
    if (ichar(answ).eq.13)   goto 900

    if ( modify .and. answ.ne.'1' ) answ = '1'

    ios = 1
    if ( answ .eq. '1' ) then

      do while (ios.ne.0)
        call lib$erase_line (21,1)
        call lib$put_screen (' (.plo) plotfile     '    ,21, 4,1)
        read(5,'(a)',iostat=ios) plotfile
      enddo

    else if ( answ .eq. '2' ) then

      call lib$put_screen(' hp7475  hewlett packard 7475  ',21,40,)
      call lib$put_screen(' hp7550  hewlett packard 7550  ',22,40,)
      call lib$put_screen(' hp7585  hewlett packard 7585  ',23,40,)
      call lib$put_screen(' hou     houston hiplot        ',21, 4,)
      call lib$put_screen(' tex     tektronix 4663        ',22, 4,)
      call lib$put_screen(' vis     visual 550 jhd        ',23, 4,)
      call lib$put_screen(' vis1    visual 550 vod        ',24, 4,)

      plotter = '   '
      do while ( ios.ne.0 )
        call lib$erase_line (24,41)
        read(5,'(a)',iostat=ios) plotter
      enddo

    else if ( answ .eq. '3' ) then

      call getf (' scale for whole picture  '  , scale ,21,4,1)

    else if ( answ .eq. '4' ) then

      icelandic = .not.icelandic

    endif

  enddo
```

```
* open plotfile and test if error *.....................................

900     open (unit=10,file=plotfile,status='old',
     £  iostat=ios,defaultfile='.plo',readonly)

        if ( ios .ne. 0 ) then

          call nonotify ('  FIG. 11.1  SETUP  ERROR    ',2,1,0)
          call lib$put_screen (' file '//plotfile        ,10, 4,)
          call lib$put_screen (' error in filename       ',12, 4,)
          call lib$put_screen (' or file not found       ',12,22,)
          call lib$put_screen (' you have 3 choices      ',15, 4,)
          call lib$put_screen (' 1  try again            ',17, 4,)
          call lib$put_screen (' 2  spawn and try again  ',18, 4,)
          call lib$put_screen (' 3  exit setup module    ',19, 4,)

          do while ( .true. )

            call lib$put_screen ('    your choice  ',21,,1)
            call noecho ( answ, 1, iq, 0 )

            if ( answ .eq. '1' ) then
              goto 100
            else if ( answ .eq. '2' ) then
              call lib$erase_page (1,1)
              call spawn(' ')
              goto 100
            else if ( answ .eq. '3' ) then
              modify    = .false.
              figure    = .false.
              return
            endif

          enddo

        endif

* call plots if figure is .true. and test if device is allocated *......

        if ( figure ) then

           plotter = 'pl_'//plotter
           istat   = str$upcase ( plotter, plotter )
           istat   = lib$set_logical ('PL_', plotter)
           if ( .not.istat ) call lib$stop( %val ( istat ) )

           inquire(file=plotter,name=name)           ! is plotter ok ?
           call lib$get_lun ( lun )
           open(unit=lun,file=name,status='new',
     £     carriagecontrol='none',recl=512,iostat=iopen)
           close ( lun )
           call lib$free_lun ( lun )
           plotter = plotter ( 4: )
```

```
            if ( iopen .eq. 0 ) then                    ! plotter is ok
              call nonotify ('  SETUP  ' ,22,2,0)
              call plots(1729,0,7)
              if ( plotter .eq. 'TEX' ) then            ! NB bug
                call factor ( scale * 1.010101 )
              else
                call factor ( scale )
              endif
            else
              call notify ('  error using plotter '//plotter   ,22,2,0)
              goto 100
            endif

          endif

* open new plotfiles .. call exit if error *.........................

990       open(unit=11,file='logplot1.plo',status='new',
     £    iostat=lun_11,carriagecontrol='list',dispose='delete')
          open(unit=12,file='logplot2.plo',status='new',
     £    iostat=lun_12,carriagecontrol='list',dispose='delete')

          if ( lun_11 .ne. 0 .or. lun_12 .ne. 0 ) then
            call notify ('  logplot open failure ' ,22,2,0)
            call exit
          endif

          return
          end
```

```
      subroutine   creator

*     purpose:     create plotfile

*     subroutines: menu
*                  set_repeat
*                  item
*                  set_verify
*                  output
*                  repeat

      include '<jd330314.logplot>commonblk.for'

      do while ( .true. )

        verify = .true.

        do while ( verify )
          call menu
          if ( endofplot )  return
          if ( n_box.eq.1)  call set_repeat
          call item
          call set_verify
        enddo

        if ( store ) then
          call output
          n_box   = n_box + 1
          y0_last = y0 + box_width
        endif

        call repeat

      enddo

      end
```

```
      subroutine   modifier

*     purpose:     modify plotfile

*     subroutines: input
*                  menu
*                  modlogic
*                  item
*                  set_verify
*                  output

      include '<jd330314.logplot>commonblk.for'

      do while ( .true. )

        call input
        if ( endofplot ) return

        verify = .true.

        do while ( verify )

          call menu
          if ( endofplot ) return
          call modlogic
          call item
          call set_verify

        enddo

        if ( store ) then
          call output
          n_box   = n_box + 1
          y0_last = y0 + box_width
        endif

      enddo

      end
```

```
      subroutine   drawfig

*     purpose:     draw figure on one page

*     subroutines: input
*                  data
*                  nonotify
*                  drawbox
*                  output
*                  lin_typ
*                  head_line
*                  os_merki

      include '<jd330314.logplot>commonblk.for'

      ydumma =  9999.
      ydummb = -9999.

      do while ( .true. )

        call input
        if ( endofplot ) goto 900

        call data

        x_scale    = ( x_last - x_first ) / box_length
        y_scale    = ( y_last - y_first ) / box_width

        if ( x_cut .lt. x_first) then
          call output
          return
        else if ( x_cut .lt. x_last ) then
          box_length = ( x_cut - x_first ) / x_scale
          x_scale    = ( x_cut - x_first ) / box_length
        endif

        if ( .not.errorplot ) then

          call nonotify (' reading '//datafile , 22,5,0)
          call drawbox
          close ( unit=13 )
          if ( endofplot  ) return

          ydumma  = min( ydumma, y0 )
          ydummb  = max( ydummb, y0 + box_width )

        endif

        call output

      enddo

900   lint = lin_typ( 0 )
      call peep     ( 3 )
      call head_line( 3.0, ( ydummb + ydumma ) * 0.5, 0.7, 90.0, 0.0 )
      call os_merki( 1.5,1.5,1.0,90.0,'JHD-BJ-9000-IM',14,'84.12.1001 T',14 )

      return
      end
```

```
      subroutine   reset

*     purpose:     close input and output files
*                  and reset common variables

*     subroutines: notify
*                  plot

      include '<jd330314.logplot>commonblk.for'

      if ( create ) then

        close ( unit=11, dispose = 'keep'   )
        close ( unit=12, dispose = 'delete' )
        endofplot = .false.
        plotfile  = 'logplot1.plo'
        call notify (' this picture logplot1.plo ',22,1,0)

      else if ( modify ) then

        close ( unit=10 )
        close ( unit=11, dispose = 'keep'   )
        close ( unit=12, dispose = 'delete' )
        endofplot = .false.
        plotfile  = 'logplot1.plo'
        call notify (' this picture logplot1.plo ',22,1,0)

      else if ( figure ) then

        close ( unit=10 )
        close ( unit=11, dispose = 'delete' )
        close ( unit=12, dispose = 'keep'   )
        endofplot = .false.
        call notify (' next picture logplot2.plo ',22,1,0)
        call plot   ( 999, 999, 999 )

      endif

      return
      end
```

```
      subroutine   menu

*     purpose:     select one item from menu
*                  set common variables draw_lit, draw_aqu,
*                  draw_min, draw_bor, draw_log and  endofplot

*     subroutines: lib$put_screen
*                  nonotify
*                  noecho
*                  esc6
*                  append

      include '<jd330314.logplot>commonblk.for'
      character*3  dumm

      call nonotify ('  FIG. 2     MENU    ' ,2,1,0)

      write(dumm,'(i3)') n_box
      if ( create ) then
        call esc6 ('  next you  CREATE  box'//dumm    ,4,1,0)
      else
        call esc6 ('  next you  MODIFY  box'//dumm    ,4,1,0)
      endif

      do while ( .true. )

        i1 = 0
        i2 = 0
        i3 = 0
        i4 = 0
        i5 = 0
        i6 = 0

        if ( draw_lit  )  i1 = 2
        if ( draw_aqu  )  i2 = 2
        if ( draw_min  )  i3 = 2
        if ( draw_bor  )  i4 = 2
        if ( draw_log  )  i5 = 2
        if ( endofplot )  i6 = 2

        call lib$put_screen (' 1) lithology   ',     13,40,i1)
        call lib$put_screen (' 2) aquifers    ',     14,40,i2)
        call lib$put_screen (' 3) minerals    ',     15,40,i3)
        call lib$put_screen (' 4) drill rate  ',     16,40,i4)
        call lib$put_screen (' 5) log         ',     17,40,i5)
        call lib$put_screen (' 6) exit menu   ',     18,40,i6)

        call lib$put_screen (' <ret> = no change  ', 21,43,)
        call noecho ( answ, 1, iq, 0 )
        if (ichar(answ).eq.13 .and. modify .and. endofplot ) call append
        if (ichar(answ).eq.13 )     return
```

```
  if ( answ.ge.'1' .and. answ.le.'6' ) then

    draw_lit  = .false.
    draw_aqu  = .false.
    draw_min  = .false.
    draw_bor  = .false.
    draw_log  = .false.
    endofplot = .false.

    if ( answ .eq. '1' )   draw_lit  = .true.
    if ( answ .eq. '2' )   draw_aqu  = .true.
    if ( answ .eq. '3' )   draw_min  = .true.
    if ( answ .eq. '4' )   draw_bor  = .true.
    if ( answ .eq. '5' )   draw_log  = .true.
    if ( answ .eq. '6' )   endofplot = .true.

  endif
enddo
end
```

```
      subroutine   item

*     purpose:     create or modify one item

*     subroutines: lithology
*                  aquifers
*                  minerals
*                  measure

      include '<jd330314.logplot>commonblk.for'

      if ( draw_lit ) call lithology
      if ( draw_aqu ) call aquifers
      if ( draw_min ) call minerals
      if ( draw_bor ) call measure
      if ( draw_log ) call measure

      return
      end
```

```
      subroutine   lithology

*     purpose:     create or modify lithology

*     subroutines: data
*                  bounds
*                  set_x_scale
*                  set_x_logic
*                  set_y_scale
*                  litlogic
*                  set_origin

      include '<jd330314.logplot>commonblk.for'

      draw_y_axis = .false.
      draw_y_grid = .false.
      draw_x_grid = .false.
      draw_y_mean = .false.
      draw_border = .false.
      upside_down = .false.

      if ( create ) then

        call data
        call bounds
        if ( n_box .eq.1 ) call set_x_scale
        call set_x_logic
        call set_y_scale
        call litlogic
        call set_origin

      else if ( modify ) then

        if ( modify_data    )  call data
        if ( modify_bounds  )  call bounds
        if ( modify_x_scale )  call set_x_scale
        if ( modify_x_logic )  call set_x_logic
        if ( modify_body    )  call set_y_scale
        if ( modify_body    )  call litlogic
        if ( modify_origin  )  call set_origin

      endif

      return
      end
```

```
      subroutine   aquifers

*     purpose:     create or modify aquifers

*     subroutines: data
*                  bounds
*                  set_x_scale
*                  set_x_logic
*                  set_y_scale
*                  set_y_logic
*                  aqubody
*                  set_origin

      include '<jd330314.logplot>commonblk.for'

      draw_y_grid = .false.
      draw_x_grid = .false.
      draw_y_mean = .false.
      draw_border = .false.

      if ( create ) then

        draw_y_axis = .false.
        upside_down = .false.

        call data
        call bounds
        if ( n_box .eq.1 ) call set_x_scale
        call set_x_logic
        call set_y_scale
        call set_y_logic
        call aqubody
        call set_origin

      else if ( modify ) then

        if ( modify_data    )  call data
        if ( modify_bounds  )  call bounds
        if ( modify_x_scale )  call set_x_scale
        if ( modify_x_logic )  call set_x_logic
        if ( modify_y_scale )  call set_y_scale
        if ( modify_y_logic )  call set_y_logic
        if ( modify_body    )  call aqubody
        if ( modify_origin  )  call set_origin

      endif

      return
      end
```

```
      subroutine   minerals

*     purpose:     create or modify minerals

*     subroutines: data
*                  bounds
*                  set_x_scale
*                  set_x_logic
*                  minbody
*                  set_origin

      include '<jd330314.logplot>commonblk.for'

      draw_y_axis = .false.
      draw_y_grid = .false.
      draw_x_grid = .false.
      draw_y_mean = .false.
      draw_border = .false.
      upside_down = .false.

      if ( create ) then

        call data
        call bounds
        if ( n_box .eq.1 ) call set_x_scale
        call set_x_logic
        call minbody
        call set_origin

      else if ( modify ) then

        if ( modify_data    )  call data
        if ( modify_bounds  )  call bounds
        if ( modify_x_scale )  call set_x_scale
        if ( modify_x_logic )  call set_x_logic
        if ( modify_body    )  call minbody
        if ( modify_origin  )  call set_origin

      endif

      return
      end
```

```
      subroutine   measure

*     purpose:     create or modify drillrate and log

*     subroutines: data
*                  bounds
*                  set_x_scale
*                  set_x_logic
*                  set_y_scale
*                  set_y_logic
*                  boxlogic
*                  set_origin

      include '<jd330314.logplot>commonblk.for'

      if ( create ) then

        call data
        call bounds
        if ( n_box .eq.1 ) call set_x_scale
        call set_x_logic
        call set_y_scale
        call set_y_logic
        call boxlogic
        call set_origin

      else if ( modify ) then

        if ( modify_data    ) call data
        if ( modify_bounds  ) call bounds
        if ( modify_x_scale ) call set_x_scale
        if ( modify_x_logic ) call set_x_logic
        if ( modify_y_scale ) call set_y_scale
        if ( modify_y_logic ) call set_y_logic
        if ( modify_body    ) call boxlogic
        if ( modify_origin  ) call set_origin

      endif

      return
      end
```

```
      subroutine   set_repeat

*     purpose:     to determine continuation ( same page or next page)
*                  set common variable n_repeat

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outi
*                  geti

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 2.1   REPEAT  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        call outi           ('                    ', n_repeat-1   ,18, 1,)
        call lib$put_screen (' 1) continue each box'              ,18, 1,)
        call lib$put_screen (' times on this page '               ,18,25,)
        call outi           ('             ', n_repeat            ,19, 1,)
        call lib$put_screen ('    ( ie total '                    ,19, 1,)
        call lib$put_screen (' box(es) with same datafile )'      ,19,18,)

        call lib$put_screen ('    <ret> = no change  '            ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then

          n_repeat = -1
          do while ( n_repeat .lt. 1 .or. n_repeat .gt. 99 )
            call geti ('    enter total number  ', n_repeat   ,21,1,1)
          enddo

        endif

      enddo
      end
```

```
      subroutine   repeat

*     purpose:     continue current box n_repeat - 1 times on same page
*                  and reset common variables

*     subroutines: set_origin
*                  output

      include '<jd330314.logplot>commonblk.for'

      x_dum1 = x_first                ! save common variables
      x_dum2 = x_last
      y_dum1 = y_first
      y_dum2 = y_last
      y_dum3 = y0_last

      do i = 2, n_repeat

        x_first = x_last
        x_last  = x_first + box_length * x_scale

        call set_origin
        call output

        n_box   = n_box + 1
        y0_last = y0 + box_width

      enddo

      x_first  = x_dum1               ! reset common variables
      x_last   = x_dum2
      y_first  = y_dum1
      y_last   = y_dum2
      y0_last  = y_dum3
      datafile = 'unknown'
      y_text   = ' '
      y_nst    = 0

      return
      end
```

```
      subroutine    modlogic

*     purpose:      control what items are to be modified
*     set common    modify_data, modify_bounds,
*                   modify_x_scale, modify_x_logic,
*                   modify_y_scale, modify_y_logic,
*                   modify_body and  modify_origin

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  notify
*                  noecho

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 2.2   MODIFY  LOGIC  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (4,1)

        if ( modify_data ) then
          call lib$put_screen (' 1  modify data               ',15,,)
        else
          call lib$put_screen (' 1  do NOT modify data        ',15,,)
        endif

        if ( modify_bounds ) then
          call lib$put_screen (' 2  modify bounds             ',16,,)
        else
          call lib$put_screen (' 2  do NOT modify bounds      ',16,,)
        endif

        if ( modify_x_scale ) then
          call lib$put_screen (' 3  modify depth scale        ',17,,)
        else
          call lib$put_screen (' 3  do NOT modify depth scale ',17,,)
        endif

        if ( modify_x_logic ) then
          call lib$put_screen (' 4  modify depth logic        ',18,,)
        else
          call lib$put_screen (' 4  do NOT modify depth logic ',18,,)
        endif

        if ( modify_y_scale ) then
          call lib$put_screen (' 5  modify log scale          ',15,40,)
        else
          call lib$put_screen (' 5  do NOT modify log scale   ',15,40,)
        endif

        if ( modify_y_logic ) then
          call lib$put_screen (' 6  modify log logic          ',16,40,)
        else
          call lib$put_screen (' 6  do NOT modify log logic   ',16,40,)
        endif
```

```
  if ( modify_body ) then
    call lib$put_screen (' 7  modify main body           ',17,40,)
  else
    call lib$put_screen (' 7  do NOT modify main body    ',17,40,)
  endif

  if ( modify_origin ) then
    call lib$put_screen (' 8  modify origin              ',18,40,)
  else
    call lib$put_screen (' 8  do NOT modify origin       ',18,40,)
  endif

  call lib$put_screen ('    t> = no change  '          ,21,,)
  call noecho (answ,1,iq,1)
  if (ichar(answ).eq.13)    return

  if ( answ .eq. '1' )   modify_data    = .not.modify_data
  if ( answ .eq. '2' )   modify_bounds  = .not.modify_bounds
  if ( answ .eq. '3' )   modify_x_scale = .not.modify_x_scale
  if ( answ .eq. '4' )   modify_x_logic = .not.modify_x_logic
  if ( answ .eq. '5' )   modify_y_scale = .not.modify_y_scale
  if ( answ .eq. '6' )   modify_y_logic = .not.modify_y_logic
  if ( answ .eq. '7' )   modify_body    = .not.modify_body
  if ( answ .eq. '8' )   modify_origin  = .not.modify_origin

enddo
end
```

```
      subroutine   data

*     purpose:     open datafile ..... if error set errorplot .true.
*                  set common datafile, dataform, curve_type and ipen

*     subroutines: lib$erase_page
*                  lib$erase_line
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  spawn
*                  outi

      include '<jd330314.logplot>commonblk.for'

      character*4      dum
      character*10     type(0:4)

* initialize *.....................................................

      type(0) = ' solid    '
      type(1) = ' points   '
      type(2) = ' dotdash  '
      type(3) = ' dashed   '
      type(4) = ' longdash '

      if ( draw_lit ) dum = '.LIT'
      if ( draw_aqu ) dum = '.AQU'
      if ( draw_min ) dum = '.MIN'
      if ( draw_bor ) dum = '.BOR'
      if ( draw_log ) dum = '.LOG'

      errorplot = .false.

      if ( figure )   goto 900                ! and skip main section

* main section *...................................................

100   call nonotify ('  FIG. 2.3   DATA  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        if ( draw_lit .or. draw_aqu .or. draw_min ) then
          call lib$put_screen (' 1  '//dum//' file        '//  datafile,18,,)
        else
          call lib$put_screen (' 1 '//dum//' file         '//  datafile,15,,)
          call lib$put_screen (' 2 format           '//       dataform,16,,)
          call lib$put_screen (' 3 curve_type      '//type(curve_type),17,,)
          call outi           (' 4 requested pen',     ipen            ,18,,)
        endif

        call lib$put_screen ('    <ret> = no change  '       ,21,,)
```

```
      call noecho ( answ, 1, iq, 0 )

      if ( answ.eq.'2' .and. (draw_lit.or.draw_aqu.or.draw_min)) answ = '0'
      if ( answ.eq.'3' .and. (draw_lit.or.draw_aqu.or.draw_min)) answ = '0'
      if ( ichar( answ ) .eq. 13 )                    goto 900

      ios = 1
      if ( answ .eq. '1' ) then

        do while ( ios .ne. 0 .or. iq .gt. 32 )
          call lib$erase_line (21,1)
          call lib$put_screen ('    '//dum//' file  '  ,,,1)
          read(5,'(q,a)',iostat=ios) iq, datafile
        enddo

      else if ( answ .eq. '2' ) then

        do while ( ios .ne. 0 .or. iq .gt. 32 )
          call lib$erase_line (21,1)
          call lib$put_screen ('    format  ' ,,,1)
          read(5,'(q,a)',iostat=ios) iq, dataform
        enddo

      else if ( answ .eq. '3' ) then

        call lib$erase_line (21,1)
        call lib$put_screen ( '    0  solid    ',21, 1,)
        call lib$put_screen ( '    1  points   ',21,15,)
        call lib$put_screen ( '    2  dotdash  ',21,30,)
        call lib$put_screen ( '    3  dashed   ',21,45,)
        call lib$put_screen ( '    4  longdash ',21,60,)

        do while ( ios .ne. 0 )
          call lib$erase_line (21,78)
          call noecho ( answ, 1, iq, 0 )
          read(answ,'(i1)',iostat=ios)  curve_type
          if ( curve_type .lt. 0 .or. curve_type .gt. 4 ) ios = 1
        enddo

      else if ( answ .eq. '4' ) then

        do while ( ios .ne. 0 )
          call lib$put_screen
£         ('    number of requested pen  ( hp only )  ', 21,,1)
          call noecho ( answ, 1, iq, 0 )
          read(answ,'(i1)',iostat=ios)  ipen
        enddo

      endif
    enddo
```

```
* open datafile and reset errorplot *.....................................

900     errorplot = .false.

        open(unit=13,file=datafile,status='old',
     £       defaultfile=dum,readonly,iostat=ios)

        if ( ios .ne. 0 ) then

          errorplot = .true.
          call nonotify ('  FIG. 11.2  DATA  ERROR      ',2,1,0)
          call lib$put_screen (' file '//datafile        ,10, 4,)
          call lib$put_screen (' error in filename       ',12, 4,)
          call lib$put_screen (' or file not found       ',12,22,)
          call lib$put_screen (' you have 3 choices      ',15, 4,)
          call lib$put_screen (' 1  try again            ',17, 4,)
          call lib$put_screen (' 2  spawn                ',18, 4,)
          call lib$put_screen (' 3  continue             ',19, 4,)

          do while ( .true. )

            call lib$put_screen ('    your choice  ',21,,1)
            call noecho ( answ, 1, iq, 0 )

            if ( answ .eq. '1' ) then
              goto 100
            else if ( answ .eq. '2' ) then
              call lib$erase_page (1,1)
              call spawn(' ')
              goto 100
            else if ( answ .eq. '3' ) then
              return
            endif

          enddo

        else if ( create .or. modify ) then

          close ( unit = 13 )

        endif

        return
        end
```

```
      subroutine   bounds

*     purpose:     set common variables: x_min, x_max, x_cut, y_min, y_max

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 2.4   BOUNDS  '   ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        if ( draw_lit .or. draw_aqu .or. draw_min ) then
          call outf(' 1) minimum depth                ', x_min ,14,,)
          call outf(' 2) maximum depth                ', x_max ,15,,)
          call outf(' 3) cutoff depth axis            ', x_cut ,16,,)
        else
          call outf(' 1) minimum depth                ', x_min ,14,,)
          call outf(' 2) maximum depth                ', x_max ,15,,)
          call outf(' 3) cutoff depth axis            ', x_cut ,16,,)
          call outf(' 4) minimum log                  ', y_min ,17,,)
          call outf(' 5) maximum log                  ', y_max ,18,,)
        endif

        call lib$put_screen ('    <ret> = no change  '        ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( (draw_lit.or.draw_aqu.or.draw_min) .and. answ.eq.'4') answ = '0'
        if ( (draw_lit.or.draw_aqu.or.draw_min) .and. answ.eq.'5') answ = '0'

        if ( answ .eq. '1' ) then
          call getf ('    minimum depth  ',  x_min  ,21,1,1)
        else if ( answ .eq. '2' ) then
          call getf ('    maximum depth  ',  x_max  ,21,1,1)
        else if ( answ .eq. '3' ) then
          call getf ('    cutoff depth  ',   x_cut  ,21,1,1)
        else if ( answ .eq. '4' ) then
          call getf ('    minimum log  ',    y_min  ,21,1,1)
        else if ( answ .eq. '5' ) then
          call getf ('    maximum log  ',    y_max  ,21,1,1)
        endif

      enddo

      end
```

```
      subroutine   set_x_scale

*     purpose:     scale x_axis   ( ie depth axis )
*                  ie common variables x_scale box_length x_first x_last

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 3     DEPTH AXIS   SCALE  ' ,2,1,0)

      do while ( .true. )

        x_scale = (x_last - x_first) / box_length

        call lib$erase_page (3,1)
        call outf('    depth scale (m/cm)  ', x_scale          ,13,,)
        call outf(' 1) box length          ', box_length       ,16,,)
        call outf(' 2) value at origin     ', x_first          ,17,,)
        call outf(' 3) value at endpoint   ', x_last           ,18,,)

        if ( box_length.gt.50. )   call lib$put_screen
     £  (' box longer than 50 cm '                             ,16,37,2)
        if ( x_last .le. x_first ) call lib$put_screen
     £  (' value at endpoint less than at origin '             ,18,37,6)

        call lib$put_screen ('    <ret> = no change  '        ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ.eq.'1' ) then
          box_length = -1.
          do while ( box_length.le.0. )
            call getf ('    box_length (cm)  ',  box_length ,21,1,1)
          enddo
        else if ( answ.eq.'2' ) then
          call getf ('    value at origin (m)  ',   x_first ,21,1,1)
        else if ( answ.eq.'3' ) then
          call getf ('    value at endpoint (m)  ',  x_last ,21,1,1)
        endif

      enddo
      end
```

```
      subroutine   set_x_logic

*     purpose:     supervisory program for x axis
*                  set common variables draw_x_axis draw_x_line
*                           draw_x_tick draw_x_numb draw_x_text

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  set_x_tick
*                  set_x_numb
*                  set_x_text

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 3.1   DEPTH AXIS   LOGIC  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page(3,1)

        if ( draw_x_axis ) then

          call lib$put_screen   (' 1) draw axis '       ,14,,)

          if ( draw_x_line ) then
            call lib$put_screen (' 2) draw line '       ,15,,)
          else
            call lib$put_screen (' 2) OMIT line '       ,15,,)
          endif

          if ( draw_x_tick ) then
            call lib$put_screen (' 3) draw tickmarks '  ,16,,)
          else
            call lib$put_screen (' 3) OMIT tickmarks '  ,16,,)
          endif

          if ( draw_x_numb ) then
            call lib$put_screen (' 4) draw numbers '    ,17,,)
          else
            call lib$put_screen (' 4) OMIT numbers '    ,17,,)
          endif

          if ( draw_x_text ) then
            call lib$put_screen (' 5) draw text '       ,18,,)
          else
            call lib$put_screen (' 5) OMIT text '       ,18,,)
          endif

        else

          call lib$put_screen   (' 1) OMIT axis '       ,14,,)

        endif
```

```
      call lib$put_screen ('    <ret> = no change  '  ,21,,)
      call noecho (answ,1,iq,1)
      if (ichar(answ).eq.13)    goto 900
      if (.not.draw_x_axis .and. answ.ne.'1') answ = '0'

      if ( answ .eq. '1' )   draw_x_axis  = .not.draw_x_axis
      if ( answ .eq. '2' )   draw_x_line  = .not.draw_x_line
      if ( answ .eq. '3' )   draw_x_tick  = .not.draw_x_tick
      if ( answ .eq. '4' )   draw_x_numb  = .not.draw_x_numb
      if ( answ .eq. '5' )   draw_x_text  = .not.draw_x_text

      enddo

900   if ( draw_x_axis .and. draw_x_tick ) call set_x_tick
      if ( draw_x_axis .and. draw_x_numb ) call set_x_numb
      if ( draw_x_axis .and. draw_x_text ) call set_x_text

      return
      end
```

```
      subroutine   set_x_tick

*     purpose:     determine how tickmars are drawn on x axis
*                  set common x_tickint x_tackint x_ticklen x_tacklen

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 3.2   DEPTH AXIS   TICKMARKS  ' ,2,1,0)

      call outf('    value at origin                    ', x_first ,5,,)
      call outf('    value at endpoint                  ', x_last  ,6,,)

      do while ( .true. )

        call lib$erase_page (7,1)

        call outf(' 1) interval between shorter tickmarks ', x_tickint ,15,,)
        call outf(' 2) interval between longer tickmarks  ', x_tackint ,16,,)
        call outf(' 3) length of shorter tickmarks        ', x_ticklen ,17,,)
        call outf(' 4) length of longer tickmarks         ', x_tacklen ,18,,)

        if ( mod(x_tackint,x_tickint) .ne. 0 ) then
          call lib$put_screen ('  short tick interval '   ,15,52,6)
          call lib$put_screen (' / long tick interval '   ,16,52,6)
        endif
        if ( abs(x_ticklen) .gt. 1. ) then
          call lib$put_screen (' longer than 1 cm '       ,17,52,2)
        endif
        if ( abs(x_tacklen) .gt. 1. ) then
          call lib$put_screen (' longer than 1 cm '       ,18,52,2)
        endif

        call lib$put_screen ('    <ret> = no change  '    ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then
          x_tickint = 0.
          do while ( x_tickint.le.0. )
            call getf ('    interval between shorter tickmarks (m)  ',
     £       x_tickint      ,21,1,1)
          enddo
        else if ( answ .eq. '2' ) then
          x_tackint = 0.
          do while ( x_tackint.le.0. )
            call getf ('    interval between longer tickmarks (m)  ',
     £       x_tackint      ,21,1,1)
          enddo
        else if ( answ .eq. '3' ) then
          call getf ('    length of shorter tickmarks (cm)  ',
     £     x_ticklen      ,21,1,1)
        else if ( answ .eq. '4' ) then
          call getf ('    length of longer tickmarks (cm)  ',
     £     x_tacklen      ,21,1,1)
        endif

      enddo
      end
```

```
      subroutine   set_x_numb

*     purpose:     determine how numbers are drawn on x axis
*                  set common variables x_numbint x_hgtno x_numbdist ndec_x

*     subroutines: lib$erase_page
*                  lib$erase_line
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  outi
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 3.3   DEPTH AXIS   NUMBERS  '   ,2,1,0)

      call outf('    value at origin              ', x_first ,5,,)
      call outf('    value at endpoint            ', x_last  ,6,,)

      if ( draw_x_axis .and. draw_x_tick ) then
        call outf('    shorter tickmarks interval  ', x_tickint  ,8,,)
        call outf('    longer tickmarks interval   ', x_tackint  ,9,,)
        call outf('    length of shorter tickmarks ', x_ticklen ,10,,)
        call outf('    length of longer tickmarks  ', x_tacklen ,11,,)
      endif

      do while ( .true. )

        call lib$erase_page (12,1)

        call outf(' 1) interval between numbers    ', x_numbint ,15,,)
        call outf(' 2) height of numbers           ', x_hgtno   ,16,,)
        call outf(' 3) distance from axis          ', x_numbdist,17,,)
        call outi(' 4) number of decimal digits    ', ndec_x    ,18,,)

        if ( draw_x_tick .and. mod(x_numbint,x_tackint).ne.0 ) then
          call lib$put_screen
     £    (' NOTE: interval between tickmarks '              ,15,44,2)
        endif

        call lib$put_screen ('    <ret> = no change  '       ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then
          x_numbint = -1.
          do while (x_numbint.le.0.)
            call getf('    interval between numbers (m)  ',
     £      x_numbint      ,21,1,1)
          enddo
        else if ( answ .eq. '2' ) then
          call getf ('    height of numbers (cm)   ',   x_hgtno ,21,1,1)
        else if ( answ .eq. '3' ) then
          call getf ('   distance from axis (cm)  ', x_numbdist ,21,1,1)
```

```
  else if ( answ .eq. '4' ) then

    ios = 1
    call lib$erase_page (19,1)
    call lib$put_screen ('    = -1  omit decimal point ',21,,)
    call lib$put_screen ('    =  0  draw decimal point ',22,,)
    call lib$put_screen ('    >  0  number of decimals ',23,,)

    do while ( ios.ne.0 .or. ndec_x.lt.-1 .or. ndec_x.gt.9 )
      call lib$erase_line (23,35)
      call lib$put_screen ('    decimal digits  '      ,,,1)
      read(5,'(i)',iostat=ios)  ndec_x
    enddo

  endif

enddo
end
```

```
      subroutine   set_x_text

*     purpose:     determine what text is drawn on x axis
*                  set common x_text x_nst x_textdist x_textheight

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  outi
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 3.4   DEPTH AXIS   TEXT  '  ,2,1,0)

      if ( draw_x_axis .and. draw_x_tick ) then
        call outf('    length of shorter tickmarks ', x_ticklen ,5,,)
        call outf('    length of longer tickmarks  ', x_tacklen ,6,,)
      endif

      if ( draw_x_axis .and. draw_x_numb ) then
        call outf('    height of numbers           ', x_hgtno     ,9,,)
        call outf('    distance from axis          ', x_numbdist ,10,,)
        call outi('    number of decimal digits    ', ndec_x     ,11,,)
      endif

      do while ( .true. )

        call lib$erase_page (12,1)

        call lib$put_screen
     £  (' 1) text                        '//x_text(1:x_nst)     ,16,,)
        call outf(' 2) distance from axis    ', x_textdist      ,17,,)
        call outf(' 3) height of characters  ', x_textheight    ,18,,)

        call lib$put_screen ('    <ret> = no change  '          ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then
          ios = 1
          do while ( ios.ne.0 .or. x_nst.gt.40 )
            call lib$erase_page (21,1)
            call lib$put_screen ('    text on  depth_axis  '  ,,,1)
            read(6,'(q,a)',iostat=ios) x_nst, x_text
          enddo
        else if ( answ .eq. '2' ) then
          call getf ('    distance (cm)  ',  x_textdist  ,21,1,1)
        else if ( answ .eq. '3' ) then
          call getf ('    height (cm)  ',  x_textheight  ,21,1,1)
        endif

      enddo
      end
```

```
      subroutine   set_y_scale

*     purpose:     scale y_axis  ( ie log axis )
*                  ie common variables y_scale box_width y_first y_last

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 4     LOG AXIS   SCALE  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        if ( draw_aqu .or. draw_lit ) then
          call outf(' 1) box width           ',     box_width ,16,,)
          if ( draw_lit .and. lit_omit ) then
            call outf ('    layers are omitted if thinner than',
     £                    x_scale*0.666*box_width/8.          ,13,,)
          endif
          if ( draw_lit .and. box_width .ne. 2.5 ) then
            call lib$put_screen ('  recommended  2.50  ',16,37,2)
          endif
          if ( draw_aqu .and. box_width .ne. 1. ) then
            call lib$put_screen ('  recommended  1.00  ',16,37,2)
          endif
        else
          y_scale = (y_last - y_first) / box_width
          call outf('    scale (data/cm)     ', y_scale    ,13,,)
          call outf(' 1) box width           ', box_width  ,16,,)
          call outf(' 2) value at origin     ', y_first    ,17,,)
          call outf(' 3) value at endpoint   ', y_last     ,18,,)
          if ( box_width .lt.  3. ) then
            call lib$put_screen (' box width less than  3 cm ',16,37,2)
          else if ( box_width .gt. 10. ) then
            call lib$put_screen (' box width more than 10 cm ',16,37,2)
          endif
          if ( y_last .le. y_first ) call lib$put_screen
     £    (' value at endpoint less than at origin '          ,18,37,6)
        endif

        call lib$put_screen ('    <ret> = no change  '        ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( ( draw_lit .or. draw_aqu ) .and. ( answ .eq. '2' ) ) answ = '0'
        if ( ( draw_lit .or. draw_aqu ) .and. ( answ .eq. '3' ) ) answ = '0'
```

```
  if ( answ .eq. '1' ) then
    box_width = -1.
    do while ( box_width.le.0. )
      call getf ('    box_width (cm)  ',       box_width ,21,1,1)
    enddo
  else if ( answ .eq. '2' ) then
    call getf ('    value at origin (data)  ',   y_first ,21,1,1)
  else if ( answ .eq. '3' ) then
    call getf ('    value at endpoint (data)  ', y_last  ,21,1,1)
  endif

enddo
end
```

```
      subroutine   set_y_logic

*     purpose:     supervisory program for y axis
*                  set common variables draw_y_axis upside_down
*                  draw_y_line  draw_y_tick draw_y_numb draw_y_text

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  set_y_tick
*                  set_y_numb
*                  set_y_text

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 4.1   LOG AXIS   LOGIC  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page(13,1)

        if ( draw_y_axis ) then
          call lib$put_screen (' 1) draw axis '              ,13,,)
        else
          call lib$put_screen (' 1) OMIT axis '              ,13,,)
        endif

        if ( upside_down ) then
          call lib$put_screen(' 2) orientation right to left ',14,,2)
        else
          call lib$put_screen(' 2) orientation left to right ',14,,)
        endif

        if ( draw_y_axis ) then

          if ( draw_y_line ) then
            call lib$put_screen (' 3) draw line '            ,15,,)
          else
            call lib$put_screen (' 3) OMIT line '            ,15,,)
          endif

          if ( draw_y_tick ) then
            call lib$put_screen (' 4) draw tickmarks '       ,16,,)
          else
            call lib$put_screen (' 4) OMIT tickmarks '       ,16,,)
          endif

          if ( draw_y_numb ) then
            call lib$put_screen (' 5) draw numbers '         ,17,,)
          else
            call lib$put_screen (' 5) OMIT numbers '         ,17,,)
          endif
```

```
          if ( draw_y_text ) then
            call lib$put_screen (' 6) draw text '          ,18,,)
          else
            call lib$put_screen (' 6) OMIT text '          ,18,,)
          endif

        endif

        call lib$put_screen ('    <ret> = no change  '     ,21,,)
        call noecho (answ,1,iq,1)
        if (ichar(answ).eq.13)    goto 900
        if ( answ.ne.'1' .and. answ.ne.'2' .and. .not.draw_y_axis ) answ='0'

        if ( answ .eq. '1' ) draw_y_axis  = .not.draw_y_axis
        if ( answ .eq. '2' ) upside_down  = .not.upside_down
        if ( answ .eq. '3' ) draw_y_line  = .not.draw_y_line
        if ( answ .eq. '4' ) draw_y_tick  = .not.draw_y_tick
        if ( answ .eq. '5' ) draw_y_numb  = .not.draw_y_numb
        if ( answ .eq. '6' ) draw_y_text  = .not.draw_y_text

      enddo

900   if ( draw_y_axis .and. draw_y_tick ) call set_y_tick
      if ( draw_y_axis .and. draw_y_numb ) call set_y_numb
      if ( draw_y_axis .and. draw_y_text ) call set_y_text

      return
      end
```

```
      subroutine   set_y_tick

*     purpose:     determine how tickmars are drawn on y axis
*                  set common y_tickint y_tackint y_ticklen y_tacklen

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 4.2   LOG AXIS  TICKMARKS ' ,2,1,0)

      call outf('    value at origin                    ', y_first ,5,,)
      call outf('    value at endpoint                  ', y_last  ,6,,)

      do while ( .true. )

        call lib$erase_page (7,1)

        call outf(' 1) interval between shorter tickmarks', y_tickint ,15,,)
        call outf(' 2) interval between longer tickmarks ', y_tackint ,16,,)
        call outf(' 3) length of shorter tickmarks       ', y_ticklen ,17,,)
        call outf(' 4) length of longer tickmarks        ', y_tacklen ,18,,)

        if ( mod(y_tackint,y_tickint) .ne. 0 ) then
          call lib$put_screen ('  short tick interval '   ,15,52,6)
          call lib$put_screen (' / long tick interval '   ,16,52,6)
        endif

        if ( abs(y_ticklen) .gt. 1. ) then
          call lib$put_screen (' longer than 1 cm '       ,17,52,2)
        endif

        if ( abs(y_tacklen) .gt. 1. ) then
          call lib$put_screen (' longer than 1 cm '       ,18,52,2)
        endif

        call lib$put_screen ('    <ret> = no change '     ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then
          y_tickint = 0.
          do while ( y_tickint.le.0. )
            call getf
     £      ('    interval between shorter tickmarks (data_units)  '
     £      , y_tickint   ,21,1,1)
          enddo
```

```
      else if ( answ .eq. '2' ) then
        y_tackint = 0.
        do while ( y_tackint.le.0. )
          call getf
£         ('    interval between longer tickmarks (data_units)  '
£         , y_tackint   ,21,1,1)
        enddo
      else if ( answ .eq. '3' ) then
        call getf ('    length of shorter tickmarks (cm)  ',
£       y_ticklen       ,21,1,1)
      else if ( answ .eq. '4' ) then
        call getf('    length of longer tickmarks (cm)  ',
£       y_tacklen      ,21,1,1)
      endif

    enddo
    end
```

```
      subroutine   set_y_numb

*     purpose:     determine how numbers are drawn on y axis
*                  set common y_numbint y_hgtno y_numbdist ndec_y

*     subroutines: lib$erase_page
*                  lib$erase_line
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  outi
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 4.3   LOG AXIS   NUMBERS  '   ,2,1,0)

      call outf('    value at origin             ', y_first ,5,,)
      call outf('    value at endpoint           ', y_last  ,6,,)

      if ( draw_y_axis .and. draw_y_tick ) then
        call outf('    shorter tickmarks interval  ', y_tickint  ,8,,)
        call outf('    longer tickmarks interval   ', y_tackint  ,9,,)
        call outf('    length of shorter tickmarks ', y_ticklen ,10,,)
        call outf('    length of longer tickmarks  ', y_tacklen ,11,,)
      endif

      do while ( .true. )

        call lib$erase_page (12,1)

        call outf(' 1) interval between numbers    ', y_numbint  ,15,,)
        call outf(' 2) height of numbers           ', y_hgtno    ,16,,)
        call outf(' 3) distance from axis          ', y_numbdist ,17,,)
        call outi(' 4) number of decimal digits    ', ndec_y     ,18,,)

        if (draw_y_tick .and. mod(y_numbint,y_tackint).ne.0)
     £  call lib$put_screen
     £  (' NOTE: interval between tickmarks '                ,15,44,2)

        call lib$put_screen ('    <ret> = no change  '        ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then
          y_numbint = -1.
          do while ( y_numbint.le.0. )
            call getf
     £      ('    interval between numbers (data units)  ',
     £      y_numbint,21,1,1)
          enddo
        else if ( answ .eq. '2' ) then
          call getf ('    height of numbers (cm)   ',    y_hgtno ,21,1,1)
        else if ( answ .eq. '3' ) then
          call getf ('    distance from axis (cm)  ', y_numbdist ,21,1,1)
```

```
  else if ( answ .eq. '4' ) then
    ios = 1
    call lib$erase_page (21,1)
    call lib$put_screen ('    = -1  omit desimal point ',21,,)
    call lib$put_screen ('    =  0  draw desimal point ',22,,)
    call lib$put_screen ('    >  0  number of desimals ',23,,)

    do while ( ios.ne.0 .or. ndec_y.lt.-1 .or. ndec_y.gt.9 )
      call lib$erase_line (23,35)
      call lib$put_screen ('    desimal digits  '      ,,,1)
      read(5,'(i)',iostat=ios)  ndec_y
    enddo

  endif

enddo
end
```

```
      subroutine   set_y_text

*     purpose:     determine what text is drawn on y axis
*                  set common y_text y_nst y_textdist y_textheight

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  outi
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 4.4   LOG AXIS   TEXT  ' ,2,1,0)

      if ( draw_y_tick .and. draw_y_tick ) then
        call outf('    length of shorter tickmarks ', y_ticklen ,5,,)
        call outf('    length of longer tickmarks  ', y_tacklen ,6,,)
      endif

      if ( draw_y_tick .and. draw_y_numb ) then
        call outf('    height of numbers           ', y_hgtno     ,9,,)
        call outf('    distance from axis          ', y_numbdist, 10,,)
        call outi('    number of decimal digits    ', ndec_y    , 11,,)
      endif

      do while ( .true. )

        call lib$erase_page (12,1)

        call lib$put_screen
     £  (' 1) text                          '//y_text(1:y_nst)    ,16,,)
        call outf(' 2) distance from log_axis ', y_textdist     ,17,,)
        call outf(' 3) height of characters   ', y_textheight   ,18,,)

        call lib$put_screen ('    <ret> = no change  '          ,21,,)
        call noecho (answ,1,iq,0)
        if   (ichar(answ).eq.13)  return

        if ( answ .eq. '1' ) then
          ios = 1
          do while ( ios.ne.0 .or. y_nst.gt.40 )
            call lib$erase_page (21,1)
            call lib$put_screen ('    text on  log_axis  '     ,,,1)
            read(6,'(q,a)',iostat=ios) y_nst, y_text
          enddo
        else if ( answ .eq. '2' ) then
          call getf ('    distance (cm)  ',    y_textdist   ,21,1,1)
        else if ( answ .eq. '3' ) then
          call getf ('    height (cm)  ',     y_textheight  ,21,1,1)
        endif

      enddo
      end
```

```
      subroutine   boxlogic

*     purpose:     supervisory program for box
*                  set common variables draw_border
*                  draw_x_grid draw_y_grid and draw_y_mean

*     subroutines: lib$put_screen
*                  nonotify
*                  noecho
*                  set_x_grid
*                  set_y_grid
*                  set_y_mean

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 5     BOX   LOGIC  ' ,2,1,0)

      do while ( .true. )

        if ( draw_border ) then
          call lib$put_screen (' 1) draw border of box '     ,15,,)
        else
          call lib$put_screen (' 1) OMIT border of box '     ,15,,)
        endif

        if ( draw_x_grid ) then
          call lib$put_screen (' 2) draw depth_grid '        ,16,,)
        else
          call lib$put_screen (' 2) OMIT depth_grid '        ,16,,)
        endif

        if ( draw_y_grid ) then
          call lib$put_screen (' 3) draw log_grid '          ,17,,)
        else
          call lib$put_screen (' 3) OMIT log_grid '          ,17,,)
        endif

        if ( draw_y_mean ) then
          call lib$put_screen (' 4) draw fixed log_value '   ,18,,)
        else
          call lib$put_screen (' 4) OMIT fixed log_value '   ,18,,)
        endif

        call lib$put_screen ('    <ret> = no change  '       ,21,,)
        call noecho (answ,1,iq,1)
        if (ichar(answ).eq.13)    goto 900

        if ( answ .eq. '1' )   draw_border = .not.draw_border
        if ( answ .eq. '2' )   draw_x_grid = .not.draw_x_grid
        if ( answ .eq. '3' )   draw_y_grid = .not.draw_y_grid
        if ( answ .eq. '4' )   draw_y_mean = .not.draw_y_mean

      enddo

900   if ( draw_x_grid )   call set_x_grid
      if ( draw_y_grid )   call set_y_grid
      if ( draw_y_mean )   call set_y_mean

      return
      end
```

```
      subroutine   set_x_grid

*     purpose:     determine how depth grid is drawn
*                  set common variables x_gridint and x_gridlen

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 5.1   DEPTH_GRID  ' ,2,1,0)

      call outf('    value at origin             ', x_first ,5,,)
      call outf('    value at endpoint           ', x_last  ,6,,)

      if ( draw_x_axis .and. draw_x_tick ) then
        call outf('    shorter tickmarks interval  ', x_tickint  ,8,,)
        call outf('    longer tickmarks interval   ', x_tackint  ,9,,)
        call outf('    length of shorter tickmarks ', x_ticklen ,10,,)
        call outf('    length of longer tickmarks  ', x_tacklen ,11,,)
      endif
      if ( draw_x_axis .and. draw_x_numb ) then
        call outf('    interval between numbers    ', x_numbint ,13,,)
      endif

      do while ( .true. )

        call lib$erase_page(14,1)

        call outf(' 1) interval between gridlines  ', x_gridint ,17,,)
        call outf(' 2) length of grid ticks        ', x_gridlen ,18,,)

        if ( draw_x_axis ) then
          if ( draw_x_tick .and. mod(x_tackint,x_gridint).ne.0. ) then
            call lib$put_screen
     £       (' tickckmarks    /    grid interval   '         ,15,45,2)
          endif
          if ( draw_x_numb .and. mod(x_numbint,x_gridint).ne.0. ) then
            call lib$put_screen
     £       (' numbers        /    grid interval   '         ,16,45,2)
          endif
          if ( mod(abs(x_last-x_first),x_gridint).ne.0. ) then
            call lib$put_screen
     £       (' (endpoint-origin) / grid interval   '         ,17,45,2)
          endif
          if ( draw_x_tick .and. x_gridlen.ne.x_tacklen ) then
            call lib$put_screen
     £       (' NOTE: length of tickmarks           '         ,18,45,2)
          endif
        endif
```

```
      call lib$put_screen('    <ret> = no change  ',21,,)
      call noecho (answ,1,iq,0)
      if (ichar(answ).eq.13)   return

      if ( answ .eq. '1' ) then
        x_gridint = -1
        do while ( x_gridint.le.0 )
          call getf('    interval between gridlines (m)  ',
£         x_gridint      ,21,1,1)
        enddo
      else if ( answ .eq. '2' ) then
        call getf ('    length of grid_ticks (cm)  ', x_gridlen ,21,1,1)
      endif

    enddo
    end
```

```
      subroutine   set_y_grid

*     purpose:     determine how log grid is drawn
*                  set common variables y_gridint and y_gridlen

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 5.2   LOG_GRID  ' ,2,1,0)

      call outf('    value at origin             ', y_first ,5,,)
      call outf('    value at endpoint           ', y_last  ,6,,)

      if ( draw_y_axis .and. draw_y_tick ) then
        call outf('    shorter tickmarks interval  ', y_tickint  ,8,,)
        call outf('    longer tickmarks interval   ', y_tackint  ,9,,)
        call outf('    length of shorter tickmarks ', y_ticklen ,10,,)
        call outf('    length of longer tickmarks  ', y_tacklen ,11,,)
      endif
      if ( draw_y_axis .and. draw_y_numb ) then
        call outf('    interval between numbers    ', y_numbint ,13,,)
      endif

      do while ( .true. )

        call lib$erase_page(14,1)

        call outf(' 1) interval between gridlines  ', y_gridint  ,17,,)
        call outf(' 2) length of grid ticks        ', y_gridlen  ,18,,)

        if ( draw_y_axis ) then
          if ( draw_y_tick .and. mod(y_tackint,y_gridint).ne.0. ) then
            call lib$put_screen
     £        (' tickckmarks    /    grid interval   '         ,15,45,2)
          endif
          if ( draw_y_numb .and. mod(y_numbint,y_gridint).ne.0. ) then
            call lib$put_screen
     £        (' numbers        /    grid interval   '         ,16,45,2)
          endif
          if ( mod(abs(y_last-y_first),y_gridint).ne.0. ) then
            call lib$put_screen
     £        (' (endpoint-origin) / grid interval   '         ,17,45,2)
          endif
          if ( draw_y_tick .and. y_gridlen.ne.y_tacklen ) then
            call lib$put_screen
     £        (' NOTE: length of longer tickmarks     '        ,18,45,2)
          endif
        endif
```

```
      call lib$put_screen('    <ret> = no change  '       ,21,,)
      call noecho (answ,1,iq,0)
      if (ichar(answ).eq.13)   return

      if ( answ .eq. '1' ) then
        y_gridint = -1.
        do while ( y_gridint.le.0 )
          call getf('    interval between gridlines (data units)  ',
£         y_gridint      ,21,1,1)
        enddo
      else if ( answ .eq. '2' ) then
        call getf ('    length of grid_ticks (cm)  ', y_gridlen ,21,1,1)
      endif
    enddo
    end
```

```
      subroutine   set_y_mean

*     purpose:     determine how and where fixed log walue is drawn
*                  set common variables y_mean and mean_type

*     subroutines: lib$erase_page
*                  lib$erase_line
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      character*10     type(0:4)

      type(0) = ' solid    '
      type(1) = ' points   '
      type(2) = ' dotdash  '
      type(3) = ' dashed   '
      type(4) = ' longdash '

      call nonotify ('  FIG. 5.3   FIXED LOG_VALUE  ' ,2,1,0)

      call outf('    log_value at origin       ', y_first ,5,,)
      call outf('    log_value at endpoint     ', y_last  ,6,,)

      do while ( .true. )

        call lib$erase_page (7,1)

        call outf(' 1) fixed log_value           ', y_mean   ,17,,)
        call lib$put_screen(' 2) mean_type'//type(mean_type) ,18,,)

        call lib$put_screen ('    <ret> = no change  '       ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then

          call getf ('    fixed log_value  ',   y_mean ,21,1,1)

        else if ( answ .eq. '2' ) then

          call lib$erase_line (21,1)
          call lib$put_screen ( '    0  solid    ',21, 1,)
          call lib$put_screen ( '    1  points   ',21,15,)
          call lib$put_screen ( '    2  dotdash  ',21,30,)
          call lib$put_screen ( '    3  dashed   ',21,45,)
          call lib$put_screen ( '    4  longdash ',21,60,)

          ios = 1
          do while (ios.ne.0 .or. (mean_type.lt.0 .or. mean_type.gt.4))
            call lib$erase_line (21,78)
            call noecho (answ,1,iq,0)
            read(answ,'(i1)',iostat=ios)  mean_type
          enddo

        endif
      enddo
      end
```

```
      subroutine   litlogic

*     purpose:     supervisory program for lithology
*                  set common variables:
*                  draw_litlog, draw_explan, draw_casing
*                  draw_bittyp, draw_weight and draw_header

*     subroutines: lib$put_screen
*                  nonotify
*                  noecho
*                  set_lit
*                  set_exp
*                  set_cas
*                  set_bit
*                  set_wgt
*                  set_hdl

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 6     LITHOLOGY   LOGIC  ',2,1,0)

      do while ( .true. )

        if ( draw_litlog ) then
          call lib$put_screen (' 1) draw geological section ',13,,)
        else
          call lib$put_screen (' 1) OMIT geological section ',13,,)
        endif

        if ( draw_explan ) then
          call lib$put_screen (' 2) draw explanations       ',14,,)
        else
          call lib$put_screen (' 2) OMIT explanations       ',14,,)
        endif

        if ( draw_casing ) then
          call lib$put_screen (' 3) draw casing             ',15,,)
        else
          call lib$put_screen (' 3) OMIT casing             ',15,,)
        endif

        if ( draw_bittyp ) then
          call lib$put_screen (' 4) draw drill_bit          ',16,,)
        else
          call lib$put_screen (' 4) OMIT drill_bit          ',16,,)
        endif

        if ( draw_weight ) then
          call lib$put_screen (' 5) draw drill_weight       ',17,,)
        else
          call lib$put_screen (' 5) OMIT drill_weight       ',17,,)
        endif

        if ( draw_header ) then
          call lib$put_screen (' 6) draw headlines          ',18,,)
        else
          call lib$put_screen (' 6) OMIT headlines          ',18,,)
        endif
```

```
      call lib$put_screen ('    <ret> = no change  '      ,21,,)
      call noecho ( answ, 1, iq, 1 )
      if (ichar(answ).eq.13)    goto 900

      if ( answ .eq. '1' )   draw_litlog = .not.draw_litlog
      if ( answ .eq. '2' )   draw_explan = .not.draw_explan
      if ( answ .eq. '3' )   draw_casing = .not.draw_casing
      if ( answ .eq. '4' )   draw_bittyp = .not.draw_bittyp
      if ( answ .eq. '5' )   draw_weight = .not.draw_weight
      if ( answ .eq. '6' )   draw_header = .not.draw_header

    enddo

900 if ( draw_litlog ) call set_lit
    if ( draw_explan ) call set_exp
    if ( draw_casing ) call set_cas
    if ( draw_bittyp ) call set_bit
    if ( draw_weight ) call set_wgt
    if ( draw_header ) call set_hdl

    end
```

```
      subroutine   set_lit

*     purpose:     logic for geological section
*                  set common variables lit_mark, lit_raster, lit_omit

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf

      include '<jd330314.logplot>commonblk.for'

      call nonotify  ('  FIG. 6.1   LITHOLOGY    ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        if ( lit_mark ) then
          call lib$put_screen (' 1) mark bounds of layers   ',16,,)
        else
          call lib$put_screen (' 1) OMIT bounds of layers   ',16,,)
        endif

        if ( lit_raster ) then
          call lib$put_screen (' 2) generate rasters        ',17,,)
        else
          call lib$put_screen (' 2) OMIT rasters            ',17,,)
        endif

        if ( lit_omit ) then
          x_dumm =  0.08325 * box_width * (x_last - x_first) / box_length
          call outf (' ',                                  x_dumm,18,26,)
          call lib$put_screen (' 3) OMIT layers thinner than',18, 1,)
          call lib$put_screen (' meter(s) '                  ,18,37,)
        else
          call lib$put_screen (' 3) do not omit thin layers ',18,,)
        endif

        call lib$put_screen ('    <ret> = no change  '       ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' )   lit_mark   = .not.lit_mark
        if ( answ .eq. '2' )   lit_raster = .not.lit_raster
        if ( answ .eq. '3' )   lit_omit   = .not.lit_omit

      enddo
      end
```

```
      subroutine   set_exp

*     purpose:     set common variables exp_height, exp_dist

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify  ('  FIG. 6.2   LITHOLOGY   EXPLANATIONS  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        call outf
     £  (' 1) height of characters in explanations', exp_height ,17,,)
        call outf
     £  (' 2) distance from right side of box     ', exp_dist   ,18,,)

        call lib$put_screen ('    <ret> = no change  '          ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then
          call getf ('    height of characters ',  exp_height ,21,1,1)
        else if ( answ .eq. '2' ) then
          call getf ('    distance  from box  ',   exp_dist   ,21,1,1)
        endif

      enddo
      end
```

```
      subroutine   set_cas

*     purpose:     set common variables cas_height, cas_dist, cas_space

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify  ('  FIG. 6.3   LITHOLOGY   CASING  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        call outf(' 1) height of characters           ', cas_height ,16,,)
        call outf(' 2) distance from left side of box  ', cas_dist   ,17,,)
        call outf(' 3) distance between casings        ', cas_space  ,18,,)

        call lib$put_screen ('    <ret> = no change  '               ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then
          call getf (' height of characters  ',         cas_height ,21,4,1)
        else if ( answ .eq. '2' ) then
          call getf (' distance from left side of box  ', cas_dist ,21,4,1)
        else if ( answ .eq. '3' ) then
          call getf (' distance between casings (cm)  ', cas_space ,21,4,1)
        endif

      enddo
      end
```

```
      subroutine   set_bit

*     purpose:     set common variables bit_height, bit_dist

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify  ('  FIG. 6.4   LITHOLOGY   DRILL_BIT  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        call outf(' 1) height of characters           ', bit_height ,17,,)
        call outf(' 2) distance from left side of box  ', bit_dist   ,18,,)

        call lib$put_screen ('    <ret> = no change  '               ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then
          call getf ('    height of characters  ',  bit_height  ,21,1,1)
        else if ( answ .eq. '2' ) then
          call getf ('    distance  '            ,  bit_dist    ,21,1,1)
        endif

      enddo
      end
```

```
      subroutine   set_wgt

*     purpose:     set common variables wgt_height, wgt_dist

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify  ('  FIG. 6.5   LITHOLOGY   DRILL_WEIGHT  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        call outf(' 1) height of characters             ', wgt_height ,17,,)
        call outf(' 2) distance from left side of box   ', wgt_dist   ,18,,)

        call lib$put_screen ('    <ret> = no change  '                ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then
          call getf ('    height of characters  ',  wgt_height  ,21,1,1)
        else if ( answ .eq. '2' ) then
          call getf ('    distance  '           ,  wgt_dist    ,21,1,1)
        endif

      enddo
      end
```

```
      subroutine   set_hdl

*     purpose:     set common variables hdl_height, hdl_dist

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify  ('  FIG. 6.6   LITHOLOGY   HEADLINES  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        call outf(' 1) height of characters in headlines', hdl_height ,17,,)
        call outf(' 2) distance from top of box         ', hdl_dist   ,18,,)

        call lib$put_screen ('    <ret> = no change  '                ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    return

        if ( answ .eq. '1' ) then
          call getf ('    height of characters  ',  hdl_height  ,21,1,1)
        else if ( answ .eq. '2' ) then
          call getf ('    distance  '            ,  hdl_dist    ,21,1,1)
        endif

      enddo
      end
```

```
      subroutine   aqubody

*     purpose:     supervisory program for aquifers
*                  set common: aqu_height and aqu_dist

*     subroutines: lib$erase_page
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  outf
*                  getf
*                  getf

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 7     AQUIFERS  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        call outf
     £  (' 1) height of characters (tip of arrow) ', aqu_height ,17,,)
        call outf
     £  (' 2) distance between text and arrow     ', aqu_dist   ,18,,)

        call lib$put_screen ('    <ret> = no change  '          ,21,,)
        call noecho (answ,1,iq,0)

        if ( ichar ( answ ) .eq. 13 ) then
          return
        else if ( answ .eq. '1' ) then
          call getf ('    height of characters  ' ,aqu_height ,21,1,1)
        else if ( answ .eq. '2' ) then
          call getf ('    distance  '             ,aqu_dist   ,21,1,1)
        endif

      enddo
      end
```

```
      subroutine   minbody

*     purpose:     supervisory program for minerals
*                  set common: min_int, min_tck, min_dis
*                  min_hgt, min_rot, min_symbhgt and min_method

*     subroutines: lib$erase_page
*                  lib$erase_line
*                  lib$put_screen
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'
      character*15 type(0:6)

      type (0) = 'default'
      type (1) = 'unknown'
      type (2) = 'unknown'
      type (3) = 'thin section'
      type (4) = 'XRD'
      type (5) = 'unknown'
      type (6) = 'cuttings'

      call nonotify ('  FIG. 8     MINERALS  ' ,2,1,0)

      do while ( .true. )

        call lib$erase_page (3,1)

        call outf (' 1) interval between tickmarks     ',  min_int     ,11,,)
        call outf (' 2) length of tickmarks            ',  min_tck     ,12,,)
        call outf (' 3) distance between names and axis',  min_dis     ,13,,)
        call outf (' 4) height of characters           ',  min_hgt     ,14,,)
        call outf (' 5) rotation of text               ',  min_rot     ,15,,)
        call outf (' 6) height of symbols              ',  min_symbhgt ,16,,)
        call outf (' 7) height of brackets             ',  min_brahgt  ,17,,)
        call lib$put_screen (' 8) method '//type( min_method )          ,18,,)

        istat  = lib$put_screen ('    <ret> = no change  '               ,21,,)
        call noecho (answ,1,iq,0)

        if ( ichar ( answ ) .eq. 13 )  then
           return
        else if ( answ .eq. '1' ) then
          call getf (' interval between tickmarks  ',     min_int,21,4,1)
        else if ( answ .eq. '2' ) then
          call getf (' length of tickmarks  ',            min_tck,21,4,1)
        else if ( answ .eq. '3' ) then
          call getf (' distance between names and axis  ',min_dis,21,4,1)
        else if ( answ .eq. '4' ) then
          call getf (' height of characters  ',           min_hgt,21,4,1)
        else if ( answ .eq. '5' ) then
          call getf (' rotation of names  ',              min_rot,21,4,1)
```

```
      else if ( answ .eq. '6' ) then
        call getf (' height of symbols  ',          min_symbhgt,21,4,1)
      else if ( answ .eq. '7' ) then
        call lib$put_screen
£       (' recommended 1.25 x height of symbols ',24,1,2)
        call getf (' height of brackets (0=omit)  ', min_brahgt,21,4,1)
      else if ( answ .eq. '8'  ) then

        ios = 1
        call lib$erase_line (21,1)
        call lib$put_screen ('  0  default      ',21 ,1,1)
        call lib$put_screen ('  3  thin section ',21,15,1)
        call lib$put_screen ('  4  XRD          ',21,35,1)
        call lib$put_screen ('  6  cuttings     ',21,50,1)

        do while ( ios.ne.0 .or. (min_method.lt.0 .or. min_method.gt.6 ))
          call noecho (answ,1,iq,0)
          read(answ,'(i1)',iostat=ios)  min_method
        enddo

      endif

    enddo
    end
```

```
      subroutine   set_origin

*     purpose:     set common variables x0 and y0
*                  ie origin of box on paper

*     subroutines: lib$put_screen
*                  lib$erase_page
*                  nonotify
*                  noecho
*                  outf
*                  getf

      include '<jd330314.logplot>commonblk.for'
      character*3 dumm

      write(dumm,'(i3)') n_box

      call nonotify ('  FIG. 9     ORIGIN OF BOX'//dumm ,2,1,0)

      x1_dum = 0.0
      x2_dum = 0.0
      x3_dum = 0.0

      call outf('    value at origin                ',   x_first,5,,)
      call outf('    value at endpoint              ',   x_last ,6,,)

      if ( draw_x_axis .and. draw_x_tick ) then
        x1_dum = max(x_ticklen,x_tacklen)
        call outf('    length of tickmarks             ', x1_dum ,8,,)
      endif

      if ( draw_x_axis .and. draw_x_numb ) then
        x2_dum = x_hgtno * (5. + real(ndec_x)) + x_numbdist
        call outf('    distance between numbers and box', x2_dum ,9,,)
      endif

      if ( draw_x_axis .and. draw_x_text ) then
        x3_dum = x_textdist + x_textheight
        call outf('    distance between text and box   ', x3_dum,10,,)
      endif

      if ( n_box .ne. 1 ) then
        x4_dum = y0_last + max(x1_dum,x2_dum,x3_dum,0.0)
        call outf('         right side of previous box ', y0_last ,12,,)
        call outf('         and any value greater than ', x4_dum  ,13,,)
        call lib$put_screen (' is recommended for y0 '       ,13,50,)
      endif
```

```
do while ( .true. )

  call lib$erase_page (14,1)
  call outf(' 1) x0  (cm)                          ', x0    ,17,,)
  call outf(' 2) y0  (cm)                          ', y0    ,18,,)

  call lib$put_screen ('    <ret> = no change  ',21,,)
  call noecho (answ,1,iq,0)
  if (ichar(answ).eq.13)    return

  if ( answ .eq. '1' ) then

    call lib$put_screen ('    if this is not the first box ' ,23,1,2)
    call lib$put_screen ('    then be careful to change x0 ' ,24,1,2)
    call getf  ('    x0 in cm  ', x0 ,21,1,1)

  else if ( answ .eq. '2' ) then

    call getf ('    y0 in cm  ', y0 ,21,1,1)

  endif

enddo
end
```

```
      subroutine   set_verify

*     purpose:     set common variables verify and store

*     subroutines: lib$put_screen
*                  nonotify
*                  notify
*                  noecho

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  FIG. 10    VERIFY AND STORE SETUP  ' ,2,1,0)

      verify = .false.
      store  = .true.

      do while ( .true. )

        if ( verify ) then
          call lib$put_screen (' 1  verify previous box         ' ,17,,2)
        else
          call lib$put_screen (' 1  do not verify previous box  ' ,17,,0)
        endif

        if ( store ) then
          call lib$put_screen (' 2  store this setup in outfile ' ,18,,0)
        else
          call lib$put_screen (' 2  OMIT this setup in outfile  ' ,18,,2)
        endif

        call lib$put_screen ('    <ret> = no change  '  ,21,,)
        call noecho (answ,1,iq,0)
        if (ichar(answ).eq.13)    goto 900

        if ( answ .eq. '1' )   verify = .not.verify
        if ( answ .eq. '2' )   store  = .not.store

      enddo

900   if ( verify ) then
        call notify (' PREPARE TO VERIFY PREVIOUS BOX ',22,2,1)
      endif

      return
      end
```

```
      subroutine   append

*     purpose:     read old plotfile until end_of_file
*                  and write to new plotfiles
*                  first backspace to beginning of box

*     subroutines: nonotify
*                  input
*                  output

      include '<jd330314.logplot>commonblk.for'

      call nonotify ('  APPEND ' ,22,1,0)

      endofplot = .false.

      do i = 1, 14

        backspace ( unit = 10 )

      enddo

      do while ( .not.endofplot )

        call input
        if ( .not.endofplot )  call output

      enddo

      return
      end
```

```
      subroutine   input

*     purpose:     write common variables to new plotfiles

*     subroutines: plot
*                  notify
*                  exit

      include '<jd330314.logplot>commonblk.for'

      read(10,'(a)',err=900,end=800)
      read(10,'(5i1,t11,5i1,t21,6i1,t31,4i1,t41,6i1,t51,3i1,t72,i2)',
     £          err=900,end=900)
     £          draw_lit, draw_aqu, draw_min, draw_bor, draw_log,
     £          draw_x_axis,              draw_x_line,
     £          draw_x_tick, draw_x_numb, draw_x_text,
     £          draw_y_axis, upside_down, draw_y_line,
     £          draw_y_tick, draw_y_numb, draw_y_text,
     £          draw_border, draw_y_mean,
     £          draw_x_grid, draw_y_grid,
     £          draw_litlog, draw_explan, draw_casing,
     £          draw_weight, draw_bittyp, draw_header,
     £          lit_mark,    lit_raster,  lit_omit, n_repeat
      read(10,'(3f9.0,t41,3f9.0)',err=900,end=900)
     £          box_length, x_first, x_last,
     £          box_width,  y_first, y_last
      read(10,'(4f9.0,t41,4f9.0)',err=900,end=900)
     £          x_tickint, x_tackint, x_ticklen, x_tacklen,
     £          y_tickint, y_tackint, y_ticklen, y_tacklen
      read(10,'(3f9.0,i6,t41,3f9.0,i6)',err=900,end=900)
     £          x_numbint, x_hgtno, x_numbdist, ndec_x,
     £          y_numbint, y_hgtno, y_numbdist, ndec_y
      read(10,'(2f9.0,t41,q,a)',err=900,end=900)
     £          x_textdist, x_textheight, x_nst, x_text
      read(10,'(2f9.0,t41,q,a)',err=900,end=900)
     £          y_textdist, y_textheight, y_nst, y_text
      read(10,'(2f9.0,t41,3f9.0,t73,i1)',err=900,end=900)
     £          x_gridint, x_gridlen, y_gridint, y_gridlen,
     £          y_mean,    mean_type
      read(10,'(f9.0,t41,f9.0)',err=900,end=900) x0, y0
      read(10,'(a,t41,a,2i1)',err=900,end=900)
     £          datafile, dataform, curve_type, ipen
      read(10,'(3f9.0,t41,2f9.0)',err=900,end=900)
     £          x_min, x_max, x_cut, y_min, y_max
      read(10,'(2f9.0,t41,3f9.0)',err=900,end=900)
     £          exp_height, exp_dist,
     £          cas_height, cas_dist, cas_space
      read(10,'(4f9.0,t41,4f9.0)',err=900,end=900)
     £          bit_height, bit_dist, wgt_height, wgt_dist,
     £          hdl_height, hdl_dist, aqu_height, aqu_dist
      read(10,'(4f9.0,t41,3f9.2,t72,i2)',err=900,end=900)
     £          min_int, min_tck, min_dis, min_hgt, min_rot,
     £          min_symbhgt, min_brahgt, min_method
      return

800   endofplot = .true.
      return

900   if ( figure ) call plot(999,999,999)
      close ( unit=11, dispose='delete' )
      close ( unit=12, dispose='delete' )
      call notify (' error in file '//plotfile  ,21,1,0)

      call exit
      end
```

```
      subroutine   output

*     purpose:     accept common variables from old plotfile

      include '<jd330314.logplot>commonblk.for'
      integer*2 lun

      do lun = 11, 12

        write(lun,'(76(''-''))')
        write(lun,'(5l1,t11,5l1,t21,6l1,t31,4l1,t41,6l1,t51,3l1,t72,i2)')
     £              draw_lit, draw_aqu, draw_min, draw_bor, draw_log,
     £              draw_x_axis,              draw_x_line,
     £              draw_x_tick, draw_x_numb, draw_x_text,
     £              draw_y_axis, upside_down, draw_y_line,
     £              draw_y_tick, draw_y_numb, draw_y_text,
     £              draw_border, draw_y_mean,
     £              draw_x_grid, draw_y_grid,
     £              draw_litlog, draw_explan, draw_casing,
     £              draw_weight, draw_bittyp, draw_header,
     £              lit_mark,    lit_raster,  lit_omit, n_repeat
        if ( lun .eq. 11 ) then
          write(lun,'(3f9.2,t41,3f9.2)')
     £              box_length, x_first, x_last,
     £              box_width,  y_first, y_last
        else
          dumm = (box_length * x_scale)*real(n_repeat)
          write(lun,'(3f9.2,t41,3f9.2)')
     £              box_length, x_first + dumm, x_last + dumm,
     £              box_width,  y_first, y_last
        endif
        write(lun,'(4f9.2,t41,4f9.2)')
     £            x_tickint, x_tackint, x_ticklen, x_tacklen,
     £            y_tickint, y_tackint, y_ticklen, y_tacklen
        write(lun,'(3f9.2,i6,t41,3f9.2,i6)')
     £            x_numbint, x_hgtno, x_numbdist, ndec_x,
     £            y_numbint, y_hgtno, y_numbdist, ndec_y
        write(lun,'(2f9.2,t41,a)')
     £            x_textdist, x_textheight, x_text(1:x_nst)
        write(lun,'(2f9.2,t41,a)')
     £            y_textdist, y_textheight, y_text(1:y_nst)
        write(lun,'(2f9.2,t41,3f9.2,t73,i1)')
     £            x_gridint, x_gridlen, y_gridint, y_gridlen,
     £            y_mean,    mean_type
        write(lun,'(f9.2,t41,f9.2)')  x0, y0
        write(lun,'(a,t41,a,2i1)')
     £            datafile, dataform, curve_type, ipen
        write(lun,'(3f9.2,t41,2f9.2)')
     £            x_min, x_max, x_cut, y_min, y_max
        write(lun,'(2f9.2,t41,3f9.2)')
     £            exp_height, exp_dist,
     £            cas_height, cas_dist, cas_space
        write(lun,'(4f9.2,t41,4f9.2)')
     £            bit_height, bit_dist, wgt_height, wgt_dist,
     £            hdl_height, hdl_dist, aqu_height, aqu_dist
        write(lun,'(4f9.2,t41,3f9.2,t72,i2)')
     £            min_int, min_tck, min_dis, min_hgt, min_rot,
     £            min_symbhgt, min_brahgt, min_method
      enddo

      return
      end
```

```
      subroutine   drawbox

*     purpose:     draw depth axis , log axis , grid, and y_mean

*     subroutines: origin
*                  new_cha
*                  newpen
*                  rectan
*                  im_tick
*                  im_numb
*                  im_text
*                  plot
*                  draw_item
*                  lin_typ

      include '<jd330314.logplot>commonblk.for'

      real*4        mirror, translate

      iasx = 0
      iasy = 0
      if ( draw_x_line ) iasx = 1
      if ( draw_y_line ) iasy = 1

      lint = lin_typ ( 0 )
      call origin ( x0, y0 )
      if ( plotter(1:2).eq.'HP' )  call newpen ( 1 )  ! new pen if plotter HP

      if ( icelandic ) then
        call new_cha ( 1 )
      else
        call new_cha ( 0 )
      endif

      if ( draw_border ) then                         ! draw border of box
        call rectan ( 0., 0., box_length, box_width )
        iasx = 0
        iasy = 0
      endif

      if ( upside_down ) then                         ! log axis upside down
        first     = y_first + y_scale * box_width
        mirror    = -1.
        translate = box_width
      else
        first     = y_first
        mirror    = 1.
        translate = 0.
      endif
```

```
   if ( draw_x_axis ) then                          ! draw depth axis

     if ( draw_x_line ) then
       call im_tick ( 0.0, 0.0, box_length, x_first, x_scale, 0.0,
£           iasx, x_tickint, x_tackint, 0.0, 0.0 )
     endif

     if ( draw_x_tick ) then
       call im_tick ( 0.0, 0.0, box_length, x_first, x_scale, 0.0,
£           0, x_tickint, x_tackint, -x_ticklen, -x_tacklen )
     endif

     if ( draw_x_numb ) then
       call im_numb ( 0.0, 0.0, box_length, x_first, x_scale, 0.0,
£           ndec_x, x_numbint, 1, x_hgtno, -x_numbdist )
     endif

     if ( draw_x_text ) then
       call im_text ( 0.0, 0.0, box_length, 0.0, x_text,
£           x_nst, -x_textheight, -x_textdist )
     endif

   endif

   if ( draw_y_axis ) then                          ! draw log axis

     if ( draw_y_line ) then
       call im_tick ( 0.0, 0.0, box_width, first, mirror*y_scale,
£        90.0, iasy, mirror*y_tickint, mirror*y_tackint, 0.0, 0.0 )
     endif

     if ( draw_y_tick ) then
       call im_tick ( 0.0, 0.0, box_width, first, mirror*y_scale,
£           90.0, 0, mirror*y_tickint, mirror*y_tackint,
£           y_ticklen, y_tacklen )
     endif

     if ( draw_y_numb ) then
       call im_numb ( 0.0, 0.0, box_width, first, mirror*y_scale,
£          90.0, ndec_y, mirror*y_numbint, 0, y_hgtno, y_numbdist )
     endif

     if ( draw_y_text ) then
       call im_text ( 0.0, 0.0, box_width, 90.0, y_text,
£                     y_nst, y_textheight, y_textdist )
     endif

   endif
```

```
   if ( draw_x_grid ) then                          ! draw depth grid
     call im_tick ( 0.0, 0.0, box_length, x_first, x_scale, 0.0,
£         0, x_gridint, x_gridint, 0.0, box_width+x_gridlen )
   endif

   if ( draw_y_grid ) then                          ! draw log grid
     call im_tick ( 0.0, 0.0, box_width, first, mirror*y_scale,
£              90.0, 0, mirror*y_gridint, mirror*y_gridint,
£               0.0, -box_length-y_gridlen )
   endif

   if ( draw_y_mean ) then                          ! draw mean value
     lint  = lin_typ ( mean_type )
     dummy = mirror * ( y_mean - y_first) / y_scale + translate
     call plot(0.,          dummy, 3)
     call plot(box_length, dummy, 2)
     lint = lin_typ ( 0 )
   endif

   call draw_item                                   ! draw remainig part

   call origin ( -x0, -y0 )
   call plot ( 0, 0, 0 )

   return
   end
```

```
      subroutine   draw_item

*     purpose:     select one item and draw figure

*     subroutines: drawlit
*                  drawaqu
*                  drawmin
*                  drawbor
*                  drawlog
*                  newpen

      include '<jd330314.logplot>commonblk.for'

      integer*2 idumm

      if ( ( draw_bor .or. draw_log ) .and. plotter(1:2).eq.'HP' ) then
        idumm = newpen ( ipen )              ! change pen if plotter HP
      endif

      if ( draw_lit ) call drawlit
      if ( draw_aqu ) call drawaqu
      if ( draw_min ) call drawmin
      if ( draw_bor ) call drawbor
      if ( draw_log ) call drawlog

      if ( ( draw_bor .or. draw_log ) .and. plotter(1:2).eq.'HP' ) then
        call newpen ( 1 )                    ! restore pen if plotter HP
      endif

      return
      end
```

```
      subroutine   drawlit

*     purpose:     draw lithology explanations casings drill_weight drillbit

*     subroutines: new_cha
*                  plot
*                  fill_box
*                  new_cha
*                  symbol
*                  im_arrow
*                  new_cha
*                  mark

      include '<jd330314.logplot>commonblk.for'

      character*80  text
      character*80  line
      character*32  litform1, litform2, litform3
      character*1   marker
      logical       done, mark1, mark2, mark_done
      logical       draw_lithdl, draw_cashdl, draw_wgthdl, draw_bithdl

* initialize *.....................................................

      litform1 = '(t2,a,t4,f9.0,t14,f9.0,t28,1i2)'
      litform2 = '(     t4,f9.0,          t28,q,a)'
      litform3 = '(     t4,f9.0,t14,f9.0,t28,q,a)'

      draw_lithdl = .false.
      draw_cashdl = .false.
      draw_wgthdl = .false.
      draw_bithdl = .false.

      dy        = 0.0
      xmin      = max( x_first, x_min )
      xmax      = min( x_last , x_max )
      mark_done = .false.
      done      = .true.

      lint = lin_typ ( 0 )
      call new_cha ( 0 )

      bit_hgt = 0.25
      wgt_hgt = 0.25
      if ( bit_height .gt. 0.0 ) bit_hgt = 0.7 * bit_height
      if ( wgt_height .gt. 0.0 ) wgt_hgt = 0.7 * wgt_height

* lithology *.....................................................

      if ( draw_litlog ) then
        call plot( 0.0, 0.0,      3)
        call plot( ( xmax - x_first ) / x_scale, 0.0,      2)
        call plot( ( xmax - x_first ) / x_scale, box_width,3)
        call plot( 0.0, box_width,2)
      endif
```

```
      line(1:1) = '_'
      do while ( line(1:1) .eq. '_' )
        read(13,'(a)',end=600) line
      enddo

100   read(13,'(a)',end=600) line
      if ( line(1:1) .eq.'*' )                          goto 190
      if ( line(1:1) .eq.'_' .or. .not.draw_litlog )  goto 100
      read(line,litform1,iostat=ios)          marker, x1, x2, k
      include '<jd330314.logplot>error.for'
      if ( x1.gt.xmax .or. x2.lt.xmin )                 goto 100

      if ( .not.done ) x1 = x1_keep

      mark1 = mark( x1, xmin )
      mark2 = mark( xmax, x2 )

      if ( marker.eq.'0' .or. marker.eq.'1')  mark1 = .false.
      if ( marker.eq.'0' .or. marker.eq.'2')  mark2 = .false.
      if ( mark_done ) mark1 = .false.

      mark_done = .false.
      if ( marker.eq.'0' .or. marker.eq.'2' )  mark_done = .true.

      x1 = max( x1, xmin )
      x2 = min( x2, xmax )

      x1_keep = x1
      x1_plot = ( x1 - x_first ) / x_scale
      x2_plot = ( x2 - x_first ) / x_scale

      dumma = 0.666 * box_width * 0.125

      done = .true.
      if ( lit_omit .and. ( x2_plot - x1_plot ) .lt. dumma ) then
        done = .false.
        goto 100
      endif

      if ( lit_mark .and. mark1 ) then
        call plot( x1_plot, 0.0,      3 )
        call plot( x1_plot, box_width, 2)
      endif

      if ( lit_raster ) then
        call fill_box( x1_plot, 0., x2_plot-x1_plot, box_width, k )
      endif

      if ( lit_mark .and. mark2 .and. x2 .eq. xmax ) then
        call plot( x2_plot, 0.0,       3)
        call plot( x2_plot, box_width, 2)
      endif

      draw_lithdl = .true.
      goto 100
```

```
190     if ( icelandic ) call new_cha(1)
        y_plot = box_width + exp_dist

* explanations *.....................................................

200     read(13,'(q,a)',end=600) iq, line
        if ( line(1:1) .eq.'*' )                          goto 300
        if ( line(1:1) .eq.'_' .or. .not.draw_explan )    goto 200
        read(line(1:iq),litform2,err=600)            x1, iq, text
        if ( x1.gt.xmax .or. x1.lt.xmin )                 goto 200

        if ( exp_height .gt. 0.0 ) then
          x_plot = ( x1 - x_first ) / x_scale + 0.5 * exp_height
          call symbol(x_plot, y_plot, exp_height, %ref(text), 90., iq)
        endif

        goto 200

* casings *..........................................................

300     read(13,'(q,a)',end=600) iq, line
        if ( line(1:1) .eq.'*' )                          goto 400
        if ( line(1:1) .eq.'_' .or. .not.draw_casing )    goto 300
        read(line(1:iq),litform3,err= 600)      x1, x2, iq, text

        if ( x1.gt.xmax .or. x2.lt.xmin ) then
          y2_plot = -cas_dist + dy
          goto 300
        endif

        mark1 = mark( x1, xmin )
        mark2 = mark( xmax, x2 )

        x1 = max( x1, xmin )
        x2 = min( x2, xmax )

        x1_plot = ( x1 - x_first ) / x_scale
        x2_plot = ( x2 - x_first ) / x_scale
        y1_plot = -cas_dist -cas_height -cas_space + dy
        y2_plot = -cas_dist + dy
        if ( cas_height .eq. 0.0 ) y1_plot = y1_plot + 0.3

        if ( mark1 ) then
          call plot( x1_plot, y1_plot, 3 )
          call plot( x1_plot, y2_plot, 2)
        endif

        call plot( x1_plot, y2_plot, 3)
        call plot( x2_plot, y2_plot, 2)

        if ( mark2 ) then
          call plot( x2_plot, y1_plot, 3)
          call plot( x2_plot, y2_plot, 2)
        endif
```

```
        dumma = x2_plot - cas_height
        dummb = dumma - real(iq) * cas_height- cas_space * 2.
        dummy = y2_plot - cas_height * 0.5

        if ( cas_height .gt. 0.0 .and. dummb .gt. x1_plot ) then
          call symbol ( dumma, dummy, cas_height, %ref(text), 180., iq )
        endif

        dy = dy + cas_space

        draw_cashdl = .true.
        goto 300

* drill_bit *.....................................................

400     read(13,'(q,a)',end=600) iq, line
        if ( line(1:1) .eq.'*' )                          goto 500
        if ( line(1:1) .eq.'_' .or. .not.draw_bittyp ) goto 400
        read(line(1:iq),litform3,err=600)      x1, x2, iq, text
        if ( x1.gt.xmax .or. x2.lt.xmin )                 goto 400

        mark1 = mark( x1, xmin )
        mark2 = mark( xmax, x2 )

        x1 = max( x1, xmin )
        x2 = min( x2, xmax )

        x1_plot = ( x1-x_first ) / x_scale
        x2_plot = ( x2-x_first ) / x_scale

        if ( mark1 ) then
          call im_arrow( x1_plot, -bit_dist, 0., 180., 0, bit_hgt, 45. )
          call im_arrow( x1_plot, -bit_dist, 0., 180., 0, bit_hgt, 90. )
        endif

        dumma = x2_plot - bit_height - 1.0
        dummb = dumma - real(iq)*bit_height - 0.5*bit_height
        dummy = -bit_dist+0.5*bit_height

        if ( dummb .gt. (x1_plot+bit_height) ) then
          call im_arrow( x1_plot, -bit_dist, dummb-x1_plot, 0., 1, 0., 0. )
          call im_arrow( x2_plot, -bit_dist,          -1., 0., 1, 0., 0. )
          if ( bit_height .gt. 0.0 ) then
            call symbol ( dumma, dummy, bit_height, %ref(text), 180., iq )
          endif
        else
          call im_arrow( x1_plot, -bit_dist, x2_plot-x1_plot, 0., 1, 0., 0. )
        endif

        if ( mark2 ) then
          call im_arrow( x2_plot, -bit_dist, 0., 0., 0, bit_hgt, 45. )
        endif

        draw_bithdl = .true.
        goto 400
```

```
* drill_weight *.....................................................

500     read(13,'(q,a)',end=600) iq, line
        if ( line(1:1) .eq.'*' )                          goto 500
        if ( line(1:1) .eq.'_' .or. .not.draw_weight )    goto 500
        read(line(1:iq),litform3,err=600)        x1, x2, iq, text
        if ( x1.gt.xmax .or. x2.lt.xmin )                 goto 500

        mark1 = mark( x1, xmin )
        mark2 = mark( xmax, x2 )

        x1 = max( x1, xmin )
        x2 = min( x2, xmax )

        x1_plot = ( x1 - x_first) / x_scale
        x2_plot = ( x2 - x_first) / x_scale

        if ( mark1 ) then
          call im_arrow( x1_plot, -wgt_dist, 0., 180., 0, wgt_hgt, 45. )
          call im_arrow( x1_plot, -wgt_dist, 0., 180., 0, wgt_hgt, 90. )
        endif

        dumma = x2_plot - wgt_height - 1.0
        dummb = dumma - real(iq)*wgt_height - 0.5*wgt_height
        dummy = -wgt_dist+0.5*wgt_height

        if ( dummb .gt. (x1_plot+wgt_height) ) then
          call im_arrow( x1_plot, -wgt_dist, dummb-x1_plot, 0., 1, 0., 0. )
          call im_arrow( x2_plot, -wgt_dist,          -1., 0., 1, 0., 0. )
          if ( wgt_height.gt.0.0 ) then
            call symbol ( dumma, dummy, wgt_height, %ref(text), 180., iq )
          endif
        else
          call im_arrow( x1_plot, -wgt_dist, x2_plot-x1_plot, 0., 1, 0., 0. )
        endif

        if ( mark2 ) then
          call im_arrow( x2_plot, -wgt_dist, 0., 0., 0, wgt_hgt, 45. )
        endif

        draw_wgthdl = .true.
        goto 500
```

```
* headlines * ..................................................

600     if ( draw_header .and. hdl_height.gt.0.0 ) then

          x = -hdl_dist

          if ( icelandic ) then

            call new_cha(1)

            if ( draw_lithdl ) then
              y = 0.5 * box_width  - 3.5 * hdl_height
              call symbol(x, y,     hdl_height, 'Jarðlög',     90., 7)
            endif

            if ( draw_cashdl ) then
              y = -cas_dist + 0.5 * hdl_height
              call symbol(x*0.5, y, hdl_height, 'Fóðring',   180., 8)
            endif

            if ( draw_bithdl ) then
              y = -bit_dist + 0.5 * hdl_height
              call symbol(x*0.5, y, hdl_height, 'Krónugerð', 180.,10)
            endif

            if ( draw_wgthdl ) then
              y = -wgt_dist + 0.5 * hdl_height
              call symbol(x*0.5, y, hdl_height, 'Álag',      180., 5)
            endif

          else

            if ( draw_lithdl ) then
              y = 0.5 * box_width - 4.5 * hdl_height
              call symbol(x, y,     hdl_height, 'Lithology',   90., 9)
            endif

            if ( draw_cashdl ) then
              y = -cas_dist + 0.5 * hdl_height
              call symbol(x*0.5, y, hdl_height, 'Casing',     180., 6)
            endif

            if ( draw_bithdl ) then
              y = -bit_dist + 0.5 * hdl_height
              call symbol(x*0.5, y, hdl_height, 'Drill bit',  180., 9)
            endif

            if ( draw_wgthdl ) then
              y = -wgt_dist + 0.5 * hdl_height
              call symbol(x*0.5, y, hdl_height, 'Weight',     180., 6)
            endif

          endif

        endif

900     return
        end
```

```
      logical*4 function mark ( x, y )

*     purpose:  determine if bounds of layers are to be drawn

      if ( x .lt. y ) then
        mark = .false.
      else
        mark = .true.
      endif

      return
      end
```

```
      subroutine   drawaqu

*     purpose:     draw aquifers ( one box )

*     subroutines: im_arrow
*                  symbol

      include '<jd330314.logplot>commonblk.for'

      character*80 line,   text
      character*32 aquform
      character*1  reflect

      aquform = '(i1,a,t4,f9.0,t28,q,a)'
      xmin = max(x_first, x_min)
      xmax = min(x_last,  x_max)

100   read(13,'(q,a)',end=900) iq, line
      if ( line(1:1).eq.'_' )  goto 100
      read(line(1:iq),aquform,iostat = ios) n_tips, reflect, x, iq, text
      include '<jd330314.logplot>error.for'

      if ( x.lt.xmin .or. x.gt.xmax ) goto 100
      if ( line(1:1) .eq. ' ' )       n_tips = 1

      if ( upside_down ) then

        if ( reflect .eq. ' ' ) then
          angle   = 90.
          y_plot  = 0.0
        else
          angle   = -90.
          y_plot  = box_width
        endif

      else

        if ( reflect .eq. ' ' ) then
          angle   = -90.
          y_plot  = box_width
        else
          angle   = 90.
          y_plot  = 0.0
        endif

      endif

      x_plot  = ( x - x_first ) / x_scale
      aqu_len = box_width
      aqu_tip = aqu_height * 0.7071
```

```
      do i = 1, n_tips
        call im_arrow ( x_plot, y_plot, aqu_len, angle, 0, aqu_tip, 45.0 )
        aqu_len = aqu_len - aqu_tip
      enddo

      call im_arrow ( x_plot, y_plot, box_width, angle, 1, 0.0, 0.0 )

      if ( iq .gt. 0 .and. aqu_height .gt. 0. ) then
        x_plot  = x_plot  + aqu_height * 0.5
        y_plot  = box_width + aqu_dist
        call symbol ( x_plot, y_plot, aqu_height, %ref ( text ), 90.0, iq )
      endif

      goto 100

900   return
      end
```

```
      subroutine   drawmin

*     purpose:     draw mineral distribution

*     minerals are selected by method of analyses (min_method=3,4,6)
*     or diffrent symbol for each method          (min_method=0)
*     else same symbol for all methods            (min_method=1,2,5)

*     subroutines: notify
*                  im_symbol
*                  symbol
*                  plot

      include '<jd330314.logplot>commonblk.for'

      character*80   line
      character*80   txt_dat(100)
      integer*2      min_dat(100), mineral(100), iq_dat(100)
      integer*2      method

* initialize *.....................................................

      n_dat = 0
      ios   = 0
      xmin  = max ( x_first, x_min)
      xmax  = min ( x_last,  x_max)

* read numbers and mineral names from .min file *..................

      do while ( line(1:1) .ne. '*' .and. ios .eq. 0 )

        read(13,'(q,a)',iostat=ios) iq, line

        if ( line(1:1).ne.'*' .and. line(1:1).ne.'_' .and. ios.eq.0 )  then
          n_dat = n_dat + 1
          read(line(1:iq),'(i3,t6,q,a)',iostat=ios)
     £      min_dat(n_dat), iq_dat(n_dat), txt_dat(n_dat)
          if ( ios.ne.0 ) call notify ('  error in file '//datafile ,22,2,0)
          if ( ios.ne.0 ) goto 900
        endif

      enddo

      box_width = real(n_dat) * min_int + min_int
      call plot ( 0.0, 0.0, 3 )
      call plot ( 0.0, box_width, 2 )

      do i = 1, n_dat

        if ( min_tck.ne.0 ) then
          y_plot = real(i) * min_int
          call plot (     0.0, y_plot, 3 )
          call plot (-min_tck, y_plot, 2 )
        endif

        if ( iq_dat(i).gt.0 .and. min_hgt.gt.0 ) then
          y_plot = real(i) * min_int + 0.5 * min_hgt
          call symbol ( -min_dis, y_plot, min_hgt,
     £    %ref( txt_dat(i) ), min_rot + 90.0, iq_dat(i) )
        endif

      enddo

      if ( min_symbhgt .le. 0. ) goto 900
```

```
* read remaining part of .min file plot mineral distribution *.........

100     read(13,'(q,a)',end=900) iq, line
        if ( line(1:1).eq.'_' )  goto 100
        read(line(1:iq),'(i1,t4,f9.2,t16,q,a)',iostat=ios) method,x,iqq,line

        include '<jd330314.logplot>error.for'
        if ( x .lt. xmin .or. x .gt. xmax )                goto 100
        if ( mod(iqq,4) .ne. 0 .or. line(iqq:iqq) .eq. ' ')  ios = 1
        include '<jd330314.logplot>error.for'

*  select one method or fixed symbol for all methods *.................

        if ( min_method .eq. 1 )     method = 1
        if ( min_method .eq. 2 )     method = 2
        if ( min_method .eq. 3 .and. method .ne. 3 ) goto 100
        if ( min_method .eq. 4 .and. method .ne. 4 ) goto 100
        if ( min_method .eq. 5 )     method = 5
        if ( min_method .eq. 6 .and. method .ne. 6 ) goto 100
        if ( method .eq. 6 )         method = 1

* plot distribution of secondary minerals *............................

        do i = 1, iqq/4

          read(line(4*i-3:4*i),'(i4)') mineral(i)

          do j = 1, n_dat

            if ( abs(mineral(i)) .eq. min_dat(j) ) then

              x_plot = ( x - x_first ) / x_scale
              y_plot = min_int * real( j )
              call im_symbol ( x_plot, y_plot, min_symbhgt, 90.0, method )

              if ( mineral(i) .lt. 0 .and. min_brahgt .gt. 0.0 ) then
                x_plot = x_plot + min_brahgt * 0.50
                y_plot = y_plot - min_brahgt * 1.25
                call symbol ( x_plot, y_plot, min_brahgt,'( )', 90.0, 3 )
              endif

            endif

          enddo

        enddo

        goto 100

900     return
        end
```

```
      subroutine   drawbor

*     purpose:     draw drill_rate

*     subroutines: symbol
*                  plot

      include '<jd330314.logplot>commonblk.for'

      character*80 line
      real*4       mirror, translate

      if ( upside_down ) then
        mirror    = -1.
        translate = box_width
      else
        mirror    = 1.
        translate = 0.
      endif

      xmin = max(x_first, x_min)
      xmax = min(x_last,  x_max)
      ymin = y_min
      ymax = y_max

      lint = lin_typ ( curve_type )

100   read(13,'(a)',end=900) line
      if ( line(1:1).eq.'_' .or. line.eq.' ') goto 100
      read(line,dataform,iostat = ios)        x1, y1
      include '<jd330314.logplot>error.for'

      if ( x1.gt.xmax ) goto 900
      if ( x1.lt.xmin .or. y1.lt.ymin .or. y1.gt.ymax ) goto 100

      x1_plot = ( x1 - x_first ) / x_scale
      y1_plot = mirror * ( y1 - y_first ) / y_scale + translate
      call plot ( x1_plot, y1_plot, 3 )
      call plot ( x1_plot, y1_plot, 2 )

      do while ( .true. )

        read(13,'(a)',end=900) line
        if ( line(1:1).eq.'_' .or. line.eq.' ' ) goto 100
        read(line,dataform,iostat = ios)         x2, y2
        include '<jd330314.logplot>error.for'

        if ( x2.gt.xmax ) goto 900
        if ( x2.lt.xmin .or. y2.lt.ymin .or. y2.gt.ymax ) goto 100

        x2_plot = ( x2 - x_first ) / x_scale
        y2_plot = mirror * ( y2 - y_first ) / y_scale + translate
        call plot( x2_plot, y1_plot, 2 )
        call plot( x2_plot, y2_plot, 2 )
        y1_plot = y2_plot

      enddo

900   return
      end
```

```
      subroutine   drawlog

*     purpose:     draw borehole measurement (log)

*     subroutines: plot

      include '<jd330314.logplot>commonblk.for'

      character*80 line
      logical      first
      real*4       mirror, translate

      if ( upside_down ) then
        mirror    = -1.
        translate = box_width
      else
        mirror    = 1.
        translate = 0.
      endif

      xmin = max(x_first, x_min)
      xmax = min(x_last,  x_max)
      ymin = y_min
      ymax = y_max

      lint = lin_typ ( curve_type )

100   first = .true.

      do while ( .true. )

        read(13,'(a)',iostat=ios,end=900)        line
        if ( line(1:1).eq.'_' .or. line.eq.' ' ) goto 100
        read(line,dataform,iostat = ios)         x, y
        include '<jd330314.logplot>error.for'

        if ( x.gt.xmax ) goto 900
        if ( x.lt.xmin .or. y.lt.ymin .or. y.gt.ymax ) goto 100

        x_plot = ( x - x_first ) / x_scale
        y_plot = mirror * ( y - y_first ) / y_scale + translate

        if ( first ) then
          first  = .false.
          call plot( x_plot, y_plot, 3 )
          call plot( x_plot, y_plot, 2 )
        else
          call plot( x_plot, y_plot, 2 )
        endif

      enddo

900   return
      end
```

```
*       module error.for is included in all draw_modules

*       purpose:     error handling

*       subroutines: lib$put_screen
*                    lib$erase_page
*                    nonotify
*                    noecho
*                    spawn
*                    append

        if ( ios .gt. 0 ) then

          call nonotify (' FIG. 11.3  FATAL ERROR          ',2,2,0)
          call lib$put_screen (' file: '//datafile          ,6,,)
          call lib$put_screen (' line: '//line              ,7,,)
          call lib$put_screen (' you have 4 choices        ',15,,)
          call lib$put_screen (' 1  continue               ',17,,)
          call lib$put_screen (' 2  spawn                  ',18,,)
          call lib$put_screen (' 3  quit this box          ',19,,)
          call lib$put_screen (' 4  exit plot module       ',20,,)

          do while ( .true. )

            call lib$put_screen ('     your choice  '     ,22,,1)
            call noecho (answ,1,iq,1)

            if ( answ .eq. '1' ) then

              call nonotify ( ' reading '//datafile , 22,2,0)
              goto 100

            else if ( answ .eq. '2' ) then

              call lib$erase_page (15,1)
              close (unit=13)
              call spawn (' ')
              open (unit=13,file=datafile,status='old',err=900,readonly)
              call nonotify ( ' reading '//datafile , 22,2,0)
              goto 100

            else if ( answ .eq. '3' ) then

              goto 900

            else if ( answ .eq. '4' ) then

              call append
              goto 900

            endif

          enddo

        endif
```

```
      program legend

* program to draw explanations to geological section drawn by program LOGPLOT
* read items from file legend.dat on default directory or <jd330314.datafiles>
* read .LIT file or type numbers to bee explained

* subroutines *.....................................................

*       lib$erase_page    symbol      gin_bxyp    nonotify   outf
*       lib$put_screen    factor      num_string  notify     outi
*       plots             wait_s      rectan      noecho     getf
*       plot              new_cha     fill_box    esc6       geti

* specification *....................................................

      integer      max_item
      parameter    ( max_item = 20 )
      character*80 line
      character*80 file
      character*80 text (max_item)
      character*1  answ
      real*4       x_first     /  8.00 /  ! lower left corner of first box
      real*4       y_first     /  3.00 /  ! lower left corner of first box
      real*4       box_height  /  1.50 /  ! height of box
      real*4       box_width   /  2.50 /  ! width  of box
      real*4       txt_height  /  0.30 /  ! height of characters
      real*4       dx          /  1.00 /  ! vertical   distance between boxes
      real*4       dy          / 12.00 /  ! horizontal distance between boxes
      real*4       scale       /  1.00 /  ! scale for whole picture
      integer*2    n_col       /  1    /  ! number of columns
      integer*2    legends     /  0    /  ! number of legends
      integer      ibox        /  1    /  ! box in column i
      integer      jbox        /  1    /  ! box in row j
      integer*2    item
      integer*2    ipen
      integer*2    iqq
      integer*2    no
      integer*2    idraw  (max_item)
      integer*2    iqtext (max_item)
      logical      draw   (max_item)
      logical      done   (2)
      logical      lit_raster  / .true. / ! draw rasters
      byte         chin

      call plots ( 1729, 0, 7 )
```

```
* open legend.dat and read items exit if error *.....................

        open(11,file='legend.dat',status='old',readonly,iostat=ios11)

        if ( ios11 .ne. 0 ) then
          file = 'osdisk1:<jd330314.datafiles>legend.dat'
          open(11,file=file,status='old',readonly,iostat=ios)
          if ( ios .ne. 0 ) call exit
        endif

        do while ( ios .eq. 0 .and. item .le. max_item )
          read(11,'(t6,q,a)',iostat=ios)  iqtext (item+1), text (item+1)
          if ( ios .eq. 0 )  item = item + 1
          if ( ios .gt. 0 )  call exit
        enddo

        close ( unit=11 )

* what items are to be drawn *.....................................

100     call nonotify ( '  PROGRAM LEGEND '                          , 10,1,0)

        if ( ios11 .ne. 0 ) then
          call lib$put_screen ('  legend.dat not found now using  ' ,2,45,2)
          call lib$put_screen ('  <jd330314.datafiles>legend.dat  ' ,3,45,2)
        endif

        legends = 0
        do i = 1, max_item
          draw(i) = .false.
        enddo

        call lib$put_screen (' 1  read .LIT file     ',16,10,)
        call lib$put_screen (' 2  type numbers       ',17,10,)
        call lib$put_screen (' 3  exit               ',18,10,)
        call lib$put_screen ('    your choice  '       ,21,10,)
        call noecho ( answ, 1, iq, 0 )
        call lib$erase_page (1,1)
```

```
if ( answ .eq. '1' )  then

  call lib$put_screen ('  enter name of .LIT file   ',21,2,)
  read (5,'(a)',end=100)  file
  open (12,file=file,status='old',defaultfile='.lit',err=100)

  do while ( .not.done(2) )

    read(12,'(q,a)',iostat=ios) iq, line

    if ( ios.eq.0 .and. line(1:1).eq.' ' ) then
      read(line,'(t28,i2)',iostat=ios)  i
      if ( ios.gt.0 .or. i.lt.1 .or. i.gt.max_item ) then
        goto 100
      else if ( .not.draw(i) )  then
        draw(i) = .true.
        legends = legends + 1
      endif
    else if ( line(1:1).eq.'*' ) then
      if ( done(1) ) done(2) = .true.
      done(1) = .true.
    else if ( ios.ne.0 ) then
      done(2) = .true.
    endif

  enddo

  close ( unit=12 )

else if ( answ .eq. '2' )  then

  do i = 1, item
    write(6,'(i4,2x,a)') i, text (i) (1:iqtext(i))
  enddo

  call lib$put_screen ('  enter numbers  (eg 1,4,9)  ' ,23,1,1)
  read (5,'(q,a)',iostat=ios)   iq, line
  if ( iq.eq.0 .or. ios.ne.0 )  goto 100

  idummy = num_string (line(1:iq),',') + 1
  nblank = num_string (line(1:iq),' ')

  read(line(1:iq),'(20i)',iostat=ios) (idraw(i),i=1,idummy)

  if ( ios.eq.0 .and. nblank.eq.0 ) then
    do i = 1, idummy
      if ( idraw(i).lt.1 .or. idraw(i).gt.max_item ) then
        goto 100
      else
        draw ( idraw (i) ) = .true.
        legends = legends + 1
      endif
    enddo
  else
    goto 100
  endif

else if ( answ .eq. '3' )  then
  call exit
else
  goto 100
endif
```

```
      if ( legends .eq. 0 ) goto 100

* setup *.....................................................

      call nonotify ( '  PROGRAM LEGEND '                      , 6,1,)
      call outi     (' ',  legends                             ,2,69,2)

      do while ( .true. )

        call lib$erase_page (9,1)
        call outf  (' 1  x coordinate          ',  x_first     ,10,1,)
        call outf  (' 2  y coordinate          ',  y_first     ,11,1,)
        call outf  (' 3  box height            ',  box_height  ,12,1,)
        call outf  (' 4  box width             ',  box_width   ,13,1,)
        call outf  (' 5  vertical distance     ',  dx          ,14,1,)
        call outf  (' 6  horizontal distance   ',  dy          ,15,1,)
        call outf  (' 7  height of characters  ',  txt_height  ,16,1,)
        call outf  (' 8  scale                 ',  scale       ,17,1,)
        call outi  (' 9  number of columns     ',  n_col       ,18,1,)

        if ( lit_raster ) then
          call lib$put_screen (' 0  draw rasters '             ,19,1,)
        else
          call lib$put_screen (' 0 '                           ,19,1,0)
          call lib$put_screen (' omit rasters '                ,19,4,2)
        endif

        call lib$erase_page (21,1)
        call lib$put_screen ('    <ret> = no change  '         ,21,1,)
        call noecho ( answ, 1, iq, 0 )

        if ( answ .eq. '1' )  then
          call lib$put_screen('   to digitize enter 999.00   '   ,24,1,2)
          call getf (' x coordinate of first box  ',  x_first    ,21,4,1)
        else if ( answ .eq. '2' )  then
          call lib$put_screen('   to digitize enter 999.00   '   ,24,1,2)
          call getf (' y coordinate of first box  ',  y_first    ,21,4,1)
        else if ( answ .eq. '3' )  then
          call getf (' box height  ',                 box_height ,21,4,1)
        else if ( answ .eq. '4' )  then
          call getf (' box width  ',                  box_width  ,21,4,1)
        else if ( answ .eq. '5' )  then
          call getf (' vertical distance between boxes  ',   dx  ,21,4,1)
        else if ( answ .eq. '6' )  then
          call getf (' horizontal distance between boxes  ', dy  ,21,4,1)
        else if ( answ .eq. '7' )  then
          call getf (' height of characters  ',       txt_height ,21,4,1)
        else if ( answ .eq. '8' )  then
          call getf (' scale for whole picture  ',    scale      ,21,4,1)
          call factor ( scale )
        else if ( answ .eq. '9' )  then
          call geti (' number of columns  ',          n_col      ,21,4,1)
        else if ( answ .eq. '0' )  then
          lit_raster = .not.lit_raster
```

```
  else if ( ichar(answ) .eq. 13 )  then

    if ( x_first .eq. 999. .or. y_first .eq. 999. ) then
      call lib$erase_page (21,1)
      call gin_bxyp ( chin, x_first, y_first, ipen)
    endif

    do no = 1, max_item

      if ( draw (no) ) then

        x = x_first + real ( ibox - 1 ) * ( dx + box_height )
        y = y_first + real ( jbox - 1 ) * ( dy + box_width  )

        if ( lit_raster ) then
          call new_cha  ( 0 )
          call rectan   ( x, y, box_height, box_width     )
          call fill_box ( x, y, box_height, box_width, no )
          call new_cha  ( 1 )
        else
          call rectan   ( x, y, box_height, box_width     )
        endif

        if ( txt_height .gt. 0. ) then
          x = x + 0.5 * ( box_height + txt_height )
          y = y + box_width + 1.25 * txt_height
          call symbol (x,y,txt_height,%ref(text(no)),90.0,iqtext(no))
        endif

        irow1 = legends / n_col
        idumm = mod (legends,n_col)
        if ( idumm.ne.0 .and. jbox.le.idumm ) irow1 = irow1 + 1
        irow2 = max(irow1,irow2)

        if ( ibox .ge. irow1 ) then
          ibox = 1
          jbox = jbox + 1
        else
          ibox = ibox + 1
        endif

      endif
    enddo

    call plot   ( 0, 0, 0 )
    goto 200

  endif
enddo
```

```
* headline *.....................................................

200     call nonotify ( '  PROGRAM LEGEND '                ,  10,1,)
        call lib$put_screen( ' want headline   <ret>=yes  ',  21,1,)
        call noecho ( answ, 1, iq, 0 )
        if ( ichar(answ) .ne. 13 )  goto 300

        x   =  x_first - 1.20
        y   =  y_first
        hgt =  1.667 * txt_height
        iqq =  27
        line= 'Skýringar við jarðlagasnið'

        do while ( .true. )

          call nonotify ( '  LEGEND '                          ,10,1,)
          call outf   (' 1  x coordinate of headline   ',  x    ,15,1,)
          call outf   (' 2  y coordinate of headline   ',  y    ,16,1,)
          call outf   (' 3  height of characters       ',  hgt  ,17,1,)
          call lib$put_screen( ' 4  headline  '//line           ,18,1,)

          call lib$put_screen ('    <ret> = no change  '        ,21,1,)
          call noecho ( answ, 1, iq, 0 )
          call lib$erase_page (21,1)

          if ( answ .eq. '1' )  then
            call lib$put_screen('   to digitize enter 999.00   ',24,1,2)
            call getf ('    x coordinate of headline  ',  x     ,21,1,1)
          else if ( answ .eq. '2' )  then
            call lib$put_screen('   to digitize enter 999.00   ',24,1,2)
            call getf ('    y coordinate of headline  ',  y     ,21,1,1)
          else if ( answ .eq. '3' )  then
            call getf ('    height of characters  ',      hgt   ,21,1,1)
          else if ( answ .eq. '4' )  then
            call lib$put_screen( '    enter new headline  '     ,21,1,1)
            read(5,'(q,a)',end=200,iostat=ios) iqq, line
          else if ( ichar(answ) .eq. 13 )  then
            if ( x.eq.999. .or. y.eq.999. ) call lib$erase_page (21,1)
            if ( x.eq.999. .or. y.eq.999. ) call gin_bxyp ( chin, x, y, ipen)
            if ( hgt .gt. 0. ) then
              call symbol ( x, y, hgt, %ref ( line ), 90.0, iqq )
              call plot   ( 0, 0, 0 )
            endif
            goto 300
          endif
        enddo
```

```
* further explanations *.....................................

300     call nonotify ( '  PROGRAM LEGEND '  ,10,1,0)
        call lib$put_screen('  want further explanations  <ret>=yes  ',21,1,0)
        call noecho ( answ, 1, iq, 0 )
        if ( ichar(answ) .ne. 13 )  goto 900

        x   =  x_first + real(irow2) * ( dx + box_height) + txt_height
        y   =  y_first
        dx  =  txt_height * 2.50
        dy  =  box_width + dy
        hgt =  txt_height

        do while ( .true. )

          call nonotify ( '  PROGRAM LEGEND '                      ,10,1,)
          call outf   (' 1  x coordinate of first line ',  x       ,13,1,)
          call outf   (' 2  y coordinate of first line ',  y       ,14,1,)
          call outf   (' 3  height of characters       ',  hgt     ,15,1,)
          call outf   (' 4  distance between lines     ',  dx      ,16,1,)
          call outf   (' 5  horizontal distance        ',  dy      ,17,1,)
          call outi   (' 6  number of columns          ',  n_col   ,18,1,)

          call lib$erase_page (21,1)
          call lib$put_screen ('    <ret> = no change  '           ,21,1,)
          call noecho ( answ, 1, iq, 0 )

          if ( answ .eq. '1' )  then
            call lib$put_screen('   to digitize enter 999.00   '     ,24,1,2)
            call getf ('    x coordinate of first line  ',       x   ,21,1,1)
          else if ( answ .eq. '2' )  then
            call lib$put_screen('   to digitize enter 999.00   '     ,24,1,2)
            call getf ('    y coordinate of first line  ',       y   ,21,1,1)
          else if ( answ .eq. '3' )  then
            call getf ('    height of characters  ',            hgt ,21,1,1)
          else if ( answ .eq. '4' )  then
            call getf ('    distance between lines  ',          dx  ,21,1,1)
          else if ( answ .eq. '5' )  then
            call getf (' horizontal distance between columns  ', dy  ,21,4,1)
          else if ( answ .eq. '6' )  then
            n_col = -1
            do while ( n_col .lt. 1 )
              call geti ('    number of columns  ',          n_col  ,21,1,1)
            enddo
```

```
          else if ( ichar(answ) .eq. 13 )  then

            if ( x.eq.999. .or. y.eq.999. ) call lib$erase_page (21,1)
            if ( x.eq.999. .or. y.eq.999. ) call gin_bxyp ( chin, x, y, ipen)

            call lib$erase_page (1,1)
            call lib$put_screen( ' enter up to 80 characters per line  ',2,1,2)
            call lib$put_screen( ' to stop  press ..... CTRL-Z  ',3,1,2)
            call lib$set_cursor (7,1)

            if ( hgt .gt. 0. .and. n_col .gt. 0 ) then
              do while ( .true. )
                do i = 1, n_col
                  y_pos  = real( i - 1 ) * dy + y
                  read(5,'(q,a)',end=900,iostat=ios) iqq, line
                  call symbol ( x, y_pos, hgt, %ref ( line ), 90.0, iqq )
                  call plot   ( 0, 0, 0 )
                enddo
                x = x + dx
              enddo
            endif

          endif
        enddo

* end section *.....................................................

900     call lib$erase_page (1,1)
        call plot   ( 0, 0, 999 )
        call exit
        end
```

```
      program textplot2

*     forrit sem skrifar texta a teiknitaeki

      call plots     ( 1729, 0, 7 )
      call newpen    ( 1 )
      call text_plot ( 1, 1.0, 0.45, 0.0, 15.0 )
      call plot      ( 999, 999, 999 )

      call exit
      end
```

```
      program osmerki

*     forrit til ad teikna merki Orkustofnunar og tilheyrandi textalinur

      call plots ( 1729, 0, 7 )
      call os_merki( 1.5,1.5,1.,90.,'JHD-BJ-9000-IM',14,'84.11.1001   T',14 )
      call plot ( 999, 999, 999 )
      call exit
      end
```

7 FORRITASAFNIÐ IMLIBRARY

TEIKNIFORRIT

SKJÁFORRIT I

SKJÁFORRIT II

ANNAÐ

* ekki ætlað til almennra nota nema að vel athuguðu máli.
** úr forritasafninu OSLIB:OSPLOTLIB/LIB og því ekki birt hér.

```
      subroutine fill_box ( x, y, width, height, no )

*     forrit sem fyllir retthyrning með taknum
*     undirforrit: plot, symbol

*     x, y       : hnit haegra nedra hornpunkts retthyrnings
*     width      : breidd retthyrnings
*     height     : haed   retthyrnings
*     no         : numer takns  1 - 20

      implicit  integer*2  (i-n)
      character*7       symb(20)

      data  symb(1),  symb(2)          /'       ','       '/
      data  symb(3),  symb(4),  symb(5)/'+ + + +','+ + + +','+ + + +'/
      data  symb(6),  symb(7),  symb(8)/' +   + ','x x x x','x x x x'/
      data  symb(9), symb(10), symb(11)/'x x x x','v v v v',' / / / '/
      data symb(12), symb(13), symb(14)/' X X X ','< > < >','< > < >'/
      data symb(15), symb(16), symb(17)/'.......','o o o o','öööööööö'/
      data symb(18), symb(19), symb(20)/'XXXXXXX','       ','- - - -'/

      if ( no .eq. 19 ) return

      if ( no.eq.1 .or. no.eq.3 .or. no.eq.7 .or. no.eq.13 ) then
        call plot(x,          y + height * 0.50, 3)
        call plot(x + width,  y + height * 0.50, 2)
      else if ( no.eq.2 .or. no.eq.4 .or. no.eq.8 ) then
        call plot(x,          y + height * 0.25, 3)
        call plot(x + width,  y + height * 0.25, 2)
        call plot(x + width,  y + height * 0.50, 3)
        call plot(x,          y + height * 0.50, 2)
        call plot(x,          y + height * 0.75, 3)
        call plot(x + width,  y + height * 0.75, 2)
      endif

      if ( no .ge. 3  .and.  no .le. 20 ) then

        hgt   = height * 0.1250                                    ! haed takna
        lines = int( width / hgt + 0.001  )                        ! linufjoldi
        xinc  = abs( width - hgt * real(lines) ) / real(lines+1) ! increment

        xpos  = x + hgt + xinc
        ypos  = y + hgt * 0.666

        if ( no .eq. 15 ) xpos = xpos - hgt * 0.5
        if ( no .eq. 17 ) xpos = xpos + hgt * 0.5

        do i = 1, lines
          call symbol ( xpos, ypos, hgt, %ref( symb(no) ), 90.0, 7 )
          xpos  = xpos + hgt + xinc
        enddo

        if ( lines.lt.1 .and. width.gt.0. ) then                   ! ein lina
          xpos = x + ( hgt + width ) * 0.5
          if ( no .eq. 15 ) xpos = xpos - hgt * 0.5
          if ( no .eq. 17 ) xpos = xpos + hgt * 0.5
          call symbol ( xpos, ypos, hgt, %ref( symb(no) ), 90.0, 7 )
        endif

      endif

      return
      end
```

```
      subroutine head_line ( x, y, hgt, rot, char_slant )

*     purpose:        interactive program to draw headlines
*                     read text from file or terminal

*     subroutines:    lib$erase_page
*                     lib$erase_line
*                     lib$put_screen
*                     lib$set_cursor
*                     lib$get_lun
*                     lib$free_lun
*                     nonotify
*                     noecho
*                     echo
*                     outf
*                     getf
*                     gin_bxyp
*                     set_slant
*                     symbol
*                     plot

* specify variables and initialize *.................................

      parameter      ( maxlines = 100 )   ! max number of headlines
      character*256  line  ( maxlines )   ! headlines
      integer*2      ncha  ( maxlines )   ! number of characters in headlines
      integer*2      lcount               ! current headline
      integer*2      lines                ! number of headlines
      character*72   file
      character*1    answ
      byte           chin
      real*4         x_pos,    y_pos,  height,  rotate, slant

      x_pos    =  x                      ! x center of headline
      y_pos    =  y                      ! y center of headline
      height   =  hgt                    ! height of characters
      rotate   =  rot                    ! rotation of headline
      slant    =  char_slant             ! slant of characters
```

```
* select action *.....................................................

100     call nonotify ( '  HEADLINE  ', 12,1,0)

        do i = 1, maxlines
          line ( i ) = ' '
          ncha ( i ) = 0
        enddo

        ios    = 0
        lines  = 0
        lcount = 1

        call lib$put_screen (' 1  read .TXT file            ',16,10,)
        call lib$put_screen (' 2  enter text from terminal  ',17,10,)
        call lib$put_screen (' 3  exit the headline module  ',18,10,)
        call lib$put_screen ('    your choice '              ,21,10,)

        call noecho ( answ, 1, iq, 0 )

        if ( answ .eq. '1' )  then                          ! read txt file

          call lib$get_lun    ( lun )
          call lib$erase_page ( 9,1 )
          call lib$put_screen ('  enter name of .TXT file  ' ,16,10,)
          read(5,'(a)',end=100)   file
          open(lun,file=file,status='old',defaultfile='.txt',err=100)

          do while ( lines .lt. maxlines .and. ios .eq. 0 )
            read(lun,'(q,a)',iostat=ios) ncha(lines+1), line(lines+1)
            if ( ios .eq. 0 ) then
              lines = lines + 1
            endif
          enddo

          close ( lun )
          call lib$free_lun ( lun )

        else if ( answ .eq. '2' )  then                     ! enter headlines

          call lib$erase_page ( 1,1 )
          call lib$put_screen ('  Enter headlines (<100 lines)  ',1,1,2)
          call lib$put_screen ('  CTRL-Z  ..... when finished  ',2,1,2)
          call lib$set_cursor(5,1)

          do while ( lines .lt. maxlines .and. ios .eq. 0 )
            read(5,'(q,a)',iostat=ios) ncha(lines+1), line(lines+1)
            if ( ios .eq. 0 ) then
              lines = lines + 1
            endif
          enddo

        else if ( answ .eq. '3' )  then                     ! return

          call lib$erase_page (1,1)
          return

        else

          goto 100

        endif
```

```
* main section *.....................................................

        do while ( lcount .le. lines )

          call nonotify ( '  HEADLINE  ', 3,1,0)

          ndumm   = -num_string(line(lcount),'f') -num_string(line(lcount),'f')
          ndumm   = ndumm + ncha(lcount)
          dummy   = height * ( real( ndumm ) * 0.5 - 0.2 )

          cosan   = cos( rotate * 0.017453292 )
          sinan   = sin( rotate * 0.017453292 )

          x_left  = x_pos - dummy * cosan
          x_right = x_pos + dummy * cosan
          y_left  = y_pos - dummy * sinan
          y_right = y_pos + dummy * sinan

          if ( x_pos.ne.999. .and. y_pos.ne.999. ) then
            call outf ( '  x left  ', x_left    ,6,60,2)
            call outf ( '  x right ', x_right   ,7,60,2)
            call outf ( '  y left  ', y_left    ,8,60,2)
            call outf ( '  y right ', y_right   ,9,60,2)
          endif

          call lib$put_screen ( ' 1  headline    '// line(lcount) ,12,,)
          call outf ( ' 2  x_center of headline ',   x_pos        ,13,,)
          call outf ( ' 3  y_center of headline ',   y_pos        ,14,,)
          call outf ( ' 4  height of characters ',   height       ,15,,)
          call outf ( ' 5  rotation of headline ',   rotate       ,16,,)
          call outf ( ' 6  character slant      ',   slant        ,17,,)
          call lib$put_screen ( ' 7  omit this headline  '        ,18,,)

          call lib$put_screen ( '    <ret> = no change    '       ,21,,)
          call noecho ( answ, 1, iq, 0 )

          if ( answ .eq. '1' ) then
            ios = 1
            do while ( ios .ne. 0 )
              call lib$erase_line ( 21,1 )
              call lib$put_screen ( '    enter new headline   ' ,21,1,1)
              read(5,'(q,a)',iostat=ios) ncha(lcount), line(lcount)
            enddo
          else if ( answ .eq. '2' ) then
            call lib$put_screen(' to digitize center'           ,24,1,2)
            call lib$put_screen(' of text enter 999 '             ,,,2)
            call getf ('    x center of headline    ',  x_pos   ,21,1,1)
          else if ( answ .eq. '3' ) then
            call lib$put_screen(' to digitize center'           ,24,1,2)
            call lib$put_screen(' of text enter 999 '             ,,,2)
            call getf ('    y center of headline    ',  y_pos   ,21,1,1)
          else if ( answ .eq. '4' ) then
            call getf ('    height of characters    ',  height  ,21,1,1)
          else if ( answ .eq. '5' ) then
            call getf ('    rotation of headline    ',  rotate  ,21,1,1)
          else if ( answ .eq. '6' ) then
            call getf ('    character slant         ',  slant   ,21,1,1)
          else if ( answ .eq. '7' ) then
            lcount = lcount + 1
```

```
          else if ( ichar(answ) .eq. 13 .and. height .gt. 0. ) then

            if ( x_pos .eq. 999. .or. y_pos .eq. 999. ) then
              call lib$erase_page (21,1)
              call gin_bxyp ( chin, x_pos, y_pos, ipen)
              x_left  = x_pos - dummy * cosan
              y_left  = y_pos - dummy * sinan
            endif

c           call set_slant ( -slant )
            call symbol ( x_left, y_left, height,
     £      %ref(line(lcount)), rotate, ncha(lcount) )
c           call set_slant ( slant )
            call plot ( 0, 0, 0 )

            call lib$erase_page (21,1)
            call lib$put_screen ('    rewrite same headline',,,)
            call lib$put_screen ('  <ret>=no   else=yes    ',,,)
            call echo ( answ, 1, iq, 0 )

            if ( ichar(answ) .eq. 13 ) then

              lcount = lcount + 1
              x_pos  = x_pos + 1.75 * height * sinan
              y_pos  = y_pos - 1.75 * height * cosan

            else

              call lib$erase_line (22,1)
              call lib$put_screen ('    same position or new ',,,)
              call lib$put_screen ('  <ret>=new  else=same   ',,,)
              call echo ( answ, 1, iq, 0 )

              if ( ichar(answ) .eq. 13 ) then
                x_pos = x_pos + 1.75 * height * sinan
                y_pos = y_pos - 1.75 * height * cosan
              endif

            endif

          endif
        enddo

        goto 100

        end
```

```
      subroutine im_arrow ( x0, y0, arrowlen, angle, line, arrowtip, theta )

*     forrit sem teiknar orvar (eina ör i hverju kalli)
*     undirforrit: plot

*     x0, y0      : upphafspunktur orvar
*     arrowlen    : lengd orvar
*     angle       : horn orvar vid x_as
*     line        : adeins oddur dreginn ef line = 0
*     arrowtip    : lengd haka i oddinum    0 sleppt
*     theta       : hornid milli orvarinnar og hakanna i oddinum

      implicit integer*2 (i-n)

      x1 = x0 + arrowlen * cos( 0.017453292 * angle )
      y1 = y0 + arrowlen * sin( 0.017453292 * angle )

      x2 = x1 + arrowtip * cos( 0.017453292 * ( 180.0 + angle - theta ) )
      y2 = y1 + arrowtip * sin( 0.017453292 * ( 180.0 + angle - theta ) )

      x3 = x1 + arrowtip * cos( 0.017453292 * ( 180.0 + angle + theta ) )
      y3 = y1 + arrowtip * sin( 0.017453292 * ( 180.0 + angle + theta ) )

      if ( line .ne. 0 .and. arrowlen .ne.0. ) then
        call plot (x0, y0, 3)
        call plot (x1, y1, 2)
      endif

      if ( arrowtip .ne. 0. ) then
        call plot (x2, y2, 3)
        call plot (x1, y1, 2)
        call plot (x3, y3, 2)
      endif

      return
      end
```

```
      subroutine im_axis ( xpos, ypos, axlen, first, scale, angle,
     £                     line, tickint, tackint, ticklen, tacklen,
     £                     ndec, spaceno, markno, hgtno, distno,
     £                     text, nst, hgtext, distext )

*     undirforrit sem teiknar as vid linurit
*     undirforrit im_tick, im_numb, im_text

*     xpos, ypos  : upphafspunktur ass
*     axlen       : lengd ass
*     first       : upphafsgildi a asnum
*     scale       : kvardi  gagnaeiningar/cm
*     angle       : horn ass vid x_as teiknitaekis
*     line        : aslina ekki dreginn ef  line = 0
*     tickint     : bil milli styttri haka a asnum
*     tackint     : bil milli lengri  haka a asnum
*     ticklen     : lengd styttri haka  <0 nedan  0 sleppt  >0 ofan
*     tacklen     : lengd lengri  haka  <0 nedan  0 sleppt  >0 ofan
*     ndec        : fjoldi aukastafa i tolum sem merkja asinn
*     spaceno     : bil milli talna a asnum i gagnaeiningum
*     markno      : snua tolum       <0 -90 deg   0 samsida as   >0 90 deg
*     hgtno       : haed tolustafa                0 sleppt
*     distno      : fjarlaegd talna fra as       <0 nedan        >0 ofan
*     text        : texti vid asinn
*     nst         : fjoldi stafa i texta       0 sleppt
*     hgtext      : haed texta    <0 gagnst.   0 sleppt   >0 samsida as
*     distext     : fjarlaegd texta fra as    <0 nedan    >0 ofan

      implicit integer*2 (i-n)
      character*(*)      text

      call im_tick ( xpos, ypos, axlen, first, scale, angle,
     £               line, tickint, tackint, ticklen, tacklen )

      call im_numb ( xpos, ypos, axlen, first, scale, angle,
     £               ndec, spaceno, markno, hgtno, distno )

      call im_text ( xpos, ypos, axlen, angle,
     £               text, nst, hgtext, distext )

      return
      end
```

```
      subroutine im_tick ( xpos, ypos, axlen, first, scale, angle,
     £                     line, tickint, tackint, ticklen, tacklen )

*     undirforrit sem dregur aslinu og hok

*     xpos, ypos  : upphafspunktur ass
*     axlen       : lengd ass
*     first       : upphafsgildi a asnum
*     scale       : kvardi  gagnaeiningar/cm
*     angle       : horn ass vid x_as teiknitaekis
*     line        : aslina ekki dreginn ef  line = 0
*     tickint     : bil milli styttri haka a asnum
*     tackint     : bil milli lengri  haka a asnum
*     ticklen     : lengd styttri haka  <0 nedan  0 sleppt  >0 ofan
*     tacklen     : lengd lengri  haka  <0 nedan  0 sleppt  >0 ofan

      implicit integer*2 (i-n)
      logical  large

      if ( scale .eq. 0. ) return

      del   = 0.0
      fpn   = first

      cosan = cos( angle * 0.017453292 )
      sinan = sin( angle * 0.017453292 )

      if ( line .ne. 0 ) then
        call plot (xpos, ypos, 3)
        call plot (xpos + axlen * cosan, ypos + axlen * sinan, 2)
        call plot (xpos, ypos, 2)
      endif

      if ( ticklen .eq. 0. .and. tacklen .eq. 0. ) return
      if ( tickint .eq. 0. .or.  tackint .eq. 0. ) return
      if ( mod(tackint,tickint) .ne. 0 )           return
      interval = int ( tackint / tickint )

      do 100 i=1,10000

         large = .false.
         if ( del .ge. axlen * 1.00001 ) return
         if ( mod(i-1,interval) .eq. 0 ) large = .true.

         xplot = xpos + del * cosan
         yplot = ypos + del * sinan

         if ( .not.large .and. ticklen .ne. 0. ) then
           call plot (xplot, yplot, 3)
           call plot (xplot - ticklen * sinan, yplot + ticklen * cosan, 2)
         endif

         if ( large .and. tacklen .ne. 0. ) then
           call plot (xplot, yplot, 3)
           call plot (xplot - tacklen * sinan, yplot + tacklen * cosan, 2)
         endif

         fpn = fpn + tickint
         del = del + tickint / scale

  100 continue

      end
```

```
      subroutine im_numb ( xpos, ypos, axlen, first, scale, angle,
     £                     ndec, spaceno, markno, hgtno, distno )

*     undirforrit sem skrifar tolur vid as

*     xpos, ypos  : upphafspunktur ass
*     axlen       : lengd ass
*     first       : upphafsgildi a asnum
*     scale       : kvardi  gagnaeiningar/cm
*     angle       : horn ass vid x_as teiknitaekis
*     ndec        : fjoldi aukastafa i tolum sem merkja asinn
*     spaceno     : bil milli talna a asnum i gagnaeiningum
*     markno      : snua tolum       <0 -90 deg   0 samsida as   >0 90 deg
*     hgtno       : haed tolustafa                0 sleppt
*     distno      : fjarlaegd talna fra as       <0 nedan        >0 ofan

      implicit integer*2 (i-n)

      if ( scale .eq. 0. .or. hgtno .le. 0. ) return

      del   = 0.0
      fpn   = first

      cosan = cos( angle * 0.017453292 )
      sinan = sin( angle * 0.017453292 )

      do 100 i=1,10000

         if ( del .gt. axlen * 1.00001 ) return

         xhat = xpos + del * cosan - distno * sinan
         yhat = ypos + del * sinan + distno * cosan

         num = num_digits(fpn) + ndec + 1

         if ( distno .ge. 0.0 ) then

           if ( markno .lt. 0 ) then
             horiz  = -0.5 * hgtno
             vert   = hgtno * ( num - 0.2 )
             rotate = angle - 90.0
           else if ( markno .eq. 0 ) then
             horiz  = -hgtno * ( num * 0.5 - 0.2 )
             vert   = 0.0
             rotate = angle
           else if ( markno .gt. 0 ) then
             horiz  = 0.5 * hgtno
             vert   = 0.0
             rotate = angle + 90.0
           endif

         else
```

```
            if ( markno .lt. 0 ) then
              horiz  = -0.5 * hgtno
              vert   = 0.0
              rotate = angle - 90.0
            else if ( markno .eq. 0 ) then
              horiz  = -hgtno * ( num * 0.5 - 0.2 )
              vert   = -hgtno
              rotate =  angle
            else if ( markno .gt. 0 ) then
              horiz  = 0.5 * hgtno
              vert   = -hgtno * ( num - 0.2 )
              rotate = angle + 90.0
            endif

          endif

          xhat = xhat + horiz * cosan - vert * sinan
          yhat = yhat + horiz * sinan + vert * cosan

          call number (xhat, yhat, hgtno, fpn * 1.000001, rotate, ndec)

          del   = del + spaceno / scale
          fpn   = fpn + spaceno

100       continue

          return
          end
```

```
      subroutine im_text ( xpos, ypos, axlen, angle,
     £                     text, nst, hgtext, distext )

*       undirforrit sem skrifar texta vid as
*       undirforrit : num_string, symbol

*       xpos, ypos  : upphafspunktur ass
*       axlen       : lengd ass
*       angle       : horn ass vid x_as teiknitaekis
*       text        : texti vid asinn
*       nst         : fjoldi stafa i texta       0 sleppt
*       hgtext      : haed texta    <0 gagnst.   0 sleppt   >0 samsida as
*       distext     : fjarlaegd texta fra as    <0 nedan    >0 ofan

        implicit integer*2 (i-n)
        character*(*)      text

        if ( nst .eq. 0 .or. hgtext .eq. 0.0 ) return

        dumnst = real(nst) - num_string(text,'´') - num_string(text,'¨')
        height = abs (hgtext)            ! telja storar og litlar kommur

        cosan  = cos( angle * 0.017453292 )
        sinan  = sin( angle * 0.017453292 )

        xplot = xpos + axlen * 0.5 * cosan - sinan * distext
        yplot = ypos + axlen * 0.5 * sinan + cosan * distext

        if ( distext .ge. 0.0) then

          if ( hgtext .lt. 0.0 ) then
            horiz  = height * ( dumnst * 0.5 - 0.2 )
            vert   = height
            rotate = angle - 180.0
          else if ( hgtext .gt. 0.0 ) then
            horiz  = -height * ( dumnst * 0.5 - 0.2 )
            vert   = 0.0
            rotate = angle
          endif

        else

          if ( hgtext .lt. 0.0 ) then
            horiz  = height * ( dumnst * 0.5 - 0.2 )
            vert   = 0.0
            rotate = angle - 180.0
          else if ( hgtext .gt. 0.0 ) then
            horiz  = -height * ( dumnst * 0.5 - 0.2 )
            vert   = -height
            rotate = angle
          endif

        endif

        xplot = xplot + horiz * cosan - vert * sinan
        yplot = yplot + horiz * sinan + vert * cosan

        call symbol (xplot, yplot, height, %ref(text), rotate, nst)

        return
        end
```

```
      subroutine im_symbol ( x, y, height, angle, isymb )

*     forrit sem teiknar midlaeg takn

*     undirforrit: im_reform
*                  im_transform
*                  im_line
*                  plot

*     x, y         : hnit miðju
*     height       : haed taknsins
*     angle        : snuningshorn takns (positiv orientation)
*     isymb        : numer takns 0 ferningur          3 +
*                                1 atthyrningur       4 x
*                                2 þrihyrningur       5 tigull

      integer*2     isymb
      integer*2     ip(50)
      real*4        dx(50),  dy(50)
      real*4        s00(16), s01(31), s02(13), s03(13), s04(13), s05(16)

      data s00 / 5.0,-0.50, -0.50, 3., -0.50,  0.50, 2.,
     £                0.50,  0.50, 2.,  0.50, -0.50, 2.,
     £               -0.50, -0.50, 2./
      data s01 / 10., 0.00,  0.50, 3.,  0.21,  0.50, 2.,
     £                0.50,  0.21, 2.,  0.50, -0.21, 2.,
     £                0.21, -0.50, 2., -0.21, -0.50, 2.,
     £               -0.50, -0.21, 2., -0.50,  0.21, 2.,
     £               -0.21,  0.50, 2.,  0.00,  0.50, 2./
      data s02 / 4.0, 0.00,  0.58, 3.,  0.50, -0.29, 2.,
     £               -0.50, -0.29, 2.,  0.00,  0.58, 2./
      data s03 / 4.0, 0.00,  0.50, 3.,  0.00, -0.50, 2.,
     £               -0.50,  0.00, 3.,  0.50,  0.00, 2./
      data s04 / 4.0, 0.35,  0.35, 3., -0.35, -0.35, 2.,
     £               -0.35,  0.35, 3.,  0.35, -0.35, 2./
      data s05 / 5.0, 0.00,  0.50, 3.,  0.50,  0.00, 2.,
     £                0.00, -0.50, 2., -0.50,  0.00, 2.,
     £                0.00,  0.50, 2./

      if ( isymb.lt.0 .or. isymb.gt.5 ) return
      n = 0

      if ( isymb .eq. 0 )  call im_reform ( s00 ,dx, dy, ip, n )
      if ( isymb .eq. 1 )  call im_reform ( s01 ,dx, dy, ip, n )
      if ( isymb .eq. 2 )  call im_reform ( s02, dx, dy, ip, n )
      if ( isymb .eq. 3 )  call im_reform ( s03, dx, dy, ip, n )
      if ( isymb .eq. 4 )  call im_reform ( s04, dx, dy, ip, n )
      if ( isymb .eq. 5 )  call im_reform ( s05, dx, dy, ip, n )

      call im_transform ( dx, dy, n, height, angle, x, y  )
      call im_line      ( dx, dy, ip, n     )

      call plot ( x, y, 3 )

      end
```

```
      subroutine os_merki (x, y, hgt, rot, texta, iqa, textb, iqb )

*     os_merki     undirforrit sem setur upp og teiknar
*                  merki OS asamt tilheyrandi textalinum.
*                  les .nea skra eda notar parametrana i kallinu.
*                  notandi getur breytt uppsetningu merkisins.

*     undirforrit: lib$put_screen     nonotify     os_teikn
*                  lib$erase_page     notify       gin_bxyp
*                  lib$erase_line     noecho       symbol
*                  lib$get_lun        getf
*                  lib$free_lun       outf

      real*4          x_os,   y_os,  os_hgt,  os_rot
      character       texta*(*), textb*(*),  text1*72, text2*72
      character       file*64,  answ*1
      integer*4       iq1,  iq2
      integer*2       iq_1,  iq_2, iqa, iqb
      logical         error
      byte            chin
      character*32    neaform    /'(t9,f9.2,t22,f9.2,t43,q,a)'/

      x_os    =  x              ! x hnit nedra vinstra horns OS merkis
      y_os    =  y              ! y hnit nedra vinstra horns OS merkis
      os_hgt  =  hgt            ! haed OS merkis
      os_rot  =  rot            ! snuningshorn  positivt fra x as
      text1   =  texta          ! textalina 1
      text2   =  textb          ! textalina 2
      iq1     =  iqa            ! fjoldi stafa i textalinu 1
      iq2     =  iqb            ! fjoldi stafa i textalinu 2

100   call nonotify ( ' Merki Orkustofnunar ',8,1,0)

      call lib$put_screen (' 1  lesa .NEA skra        ',16,10,)
      call lib$put_screen (' 2  nota sjalfgefin gildi ',17,10,)
      call lib$put_screen (' 3  haetta merkingu       ',18,10,)
      call lib$put_screen ('    ritadu valnumer       ',21,10,)

      call noecho (answ,1,iq,0)

      if ( answ .lt. '1' .or. answ .gt. '3'  ) goto 100
      if ( answ .eq. '1' )  then

        call lib$erase_page (9,1)
        call lib$put_screen ('  nafn a .NEA skra  '    ,16,10,)
        read(5,'(a)',iostat=ios)  file
        if  ( ios.lt.0 )          goto 100
        call lib$get_lun(lun)
        open(lun,file=file,status='old',defaultfile='.nea',err=100)

        error = .false.
        read(lun,neaform,iostat=ios) x_os,  y_os,  iq1, text1
        if ( ios.ne.0 )  error = .true.
        read(lun,neaform,iostat=ios) os_hgt,os_rot,iq2, text2
        if ( ios.ne.0 )  error = .true.
        close (unit = lun)
        call lib$free_lun(lun)
```

```
      if ( error )  then
        call notify (' villa  i  skra '//file  ,22,1,0)
        goto 100
      endif

    else if ( answ .eq. '3' )  then

      call lib$erase_page (1,1)
      return

    endif

    do while ( .true. )

      call lib$erase_page(9,1)
      call outf          (' 1) x hnit         ',      x_os         ,12,1,)
      call outf          (' 2) y hnit         ',      y_os         ,13,1,)
      call outf          (' 3) haed OS merkis ',      os_hgt       ,14,1,)
      call outf          (' 4) snuningshorn   ',      os_rot       ,15,1,)
      call lib$put_screen(' 5) efri  lina i merki  '//text1(1:iq1) ,16,1,)
      call lib$put_screen(' 6) nedri lina i merki  '//text2(1:iq2) ,17,1,)
      call lib$put_screen(' 7) haetta við '                        ,18,1,)
      call lib$put_screen('    <ret> = engin breyting  '           ,21,1,)
      call noecho (answ,1,iq,0)

      if ( answ .eq. '1' )  then
        call lib$put_screen(' vinstra nedra horns OS merkis ',23,1,2)
        call lib$put_screen('   999.00 til ad digitisera    ',24,1,2)
        call getf ('    x hnit  ',              x_os         ,21,1,1)
      else if ( answ .eq. '2' )  then
        call lib$put_screen(' vinstra nedra horns OS merkis ',23,1,2)
        call lib$put_screen('   999.00 til ad digitisera    ',24,1,2)
        call getf ('    y hnit  ',              y_os         ,21,1,1)
      else if ( answ .eq. '3' )  then
        call getf ('    haed OS merkis i cm  ', os_hgt       ,21,1,1)
      else if ( answ .eq. '4' )  then
        call lib$put_screen
£       (' positivt rangsaelis fra x as teiknara '           ,23,1,2)
        call getf ('    snuningshorn  ',        os_rot       ,21,1,1)
      else if ( answ .eq. '5' )  then
        call lib$erase_line (21,1)
        call lib$put_screen ('    efri lina i merki  ' ,,,1)
        call echo(text1,72,iq1,0)
      else if ( answ .eq. '6' )  then
        call lib$erase_line (21,1)
        call lib$put_screen ('    nedri lina i merki  ',,,1)
        call echo(text2,72,iq2,0)
      else if ( answ .eq. '7' )  then
        goto 100
      else if ( ichar(answ) .eq. 13 )  then
        if ( x_os.eq.999. .or. y_os.eq.999. ) then
          call lib$erase_line (21,3)
          call gin_bxyp ( chin, x_os, y_os, ipen)
        endif
        iq_1 = iq1
        iq_2 = iq2
        call os_teikn(x_os, y_os, os_hgt, os_rot, text1, iq_1, text2, iq_2)
        goto 100
      endif

    enddo
    end
```

```
      subroutine os_teikn ( x, y, hgt, angle, text1, iq1, text2, iq2 )

*     os_teikn     undirforrit sem teiknar merki OS
*                  og tvaer tilheyrandi textalinur.
*     undirforrit: im_tilf, symbol

*     x, y         : hnit vinstra nedra horns OS merkis
*     hgt          : haed merkisins
*     angle        : snuningshorn merkis (positiv orientation)
*     text1        : fyrri  textalina i OS merki
*     text2        : seinni textalina i OS merki

      implicit integer*2 (i-n)
      character*(*) text1, text2

      real*4         os0(43), os1(4),  os2(4)
      real*4         dx(50),  dy(50)
      dimension      ip(50)

      data os0 / 14., 0.00,  0.00, 3.,  1.41,  0.00, 2.,
     £                0.00,  0.00, 2.,  0.00,  1.00, 2.,
     £                1.41,  1.00, 2.,  1.41,  0.00, 2.,
     £                0.35,  0.33, 3.,  0.35,  0.67, 2.,
     £                0.70,  1.00, 3.,  0.70,  0.00, 2.,
     £                0.70,  0.33, 3.,  1.05,  0.33, 2.,
     £                1.05,  0.67, 3.,  1.41,  0.67, 2./
      data os1 / 1.0, 1.73,  0.67, 3./
      data os2 / 1.0, 1.73,  0.00, 3./

      call im_reform ( os0, dx, dy, ip, n )
      call im_transform   ( dx, dy, n, hgt, angle, x, y)
      call im_line        ( dx, dy, ip, n )

      call im_reform ( os1, dx, dy, ip, n )
      call im_transform   ( dx, dy, n, hgt, angle, x, y)
      call im_line        ( dx, dy, ip, n )
      call symbol (999.,999., hgt*0.33, %ref(text1), angle, iq1 )

      call im_reform ( os2, dx, dy, ip, n )
      call im_transform   ( dx, dy, n, hgt, angle, x, y)
      call im_line        ( dx, dy, ip, n )
      call symbol (999.,999., hgt*0.33, %ref(text2), angle, iq2 )

      return
      end
```

```
      subroutine im_reform ( svec, xvec, yvec, ip, n )

*     undirforrit sem tilheyrir im_symbol
*     brytur vektorinn svec upp i frumparta sina !!!

      real*4     svec(1) ! vektor sem geymir x , y , ipen
*                svec(1) fjoldi punkta (x,y,ipen), sem geymdir eru i svec
*                svec(2+3j) geymir x hnit punktanna j = 1...n
*                svec(3+3j) geymir y hnit punktanna j = 1...n
*                svec(4+3j) geymir gildin a ipen    j = 1...n
      real*4     xvec(1) ! vektor sem faer x hnit punktanna
      real*4     yvec(1) ! vektor sem faer y hnit punktanna
      integer*2  ip(1)   ! vektor sem faer gildin a ipen
      integer*2  n       ! fjoldi punkta sem geymdir eru i xvec og yvec

      n = svec(1)

      do i = 1, n

        idumm   =  3 * i + 1
        xvec(i) =  svec(idumm-2)
        yvec(i) =  svec(idumm-1)
        ip(i)   =  svec(idumm)

      enddo

      return
      end
```

```
      subroutine im_transform ( xvec, yvec, n, scale, angle, x, y  )

*     framkvæmir linulega vorpun - snuning, kvordun og tilfaerslu

      real*4     xvec(1) ! vektor sem geymir x hnit punktanna
      real*4     yvec(1) ! vektor sem geymir y hnit punktanna
      integer*2  n       ! fjoldi punkta sem geymdir eru i xvec og yvec
      real*4     scale   ! margfoldunarstudull hnitanna
      real*4     angle   ! hornið sem snua a punktunum um (positiv orientation)
      real*4     x       ! faersla i stefnu x ass
      real*4     y       ! faersla i stefnu y ass

      if ( angle .ne. 0.0 ) then
        call im_rotation  ( xvec, yvec, n, angle )
      endif

      if ( scale .ne. 1.0 ) then
        call im_scaling   ( xvec, yvec, n, scale )
      endif

      if ( x .ne. 0.0 .or. y .ne. 0.0 ) then
        call im_translate ( xvec, yvec, n, x, y )
      endif

      return
      end
```

```
      subroutine im_rotation ( xvec, yvec, n, angle )

*     framkvæmir snuning  (rotation)

      real*4     xvec(1) ! vektor sem geymir x hnit punktanna
      real*4     yvec(1) ! vektor sem geymir y hnit punktanna
      integer*2  n       ! fjoldi punkta sem geymdir eru i xvec og yvec
      real*4     angle   ! hornið sem snua a punktunum um (positiv orientation)

      rad   = angle * 0.017453292
      sinan = sin (rad)
      cosan = cos (rad)

      do i = 1, n

        dummx = xvec(i)
        dummy = yvec(i)

        xvec(i) = dummx * cosan - dummy * sinan
        yvec(i) = dummx * sinan + dummy * cosan

      enddo

      return
      end
```

```
      subroutine im_scaling ( xvec, yvec, n, scale )

*     framkvæmir kvordun (scaling)

      real*4     xvec(1) ! vektor sem geymir x hnit punktanna
      real*4     yvec(1) ! vektor sem geymir y hnit punktanna
      integer*2  n       ! fjoldi punkta sem geymdir eru i xvec og yvec
      real*4     scale   ! margfoldunarstudull hnitanna

      do i = 1, n

        xvec(i) = xvec(i) * scale
        yvec(i) = yvec(i) * scale

      enddo

      return
      end
```

```
      subroutine im_translate ( xvec, yvec, n, x, y )

*     framkvæmir tilfaerslu (translation)

      real*4     xvec(1) ! vektor sem geymir x hnit punktanna
      real*4     yvec(1) ! vektor sem geymir y hnit punktanna
      integer*2  n       ! fjoldi punkta sem geymdir eru i xvec og yvec
      real*4     x       ! faersla i stefnu x ass
      real*4     y       ! faersla i stefnu y ass

      do i = 1, n

        xvec(i) = x + xvec(i)
        yvec(i) = y + yvec(i)

      enddo

      return
      end
```

```
      subroutine im_line ( xvec, yvec, ip, n )

*     dregur feril i gegnum punkta sem geymdir er i vektorum xvec og yvec

      real*4     xvec(1) ! vektor sem geymir x hnit punktanna
      real*4     yvec(1) ! vektor sem geymir y hnit punktanna
      integer*2  ip(1)   ! vektor sem geymir x hnit punktanna
      integer*2  n       ! fjoldi punkta sem geymdir eru i xvec og yvec

      do i = 1, n

        call plot ( xvec(i), yvec(i), ip(i) )

      enddo

      return
      end
```

```
      subroutine text_plot ( nchset1, dline1, height1, angle1, slant1 )

*     text_plot    undirforrit sem skrifar texta a teiknitaeki
*                  les textann ur skra eða af skermi
*                  notandi stjornar uppsetningu textans

*     undirforrit: lib$put_screen    nonotify    new_cha
*                  lib$set_cursor    noecho      set_slant
*                  lib$erase_page    echo        gin_bxyp
*                  lib$get_lun       outf        plot
*                  lib$free_lun      getf        symbol
*                                    esc6

      parameter      ( maxlines = 100 )
      character*1    answ, dumm
      character*10   alphabet(0:2)
      character*64   file
      character*256  line ( maxlines )
      integer*2      ntex ( maxlines )
      integer*2      ipen
      byte           dummy

      data alphabet  / 'enskt', 'islenskt', 'griskt' /

      nchset = nchset1    ! stafa sett ( 0=enskt 1=islenskt 2=griskt)
      dline  = dline1     ! linubil i cm
      height = height1    ! haed stafa i cm
      angle  = angle1     ! snuningshorn texta positivt rangsaelis fra x as
      slant  = slant1     ! stafa halli i gradum

100   ios   = 0
      lines = 0

      do i = 1, maxlines
        line ( i ) = ' '
        ntex ( i ) = 0
      enddo

      call nonotify ( ' skrifa texta a teiknitaeki  ', 7,1,0)

      call lib$put_screen (' 1  lesa texta ur skra     ',16,10,)
      call lib$put_screen (' 2  rita texta a skjainn   ',17,10,)
      call lib$put_screen (' 3  haetta                 ',18,10,)
      call lib$put_screen ('    ritadu valnumer        ',21,10,)

      call noecho (answ,1,iq,0)

      if ( answ .lt. '1' .or. answ .gt. '3'  ) goto 100
      if ( answ .eq. '1' )  then

        call lib$erase_page (9,1)
        call lib$put_screen ('  nafn a .TXT skra   '    ,16,10,)
        read(5,'(a)',end=100)   file
        call lib$get_lun ( lun )
        open(lun,file=file,status='old',defaultfile='.txt',err=100)
```

```
      do while ( lines.le.maxlines .and. ios.eq.0 )
        read(lun,'(q,a)',iostat=ios) ntex(lines+1), line(lines+1)
        if ( ios.eq.0 ) then
          lines = lines + 1
        endif
      enddo

      close ( lun )
      call lib$free_lun ( lun )

    else if ( answ .eq. '2' )  then

      call nonotify ('Ritadu texta (<100 linur)  ',1,1,0)
      call esc6     ('CTRL-Z til ad haetta       ',2,1,0)
      call lib$set_cursor(5,1)

      do while ( lines.le.maxlines .and. ios.eq.0 )
        read(5,'(q,a)',iostat=ios) ntex(lines+1), line(lines+1)
        if ( ios.eq.0 ) then
          lines = lines + 1
        endif
      enddo

    else if ( answ .eq. '3' )  then

      call lib$erase_page (1,1)
      return

    endif

    do while (.true.)
200   call nonotify ( ' skrifa texta a teiknitaeki  ',2,1,0)

      call lib$put_screen (' 1) skoda texta  '                ,12,,)
      call lib$put_screen (' 2) stafrof '//alphabet(nchset)  ,13,,)
      call outf ( ' 3) linubil i cm            ', dline      ,14,,)
      call outf ( ' 4) haed stafa i cm         ', height     ,15,,)
      call outf ( ' 5) snuningshorn textalinu  ', angle      ,16,,)
      call outf ( ' 6) stafa halli             ', slant      ,17,,)
      call lib$put_screen ( ' 7) haetta vid '                 ,18,,)

      if (dline.le.height)  call lib$put_screen ('  ath  ',13,45,6)
      if (dline.le.height)  call lib$put_screen ('  ath  ',14,45,6)
      if (abs(slant.gt.40)) call lib$put_screen ('  ath  ',16,45,6)

      call lib$put_screen ( '    <ret> = engin breyting  '   ,21,,)
      call noecho(answ,1,iq,0)

      if (answ.eq.'1') then
        call lib$erase_page(1,1)
        do i = 1, lines
          write(6,'('' '',a)') line(i)(1:ntex(i))
          if (mod(i,20).eq.0) then
            call lib$put_screen (' <ret> til ad sja meira  ',23,30,1)
            call lib$put_screen (' annad til ad haetta  '      ,,,1)
            call noecho (dumm,1,iq,0)
            call lib$erase_page(1,1)
            if ( ichar(dumm).ne.13) goto 200
          endif
        enddo
        call lib$put_screen ('  <ret> til ad halda afram  '  ,23,50,1)
        call noecho (dumm,1,iq,0)
```

```
      else if (answ.eq.'2') then
        ios = 1
        do while ( ios.ne.0 )
          call lib$put_screen ('  enskt (0)     ', 21,,)
          call lib$put_screen ('  islenskt (1)     ',,,)
c         call lib$put_screen ('  griskt (2)    '   ,,,)
          call noecho (dumm,1,iq,0)
          read(dumm,'(i)',iostat=ios)  nchset
        enddo

      else if (answ.eq.'3') then
        call getf ('    linubil i cm  '     , dline  ,21,1,1)
      else if (answ.eq.'4') then
        call getf ('    haed stafa i cm  '  , height ,21,1,1)
      else if (answ.eq.'5') then
        call getf ('    snuningshorn  '     , angle  ,21,1,1)
      else if (answ.eq.'6') then
        call getf ('    stafa_halli  '      , slant  ,21,1,1)
      else if (answ.eq.'7') then
        go to 100
      else if (ichar(answ).eq.13) then

        call lib$erase_page(21,1)

        call set_slant( slant )
        call new_cha  ( nchset )
        call gin_bxyp ( dummy, xx, yy, ipen )

300     x = xx
        y = yy

        do i = 1, lines
          if ( ntex ( i ) .gt. 0 ) then
            call symbol ( x, y, height, %ref(line(i)), angle, ntex(i) )
          endif
          x = x + dline * sind (angle)
          y = y - dline * cosd (angle)
        enddo

        call plot (999,999,0)

        call lib$erase_page (21,1)
        call lib$put_screen ('  skrifa textann aftur ',,,)
        call lib$put_screen ('  <ret>=nei  annad=ja    ',,,)
        call echo (answ,1,iq,0)

        if (ichar(answ).ne.13) then
          call lib$erase_page (22,1)
          call lib$put_screen ('  sami stadur eda nyr  ',,,)
          call lib$put_screen ('  <ret>=nyr  annad=sami  ',,,)
          call echo (dumm,1,iq,0)
          if (iq. ne. 0) goto 300
          call set_slant ( -slant )
          goto 200
        else
          call set_slant ( -slant )
          goto 100
        endif

      endif
    enddo
    end
```

```
      subroutine echo ( buffer, nbuf, nc, ipurge )

*     subroutine to read character data to variable buffer
*     input displayed on terminal   (c) asmundur jakobsson

*     nbuf    maximum numbers of characters to be read to variable buffer
*     nc      numbers of characters read to buffer
*     ipurge  clear type_ahead buffer if ipurge = 0

      character   buffer*(*)
      integer*2   ttchan, iosb(4)
      integer*4   iokode, ipurge
      integer*4   sys$assign, sys$dassgn, sys$qiow
      external    io$_readvblk, io$m_purge

      iostat = sys$assign ('tt',ttchan,,)
      if (.not.iostat)  call lib$stop ( %val (iostat) )

      if ( ipurge .ne. 0 ) then
        iokode = %loc(io$_readvblk)
      else
        iokode = %loc(io$_readvblk) .or. %loc(io$m_purge)
      endif

      iostat = sys$qiow (,%val(ttchan),%val(iokode),iosb,,,
     £                    %ref(buffer),%val(nbuf),%val(0),,,)
      if (.not.iostat)  call lib$stop (%val(iostat))
      nc = iosb(2)

      iostat = sys$dassgn (%val(ttchan))
      if (.not.iostat)  call lib$stop (%val(iostat))

      return
      end
```

```
      subroutine noecho ( buffer, nbuf, nc, ipurge )

*     subroutine to read character data to variable buffer
*     input not displayed on terminal ( see subroutine echo )

*     nbuf    maximum numbers of characters to be read to variable buffer
*     nc      numbers of characters read to buffer
*     ipurge  clear type_ahead buffer if ipurge = 0

      character   buffer*(*)
      integer*2   ttchan, iosb(4)
      integer*4   iokode, ipurge
      integer*4   sys$assign, sys$dassgn, sys$qiow
      external    io$_readvblk, io$m_noecho, io$m_purge

      iostat = sys$assign ('tt',ttchan,,)
      if (.not.iostat)  call lib$stop ( %val (iostat) )

      if ( ipurge .ne. 0 ) then
        iokode = %loc(io$_readvblk) .or. %loc(io$m_noecho)
      else
        iokode = %loc(io$_readvblk) .or. %loc(io$m_noecho)
     £                              .or. %loc(io$m_purge)
      endif

      iostat = sys$qiow (,%val(ttchan),%val(iokode),iosb,,,
     £                   %ref(buffer),%val(nbuf),%val(0),,,)
      if (.not.iostat)  call lib$stop (%val(iostat))
      nc = iosb(2)

      iostat = sys$dassgn (%val(ttchan))
      if (.not.iostat)  call lib$stop (%val(iostat))

      return
      end
```

```
      subroutine getf ( prompt, x, i1, i2, i3 )

*     subroutine to prompt for real*4 number   x
*     and accept it at specified screen position

      real*4          x
      integer*4       lib$put_screen, lib$erase_line
      character*(*)   prompt

      x_dummy = x
      ios     = 1

      do while ( ios .gt. 0 )

        istat = lib$erase_line (i1,i2)
        istat = lib$put_screen ( prompt , i1, i2, i3 )

        read(5,'(f9.0)',iostat=ios)  x

      enddo

      if ( ios .lt. 0 ) x = x_dummy

      return
      end
```

```
      subroutine geti ( prompt, number, i1, i2, i3 )

*     subroutine to prompt for integer*2 number
*     and accept it at specified screen position

      integer*2       number
      integer*4       lib$put_screen, lib$erase_line
      character*(*)   prompt

      i_dummy = number
      ios     = 1

      do while ( ios .gt. 0 )

        istat = lib$erase_line (i1,i2)
        istat = lib$put_screen ( prompt , i1, i2, i3 )

        read(5,'(i)',iostat=ios)  number

      enddo

      if ( ios .lt. 0 ) number = i_dummy

      return
      end
```

```
      subroutine outf ( message, x, i1, i2, i3 )

*     subroutine to write text and real*4 number
*     at specified screen position using lib$put_screen

      real*4          x
      integer*4       lib$put_screen
      character*(*)   message
      character*80    line

      write(line,'(a,f9.2)',iostat=ios) message, x

      if (ios.eq.0) then
        istat = lib$put_screen ( line, i1, i2, i3 )
      else
        istat = lib$put_screen ( message//'    ERROR ', i1, i2, 6 )
      endif

      return
      end
```

```
      subroutine outi ( message, number, i1, i2, i3 )

*     subroutine to write text and integer*2 number
*     at specified screen position using lib$put_screen

      integer*2       number
      integer*4       lib$put_screen
      character*(*)   message
      character*80    line

      write(line,'(a,i6)',iostat=ios) message, number

      if (ios.eq.0) then
        istat = lib$put_screen ( line, i1, i2, i3 )
      else
        istat = lib$put_screen ( message//'    ERROR ', i1, i2, 6 )
      endif

      return
      end
```

```
      subroutine esc6 ( message, i1, i2, i3 )

*     subroutine to write single_height double_width message
*     at specified screen position using lib$put_screen

      character*(*)   message
      integer*4       lib$put_screen

      istat = lib$put_screen (char(27)//'£6'//message,i1,(i2+1)/2,i3)

      return
      end
```

```
      subroutine nonotify ( message, i1, i2, i3 )

*     subroutine to erase screen,
*     write text at specified screen position

      character*(*)   message

      call lib$erase_page (1,1)
      call esc6 ( message, i1, i2, i3 )
      call lib$set_cursor (24,1)

      return
      end
```

```
      subroutine notify ( message, i1, i2, i3 )

*     subroutine to erase screen,
*     write text at specified screen position
*     ring terminal bell, and wait 4 seconds

      character*(*)   message

      call lib$erase_page (1,1)
      call esc6 ( message, i1, i2, i3 )
      call lib$set_cursor (24,1)

      call peep   ( 2 )
      call wait_s ( 4.0 )

      return
      end
```

```
      subroutine peep ( n_peep )

*     rings terminal's bell n_peep times

      integer*4   lib$put_screen

      do i = 1, n_peep
        istat = lib$put_screen ( char ( 7 ) )
      enddo

      return
      end
```

```
      logical*4 function err ( line )

*     write error line on sys$output

      character*(*) line

      err = .true.
      write(6,'('' '',a)') line

      return
      end
```

```
      integer*4 function num_digits ( x )

*     returns the number of digits in real numer x
*     desimal digits and desimal point not included

      if ( x .lt. 0. ) num_digits = 2 + alog10( -x )
      if ( x .eq. 0. ) num_digits = 1
      if ( x .gt. 0. ) num_digits = 1 + alog10( x )

      return
      end
```

```
      integer*4 function num_string ( string, sub_str )

*     returns the number of occurences of sub_str in string

      character*(*) string, sub_str
      integer*4     str$position
      integer*4     beg_pos, end_pos, rel_pos

      num_string = 0
      beg_pos    = 1
      end_pos    = len (string)
      rel_pos    = str$position(string(beg_pos:end_pos),sub_str)

      if  ( len(sub_str) .le. 0 ) return

      do while ( rel_pos .gt. 0 )
        num_string = num_string + 1
        beg_pos    = beg_pos    + rel_pos
        rel_pos    = str$position(string(beg_pos:end_pos),sub_str)
      enddo

      return
      end
```

```
      subroutine spawn ( dcl )

*     program to faciliate use of DCL at runtime

      external      ss$_normal
      integer*4     lib$spawn, lib$signal, lib$put_output
      character     dcl*(*), command*(80)

      if ( dcl .eq. ' ' ) then

        write(6,'(a)')
        do while ( .true. )

          write(6,'(a,$)') ' spawn> '
          read(5,'(q,a)',iostat=ios) iq, command
          if ( ios.lt.0 )  return

          istat = lib$spawn ( command,,,0 )

          if ( .not.istat ) then
            call lib$put_output ( ' error ' )
            call lib$signal ( %val(istat) )
          endif

        enddo

      else

        istat = lib$spawn ( dcl,,,0 )

        if ( .not.istat ) then
          call lib$put_output( ' error ')
          call lib$signal ( %val(istat) )
        endif

      endif

      return
      end
```