UNITED STATES DEPARTMENT OF INTERIOR
GEOLOGICAL SURVEY

GEOPLOT

An Advanced Graphics Package for Minicomputers

Part 2: Computer Source Code

Peter L. Ward

Open-File Report 83- 807 – B

Office of Earthquake Studies
345 Middlefield Road
Menlo Park, Ca. 94025

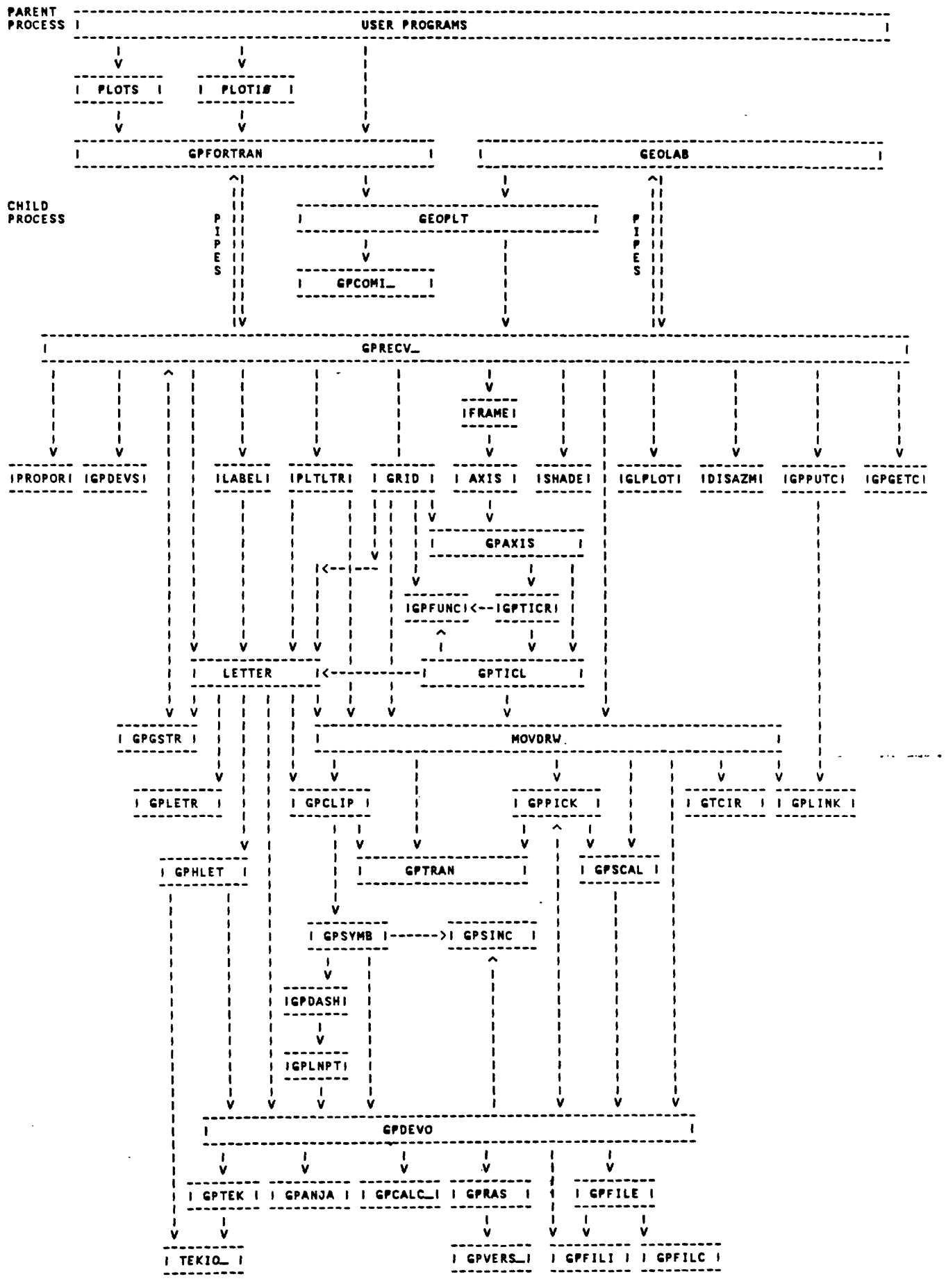
Table of Contents

Introduction

Geoplot was initially written in Fortran but large amounts have been rewritten in C which is far better suited for writing compact, efficient, maintainable code. Most code was written by Peter L. Ward except for gpcalc¬c, gpras.c, gpvers¬c, and plot10.f which were written by Barbara Bekins, U. S. Geological Survey, formerly of Electronic Data Systems, Inc. working on contract with the USGS. Plots.f was written by Dave Oppenheimer of the USGS.

Geoplot subroutines are organized in layers with high-level routines at the top and low-level device "drivers" at the bottom. Error tracing can be turned on at any given level. The levels and interrelationships of subroutines are shown in Figure 1. When multiple small subroutines exist within the same file, only the file name without suffix is shown. User programs can call geoplot through subroutine calls directly or through the plots (versatec) and plot10 (tektronix) emulation packages. The geoplot calls start up the geoplt child process and communicate with it through pipes.

Gpcom is the main common block. Each variable as accessed in Fortran or C is listed with its position in gpcom, its name, its description, and a list of subroutines where is is defined and used. Gpdevs refers to different device dependent routines like gptek, gpvers, etc. User means simply available for use by user.


PARENT PROCESS
USER PROGRAMS
PLOTS
PLOTI#
GPFORTRAN
GEOLAB
CHILD PROCESS
PIPES
GEOPLT
GPCOMI_
PIPES
GPRECV_
IFRAMEI
IPROPORI
IGPDEVSI
ILABELI
IPLTLTRI
I GRID I
I AXIS I
ISHADEI
IGLPLOTI
IDISAZMI
IGPPUTCI
IGPGETCI
GPAXIS
IGPFUNCI
IGPTICRI
LETTER
GPTICL
I GPGSTR I
MOVDRW
I GPLETR I
I GPCLIP I
I GPPICK I
I GTCIR I
I GPLINK I
I GPHLET I
GPTRAN
I GPSCAL I
I GPSYMB I
I GPSINC I
IGPDASHI
IGPLNPTI
GPDEVO
I GPTEK I
I GPANJA I
I GPCALC_I
I GPRAS I
I GPFILE I
I TEKIO_ I
I GPVERS_I
I GPFILI I
I GPFILC I

Variables in gpcom. Gpdevs refers to different device dependent routines like gptek, gpvers, etc. User means simply available for use by the user.

NAME	DESCRIPTION	DEFINED	USED
REAL NUMBERS:	USER MAY READ ONLY		
1 devmax(2)	Maximum number of page units in x and y.	gpdevs	gpdevo
3 dpage(2)	Number of device units per page unit in x and y.	gpdevs	gpdevs
5 phome(2)	Page coordinates of home for pen, place where pen sits after a page command.	gpdevs	movdrw
7 dpgin	Inches per page unit on display.	gpdevs	user
8 hpgin	Inches per page unit on the hardcopy.	gpdevs	user
9 rasmin	Page units per device unit, ie minimum raster resolution in pageunits.	gpdevs	movdrw gpputc gpclip gplnpt gpdevo gpdash
10 pshade	Line intensity or color in number system recognized by the device.	gpdevs	letter movdrw gpsymb
11 ratchw			
12 ratsph			
13 spalin(3)	Number of data units between interpolated points for curvilinear transformations.	gptran	movdrw
16 pointp(2)	Present pen position from software, x,y in page units.	gpdevo	gpdash gplnpt gpdevo axis gpdevo gpfile gpsymb grid
18 setsca	Set map scale to this. Usually 0.	gpputc movdrw	gptran
19 pad1(2)			
REAL NUMBERS:	USER MAY READ OR WRITE		
21 pspace(24)	Plot spaces. 1-6 pictsp, 7-12 gridsp, 13-18 subjsp, 19-24 datasp. Order is xa,xz,ya,yz,za,zz. All are in page units except data space.	movdrw	gpclip gptran letter map axis gpaxis gpgetc gppick gplink propor
45 omitsp(16)	Areas of the plot to leave blank. Up to 4 areas xa,xz,ya,yz in page units may be specified.	movdrw	gpdevo
61 trans(9)	Constants used in the transformations.	movdrw	gptran
70 offset(2)	Amount plot to be offset in page units for x and y.	movdrw	gpdevo
72 pangle	Angle plot to be rotated by in degrees.	movdrw	gpdevo letter
73 pgain(2)	Amount plot to be multiplied by in page units for x and y.	movdrw	gpdevo letter

	NAME	DESCRIPTION	DEFINED	USED
75	ticbas	Base line for tic marks. Either x or y as appropriate in data units or page units as appropriate.	movdrw	pltltr
76	symshp	Type or shape of symbol.	movdrw	gpsymb
77	symlen	Length of symbol in page units.	movdrw	gpsymb
78	symspa	Spacing between concentric symbols in page units. If < 0 move to point.	movdrw	gpsymb
79	symang	Angle symbol draw at relative to line.	movdrw gpputc	gpsymb gpgetc
80	symfac	Factor by which to multiply vector length and add to symlen to get symbol length.	movdrw	gpsymb
81	space(3)	Spacing between parallel lines in page for lines, characters, and symbols.	movdrw	gplnpt
84	width(3)	Width of lines in page units for lines, characters, and symbols.	movdrw	gplnpt
87	shade(3)	Line shade, intensity, or color for for lines, characters, and symbols.	movdrw	gpdevs gpsymb letter movdrw
90	curinc	Spacing in page units between inter-polated points along a curve drawn in curvilinear coordinates.	movdrw	movdrw gptran
91	height	Character height in page units.	gpdeva movdrw	letter gpaxis gpgetc gpdevs
92	aspect	Ratio of character width to height.	gpdevs movdrw	letter gpaxis gpgetc
93	slopel	Angle of lettering in degrees from perpendicular to base of lettering.	gpdevs movdrw	letter
94	spacel	Ratio of width of space between characters to width of characters.	movdrw gpdevs	letter gpaxis gpgetc
95	spaceh	Ratio of height of apace above characters to height of characters.	movdrw gpdeva	letter
96	chrerr	Maximum difference in page units between specified character height and hardware character height tolerated for hardware lettering.	movdrw	gpdevs
97	pole(2)	Pole lon and lat about which to rotate map coordinates	movdrw mapdig	maputil newpol
99	pad2(2)			
101	ticfac(6)	Factor by which tic length to be multiplied for this axis.	movdrw	gpaxis
107	ticin(10)	Tic lengths inside for x(5) and y(5).	movdrw	gpticl
117	ticout(10)	Tic lengths outside for x(5) and y(5)	movdrw	gpticl
127	ticmin(6)	Minimum tic spacing in page units.	movdrw	gpaxis
133	mapsca	Scale of map now in use	gptran movdrw	gpgetc gptran
134	pad3(2)			
136	dislab(6)	Distance of label from axis in page units. If =0.0 then do not plot a label.	movdrw	gpaxis gpticl
142	spalab(6)	Minimum spacing between labels in page units.	movdrw	gpaxis
148	faclab(6)	Amount to be factored out of label.	movdrw	gpticl
154	anglab(6)	Angle of label with axis.	movdrw	gpticl

	NAME	DESCRIPTION	DEFINED	USED
160	usym	Missing data symbol.	device	gptran pltltr
161	fullsp(2)	User adjustible max plot size < devmax.	movdrw	movdrw
163	clip(3)	Clipping factor used in propor.	movdrw	propor
166	pad4(14)			
180	timbas	Time base, double precision.	device	gpticl
INTEGERS: USER MAY READ OR WRITE				
181	lintyp(3)	Line type, integer*4, for lines, characters, and symbols.	movdrw	gpsymb letter movdrw
184	kside(3)	Side of line stipples drawn on for lines, characters, and symbols. =1 both, =2 left, =3 right.	movdrw	gplnpt
187	kltrsp	Clip space for letters.	movdrw	letter movdrw
188	kpltsp	Clip space for lines.	movdrw	movdrw gpticl letter
189	linclp	=1 no lines on clip boundary, =2 lines.	movdrw	gpclip
190	ktrans	Transformation number.	movdrw	gptran
191	lablsp(6)	Number of character spaces between pictsp and gridsp.	movdrw	gpgetc
197	kbmtyp	=1 restore beam, =2 put beam at top of the page, =3 leave beam where it is, =4 do nothing.	movdrw	gpbufo gpputc gpdevs tekin_
198	idbug	Level of dbugging printout. 1 to 10	device	all
199	intact	=1 for interactive dumping of buffer =0 otherwise	movdrw gpdevs	gprecv
200	gtrans	Transformation type for grid.	movdrw	grid
201	m1pad			
202	iupad			
203	julian	= 1 dates are julian.	movdrw	gpticl
204	izone	Time zone of dates.	device	gpticl
205	kpad			
206	iup	Up direction on plot. 1 to 4.	movdrw	letter
207	noaxis(6)	=1 if no axis line is to be drawn.	movdrw	axis grid
213	itcpag	if =1 force tics to be in page units.	movdrw	gpticl
214	lpad			
215	mpad			
216	maxpow(6)	Maximum power of 10 to be plotted on this axis.	movdrw	gpaxis
222	minlab(6)	Minimum number of characters in the label before changing to E format.	movdrw	gpticl
228	idtype(6)	Data type =0 data, =1 date, =2 degrees and minutes, =3 degrees, minutes, and seconds.	movdrw	gpticl gpaxis
234	lineid	Line identification number used in gppick	movdrw	gppick
235	nfield	Number of fields stored in gppick.	movdrw	gppick
236	ksubj	=0 if subjsp is initialized =1 if subsp needs to be saved in gppick	movdrw gpputc axis grid propor	gpdevo
237	nfare	Type of pick distance used in gppick	movdrw	gppick
238	ipad1(3)			
241	ipad2(9)			
INTEGERS: USER MAY READ ONLY				
250	nosym	Turn off symbols while interpolating	movdrw	gpsymb

	NAME	DESCRIPTION	DEFINED	USED
251	kkchar	Last hardware letter size set.	gpdevo	gpdevo gpbufo
252	nomit	Number of areas to be omitted, see omitsp.	movdrw gpputc	gpdevo
253	numdev(3)	Device number, 1 is number, 2 is model, 3 is position on device (1 to 3)	movdrw device	gpdevo gpbufo gpdevs gprecv
256	letchg	If =1 reinitialize letter related things in letter.	movdrw gpputc	letter
257	ibreak	If -1, break key has been hit.	gprecv	gpbufo gprecv letter map movdrw
258	linflg	Present line type 0,1,2,or 3.	movdrw gplnpt gpputc	gplnpt
259	kbmflg	If =1 dump the plot buffer before new prompt is given in geolab.	gpbufo movdrw gpclip	gpbufo
260	ifirst	If =1 reinitialize common block from scratch.	gpcomi	movdrw
261	ktrflg	If =1 reinitialize the transformations.	gpputc movdrw gptran	gptran movdrw
262	kscale	If =1 reinitialize before scaling, rotating, or offsetting plot. =2 scaling etc. initialized and in use. =0 scaling etc. not in use.	movdrw gpputc gpdevo	gpdevo gpscal movdrw
263	kerror	=0 no error, =1 warning, =2 fatal error	movdrw all	gperr gprecv
264	nchar	Number of hardware lettering sizes avail.	gpdevs	gpdevs
265	kchar	Number of hardware lettering now in use.	gpdevs movdrw	letter gpbufo gpdevs
266	linspa	If =1 this is a curvilinear transformation.	gptran grid movdrw	movdrw gpticl
267	linmod	Linemode 1=line, 2=character, 3=symbol.	movdrw letter gpsymb	gplnpt
268	lmode	=1 move, =2 vector line, =3 points, =4 line of points, =5 defocused line of points.	gpdevs	gpsymb gpdevs
269	iprnbf	If =1 print output buffer as integers.	gpdevo	gpbufo gpputc
270	ltype	Present line type. Integer*4.	movdrw gpdevs	gplnpt gpsymb gpdevs movdrw letter gpdash
271	filnam	File name character*60	pfile	gpfile
272	pspaces	Print pspaces	gpgetc	
272	i0 to i20	Miscellaneous constants.	blockd	many

```
#  Makefile for overlayed version of geoplt
#          September 1983  PLWard

#     NOTE:
#     If the routines gpvers_.c or gpqwc_.c are changed, it may be
#     necessary to install new spoolers and daemons for lpr and vpr.
#     To do this, run the makefile in /usr/src/cmd/lpr by typing:
#         make all install
#     You must have root permission to do this.

SOURCES = axis.f blockd.f cormap.f corout_.c frame.f geoplt.f glplot.f \
        gpanja.f gpaxis.f gpcalc_.c gpclip.f gpcomi_.c gpdash.f gpdevo.f \
        gpdevs_.c gperr_.c gpfile.f gpfili_.c gpfortran.c gpfunc.f \
        gpgetc.f gphlet.f gpletr_.c gplnpt.f gppick.f gpputc.f gpras.f \
        gprecv_.c gpscal.f gpsinc.f gpsymb.f gptek.f gpticl.f gpticr.f \
        gptran_.c gpvers_.c grid.f gtcir_.c ibaudr_.c label.f letter.f loadletr.c \
        map_.c mapdig.c maputil.c movdrw.f mscale_.c newpol_.c  \
        pltfor.f pltfor.ops pltltr.f propor.f scale.f shade.f tekio_.c tek2.c

INCLUDES = common GPCOM.h gpaxis.h gpbuf.h gpbufc.h \
        gpcom.h gpcomc.h gpletr.h lmccom.h version.h

DATA = letter.codes corsmap.d

ROOT =  blockd.o geoplt.o glplot.o gpaxis.o \
        gpclip.o gpdash.o gpdevo.o gpfunc.o gphlet.o \
        gpletr_.o gplnpt.o gprecv_.o gpscal.o gpsinc.o \
        gpsymb.o  gpticl.o gpticr.o gptran_.o  \
        letter.o movdrw.o pltltr.o

OVL1 = gptek.o tekio_.o tek2.o gppick.o ibaudr_.o grid.o

OVL2 = gpras.o gpvers_.o gpdevs_.o gpfile.o gpfili_.o  gpanja.o  \
       shade.o gpcomi_.o gpputc.o

OVL3 = axis.o frame.o gpcalc_.o gperr_.o propor.o label.o  \
        map_.o gtcir_.o mscale_.o newpol_.o gpgetc.o

OBJECTS = $(ROOT) $(OVL1) $(OVL2) $(OVL3)

USRDIR = /usr/src/lib/geop

LIBES = -lovprog -lovF77 -lovI77 -lovm -lovc

FFLAGS = -O -U

CFLAGS  = -O

CC = cc -V

EC = f77 -V

LDFLAGS = -X -u

gp : $(OBJECTS)
        @echo "LOADING GEOPLT"
        @ld $(LDFLAGS) _MAIN__ /lib/crt0.o -i -o geoplt -Z $(OVL1) \
        -Z $(OVL2) -Z $(OVL3) -L $(ROOT)  $(LIBES)
        @size geoplt
        @echo "allow 57343(8191+8191+8191)"
        @echo "GEOPLT READY"^G^G^G

libgeop: gpfortran.o scale.o
        @echo Making libgeop
```

```
	-rm -f libgeop.a
	@ar x /usr/lib/libprog.a makechild.o pipack.o
	@ar cr libgeop.a gpfortran.o  scale.o makechild.o pipack.o
	@rm makechild.o pipack.o
	@chmod 664 libgeop.a
	@echo Done making libgeop.

libgeop4:
	cc -O -c -DLENGTH=LONG gpfortran.c
	f77 $(FFLAGS) -I4 -c scale.f
	@echo Making libgeop4
	-rm -f libgeop4.a
	@ar x /usr/lib/libprog.a makechild.o pipack.o
	@ar cr libgeop4.a gpfortran.o scale.o makechild.o pipack.o
	@rm makechild.o pipack.o gpfortran.o scale.o
	@chmod 664 libgeop4.a
	@echo Done making libgeop4.

install:
	-cp geoplt      /usr/geolab/geoplt
	-cp libgeop.a   /usr/lib/libgeop.a
	-cp libgeop4.a  /usr/lib/libgeop4.a
	-cp GPCOM.h     /usr/include/GPCOM.h
	-cp world.co    /usr/maps/world.co
	-cp mapdig      /usr/bin/mapdig
	-cp maputil     /usr/bin/maputil

pltfor:  pltfor.o
	@f77 -O -o pltfor pltfor.o -lgeop
	@echo "Pltfor ready ^G^G^G"

map : cormap.o corout_.o
	@f77 -o cormap -i cormap.o corout_.o
	@echo "cormap ready"
	-cormap
	@echo "world.co ready"

maputil: maputil.o newpol_.o
	@$(CC) -X -o maputil maputil.o newpol_.o $(LIBES)
	@echo "MAPUTIL READY ^G^G^G"

gpletr:
	-cc -o loadletr loadletr.c
	@echo Making gpletr.h from letter.codes
	@loadletr

listing:
	@ls -lFa > liss
	@size *.o >> liss
	@/usr/bin/num "+`pwd`---------`date`" README Makefile liss  \
	$(INCLUDES) $(SOURCES)  | lpr
	@rm liss

mapdig:  mapdig.o gptran_.o newpol_.o
	@ echo "MAKING MAPDIG"
	@ld $(LDFLAGS) _gperr_ /lib/crt0.o -i -o mapdig gptran_.o \
	pol_.o mapdig.o $(LIBES)
	@size mapdig
	@echo MAPDIG READY

mapdiglist:
	@/usr/bin/num "+`pwd`---------`date`" Makefile gpcom.h gpcomc.h \
	mapdig.c gptran_.c | lpr
```

```
maputillist:
	@/usr/bin/num "+`pwd`---------`date`" Makefile maputil.c newpol_.c | lpr

usr:
	@cp $(SOURCES) $(USRDIR)
	@cp $(DATA)    $(USRDIR)
	@cp $(INCLUDES) $(USRDIR)
	@cp $(OBJECTS)  $(USRDIR)
	@cp Makefile    $(USRDIR)
	@echo GEOPLOT COPIED TO $(USRDIR) ^G^G^G

gpfile.o              gphlet.o  gptek.o       : gpcom.h gpbuf.h
gpras.o  gpanja.o               propor.o      : gpcom.h
gpdash.o  gpdevo.o  gpgetc.o  pltltr.o       : gpcom.h
gplnpt.o  gpputc.o  gpsymb.o  label.o        : gpcom.h
blockd.o  frame.o   gpscal.o  gppick.o       : gpcom.h
axis.o    grid.o    gpaxis.o  gpticl.o       : gpcom.h gpaxis.h
gpticr.o                                     : gpcom.h gpaxis.h
movdrw.o  gpclip.o  letter.o                 : gpcom.h lmccom.h
gprecv_.o map_.o    mapdig.o  gpcomi_.o      : gpcomc.h
gpfili_.o gpvers_.o gpdevs_.o gpcalc_.o      : gpcomc.h
gptran_.o tekio_.o                           : gpcomc.h
gpvers_.o                                    : gpbufc.h
gpletr_.o                                    : gpletr.h
gpcomi_.o                                    : version.h
pltfor.o                                     : /usr/include/GPCOM.h
gpputc.o gpgetc.o                            : GPCOM.h

gptran_.o       : gpcomc.h gptran_.c
	$(CC) $(CFLAGS) -c -DGEOPLOT gptran_.c

gpfortran.o     : gpfortran.c
	cc -O -c gpfortran.c

scale.o         : scale.f GPCOM.h
	f77 -O -U -c scale.f
```

```
c----This include block allows a fortran user of geoplot to reference
c     each element of gpcom by the first 6 letters of the name given in
c     the manual but in all capital letters. letspacev is letspv in order
c     not to conflict with letspace.
c
c-----Programmer: PLWard, USGS, Menlo Park, 5/7/80
c
c
      integer*2 PICTSP,GRIDSP,SUBJSP,DATASP,LABLSP,OMITSP,
     1 FULLSP,PSPACE,CLIP,TRANS,TRANSC,CURINC,LINTYP,
     2 LINWID,LINSPA,LINSID,LINSHA,LINCLI,CLIPLI,LETHEI,LETERR,LETCLI,
     3 LETASP,LETANG,LETSPA,LETSPV,UP,SYMSHA,SYMLEN,SYMSPA,SYMANG,
     4 SYMFAC,TICXIN,TICYIN,TICXOU,TICYOU,TICFAC,TICMIN,LABMIN,LABDIS,
     5 LABFAC,LABANG,MAXLEV,MINLAB,DATAY,NOAXIS,TICPAG,POFFSE,PANGLE,
     6 PGAIN,BEAMTY,GPUSYM,JULIAN,GPZONE,GPRINT,TIMEBA,TICBAS,LINEID,
     7 NUMFIE,PICKTY,DEVMAX,PAGEMA,HOMEI,INCHSC,INCHRE,RASMIN,NCHAR,
     8 KCHAR,NUMDEV,POINTP,POLE,MAPSCA
c
      parameter(PICTSP=21)
      parameter(GRIDSP=27)
      parameter(SUBJSP=33)
      parameter(DATASP=39)
      parameter(LABLSP=191)
      parameter(OMITSP=45)
      parameter(FULLSP=161)
      parameter(PSPACE=272)
      parameter(CLIP=163)
      parameter(TRANS=190)
      parameter(TRANSC=61)
      parameter(CURINC=90)
      parameter(LINTYP=181)
      parameter(LINWID=84)
      parameter(LINSPA=81)
      parameter(LINSID=184)
      parameter(LINSHA=87)
      parameter(LINCLI=188)
      parameter(CLIPLI=189)
      parameter(LETHEI=91)
      parameter(LETERR=96)
      parameter(LETCLI=187)
      parameter(LETASP=92)
      parameter(LETANG=93)
      parameter(LETSPA=94)
      parameter(LETSPV=95)
      parameter(POLE=97)
      parameter(MAPSCA=133)
      parameter(UP=206)
      parameter(SYMSHA=76)
      parameter(SYMLEN=77)
      parameter(SYMSPA=78)
      parameter(SYMANG=79)
      parameter(SYMFAC=80)
      parameter(TICXIN=107)
      parameter(TICYIN=112)
      parameter(TICXOU=117)
      parameter(TICYOU=122)
      parameter(TICFAC=101)
      parameter(TICMIN=127)
      parameter(LABDIS=136)
      parameter(LABMIN=142)
      parameter(LABFAC=148)
      parameter(LABANG=154)
      parameter(MAXLEV=216)
      parameter(MINLAB=222)
```

```
      parameter(DATATY=228)
      parameter(NOAXIS=207)
      parameter(TICPAG=213)
      parameter(POFFSE=70)
      parameter(PANGLE=72)
      parameter(PGAIN=73)
      parameter(BEAMTY=197)
      parameter(GPUSYM=160)
      parameter(JULIAN=203)
      parameter(GPZONE=204)
      parameter(GPRINT=198)
      parameter(TIMEBA=180)
      parameter(TICBAS=75)
      parameter(LINEID=234)
      parameter(NUMFIE=235)
      parameter(PICKTY=237)
      parameter(DEVMAX=1)
      parameter(PAGEMA=3)
      parameter(HOME=5)
      parameter(INCHSC=7)
      parameter(INCHRE=8)
      parameter(RASMIN=9)
      parameter(NCHAR=264)
      parameter(KCHAR=265)
      parameter(NUMDEV=253)
      parameter(POINTP=16)
c
```

```
c--*--Common block used between axis, grid, and gpaxis.------------------
c
      integer ltrans,iaxis,iax,iother,numtic(5),numax,labps
      real drange(2),dother,pbegin(2),pend(2),
     1   c1,c2,sinx,cosx,tdist,dbegin(2),dend(2),
     2   axsign,pold(2),distic,dislb,widlab,differ(5),
     3   poslab(5),ticbax,ticbay,ticdat,ang
c
      common /gpax/ drange(2),dother,pbegin(2),pend(2),ltrans,iaxis,
     1   iax,iother,c1,c2,sinx,cosx,tdist,numax,dbegin(2),dend(2),
     2   axsign,pold(2),distic,dislb,widlab,differ(5),
     3   numtic(5),poslab(5),ticbax,ticbay,ticdat,ang,labps
c
```

```
c-*--GPBUFF--Geoplot output buffer and related variables.--------------
      integer lenbuf,lenmax,iegm,kzaxis,kline,intens,kplt,
     &   iplt,ibuffr,in,iout
      common/gpbuff/lenbuf,lenmax,iegm,kzaxis,kline,intens,kplt,
     &   iplt(5),ibuffr(2048),in(5),iout(2)
c
```

```
/*-----GPBUFC.H----C version of geoplot output buffer-----------------*/
struct bcom{
     int pad[12];
     int bfl[2048];
     int end[7];
} *bufcom;
```

```
c--*--GPCOM-----Geoplot common block.----------------------Total words (292)
      real devmax(2),dpage(2),phome(2),dpgin,hpgin,rasmin,
     1  pshade,ratchw,ratsph,spalin(3),pointp(2),setsca,pad1(2),
     2  pspace(24),omitsp(16),
     3  trans(9),offset(2),pangle,pgain(2),ticbas,
     4  symshp,symlen,symspa,symang,symfac,
     5  space(3),width(3),shade(3),curinc,
     6  height,aspect,slopel,spacel,spaceh,chrerr,pole(2),pad2(2),
     7  ticfac(6),ticin(10),ticout(10),ticmin(6),mapsca,pad3(2),
     8  dislab(6),spalab(6),faclab(6),anglab(6),usym,fullsp(2),clip(3),
     9  pad4(14)
      double precision timbas
      integer*4 lintyp,ltype
      integer kside,kltrsp,kpltsp,linclp,ktrans,
     1  lablsp,kbmtyp,idbug,intact,gtrans,
     2  m1pad,iupad,julian,izone,kpad,
     3  iup,noaxis,itcpag,lpad,mpad,
     4  maxpow,minlab,idtype,lineid,nfield,ksubj,nfare,ipad1,
     5  ipad2,
     6  kkchar,nomit,numdev,letchg,ibreak,linflg,kbmflg,ifirst,
     7  ktrflg,kscale,kerror,nchar,kchar,
     8  linspa,linmod,lmode,iprnbf,
     9  i0,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,i16,i17,
     A  i18,i19,i20
      integer LOUSR,INTG,HIUSR,MAXCOM
      character*60 filnam
c
c******Geolab and fortran level users may only read from this part. (20)
      common /gpcom/ devmax(2),dpage(2),phome(2),dpgin,hpgin,rasmin,
     1  pshade,ratchw,ratsph,spalin(3),pointp(2),setsca,pad1(2),
c******Geolab and fortran level users may read and write from here.(160)
     2  pspace(24),omitsp(16),
     3  trans(9),offset(2),pangle,pgain(2),ticbas,
     4  symshp,symlen,symspa,symang,symfac,
     5  space(3),width(3),shade(3),curinc,
     6  height,aspect,slopel,spacel,spaceh,chrerr,pole(2),pad2(2),
     7  ticfac(6),ticin(10),ticout(10),ticmin(6),mapsca,pad3(2),
     8  dislab(6),spalab(6),faclab(6),anglab(6),usym,fullsp(2),clip(3),
c-------Begin integer part of common.(after timbas)------------------ (70)
     9  pad4(14),timbas,lintyp(3),kside(3),kltrsp,kpltsp,linclp,ktrans,
     A  lablsp(6),kbmtyp,idbug,intact,gtrans,
     B  m1pad,iupad,julian,izone,kpad,iup,noaxis(6),itcpag,lpad,mpad,
     C  maxpow(6),minlab(6),idtype(6),lineid,nfield,ksubj,nfare,ipad1(3)
     D  ,ipad2(9),
c******Geolab and fortran level users may only read from this part. (21)
     E  nosym,kkchar,nomit,numdev(3),letchg,ibreak,linflg,kbmflg,
     F  ifirst,ktrflg,kscale,kerror,nchar,kchar,
     G  linspa,linmod,lmode,iprnbf,ltype,
     H  filnam,
c-----Constants 0 to 20 used in various call statements.------------- (21)
     I  i0,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,i16,i17,
     J  i18,i19,i20
c
c-----When changing size of common be sure to change dimension and
c------parameter statements.
      real rcom(180)
      integer icom(91)
      equivalence (devmax(1),rcom(1)),(kside(1),icom(1))
      parameter (LOUSR=21, INTG=181, HIUSR=249, MAXCOM=270)
c
```

```
/* GPCOMC----Geoplot common block. C version----------292 words*/
#define COMLEN  1014
#define FILNAMLEN 60
struct scom {
      float devmax[2],dpage[2],phome[2],dpgin,hpgin,rasmin,
        pshade,ratchw,ratsph,spalin[3],pointp[2],setsca,pad1[2],
        pspace[24],omitsp[16],
        trans[9],offset[2],pangle,pgain[2],ticbas,
        symshp,symlen,symspa,symang,symfac,
        space[3],width[3],shade[3],curinc,
        height,aspect,slopel,spacel,spaceh,chrerr,pole[2],pad2[2],
        ticfac[6],ticin[10],ticout[10],ticmin[6],mapsca,pad3[2],
        dislab[6],spalab[6],faclab[6],anglab[6],usym,fullsp[2],clip[3],pad4[14];
      double timbas;
      long lintyp[3];
      int kside[3],kltrsp,kpltsp,linclp,ktrans,
        lablsp[6],kbmtyp,idbug,intact,gtrans,
        m1pad,iupad,julian,izone,kpad,
        iup,noaxis[6],itcpag,lpad,mpad,
        maxpow[6],minlab[6],idtype[6],lineid,nfield,ksubj,nfare,ipad1[3],
        ipad2[9],
        nosym,kkchar,nomit,numdev[3],letchg,ibreak,linflg,kbmflg,ifirst,
        ktrflg,kscale,kerror,nchar,kchar,
        linspa,linmod,lmode,iprnbf;
      long ltype;
      char filnam[FILNAMLEN];
        int i0,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,i16,i17,i18,i19,i20;
        } *common;
```

```
/* Include block for letter codes used by gpletr_.c */
#define NCHARS 128
long index[]={
 1,33,65,97,129,161,198,389,548,677,
 838,1032,1289,1574,1769,2050,2116,2245,2405,2574,
 3025,3564,3945,4236,4615,4837,5000,5257,5549,5961,
 6252,6633,6913,6951,7173,7339,7698,8270,8717,9122,
 9188,9316,9448,9701,9862,10050,10117,10274,10347,10694,
 10891,11246,11689,11977,12268,12644,12784,13292,13675,14028,
 14403,14501,14659,14766,15214,15657,15951,16424,16681,16967,
 17190,17386,17704,17960,18216,18470,18659,18757,18916,19049,
 19335,19564,19946,20268,20645,20806,20995,21093,21253,21416,
 21668,21796,21922,21988,22115,22210,22274,22348,22727,22952,
 23207,23435,23786,24108,24488,24746,25066,25384,25636,25774,
 26216,26473,26763,27115,27462,27660,28039,28262,28451,28551,
 28773,28939,29286,29479,29701,29863,30084,0
};

unsigned int nodes[]={
 0x5f2f,0x1f6f,0x3f4f,0x878,0x6aa8,0x6678,0x5308,0x5da8,0x5305,0x5ea5,
 0x503,0xdad,0xa303,0x235d,0x830a,0xaa23,0x357b,0xf089,0x29f0,0x2787,
 0x8978,0x3a29,0xf027,0x3876,0x8726,0x7687,0x897a,0x2a76,0x4637,0x394a,
 0x7af0,0x5c54,0x2eae,0x2244,0x6c8e,0x2838,0x535e,0xae8d,0x2d78,0x2383,
 0x6468,0x79f0,0x6859,0x3928,0x2433,0x5364,0x7384,0x3039,0xf038,0x4969,
 0x7877,0x6636,0xf066,0x7574,0x6343,0x3457,0x3634,0x4363,0x7476,0x393a,
 0x4b5b,0x7936,0x56f0,0x7959,0x4745,0x5373,0x316a,0xf078,0x6949,0x3824,
 0x3353,0x6478,0x3849,0x66f0,0x7978,0x4333,0x66f0,0x4b73,0x2049,0xf035,
 0x4353,0x6579,0x3849,0x5845,0xf058,0x6978,0x5049,0x3825,0x2433,0x4456,
 0xf044,0x5364,0x8889,0x3349,0xf039,0x79f0,0xf069,0x7326,0x76f0,0x6478,
 0x6949,0x3824,0x3353,0x6443,0x6af0,0x3937,0x4666,0x7779,0x7f5d,0x57f0,
 0x5443,0x6354,0x494d,0xf06d,0x6943,0x4df0,0x6d63,0xf08a,0x2af0,0x2686,
 0x2534,0x7485,0x8778,0x3829,0x2b3c,0x7c8b,0xf04d,0x43f0,0x6d63,0x238d,
 0xf03b,0x4c3d,0x2c3b,0xf073,0x6475,0x8473,0x7774,0x6343,0x3436,0x7b7c,
 0x6d4d,0x3c3b,0x836d,0x496d,0x4b45,0x634d,0x6b65,0x438c,0x24f0,0x535d,
 0xf02c,0x8428,0x88f0,0x5b55,0x5343,0x4454,0x5331,0x2888,0x5343,0x4454,
 0x5323,0x8d34,0x7c6d,0x4d2b,0x2543,0x6385,0x8b7c,0x4c5d,0x53f0,0x4363,
 0x2c3d,0x7d8c,0x8978,0x3827,0x2383,0x842c,0x3d7d,0x8c89,0x7848,0xf078,
 0x8784,0x7333,0x242d,0x2888,0xf06d,0x63f0,0x5373,0x2433,0x7384,0x8778,
 0x282d,0x8d27,0x3878,0x8784,0x7333,0x242c,0x3d7d,0x8c2d,0x8d35,0x3338,
 0x292c,0x3d7d,0x8c89,0x7887,0x8473,0x3324,0x2738,0x7824,0x3373,0x848c,
 0x7d3d,0x2c29,0x3878,0x8948,0x4959,0x5848,0xf053,0x4344,0x5453,0x4849,
 0x5958,0x48f0,0x5343,0x4454,0x5331,0x8d28,0x8389,0x29f0,0x2787,0x2d88,
 0x232a,0x2c3d,0x7d8c,0x8978,0x5856,0xf054,0x4363,0x5467,0x6b5b,0x4a47,
 0x878c,0x7d4d,0x2b25,0x4363,0x8523,0x2a4d,0x6d8a,0x83f0,0x2888,0x333d,
 0xf02d,0x7d8c,0x8978,0x38f0,0x7887,0x8473,0x238b,0x6d4d,0x2b25,0x4363,
 0x8533,0x3df0,0x2d6d,0x8b85,0x6323,0x8323,0x2d8d,0xf028,0x6823,0x2d8d,
 0xf028,0x688b,0x6d4d,0x2b25,0x4363,0x8588,0x6823,0x2df0,0x2888,0xf08d,
 0x833d,0x7df0,0x5d53,0xf033,0x7325,0x4353,0x757d,0xf06d,0x8d23,0x2df0,
 0x8d28,0x832d,0x2383,0x232d,0x588d,0x8323,0x2d83,0x8d43,0x252b,0x4d6d,
 0x8b85,0x6343,0x232d,0x6d8b,0x8a68,0x2843,0x252b,0x4d6d,0x8b85,0x6343,
 0xf066,0x8323,0x2d6d,0x8b8a,0x6828,0xf048,0x8324,0x3363,0x8587,0x7838,
 0x292b,0x4d7d,0x8c2d,0x8df0,0x5d53,0x2d25,0x4363,0x858d,0x2d53,0x8d2d,
 0x2358,0x838d,0x238d,0xf02d,0x832d,0x2b59,0xf053,0x598b,0x8d2d,0x8d23,
 0x838d,0x2d23,0x832d,0x832d,0x8d83,0x2323,0x5983,0x22a2,0x3d69,0x2839,
 0x7988,0x8473,0x3324,0x2637,0x7786,0x2d23,0x6385,0x8769,0x2988,0x6949,
 0x2725,0x4363,0x848d,0x8343,0x2527,0x4989,0x2676,0x8788,0x7939,0x2824,
 0x3373,0x847c,0x6d5c,0x53f0,0x4363,0xf048,0x6821,0x3070,0x8188,0x7939,
 0x2824,0x3373,0x8423,0x2df0,0x2749,0x6987,0x835a,0x4b6b,0x5af0,0x5853,
 0xf043,0x6321,0x3060,0x7178,0xf07a,0x8b6b,0x7a2d,0x23f0,0x2689,0xf047,
 0x834d,0x5c54,0x6323,0x29f0,0x2839,0x4958,0x53f0,0x5869,0x7988,0x8323,
 0x29f0,0x2749,0x6987,0x8343,0x2527,0x4969,0x8785,0x6343,0x2029,0xf028,
 0x3969,0x8785,0x6333,0x2480,0x89f0,0x8879,0x4927,0x2543,0x7384,0x3339,
 0xf037,0x5878,0x2433,0x7384,0x8576,0x3627,0x2839,0x7988,0x3979,0xf05d,
 0x5463,0x7429,0x2543,0x6385,0x8929,0x5389,0x2925,0x4355,0x6385,0x8923,
```

```
 0x89f0,0x2983,0x2130,0x7081,0x89f0,0x8574,0x3425,0x2928,0x3989,0x2373,
 0x846d,0x5c59,0x4857,0x5463,0x5d59,0xf057,0x534d,0x5c59,0x6857,0x5443,
 0x2839,0x7788,0x5746,0x3627,0x293a,0x4a59,0x6a7a,0x8987,0x7666,0x0
};
```

```
c--*--Common for movdrw,gpclip, and letter.--------------------------------
      integer lastyp,kmode,kpenl,jkl
      real point(4),pointl(2),penl(2)
      common/lmccom/point(4),pointl(2),lastyp,kmode,penl(2),kpenl,
     &  jkl
c
```

plot10.h

```
      common/plot10/ page(2),kline,kcode,ntran,nterm,
     & pointp(2),pspace(24),dpgin
```

```
char version[] = "14 Sept 1983";
```

```
c-*--AXIS----PLOT AND LABEL AN AXIS-----------------------------------
c
      subroutine axis(numbax)
c
c-----Programmer: PLWard,USGS, Menlo Park, California 94025, January 1980
c
c-*--Numax is the axis number where
c      1  lower x axis
c      2  upper x axis
c      3  lower or left  hand y axis
c      4  upper or right hand y axis
c
      save
c
      include 'gpcom.h'
      include 'gpaxis.h'
c
      integer numaxi(4),labpo(8),ksubjs,numbax,modes,linsa,MONE
      data numaxi/0,2,3,1/
      data labpo /6,7,4,5,7,6,5,4/
      data MONE/-1/
c
      if(idbug.ge.3)call printn("axis  : %d\n",numbax)
c
      numax=numbax
c
c-*--Check to be sure that the axis code (numax) is valid.--------------
      if(numax.ge.1.and.numax.le.4) goto 10
      kerror=2
      call gperr(kerror,"axis","Illegal axis code of",numax)
      return
c
c-*--Set axis constants.-----------------------------------------------
 10   iax=(numax/3)*2
      ksubjs=ksubj
      ksubj=0
      iaxis=numax/3+1
      iother=1
      if(iaxis.eq.1) iother=2
      if(idbug.ge.3)call printn("axis  : iax=%d iaxis=%d iother=%d\n",
     1   iax,iaxis,iother)
c
c-*--Set data ranges.--------------------------------------------------
      drange(1)=pspace(19+iax)
      drange(2)=pspace(20+iax)
      dother=pspace(20+numax-iax*2)
      dbegin(1)=pspace(19)
      if(numax.eq.4) dbegin(1)=pspace(20)
      dbegin(2)=pspace(21)
      if(numax.eq.2)dbegin(2)=pspace(22)
      dend(1)  =pspace(20)
      if(numax.eq.3)dend(1)=pspace(19)
      dend(2)  =pspace(22)
      if(numax.eq.1)dend(2)=pspace(21)
c
c-*--Change lintype to 1 for duration of axis.-------------------------
      linsa=lintyp(1)
      lintyp(1)=1
c
c-*--Draw line and set page ranges.------------------------------------
      call movdrw(dbegin(1),dbegin(2),MONE)
      pbegin(1)=pointp(1)
      pbegin(2)=pointp(2)
      modes=-2
```

```
      if(noaxis(numax).eq.1) modes=-1
      call movdrw(dend(1),  dend(2),modes)
      pend(1)=pointp(1)
      pend(2)=pointp(2)
c
c--*--Determine angle of line for non curvilinear transformations.--------
      ang=numaxi(numax)*90.0
c-----Gross fudge for polar transformations
      if(ktrans.ne.10.and.ktrans.ne.12.and.ktrans.ne.13) goto 100
      ltrans=4
      if(dislab(numax).lt.0)ltrans=0
      goto 101
100   ltrans=0
      if(dislab(numax).lt.0)ltrans=4
101   labps=-labpo(numax+ltrans)
c
c--*--Initialize the transformations.-------------------------------------
      ltrans=0
c
c--*--Tic and label the axis.---------------------------------------------
      call gpaxis
      lintyp(1)=linsa
      ksubj=ksubjs
      end
```

```
      block data
      include 'gpcom.h'
      data i0,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,
     1   i15,i16,i17,i18,i19,i20/0,1,2,3,4,5,6,7,8,9,10,11,12,
     2   13,14,15,16,17,18,19,20/
      end
```

```
c-*--CORMAP. Load coarse map data into file world.co for use by map_.c
c
      program cormap
c
c     Programmer PLWard, US Geological Survey, Menlo Park, Ca. 11/25/79
c
c-----Input data are in card format and read from file corsmap.d.
c
      real penup,dat(10600)
      integer*2 ii,j,j1,j2,m
c
3000  format(i3,f6.1,f7.1,32f2.1)
3001  format(40f2.1)
c
      open (4,file='corsmap.d')
      rewind 4
c
      penup=999.0
      ii=0
31        j1=ii+1
          j2=j1+32
c
c         Store data in dat lon(i.e. x) first, lat (i.e. y) second
          read(4,3000) m,(dat(j+1),dat(j),j=j1,j2,2)
          if (m.le.0) go to 33
          j1=j2+2
          j2=j1+(m-36)
          if (j2.ge.j1) read (4,3001) (dat(j+1),dat(j),j=j1,j2,2)
          ii=ii+2
          do 32 j=3,m
          ii=ii+1
          dat(ii)=dat(ii-2)-dat(ii)+5.
32        continue
          ii=ii+1
          dat(ii)=0.0
          ii=ii+1
          dat(ii)=penup
          go to 31
33    continue
      close (4)
      call corout(ii,dat)
      end
```

```
#include <stdio.h>

corout_(num,dat)
    int *num;
    float dat[];
{
    int fd;
    long out;
    if ((fd=creat("world.co",0664)) == -1){
        printf("Cormap: Unable to create world.co\n");
        exit(1);
    }
    if ((fd=open("world.co",1))== -1){
        printf("Cormap: Unable to open world.co\n");
        exit(1);
    }
    printf("Cormap: %d map points written into world.co\n",(*num)/2);
    out = (*num)*4;
    while(out>=512) {
        write(fd,dat,512);
        dat+=128;
        out-=512;
    }
    write(fd,dat,(int)out);
    close(fd);
}
```

```
c--*--FRAME------CALLS AXIS 4 TIMES FOR SIDES OF FRAME.--------------------
      subroutine frame
      include 'gpcom.h'
      integer i
      do 5 i=1,4
      if(ibreak.eq.-1.or.kerror.ne.i0) return
 5    call axis(i)
      end
```

```
c--*--GEOPLT---THIS IS THE MAIN PROGRAM OF THE GEOPLOT PROCESS.----------
      integer i1,i2
      data i1,i2/1,2/
c
c     <Read in geoplot common from home directory.>
      call gpcomi(i1)
c     <Start up the pipe receiving package.>
      call gprecv()
c*****close any open files
c     <Write out geoplot common to home directory.>
      call gpcomi(i2)
      end
```

```
c------glplot----High level geolab plotting command accepting freeform
c-------command string and data.
c-------Not implemented yet.      Programmer PLWard.
      subroutine glplot
      call printn("glplot: not implemented yet.\n")
      return
      end
```

```
c-----gpanja----Plotting on Anderson-Jacobsen printer---------------
c-------Not implemented yet.  Programmer PLWard.
      subroutine gpanja(mode,point)
c
      include 'gpcom.h'
c
      integer i,mode
      real point(2)
      kerror=1
      i=0
      call gperr(kerror,"gpanja",
     1  "Anderson-Jacobsen software not included yet",i)
      return
      end
```

```
c-*--GPAXIS---GRID A LINE, DECIDING ON TIC MARKS AND LABELS-------------
c
      subroutine gpaxis
c
c----Programmer: PLWard, USGS, Menlo Park, Calif. 94025 January, 1980
c
      save
c
      include 'gpcom.h'
      include 'gpaxis.h'
      real POSMAX
      real d,ticscl,power,start
      integer MAXLEV,ONE,mxpow,i,j,lchars,levl,levt
      parameter (POSMAX=0.4, MAXLEV=5)
      data ONE/1/
c
c
      if(idbug.ge.3)call printn("gpaxis: iax=%d iaxis=%d iother=%d\n",
     1    iax,iaxis,iother)
c-*--Set the scaling factors for tics.---------------------------------
      ticscl=pspace(10-iax)-pspace(9-iax)
      ticbax=-sin(ang*0.0174533)*ticscl*ticfac(numax)
      ticbay= cos(ang*0.0174533)*ticscl*ticfac(numax)
      d=1.0
      if(numax.eq.2.or.numax.eq.4)d=-1.0
      ticdat=(pspace(22-iax)-pspace(21-iax))*ticfac(numax)*d
      if(maxpow(numax).gt.MAXLEV)maxpow(numax)=MAXLEV
      axsign=1.0
      if(drange(2).lt.drange(1)) axsign=-1
c
c-*--Initialize tic heirarchy for date variables.----------------------
      if(idtype(numax).ne.1) goto 100
      goto 130
c
c-*--Find the maximum power on this axis.------------------------------
100   power=alog10(abs(drange(2)-drange(1)))
      if(sign(1.0,drange(1)).ne.sign(1.0,drange(2)))
     1  power=max(alog10(abs(drange(1))),alog10(abs(drange(2))))
      if(idbug.ge.2)call printn("gpaxis: power=%f maxpow=%d\n",
     1    power,maxpow(numax))
      if(power.le.0.0)power=power-0.999999
      mxpow=power
c
c-*--Find the largest grid increment.----------------------------------
      differ(1)=10.0**mxpow*axsign
c
c-*--Find the starting value.------------------------------------------
      start=int(drange(1)/differ(1))*differ(1)
      if((drange(1)-axsign*start).lt.0.0) start=start-axsign*differ(1)
      numtic(1)=1.0+abs(drange(2)-start)/differ(1)
      if(maxpow(numax).gt.MAXLEV)maxpow(numax)=MAXLEV
      do 120 i=2,maxpow(numax)
           differ(i)=differ(i-1)/10.0
120        numtic(i)=10
c
c-*--Debugger.---------------------------------------------------------
      if(idbug.ge.2)call printn("gpaxis: from %f to %f start=%f\n
     1diff=%f maxpow=%d sign=%f\n",drange(1),drange(2),start,differ(1),
     1mxpow,axsign)
c
c-*--Find the longest label length.------------------------------------
130   lchars=abs(mxpow)+2
      widlab=spalab(numax)+lchars*height*aspect*(spacel+1.0)
      if(idbug.ge.2)
```

```
      lcall printn("gpaxis: lchars=%d widlab=%f\n",lchars,widlab)
c
c--*--Find label positions for each level.-------------------------------
      do 140 i=1,MAXLEV
           poslab(i)=dislab(numax)
           if(abs(poslab(i)).lt.1.0.or.poslab(i).eq.0.0) goto 140
           do 139 j=i,MAXLEV
                if(poslab(i).lt.0.0) goto 133
                if(poslab(i).lt.ticin(j).and.ticin(j).gt.0.0.and.
     1             ticin(j).lt.POSMAX) poslab(i)=ticin(j)
                if(poslab(i).lt.-ticout(j).and.ticout(j).lt.0.0.and.
     1             ticout(j).gt.-POSMAX) poslab(i)=ticout(j)
                goto 139
133             if(poslab(i).gt.ticin(j).and.ticin(j).lt.0.0.and.
     1             ticin(j).gt.-POSMAX) poslab(i)=ticin(j)
                if(poslab(i).gt.-ticout(j).and.ticout(j).gt.0.0.and.
     1             ticout(j).lt.POSMAX) poslab(i)=ticout(j)
139             continue
                if(abs(poslab(i)).gt.POSMAX) poslab(i)=0.0
140        continue
c--*--Move to the low end of the grid.-----------------------------------
      call movdrw(pbegin(1),pbegin(2),ONE)
      if(kerror.ne.i0)return
c
c--*--Initialize pold, etc.----------------------------------------------
      pold(1)=pbegin(1)
      pold(2)=pbegin(2)
c
c--*--If start is a tic, draw it and label it if labels outside.---------
      levl=1
      levt=1
      if(dislab(numax).ge.0.0)levl=0
      if(start.eq.drange(1))call gpticl(start,levl,levt,i1)
c
c--*--Now start recursively descending tic hierarchy.--------------------
      levl=1
      call gpticr(i0,levl,levt,start)
      if(ibreak.eq.-3)ibreak=0
      end
```

```
/* GPCALC_  --Calcomp interface. Output spooled for transmission
      to rsx11-70 where calcomp resides. Programmer Barbara Bekins
*/
#include "gpcomc.h"
#include <pwd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>

char fname[14];
char filname[30]="/usr/spool/rpd/";
extern gpcom_;
int i;
struct {
   int mod;
   float poi[2];} ob;
int simtype[5]={1,12,31212,32,512};
FILE *nfd;

gpcalc_(mode,point)
int *mode;
float point[2];
{
long lt;
   common= &gpcom_;
   if(common->idbug>9) printf("gpcalc_: %d %f %f\n",*mode,point[0],point[1]);
   switch(*mode){
      case 1:case 2:case 3:
         ob.mod= *mode;
         ob.poi[0]=point[0]*33.0;
         ob.poi[1]=point[1]*33.0;
         fwrite(&ob,10,1,nfd);
         break;
      case 6:
         point[0]=common->pointp[0];
         point[1]=common->pointp[1];
         break;
      case 8:
         begincalc();
         common->devmax[0]=10.0;
         common->devmax[1]=1.0;
         common->dpage[0]=common->dpage[1]=33.0/.0002;
         common->rasmin=1./common->dpage[0];
         for(i=0;i<3;i++)common->width[i]=common->rasmin;
         common->dpgin=common->hpgin=33.0;
         common->phome[0]=common->phome[1]=0.0;
         break;
      case 10:
         if(point[0]>0.&&point[0]<99.) common->height=point[0];
         else common->height=.01;
         common->kchar=0;
         break;
      case 11:
         lt=common->ltype=point[0];
         if(!lt) common->lmode=3;
         else if(lt>0 && lt<6){
            common->lmode=2;
            common->ltype=simtype[common->ltype-1];
            }
            else {
               common->lmode=2;
                     common->ltype=1;
            }
         break;
```

```
        case 12:
            ob.mod= *mode;
            ob.poi[0]=point[0];
            fwrite(&ob,10,1,nfd);
            break;
        case 13:
            endcalc();
            begincalc();
            break;
        case 15:
            endcalc();
            break;
        default:
            printf("gpcalc_: Illegal mode of %d\n", *mode);
            return;
    }
    return(0);
}

struct stat *buf;
struct passwd *getpwuid();
int id;

endcalc()
{
float pspac[4];
    ob.mod=23;
    fwrite(&ob,10,1,nfd);
    fseek(nfd,0L,0);
    for(i=0;i<4;i++) pspac[i]=33.*common->pspace[i];
    fwrite(pspac,16,1,nfd);
    fclose(nfd);
            nfd=NULL;
    filname[15]=0;
}

begincalc()
{
    if(!nfd){
/*
        printf("Input filename: ");
        scanf("%s",fname);
*/
    }
/*
    strcat(filname,common->filnam);
*/
    if(!(nfd=fopen(common->filnam,"w"))){
        printf("Cant create plotfile for calcomp:%s\n",common->filnam);
        common->kerror=2;
        return(1);
    }
    fseek(nfd,16L,0);
}
```

```
c-*--GPCLIP---CLIP THE POINT TO BE WITHIN THE PLOT SPACE AND PLOT.-------
c
      subroutine gpclip(kspace)
c
c-----Programmer  PLWard, USGS, Menlo Park, California, February, 1979.--
c
c-*--This subroutine inputs an x,y point through pltusr common and clips
c       it to be within picture (kspace=1), grid (kspace=7), subject
c       (kspace=13), or data (kspace=19) space and then outputs the move
c       or draw to the device via the subroutine gpsymb. It sets penl in
c       common to the present point where the pen is in page coordinates.
c       If linclp = 1 no lines are drawn between successive points
c       along the boundary of clipspace. If linclp = 2 then lines
c       are drawn.
c
      save
c
      include 'gpcom.h'
c-*--Common for movdrw, clpplt, and letter.-----------------------------
      include 'lmccom.h'
      real point3(2)
      integer kspace,jd,jk,num,j,k,jspace,jkmode,jkfo,i,l
      equivalence (point3(1),point(3))
c
c
      if(idbug.ge.5) call printn("gpclip: %d %f %f kspace=%d\n",kmode,
     1  point(1),point(2),kspace)
c
c
c-*--CHECK TO SEE IF DATA IS IN PLOT SPACE, CLIP IF NECESSARY.----------
c     Set indecies for loops: j=1 for x boundaries, =2 for y boundaries;
c       point(1&2) are the coordinates of the incoming point, point(3&4)
c       any intersection point along a boundary, penl(1&2)the last point
c       jkl is the boundary that the last point is outside of, jk is the
c       boundary that this point is outside of,jkfo is the boundary that
c       this data stream first went outside of if it is still outside.
210   jd=1
      jk=0
      num=0
215   j=1
      k=kspace
c
c----If data point is above the upper boundary goto 255.----------------
220   if(point(jd).gt.pspace(k+1)) goto 255
c
c----If data point is below lower boundary go to 250.-------------------
222   if(point(jd).lt.pspace(k)) goto 250
c
c----Have we checked y bounds yet, if so goto 225.----------------------
224   if(j.eq.i2) goto 225
      j=2
      k=k+2
      jd=jd+1
      if(jd.gt.4)jd=4
      goto 220
c
c----If last data point was inside of plot space goto 230.--------------
225   if(jkl.le.i0.or.kmode.ne.2) goto 230
c
c----Draw line along a boundary of plot space.--------------------------
      jspace=((jkl-1)/6)*6+1
      jk=jkl-jspace+kspace
      j=(jkl-jspace)/2+1
      if(j.ne.1.and.j.ne.2)call printn("gpclip: j out of bounds %d\n",j)
```

```
      jkl=-jkl
      goto 260
c
c--*--Set flag to say some data is being plotted.-----------------------
230   kbmflg=1
c
c-----If this is a boundary intersection point goto 240.-----------------
      if(jk.ne.i0) goto 240
c
c-----Plot the point.-----------------------------------------------PLOT-
      if(kspace.eq.i19) call gptran(i1,point)
      call gpsymb(kmode,point)
      jkl=0
c
c-----Update value of "last" data point.----------------------------------
235   penl(1)=point(1)
      penl(2)=point(2)
      kpenl=kspace
      return
c
c-----Plot an intersection point along the boundary of plot space.--------
c-----If the new data point is inside of plot space and old point is out
c      goto 242.
240   penl(1)=point(3)
      penl(2)=point(4)
      kpenl=kspace
      if(kspace.eq.i19) call gptran(i1,point3)
      if (jkl.eq.i0) goto 242
      jkmode=kmode
c
c-----But if the last point on the boundary was not on same boundary
c      then move to the new boundary point.
      if(jk.ne.jkfo.or.linclp.ne.i2) jkmode=1
c
c-----Plot an intersection point on boundary going in.---------------PLOT-
c-----Reset value of last data point to value of intersection point.-----
      call gpsymb(jkmode,point3)
      jkl=0
      goto 210
c
c-----Plot an intersection point on the boundary going out.----------PLOT-
242   jkfo=jk
      call gpsymb(kmode,point3)
244   jkl=jk
      goto 235
c
c-----If point is outside of plot space we end up here. jk is index of
c      boundary that point is outside of.
c-----If we have already tried twice to find an intersection with bound-
c      ary then goto 244 to update values of penl and return.
250   if(point(jd).lt.(pspace(k)-rasmin)) goto 251
      point(jd)=pspace(k)
      goto 224
251   jk=k
      goto 258
255   if(point(jd).gt.(pspace(k+1)+rasmin)) goto 256
      point(jd)=pspace(k+1)
      goto 222
256   jk=k+1
258   if(num.ge.i2) goto 244
      num=num+1
c
c-----Find the intersection point of line between last and this point.
c-----Check if division by zero, if so update penl and return.-----------
```

```
c---If penl is not in the same units (page or data) as point, transform
c       it.
260     if(kspace.eq.kpenl) goto 262
        if(kspace.ne.i19.and.kpenl.ne.i19) goto 261
        i=1
        if(kpenl.eq.i19) i=2
        call gptran(i,penl)
261     kpenl=kspace
262     if (penl(j).eq.point(j)) goto 244
c
c-----If intersection is not  between this & last data point goto 244.---
        if(penl(j).gt.point(j)) goto 265
        if(pspace(jk).lt.penl(j).or.pspace(jk).gt.point(j)) goto 244
        goto 266
265     if(pspace(jk).gt.penl(j).or.pspace(jk).lt.point(j)) goto 244
266     l=5-j
        k=3-j
        jd=3
        point(j+2)=pspace(jk)
        point(l)=penl(k)+(pspace(jk)-penl(j))*(penl(k)-point(k))/
       &(penl(j)-point(j))
        goto 215
        end
```

```
/* GPCOMI_  The routine stores the gpcom in home_directory/plot_com
            and restores it next time geoplot is called.

Programmer: PLWard, USGS, Menlo Park, Ca.  March 14,1980
*/

#include "version.h"
#include  <stdio.h>
#include "gpcomc.h"
#include <pwd.h>
struct passwd *getpwuid();

gpcomi_(direc)
    int *direc;
{
    register struct passwd *pp;
    extern gpcom_;
    char name[40];
    int fd;

    common = &gpcom_;
    pp = getpwuid(getuid());
    strcpy(name,pp->pw_dir);
    strcat(name,"/geoplot.w");
    if(*direc == 1) {
        if((fd=open(name,0)) != -1){
            read(fd,common->devmax,COMLEN);
            close(fd);
            common->ktrflg=1;
            common->kscale=1;
        }
        else {
            printf("(new) ");
            common->ifirst=1;
            common->idbug=0;
        }
        printf("Geoplot-- %s\n",version);
    }
    else {
        if((fd=creat(name,0664)) == -1) {
            fprintf(stderr,"gpcomi: Unable to open %s\n",name);
            return;
        }
        write(fd,common->devmax,COMLEN);
        close(fd);
    }
}
```

```
c-*--GPDASH--SOFTWARE DASHING OF LINES.---- ----------------------------
c
      subroutine gpdash(mode,point)
c
c-*--Programmer: PLWard, US Geological Survey, Menlo Park, California
c       March, 1979
c
c-*--Draw dashed and dotted lines in device coordinates using
c       software.
c
c-*--The dash code is input as ltype and is always greater than 5. It
c       must be greater than 11 to cause any dashing. The dash code is a
c       concatenation of up to 9 integers from 1 to 8 as follows:
c            1    Draw 0.005 page units
c            2    Move 0.005 page units
c            3    Draw 0.010 page units
c            4    Move 0.010 page units
c            5    Draw 0.020 page units
c            6    Move 0.020 page units
c            7    Draw 0.040 page units
c            8    Move 0.040 page units
c       Thus 1476 means draw 0.005, then move 0.01, then draw 0.04,
c       then move 0.02, then begin again.
c
      save
c
      include 'gpcom.h'
c
      integer*4 lltype
      integer mode,num,i,ichar,imode
      real dshtab(10), dshlen(9),plast(2),point(2),pointn(2),
     1  seglen,dx,dy,dist
      data dshlen/ 0.005,-0.005,0.01,-0.01,0.02,-0.02,0.04,-0.04,0.0/
      data plast(1),plast(2)/-2.0,-2.0/
      data dshtab(2)/0./
      data lltype/1/
c
c
      if (idbug.ge.7)call printn("gpdash: %d %f %f ltype=%ld\n",
     1  mode,point(1),point(2),ltype)
c

      if(ltype.gt.i11) goto 10
      call gplnpt(mode,point)
      return
 10   if(lltype.eq.ltype) goto 170
c
c-*--DECODE THE DASH CODE INTO ARRAY DSHTAB (in device units) FROM THE
c       TOP DOWN.
      lltype=ltype
      num=0
100   i=mod(lltype,10)
      lltype=lltype/10
      if(i.eq.i0) goto 110
      num=num+1
      if(num.lt.0.or.num.gt.10)num=10
      dshtab(10-num)=dshlen(i)
110   if(lltype.ne.i0) goto 100
      lltype=ltype
c
c-*--SLIDE VALUES IN DSHTAB DOWN TO BOTTOM OF ARRAY.--------------------
      do 160 i=1,num
160       dshtab(i)=dshtab(9-num+i)
c
```

```
c-*--MODIFY TABLE SLIGHTLY TO ADJUST FOR FLARE OF BEAM.-----------------
      ichar=num-1
      do 165 i=1,ichar
           if(dshtab(i)*dshtab(i+1).gt.0.0.or.dshtab(i)*dshtab(1).lt.0.0)
     &        goto 165
           dshtab(i)=dshtab(i)-rasmin
           dshtab(i+1)=dshtab(i+1)-rasmin
165        continue
      j=0
      seglen=0
c
c-*--DRAW THE DASHED LINE.  BEGIN HERE.-----------------------------------
170   imode=2
c-----Save the coordinates to the next point to which dashed line will
c      be drawn.
      pointn(1)=point(1)
      pointn(2)=point(2)
      if(num.le.i1) goto 330
c
c-*--START A NEW DASH SEQUENCE IF NOT INTERRUPTED BY A MOVE.------------
      if(plast(1)-pointp(1).ne.0.0) goto 200
      if(plast(2)-pointp(2).eq.0.0) goto 210
200   seglen=0
      i=0
c
c-*--COMPUTE THE DISTANCE TO THE NEXT POINT.----------------------------
210   dx=pointn(1)-pointp(1)
      dy=pointn(2)-pointp(2)
      dist=sqrt(dx*dx+dy*dy)
      dx=dx/dist
      dy=dy/dist
      if (seglen.gt.1.5) goto 230
c
c-*--DRAW A DASHED PATTERN TO THE NEXT POINT.---------------------------
220   i=mod(i,num)+1
      seglen=abs(dshtab(i))
230   if (seglen.ge.dist) goto 270
      pointn(1)=pointp(1)+dx*seglen
      pointn(2)=pointp(2)+dy*seglen
      imode=1
      if (dshtab(i).gt.0.0) imode=2
      call gplnpt(imode,pointn)
      dist=dist-seglen
      seglen=0
      go to 220
c
c-*--WHEN THE DISTANCE TO THE NEXT POINT IS <= THE LENGTH OF A DASH
c      OR SPACE, PLOT TO THE POINT AND CALCULATE SEGMENT LENGTH LEFT.
270   imode=1
      if(dshtab(i).gt.0.0)imode=2
      call gplnpt(imode,point)
      plast(1)=pointn(1)
      plast(2)=pointn(2)
      seglen=seglen-dist
      return
c
c-*--IF LTYPE IS LESS THAN OR EQUAL TO 11,  THEN SIMPLY DRAW A LINE.-----
330   call gplnpt(imode,point)
      return
      end
```

```
c--*--GPDEVO---OUTPUT PLOT VECTORS TO ANY DEVICE INCLUDING DISK.---------
c
      subroutine gpdevo(mode,point)
c
c-----Programmer PLWard, USGS, Menlo Park, California, February, 1979.---
c
c--*--This subroutine inputs data from the plot package or a disk file,
c       scales, rotates, and offsets the plot vectors, clips the plot
c       to fit the device, and outputs the plot vectors and other plot
c       commands to the device.
c
c      point is the x,y coordinates in page units.
c
c      mode or jmode can be one of the following:
c        1 move to the point
c        2 draw a vector to the point
c        3 draw a point at the point
c        4 never called at this level
c        5 never called at this level
c        6 ask device for the present pen position
c        7 ask device for the present crosshair position
c        8 initialize this plot device
c        9 set the device for hardware lettering
c       10 set the character size
c       11 set the line type
c       12 set the line shade (intensity or color)
c       13 new page, erase, or start a new frame
c       14 make a hardcopy of this display
c       15 dump the buffer, end this plot
c       16 input plot data from a file.
c
c     Devices may be as follows:
c
c          DEVICE NAME          NUMDEV  MODEL   DESCRIPTION
c       Tektronix                1       1     4014 with enhanced graphics
c                                        2      4014
c                                        3      4012
c                                        4      4010
c                                        5      4025 (4012 emulation)
c                                        6      4025
c       Calcomp                   2       1     1051
c                                        2      1012
c       Houston Instruments       3       1     Complot
c       Anderson-Jacobson         4       1     832
c       Hewlett Packard           5       1     7221A
c       Output a *.PL file       20       1     File in the working directory
c       Versatek                 21       1     1200A
c       Printronix               22       1     600
c       Florida Data             23       1     BNY-78
c
c     ldirec may be
c        1 x-axis is horizontal or parallel to hardware raster line.
c        2 x-aixs is vertical or perpendicular to hardware raster line.
c        3 same as 2 but on raster devices use quickplot rasterization.
c
c
c
      save
c
      real point(2),points(2)
      integer inout,modes,kdisk,mode,i,ii,j,kout
c
      include 'gpcom.h'
c
```

```
      data inout/1/,modes/1/
      data kdisk/0/
c
c
      if(idbug.ge.9)call printn("gpdevo: %d %f %f %d %d %d\n",mode,
     1 point(1),point(2),numdev(1),numdev(2),numdev(3))
c
c--*--Branch according to the proper mode.--------------------------------
c              1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
  5   goto(200,200,200,  9,  9,400,400,400,400,400,400,400,400,400,400,
     &  100) mode
c         16  17  18
c
c--*--Improper call.
  9   return
c
c--*--Input plot data from a disk file.-----------------------------------
100   kdisk=1
      mode=0
      call gpfilc(mode)
c-----If gpfilc was unable to open plotfile -1 is returned ------------
      if(mode.ge.i0)goto 110
           kerror=2
           call gperr(kerror,"gpdevo","Unable to open file "// filnam,
     1       mode)
           return
110   continue
      call gpfili(modes,point)
      mode=modes
      if(mode.ge.1.and.mode.le.16) goto 5
      if(mode.eq.99) goto 115
      kerror=1
      call gperr(kerror,"gpdevo","Unknown mode read from disk file of",
     1 mode)
c-----End of file.
115   mode=15
      kdisk=0
      goto 5
c
c--*--Scale, rotate, and offset plot vectors if kscale=>1.----------------
200   if(kscale.ne.i0)call gpscal(i1,point)
c
c--*--Clip plot to fit this device.---------------------------------------
300   do 320 i=1,2
           if(point(i).ge.0.0)goto 310
           if(point(i).lt.-rasmin) goto 305
           point(i)=0.0
           goto 310
305        ii=0
           goto 330
310        if(point(i).le.devmax(i)) goto 320
           if(point(i).gt.(devmax(i)+rasmin)) goto 315
           point(i)=devmax(i)
           goto 320
315        ii=i
           goto 330
320        continue
      if(inout.ne.i2) goto 360
      inout=1
      if(mode.eq.i1) goto 360
c-----Plot a point on the boundary on the way in.-------------------------
      inout=3
      modes=mode
      mode=1
```

```
      i=kout
      goto 350
c----Simply remember another point outside of boundary if already out.--
330   kout=i
      if(inout.ne.i2) goto 340
      pointp(1)=point(1)
      pointp(2)=point(2)
      if(idbug.ge.9)call printn("gpdevo: statement 330+3\n")
      return
c----Plot a point on the boundary on the way out.-----------------------
340   inout=2
350   points(1)=point(1)
      points(2)=point(2)
      j=3-i
      if(i.ne.1.and.i.ne.2)call printn("gpdevo: i out of range %d\n",i)
      point(i)=0.0
      if(ii.ne.0)point(i)=devmax(i)
      point(j)=((points(j)-pointp(j))*(point(i)-pointp(i)))/
     &  (points(i)-pointp(i))+pointp(j)
      pointp(1)=points(1)
      pointp(2)=points(2)
      goto 300
c----Plot a point within the boundaries.-------------------------------
360   pointp(1)=point(1)
      pointp(2)=point(2)
      if(ksubj.ne.0)call gppick(i1,point)
c
c--*--numdev(1) 1    2     3     4     5     6     7     8     9    10
400   goto (1010,1020,1030,1040,1010,1010,1010,1010,1010,1010,
c------- 11    12    13    14    15    16    17    18    19    20
     1 1010,1010,1010,1010,1010,1010,1010,1010,1010,1200,
     2 1210,1210,1230)numdev(1)
c------ 21    22    23
c
c--*--Tektronix 4014,4012,4010.
1010  call gptek(mode,point)
      goto 1900
c----Calcomp
1020  call gpcalc(mode,point)
      goto 1900
c----Houston Instruments Complot
1030  continue
      goto 1900
c----Anderson Jacobson
1040  call gpanja(mode,point)
      goto 1900
c----File output
1200  call gpfile(mode,point)
      goto 1900
c----Versatek.
c----Printronix.
1210  call gpras(mode,point)
      goto 1900
c----Florida data printer
1230  call gpras(mode,point)
      goto 1900
c
c--*--If reading from the disk, goto 110.-------------------------------
1900  if(ibreak.eq.i0.and.kerror.eq.i0) goto 1901
      if(idbug.ge.9)call printn("gpdevo: ibreak=%d kerror=%d\n",
     1 ibreak,kerror)
      return
1901  if(kdisk.eq.i1) goto 110
c
```

```
c---If clipping is in effect and last point plotted was on the bound-
c       ary on the way in, then plot point within the boundary.
       if(inout.eq.i3)goto 1910
       if(idbug.ge.9)call printn("gpdevo: inout=%d\n",inout)
       return
1910   inout=1
       point(1)=points(1)
       point(2)=points(2)
       mode=modes
       goto 360
       end
```

```
/* GPDEVS_ -- Decode name and model of plotting device called in ploton.
      Programmer - PLWard
*/
#define NUMDEVICES 9
#define NUMMODELS  7
#define DEFAULTDEV 1
#define DEFAULTMOD 1

struct devices{
    char *name;
    int num;
    }dev[]={
    "????",1,
    "tekt",1,
    "calc",2,
    "hous",3,
    "ande",4,
    "hewl",5,
    "vers",21,
    "prin",22,
    "flor",23
    };

struct models{
    char *number;
    int code;
    }mod[]={
    "????",1,
    "4014",2,
    "4012",3,
    "4010",4,
    "4011",5,        /* Retrographics emulation of 4010 */
    "4025",6,
    "1051",1,
    "1012",2
    };

#include "gpcomc.h"
int INIT=8;
int STRGLEN=FILNAMLEN;
int MIONE = -1;
int active=0;
float FAKE=0.0;
int DUBUF=15;
int pdbug;
int termread,termwrite;

gpdevs_(expl,dollar,tbase,itzone,gusym)
    long *expl, *dollar, *itzone;
    float *gusym;
    float *tbase;
{
    extern gpcom_;
    char strg[FILNAMLEN];
    int lengot;
    char name[4];
    float x,y;
    int i,j;
    int standardio;

    standardio=0;
    if(active)movdrw_(&FAKE,&FAKE,&DUBUF);
    active=1;
    common = &gpcom_;
```

```
common->iprnbf=0;
common->kbmtyp=1;
for(i=0;i<STRGLEN;i++) strg[i]='\0';
gpstrg(&STRGLEN,strg,&lengot);
if(common->idbug > 0)printf("gpdevs: string is %s with length %d.\n",strg,lengot);

/* If plotting to or from a file, get file name and open it. */
if( *dollar == 2 || *dollar == 3){
    gpfilc_(&MIONE);
    if(strg[0] == '\0'){ strcpy(strg,"plot.PL\0"); lengot=7;}
    strcpy(common->filnam,strg);
    /* be sure filename ends in .PL  */
    if(strcmp(&common->filnam[lengot-3],".PL\0") != 0) {
        if(lengot > FILNAMLEN-4) lengot=FILNAMLEN-4;
        strcpy(&common->filnam[lengot],".PL\0");
    }
    if(common->numdev[0]==0 || common->rasmin <= 0.0) {
        common->numdev[0]=dev[DEFAULTDEV].num;
        common->numdev[1]=mod[DEFAULTMOD].code;
        common->numdev[2]=0;
        x=common->numdev[0];
        movdrw_(&x,&x,&INIT);
    }
    if(*dollar == 2){x=common->numdev[0]=20; movdrw_(&x,&x,&INIT);}
    else { lengot=16; x=1; movdrw_(&x,&x,&lengot);}
    return;
}

/* Parse device name. */
for(i=0; i<4; i++)
    if(strg[i] >= 'a' && strg[i] <= 'z') name[i]=strg[i];
    else name[i]='?';
for(i=0; (i<NUMDEVICES)&&(pstreq(name,dev[i].name)<4);i++);
if(i==NUMDEVICES) i=DEFAULTDEV;
common->numdev[0]=dev[i].num;

/* Parse device model. */
for(i=lengot-1;i>=0 && (strg[i]<'0' || strg[i]>'9');i--);
for(j=3; j>=0; j--,i--)
    if(strg[i] >='0' && strg[i]<='9')name[j]=strg[i];
    else name[j]='?';
for(i=0;(i<NUMMODELS) && (pstreq(name,mod[i].number)<4);i++);
if(i == NUMMODELS) i=DEFAULTMOD;
common->numdev[1]=mod[i].code;

/* Parse device type or direction. */
common->numdev[2]=0;
for(i=0; i<lengot; i++)
    switch (strg[i]) {
        case 'E' : if(common->numdev[1]==6)common->numdev[1]=5; break;
        case 'L' : common->numdev[2]=1; break;
        case 'Q' : common->numdev[2]=2; break;
        case 'T' : common->iprnbf=1; break;
        case 'R' : common->ifirst=1; printf("Reset common\n"); break;
        case 'B' : common->kbmtyp=2; break;
        case 'D' : common->idbug=10; pdbug=1; break;
        case 'U' : common->numdev[1]=2; break;
        case 'S' : standardio=1; break;
    };

 if(common->idbug > 0) {
    printf("gpdevs: numdev is %d %d %d\n",
        common->numdev[0],common->numdev[1],common->numdev[2]);
    printf("gpdevs: gusym=%f itzone=%ld tbase=%f\n",
```

```
            *gusym, *itzone, *tbase);
    }
    common->usym = *gusym;
    common->izone = *itzone;
    common->timbas = *tbase;
    termread =0;
    termwrite=2;
    if(common->numdev[0]==1 && standardio == 0) {
        termread = open("/dev/tty",0);
        termwrite= open("/dev/tty",1);
        if(termread == -1) termread=0;
        if(termwrite== -1) termwrite=2;
    }
    gpfilc_(&MIONE);
    x=common->numdev[0];
    movdrw_(&x,&x,&INIT);
}

int pstreq(test,base)
    char *test, *base;
{
    int i;
    for(i=0;i<4 && (test[i]==base[i] || test[i]=='?');i++);
    return(i);
}
```

```
/* GPERR_ - report errors for geoplot. Programmer PLWard.
*/
#include <stdio.h>

gperr_(level,program,message,value)
    int *level,*value;
    char program[],message[];
{
    float x,y; int z=15;

    x=y=0.0;
    /* dump plot buffer and reset to character mode */
    movdrw_(&x,&y,&z);
    switch(*level){
        case 0:  fprintf(stderr," WARNING in geoplot on %s: %s %d.\n",
                  program,message,*value); break;
        case 1:  fprintf(stderr," ERROR in geoplot on %s: %s %d.\n",
                 program,message,*value); break;
        case -1: fprintf(stderr,"\nBREAK hit in geoplot.\n"); break;
        default: fprintf(stderr," FATAL ERROR in geoplot on %s: %s %d.\n",
        program,message,*value);
    }
    return;
}
```

```
c-*--GPFILE--OUTPUT PLOT VECTORS TO A FILE READABLE BY GPDEVO----------
c
      subroutine gpfile(jmode,point)
c
c     Programmer: PL Ward, USGS, Menlo Park, California 3/9/80
c
c-*--This subroutine outputs plot commands to a plotfile that can be
c       read by gpdevo and directed to any plot device.
c
c      point is the x,y coordinates in page units.
c
c      mode or jmode can be one of the following:
c        1 move to the point
c        2 draw a vector to the point
c        3 draw a point at the point
c        4 never called at this level
c        5 never called at this level
c        6 ask device for the present pen position
c        7 ask device for the present crosshair position
c        8 initialize this plot device
c        9 set the device for hardware lettering
c       10 set the character size
c       11 set the line type
c       12 set the line shade (intensity or color)
c       13 new page, erase, or start a new frame
c       14 make a hardcopy of this display
c       15 dump the buffer, end this plot
c       16 never called at this level
c
      real point(2)
      integer jmode
      include 'gpcom.h'

      if(idbug.ge.9)
     1       call printn("gpfile: %d,%f,%f\n",jmode,point(1),point(2))
c       jmode=1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 16
      goto (10,10,10, 5, 5,60, 5,80, 5,100, 10, 10, 10, 10, 150, 5)jmode
c
c-*--Illegal call.-----------------------------------------------------
  5   kerror=1
      call gperr(kerror,"gpfile","Illegal mode of",jmode)
      return
c
c-*--Output to file.---------------------------------------------------
10    continue
      call gpfili(jmode,point)
      return
c
c-*--Get the present pen position.-------------------------------------
 60   point(1)=pointp(1)
      point(2)=pointp(2)
      return
c
c-*--Initialize this device.-------------------------------------------
 80   kchar=0
      devmax(1)=1000.
      devmax(2)=1000.
      call gpfilc(2)
      return
c
c-*--Set the character size.-------------------------------------------
100   height=point(1)
      goto 10
c
```

gpfile.f

```
c--*--End the plot-------------------------------------------------------
150     call gpfilc(-1)
        end
```

```
/* GPFILI_ - create, open, read and write intermediate plot files.
    Programmer PLWard
*/
#include "stdio.h"
#include "gpcomc.h"
int fd=0,nbyte;
int pmode=0666,two=2,ten=10;
static int rwmode;

gpfilc_(fmode)
int *fmode;

/* This routine connects a file for gpfili according to fmode:
       -1:  close the file
   0:  open for read only
   1:  create for read and write-do not overwrite an existing file.
   2:  create for read and write-ok to overwrite.
*/
{
   int i,j;
   extern gpcom_;

   common= &gpcom_;
   if(common->idbug!=0) printf("gpfilc: %d %s \n",*fmode,common->filnam);
   rwmode= *fmode;
   if(fd != 0)close(fd);
   fd=0;
   switch(rwmode){
      case -1: return;
      case  0: fd=open(common->filnam,0);
               if(fd<0) {
                  printf("gpfili: cant open %s\n",common->filnam);
                  *fmode= -1;return;
               }
               lseek(fd,0L,0);
               return;
      case  1: fd=open(common->filnam,0);
               if(fd>0) {
                  printf("gpfili: file %s exists already.\n",common->filnam);
                  close(fd);
                  *fmode= -1;
                  return;
               }
      case  2: fd=creat(common->filnam,pmode);
               if(fd<0) {
                  printf("cant create %s\n",common->filnam);
                  *fmode= -1;
               }
               return;
   }
}


gpfili_(m,p)
int *m;
float p[2];
{
   struct {
      int mode;
      float point[2];
   } buf;

   switch(rwmode){
      case 0: nbyte=read(fd,&buf,ten);
```

```
                if(nbyte==0){*m=99; return;}
                *m= buf.mode;
                p[0]= buf.point[0];
                p[1]= buf.point[1];
                break;

        case 2:
        case 1: buf.mode= *m;
                buf.point[0]= p[0];
                buf.point[1]=p[1];
                write(fd,&buf,ten);
    }
    if (common->idbug>9) printf("fili: %d %f %f \n", *m,p[0],p[1]);
}
```

```
/*----GPFORTRAN---FORTRAN OR C CALLABLE INTERFACE TO THE GEOPLOT PROCESS

  Programmers: J W Herriot and P L Ward, USGS, Menlo Park, California
               Fall, 1979

  This group of programs interfaces the program calls in the users
  program for plotting to the pipes of a geoplot child process forked
  when ploton is called.
*/

#ifdef LENGTH
   typedef long FORTINT;
#else
   typedef int FORTINT;
#endif

#include <signal.h>
#include <stdio.h>
int child;
int gpdev = -1;
int async = 0;
int errflag = -2;
float ZERO = 0.0;
FORTINT  PAGE = 13;
FORTINT  REPRO= 14;
FORTINT  DBUF = 15;
FORTINT  ONE  = 1;
int readint;
long long0,long1,long2,long3;

/*-----ploton-----fork geoplt (if not already) & init new device-----*/
ploton_(mode,string,timebase,timezone,usym,length)
   FORTINT *mode,*timezone;
   long length;
   float *timebase,*usym;
   char string[];
{
   int pid;
   extern geterr(),gpfin_();

   signal(SIGINT,gpfin_);
   async= *mode >= 100;
   if(gpdev == -1) {
      child = makechild("/usr/geolab/geoplt");
      if (child < 0) {
         fprintf(stderr,"ploton: Unable to fork geoplt. Makechild returns %d. \n",child);
         exit(1);
      }
   }
   gpdev = *mode;
   long0 = *mode;
   long1 = *timezone;
   gpsend(15,&long0,&long0,timebase,&long1,usym);
   gppwrite(&length,4);
   gppwrite(string,(int)length);
   geterr();
}
/*-----putcom-----put n words into geoplt common block---------------*/
putcom_(index,n,vec)
   FORTINT *index,*n;
   float vec[];
{
   extern geterr();
   int i;
```

```
    long0 = *index;
    long1 = *n;
    gpsend(22,&long0,&long1);
    gppwrite(vec,(int)(*n*4));
    geterr();
}
/*-----getcom-----get n words from geoplt common block---------------*/
getcom_(index,n,vec)
    FORTINT *index, *n;
    float vec[];
{
    extern geterr();
    long0 = *index;
    long1 = *n;
    gpsend(32,&long0,&long1);
    gppread(vec,(int)(*n*4));
    geterr();
}
/*-----pltcom-----function to get 1 work from plot common-----------*/
float pltcom_(n)  FORTINT *n;
{
    float num;
    getcom_(n,&ONE,&num);
    return(num);
}
/*-----movdrw-----basic geoplot subroutine -- mode=5,6,7 returns x,y-*/
movdrw_(x,y,mode)
    float *x, *y;
    FORTINT *mode;
{
    extern geterr();
    long0 = *mode;
    gpsend(43,x,y,&long0);
    if(long0 < 0) long0 = -long0;
    if(long0 >= 5 && long0 <=7) {
        gppread(x,4);
        gppread(y,4);
        gppread(&long0,4);
        *mode = long0;
    }
    geterr();
}
/*-----letter-----letter version of movdrw---------------------------*/
letter_(x,y,mode,strg,ang,posn,length)
    float *x, *y, *ang;
    FORTINT *mode, *posn;
    long length;
    char strg[];
{
    extern geterr();
    long nchar;
    int lenstr_();
    nchar=lenstr_(strg,length);
    long0 = *mode;
    long1 = *posn;
    gpsend(55,x,y,&long0,ang,&long1);
    gppwrite(&nchar,4);
    gppwrite(strg,(int)nchar);
    geterr();
}
```

```
/*-----map--------map projection-----------------------------------*/
map_(string,length) char *string; long length;
{
    extern geterr();
    gpsend(60,&long0);
    gppwrite(&length,4); gppwrite(string,(int)length);
    geterr();
}
/*-----frame------draw a frame-------------------------------------*/
frame_()
{
    extern geterr();
    gpsend(70);
    geterr();
}
/*-----shade------shade in an enclosed area defined by xarr,yarr-----*/
shade_(xarr,yarr,len)
    float xarr[],yarr[];
    FORTINT *len;

{
    int i;
    extern geterr();

    long0 = *len;
    gpsend(81,&long0);
    for(i=0; i < *len; i++)gpmpx(2,xarr+i,yarr+i);
    geterr();
}
/*-----plt-------general fortran-style plot call-------------------*/
plt_(xarr,xlen,yarr,ylen,type)
    float *xarr, *yarr;
    FORTINT *xlen, *ylen, *type;
{
    extern geterr();
    int i;
    long len;
    float xx;

    len= *xlen;
    if (*xlen <= 1) len= *ylen;
    else if(*ylen > 1 && *ylen < len) len= *ylen;
    if (len < 1) len = 1;
    long0 = *type;
    gpsend(92,&len,&long0);
    if(*xlen <= 1) for(i=0; i < len; i++) {
        xx = xarr[0] + i;
          gpmpx(2,&xx,yarr+i);
    }
    else if (*ylen <= 1) for(i=0; i< len; i++) {
        xx = yarr[0] + i;
        gpmpx(2,xarr+i,&xx);
    }
    else for(i=0; i< len; i++)gpmpx(2,xarr+i,yarr+i);
    geterr();
}
/*-----pltltr-----plot with lettering-------------------------------*/
pltltr_(xarr,xlen,yarr,ylen,type,carr,cwid,clen,ang,nang,pos,npos)
    char carr[];
    float xarr[],yarr[],ang[];
    FORTINT *xlen, *ylen, *type, *cwid, *clen, *nang, *npos,pos[];
{
    extern geterr();
    float xx,posn;
```

```
    int i,slen,akick,pkick,ckick,lenstr_();
    long len,chwid;

    chwid= *cwid;
    len= *xlen;
    if (*xlen <= 1)len= *ylen;
    else if(*ylen > 1 && *ylen < len) len= *ylen;
    if(len < 1) len = 1;
    akick = *nang >= len;
    pkick = *npos >= len;
    ckick = *clen >= len;
    if (*nang == 0) *nang=1;
    if (*npos == 0) *npos=1;
    if(*nang!=1 && *nang!= len) {
        fprintf(stderr,"pltltr: %ld is illegal value for nang\n", *nang);
        exit();
    }
    if(*npos!=1 && *npos!= len) {
        fprintf(stderr,"pltltr: %ld is illegal value for npos\n", *npos);
        exit();
    }
    long0 = *type;
    gpsend(102,&len,&long0);
    if(*xlen<=1)
        for(i=0;i<len;i++) {
            slen=lenstr_(carr+i*chwid,chwid);
            xx=xarr[0]+i;
            posn = *(pos+i*pkick);
            gpmpx(5,&xx,yarr+i,ang+i*akick,&posn,carr+i*ckick*chwid,slen);
        }
    else if(*ylen<=1)
        for(i=0;i<len;i++) {
            slen=lenstr_(carr+i*ckick*chwid,chwid);
            xx=yarr[0]+i;
            posn = *(pos+i*pkick);
            gpmpx(5,xarr+i,&xx,ang+i*akick,&posn,carr+i*ckick*chwid,slen);
        }
    else
        for(i=0;i<len;i++) {
            slen=lenstr_(carr+i*ckick*chwid,chwid);
            posn = *(pos+i*pkick);
            gpmpx(5,xarr+i,yarr+i, ang+i*akick,&posn,carr+i*ckick*chwid,slen);
        }
    geterr();
}
/*-----axis------draw an axis-----------------------------------------*/
axis_(type) FORTINT *type;
    {
    extern geterr();

    long0 = *type;
    gpsend(131,&long0);
    geterr();
}
/*-----grid------draw a gridded line-------------------------------*/
grid_(x,y,mode,string,datlo,dathi,trans,length)
    float *x, *y, *datlo, *dathi;
    FORTINT *mode, *trans,length;
    char string[];
{
    extern geterr();

    long0 = *mode;
    long1 = *trans;
```

```
    gpsend(146,x,y,&long0,datlo,dathi,&long1);
    gppwrite(&length,4);
    gppwrite(string,(int)length);
    geterr();
}

/*-----propor-----proportion subject space----------------------------*/
propor_(center,total,lodat,hidat,axis)
    float *center, *total, *lodat, *hidat;
    FORTINT *axis;
{
    extern geterr();

    long0 = *axis;
    gpsend(155,center,total,lodat,hidat,&long0);
    geterr();
}

/*-----disazm----find distance and azimuth----------------------------*/
disazm_(lo1,la1,lo2,la2,type)
    float *lo1, *la1, *lo2, *la2;
    FORTINT *type;
{
    extern geterr();

    long0 = *type;
    gpsend(175,lo1,la1,lo2,la2,&long0);
    gppread(lo1,4);
    gppread(la1,4);
    gppread(lo2,4);
    geterr();
}

/*-----label-----label a plot axis-----------------------------------*/
label_(axis,string,length)
    FORTINT *axis;
    long length;
    char string[];
{
    extern geterr();
    long nchar;

    long0 = *axis;
    gpsend(161,&long0);
    nchar = lenstr_(string,length);
    gppwrite(&nchar,4);
    gppwrite(string,(int)nchar);
    geterr();
}


/*-----glplot-----geolab style plotting with command-string----------*/
glplot_(cmdstr,typstr,len,x1,x2,x3,x4,x5)
    char cmdstr[],typstr[];
    FORTINT *len,x1[],x2[],x3[],x4[],x5[];
{
    fprintf(stderr,"glplot: not implemented yet\n");
}

/*-----gperr------get the error flag from geoplot-------------------*/
FORTINT gperr_() {
    FORTINT err;
    err=errflag;
    return(&err);
```

```
}

/*-----gpfin------terminate the child geoplot process----------------*/
gpfin_() {
    long0=0;
    gppwrite(&long0,4);
    gppread(&errflag,2);
    killchild(child);
    exit(1);
}

/*-----gpsend-----send args down pipe to geoplot process-------------*/
gpsend(cmd,params) int cmd;
    long *params;

{
    int i,nsend;
    long **stk,vec[10];

    nsend=cmd%10;
    vec[0]=cmd;
    stk= &params;
    for(i=0;i<nsend;i++)vec[i+1]= *stk[i];
    gppwrite(vec,nsend*4+4);
}

/*-----gpmpx------pipe one element of several arrays to geoplot------*/
gpmpx(nchan,x1,x2,x3,x4,string,nchar)
    int nchan;
    long *x1, *x2, *x3, *x4;
    char string[];
    int nchar;
{
    int i,nloop;
    long vec[5],**stk;

    nloop=nchan;
    if(nchan>4)nloop=4;
    stk= &x1;
    for(i=0;i<nloop;i++)vec[i]= *stk[i];
    if(nchan==5)vec[4]=nchar;
    gppwrite(vec,nchan*4);
    if(nchan==5)gppwrite(string,nchar);
}

/*-----geterr--------get error code sent back up the pipe for synchronous
                    processes
*/
geterr()
{
    if (async == 0) {
        gppread(&errflag,2);
        if (errflag >= 2) exit(1);
        if (errflag <  0) gpfin_;
    }
}
/*-----page------erase screen on plot device----------------------*/
page_()
{
    movdrw_(&ZERO,&ZERO,&PAGE);
}
```

```
/*-----clrbuf----clear the plot buffer-----------------------------*/
clrbuf_()
{
    movdrw_(&ZERO,&ZERO,&DBUF);
}
/*-----part-----divide pict space into parts---------------------*/
part_(axis,part,total)
FORTINT *axis, *part, *total;
{
FORTINT ax;
float to,pa;
    if( *axis<1||*axis>2) {
        printf("ERROR: axis %d may not be partioned(only 1 and 2)\n"
        ,*axis);
        return(-1);
    }
          to= *total;pa= *part;
    ax= *axis+16;
      movdrw_(&pa,&to,&ax);
}

/*-----repro-----make a hard copy----------------------------------*/
repro_()
{
    movdrw_(&ZERO,&ZERO,&REPRO);
}

/*-----gppread----read a vector from the pipe----------------------*/
int gppread(vec,length)
    char *vec;
    int length;
{
    int i;

    i = piread(child,vec,length,&readint);
    if (i != length) {
        fprintf(stderr,"gppread: Unable to read on pipes. piread returns %d.\n",i);
    }
    return(i);
}

/*-----gppwrite---write a vector to the pipe-----------------------*/
int gppwrite(vec,length)
    char *vec;
    int length;
{
    int i;

    i = piwrite(child,vec,length);
    if (i != length) {
        fprintf(stderr,"gppwrite: Unable to write on pipes. piwrite returns %d.\n",i);
    }
    return(i);
}
```

```
c---GPFUNC-----FUNCTIONS USED BY GRID--------------------------------
c
      function gpfunc(ltyp,in)
c
      integer*2 ltyp,MAXTYP
      real gpfunc,in
c
      parameter (MAXTYP=3)
c
      if(ltyp gt 0)goto 1000
      gpfunc=MAXTYP
      return
c
1000  goto (1,2,3) ltyp
  1   gpfunc=in
      return
  2   gpfunc=alog10(in)
      return
  3   gpfunc=alog(in)
      return
      end
```

```
c-----gpgetc-----------------------------------------------------------
c
      subroutine gpgetc(index,n)
c
c---Get values from geoplot common and send back to user process.
c
      include 'gpcom.h'
      include 'GPCOM.h'
c
      real rrr
      integer index,n
      integer i,j,k
c
      do 1000 i=1,n
           j=index+i-1
           if (j.ge.1.and.j.le.(MAXCOM+2)) goto 5
                call gperr(kerror,"getcom",
     1           "Common does not exist for element",j)
                goto 1000
  5        if(j.le.MAXCOM) goto 950
           goto (10,20) j-MAXCOM
 10        call printn("\nPlotting file is %s\n",filnam)
           goto 970
 20        call printn("                    xa            xz            ya
     1   yz\n")
           call printn("pictsp %4(%12.5f %)\n",pspace(1),pspace(2),
     1       pspace(3),pspace(4))
           call printn("gridsp %4(%12.5f %)\n",pspace(7),pspace(8),
     1       pspace(9),pspace(10))
           call printn("subjsp %4(%12.5f %)\n",pspace(13),pspace(14),
     1       pspace(15),pspace(16))
           call printn("datasp %4(%12.5f %)\n",pspace(19),pspace(20),
     1       pspace(21),pspace(22))
           call printn("lablsp %4(%6d          %)\n",lablsp(1),lablsp(2),
     1       lablsp(3),lablsp(4))
           if(nomit.eq.0) goto 970
           do 25 k=1,nomit
           j=(k*4)-3
 25        call printn("omitsp %4(%12.5f %)\n",omitsp(j),omitsp(j+1),
     1       omitsp(j+2),omitsp(j+3))
           goto 970
950        if (j.eq.MAPSCA.and.ktrflg.eq.1) call gptran(3,pole)
           if (j.lt.INTG) rrr=rcom(j)
           if (j.eq.SYMANG) rrr=57.295779*asin(symang)
           if (j.ge.INTG.and.j.le.(INTG+2))rrr=lintyp(j-INTG+1)
           if (j.ge.(INTG+3)) rrr=icom(j-INTG-2)
           if (j.eq.MAXCOM)rrr=ltype
           if (j.eq.LETCLI)rrr=(kltrsp+6)/6
           if (j.eq.LINCLI)rrr=(kpltsp+6)/6
970        call gpppip(rrr,i1)
           if(idbug.ge.2)call printn("gpgetc: %d %d %f\n",index,n,rrr)
1000       continue
      end
```

```
c-*--GPHLET---PRINT A CHARACTER STRING ON ANY DEVICE SUPPORTING IT.-----
c
      subroutine gphlet(length,itext)
c
c----Programmer: PLWard, USGS, Menlo Park, California, March, 1979.-----
c
      include 'gpbuf.h'
      include 'gpcom.h'
c
      integer i,length,itext(1)
      real fake(2)
c
      if (idbug.lt.i9) goto 100
         call printn("gphlet: length=%d  '",length)
         do 50 i=1,length
 50      call printn("%c",itext(i))
         call printn("'\n")
c
100   if((length+2+lenbuf).ge.lenmax)  call gpbufo(i3)
      call gpdevo(i9,fake)
      do 150 i=1,length
          ibuffr(lenbuf)=itext(i)
150       lenbuf=lenbuf+1
      return
      end
```

```
/* gpletr   fortran callable. Programmer PLWard,USGS 10/26/79 */

/* This program decodes the letter codes in gpletr.h written
   from a file letter.codes by loadletr.c. See loadletr.c
   for a description of the coding scheme.                  */

unsigned MSK4 = 15;
unsigned MSK5 = 31;
unsigned MSK11= 2047;
#include "gpletr.h"

gpletr_(letter,nnodes,ix,iy)
    int *letter,*nnodes,ix[],iy[];
{
    int i,j,let; unsigned msk;

    let= *letter % 128;
    i=index[let] & MSK5;
    *nnodes = i < *nnodes ? i : *nnodes;
    i=(index[*letter] >> 5) & MSK11;
    for(j=0; j<*nnodes; j++,i++){
        msk=0;
        if(i==((i/2)*2)) msk=8;
        msk=nodes[i/2] >> msk;
        iy[j]=msk & MSK4;
        ix[j]=(msk >> 4) & MSK4;
    }
    return;
}
```

```
c-*--GPLNPT---GENERATES LINES AND POINTS OF ANY WIDTH.------------------
c
      subroutine gplnpt(mode,point)
c
c-----Programmer PLWard, USGS, Menlo Park, California, February, 1979.---
c
c-*--This subroutine draws fat lines and points. point(2) is the
c       coordinates of the point to be drawn to.  width is the width
c       of the line. If multiple lines are drawn they are spaced apart
c       by space(linmod).
c
      save
c
      real xs(3),ys(3),point(2),spacen,wdtln2,wdtlsq,wdtlef,wdtrig,
     1   xi,xj,base,sinx,cosx
      integer mode,k,jplt,iegm
c
c
      include 'gpcom.h'
c
      if (idbug.ge.8)call printn("gplnpt: %d %f %f linmod=%d  linflg=%d
     1   \n",mode,point(1),point(2),linmod,linflg)
c
      spacen=space(linmod)
c
C-*--Set values related to line width if it has changed.----------------
      if(linmod.eq.linflg) goto 9
      wdtln2=width(linmod)-amod(width(linmod),space(linmod))*0.5
      wdtlsq=width(linmod)*width(linmod)*0.25
      wdtlef=-wdtln2
      wdtrig=wdtln2
      if(kside(linmod).eq.i2) wdtrig=0.0
      if(kside(linmod).eq.i3) wdtlef=0.0
      linflg=linmod
c
c-*--Branch according to the mode which is lmode.----------------------
   9  goto (50,100,10,10) mode
c
c-*--Plot in point modes.-----------------------------------------------
  10  xs(2)=point(1)
      ys(2)=point(2)
      if(mode.eq.i4) goto 30
      if(width(linmod).le.rasmin) goto 50
c-----Draw a point of diameter width(linmod).---------------------------
      do 20 xj=wdtlef,wdtrig,spacen
           point(1)=xs(2)+xj
           do 20 xi=-wdtln2,wdtln2,spacen
                point(2)=ys(2)+xi
                if(xi*xi+xj*xj.le.wdtlsq) call gpdevo(mode,point)
  20            continue
      goto 45
c-----Draw a line of points of width width(linmod).---------------------
  30  call gpsinc(pointp,point,sinx,cosx,base)
      if(base.lt.0.0000001) goto 50
      xs(1)=pointp(1)
      ys(1)=pointp(2)
      do 40 xj=0.0,base,spacen
           do 40 xi=wdtlef,wdtrig,spacen
                point(1)=xs(1)+xi*sinx+xj*cosx
                point(2)=ys(1)-xi*cosx+xj*sinx
  40            call gpdevo(mode,point)
  45  point(1)=xs(2)
      point(2)=ys(2)
  50  call gpdevo(mode,point)
```

```
      return
c
c---Plot multiple parallel lines to increase line width. Always draw
c      from near old point to near new point so that the hardware dash
c      generated lines match up.
100   if(width(linmod).le.rasmin) goto 50
      xs(3)=point(1)
      ys(3)=point(2)
      call gpsinc(pointp,point,sinx,cosx,base)
      if(base.lt.0.0000001) goto 50
      xs(1)=pointp(1)-cosx*wdtln2*0.6
      ys(1)=pointp(2)-sinx*wdtln2*0.6
      xs(2)=point(1)+cosx*wdtln2*0.6
      ys(2)=point(2)+sinx*wdtln2*0.6
      do 150 xj=wdtlef,wdtrig,spacen
          do 150 k=1,2
              point(1)=xs(k)-xj*sinx
              point(2)=ys(k)+xj*cosx
              jplt=2
              if(k.eq.i2) goto 150
              if(iegm.gt.i0) iegm=0
              jplt=1
150           call gpdevo(jplt,point)
      point(1)=xs(3)
      point(2)=ys(3)
      call gpdevo(i1,point)
      return
      end
```

```
c--*--GPPICK: PICKING DATA POINTS FROM MULTIPLE SUBJECT SPACES ON SCREEN.
c
      subroutine gppick(mode,point)
c
c       mode = 1 store spaces
c              2 retieve data point
c              3 restore spaces to that saved
c
c       nfare= 1 choose subjsp based on x distance
c              2 choose subjsp based on y distance
c              3 choose subjsp based on x and y distance
c              4 same as 3 but turn on crosshairs again to get point
c
c      ktrflg= 0 transformations initialized
c              1 need to initialize the transformations
c              2 need to reset the spaces and initialize transformations
c              3 same as 2 but gptran called from gppick
c
c      nfield is reset to zero in movdrw mode 13 or by user
c
       save
       include 'gpcom.h'
c
       real dist,temp
       integer MFIELD,mode,ioline,i,nthis,ksave
       parameter (MFIELD = 32)
       real subjsv(2,MFIELD),datasv(2,MFIELD),save(2,MFIELD),point(2)
       real subsav(4),datsav(4)
       integer*2 ntrans(MFIELD),id(MFIELD)
       data ioline/-9999/
c
       if(idbug.gt.4)call printn("gppick: %d %f %f lineid=%d nfield=%d\n"
     1 ,mode,point(1),point(2),lineid,nfield)
c
       goto (100,200,600)mode
c
c--*--Store subjsp, datasp, ktrans, and lineid for this subject space.---
100    nfield=nfield+1
       if(nfield.ge.1.and.nfield.le.MFIELD) goto 150
       kerror=2
       call gperr(kerror,"gppick","Too many lines identified. Limited to"
     1 ,MFIELD)
       return
150    ntrans(nfield)=ktrans
       save(1,nfield)=point(1)
       save(2,nfield)=point(2)
       id(nfield)=lineid
       ioline=lineid
       do 175 i=1,4
            subjsv(i,nfield)=pspace(i+12)
175         datasv(i,nfield)=pspace(i+18)
       ksubj=0
       return
c
c--*--Find which subjsp to use for nfare = 1 or 2 ( x or y distance).-----
200    nthis=point(1)+0.5
       mode=7
       call gpdevo(mode,point)
       if(nthis.gt.0.and.nthis.le.MFIELD) goto 400
       dist=10000000.0
       if(nfare.ne.1.and.nfare.ne.2) goto 300
       do 250 i=1,nfield
            temp=abs(save(nfare,i)-point(nfare))
            if(temp.ge.dist) goto 250
```

```
            dist=temp
            nthis=i
250         continue
        goto 400
c
c--*--Find which subjsp to use for nfare = 3 (x and y distance).----------
300     do 350 i=1,nfield
            temp=abs((save(1,i)-point(1))**2+(save(2,i)-point(2))**2)
            if(temp.ge.dist) goto 350
            dist=temp
            nthis=i
350         continue
c
c--*--For nfare = 4 get point now.---------------------------------------
400     if(nfare.ne.4) goto 420
        mode=7
        call gpdevo(mode,point)
c
c--*--Now convert to data coordinates.-----------------------------------
420     if(kscale.ne.i0) call gpscal(i2,point)
        if(idbug.gt.4)call printn("gppick: nthis=%d ktrflg=%d\n",nthis,
       1 ktrflg)
        do 500 i=1,4
            if(ktrflg.gt.1) goto 490
            subsav(i)=pspace(i+12)
            datsav(i)=pspace(i+18)
490         pspace(i+12)=subjsv(i,nthis)
500         pspace(i+18)=datasv(i,nthis)
        if(ktrflg.lt.2)ksave=ktrans
        ktrans=ntrans(nthis)
        lineid=id(nthis)
        ktrflg=3
        call gptran(i2,point)
        return
c
c--*--Reset subject and data spaces.-------------------------------------
600     do 610 i=1,4
            pspace(i+12)=subsav(i)
610         pspace(i+18)=datsav(i)
        ktrans=ksave
        ktrflg=1
        end
```

```
c-----GPPUTC: Put values into geoplot common when allowed and set flags.-
c
      subroutine gpputc(index,n)
c
c-----Programmer: PLWard, USGS, Menlo Park, October,1979.
c
      real rrr
      integer index,n,TEN
      integer i,j,noflag,ii,k,kk
      include 'gpcom.h'
      include 'GPCOM.h'
      data TEN/10/
c
      noflag=0
      if(n.lt.1)n=1
      do 20 i=1,n
           call gpgpip(rrr,i1)
           j=index+i-1
           if(idbug.ge.2)call printn("gpputc: %d %d %f\n",j,n,rrr)
           if (j.ge.LOUSR .and. j.le.HIUSR) goto 15
           call gperr(kerror,"putcom",
     1          "Not allowed to change common element",j)
           goto 20
 15        if (j.lt.INTG) rcom(j)=rrr
           if(j.ge.INTG.and.j.lt.(INTG+3)) lintyp(j-INTG+1)=rrr+0.5
           if (j.ge.(INTG+3)) icom(j-INTG-2)=rrr+0.5
           if(j.eq.SYMANG) symang=sin(rrr*0.01745329)
           if(j.eq.LETHEI) call movdrw(rrr,rrr,TEN)
c-*--Set ktrflg if changing anything to do with initialized
c     transformations in gptran.f.
           if ((j.ge.PICTSP.and.j.le.(TRANSC+8)).or.j.eq.TRANS) ktrflg=1
           if (j.ne.MAPSCA) goto 988
              ktrflg=1
              setsca=mapsca
              mapsca=0.0
988        if ((j.ge.PICTSP.and.j.le.(TRANSC+8)).or.j.eq.TRANS)mapsca=0.0
           if(j.eq.TRANS) call nameit
           if ((j.eq.POLE) .or. (j.eq.(POLE+1))) ktrflg=1
c-*--Set kscale is changing anything to do with positioning on device
c    in gpdevo.f
           if (j.gt.POFFSE.and.j.le.(PGAIN+1)) kscale=1
c-*--Set linflg if line properties have changed (space,width,kside)
           if((j.ge.LINWID.and.j.le.(LINSHA+2)).or.
     1          (j.ge.LINSID.and.j.le.(LINSID+2)))linflg=0
           if((j.ge.LINWID.and.j.le.(LINSHA+2)).and.(rcom(j).lt.rasmin))
     1          rcom(j)=rasmin
           if(j.ne.GPRINT) goto 16
              if(idbug.gt.8) call pidbug(1)
              if(idbug.le.8) call pidbug(0)
 16        if(j.ge.LETHEI.and.j.le.LETSPV)letchg=1
           if(j.ge.OMITSP.and.j.le.(OMITSP+15))noflag=1
           if(j.eq.LETCLI)kltrsp=kltrsp*6-5
           if(j.eq.LINCLI)kpltsp=kpltsp*6-5
           if(j.lt.TICFAC.or.j.gt.(LABANG+5))goto 18
           if(j.gt.(TICYOU+4).and.j.lt.LABDIS)goto 18
           if(j.gt.LABMIN.and.j.lt.LABANG)goto 18
           do 17 ii=1,4
 17        call gplink(ii)
 18        if(j.ge.PICTSP.and.j.le.(PICTSP+3)) call gplink(j-i20)
           if(j.ge.GRIDSP.and.j.le.(GRIDSP+3))pspace(j-14)=pspace(j-20)
           if(j.ge.FULLSP.and.j.le.(FULLSP+1).and.
     1       fullsp(j-160).gt.devmax(j-160)) fullsp(j-160)=devmax(j-160)
 20        continue
       if(noflag.eq.0) return
```

```
      kk=0
      do 30 k=1,16
           if(omitsp(k).ne.0.0) kk=k
 30        continue
      nomit=(kk+3)/4
      end

c--*--GPLINK--Link gridsp and subjsp to pictsp.--------------------------
c
      subroutine gplink(i)
c
      include 'gpcom.h'
c
      save
c
      real dtic,dlab,d
      integer i,j,k
c
      if(i.lt.1.or.i.gt.4)call printn("gplink: i out of range %d\n",i)
      ktrflg=1
      mapsca=0.0
      d=1.0
      if(i.eq.2.or.i.eq.4)d=-1.0
      dtic=0.0
      j=1
      if(i.le.2)j=5
      do 100 k=j,j+4
100   dtic=min(dtic,-ticout(k)*ticfac(i),ticin(k)*ticfac(i))
      dlab=0.0
      j=i+2
      if(j.gt.4)j=j-4
      if(lablsp(j).ne.0)dlab=abs(lablsp(j)*height*aspect*(1.0+spacel)
     1  *sin(0.01745329*anglab(j)))+abs(height*3.0*cos(0.01745329
     2  *anglab(j)))
      if(dislab(i).gt.-1.0.and.dislab(i).lt.0.0) dlab=dlab-dislab(i)
200   if(idbug.ge.2) call printn("gplink: %d %d %f %f %f\n",i,j,dlab,
     1 dtic,d)
      pspace(6+i)=pspace(i)+d*(dlab-dtic)
      pspace(12+i)=pspace(6+i)
      end
```

```
c-*--GPRAS -  format the commands and output them into a
c       buffer for controlling a Versatek printer-plotter.
c       Programmer Barbara Bekins
c
      subroutine gpras(jmode,point)
c
c     point is the x,y coordinates in page units.
c
c     mode or jmode can be one of the following:
c       1 move to the point
c       2 draw a vector to the point
c       3 draw a point at the point
c       4 never called at this level
c       5 never called at this level
c       6 ask device for the present pen position
c       7 ask device for the present crosshair position
c       8 initialize this plot device
c       9 set the device for hardware lettering
c      10 set the character size
c      11 set the line type
c      12 set the line shade (intensity or color)
c      13 new page, erase, or start a new frame
c      14 make a hardcopy of this display
c      15 dump the buffer, end this plot
c      16 never called at this level
c
c-*--This routine supports devices 21 and 22 with numdev(3)=0,1 or 2
c
c        0  A single page plot where x is the long axis and the
c           origin is the lower left corner as the plot comes out.
c
c        1  A multi page plot where y is the short axis and x spans
c           many pages.  The origin is the upper left corner of the
c           first page plotted.
c
c   2  A quickplot: expects pairs of x and y values of a time series
c           x values run down the page, time series are plotted parallel
c       to each other.
c
      parameter(NDEV=2,NPOSS=3)
      include 'gpcom.h'
      include 'gpbuf.h'
      integer*2 jmode,ix,iy,lastx,lasty,iqwc,ntr,mode
      integer*2 ltypeh(4),ndev,npos,i,ierr,opnqwc
c-----below are the initial values for the various devices----------
      real devm(2,NPOSS,NDEV),dpag(2,NDEV),dpgi(NDEV),phom(NDEV)
      real point(2),savep(2),qlast
c               max x pos 1&2,      max y,raster/page,phome:x&y
      data devm/1.2424,1.00,15.5147,1.2424,15.5147,1.2424,
     1           1.2,1.00,41.3737,1.2,41.3737,1.2/
      data dpag/1536.,1536.,660.,782./,dpgi/8.5,11.0/,phom/0.0,0.9843/
      data ltypeh/12,31212,32,512/
c
      if(idbug.ge.9)call printn("gpras: %d %f %f\n",jmode,point(1),
     1  point(2))
c
      mode=jmode
c
c-*--Branch to the proper value of mode. Note that mode is not checked
c       for the proper value to save time since users are not supposed to
c       call this routine directly.
c
c       mode= 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
      goto ( 10,10,10,10,5,60, 5,80, 5,100, 110,120,130, 130,150,
```

```
     &    5,  5,  5) mode
c         16  17  18  branch to label shown.
c
c-*--Improper call: return ----------------------------------------------
5       go to 500
10     continue
       if(iqwc.eq.0) go to 11
c----- if quickplot output only modes 2 and 3 to a plotfile-------------
c      allow only 132 traces(1 trace is a sequence with monotonically
c      increasing values in x)
       if(point(1).lt.qlast) ntr=ntr+1
       if(ntr.lt.132) go to 12
       kerror=2
       call gperr(kerror,"gpras","Too many traces in quickplot, ntr=",
     &         ntr)
       return
12     continue
       qlast=point(1)
       call putqwc(jmode,point(1),point(2))
       return
11     continue
       if(mode.eq.1) go to 500
c----convert page units to raster units---------------------------------
20     if(numdev(3).eq.1) go to 21
c-*--reverse y for single page plot so that origin is in the trailing --
c     left hand corner
       ix=point(1)*dpage(1)+.5
       iy=dpage(2)-(point(2)*dpage(2))+.5
       lastx=savep(1)*dpage(1)+.5
       lasty=dpage(2)-savep(2)*dpage(2)+.5
       go to 22
c----exchange x and y for multi page plot-------------------------------
21     continue
       ix=point(2)*dpage(2)+.5
       iy=point(1)*dpage(1)+.5
       lastx=savep(2)*dpage(2)+.5
       lasty=savep(1)*dpage(1)+.5
22     continue
       if(idbug.ge.10) call printn("line:%4(%d %)\n", ix,iy,lastx,lasty)
       if(mode.ne.3) go to 25
c-*--plot a point at the point------------------------------------------
       call ppoint(ix,iy)
       go to 500
c-*--plot a vector to the point-----------------------------------------
25     continue
       call line(lastx,lasty,ix,iy)
       go to 500
c-*--return with present pen position-----------------------------------
60     point(1)=savep(1)
       point(2)=savep(2)
       go to 500
c-*--initialize the device----------------------------------------------
 80    continue
       ndev=numdev(1)-20
       npos=numdev(3)+1
       if(ndev.lt.1.or.ndev.gt.2) ndev=1
       if(numdev(2).eq.2) ndev=3
c----- ndev=3 for upstairs printronix------
       if(npos.lt.1.or.npos.gt.3) npos=1
       do 81 i=1,2
        devmax(i)=devm(i,npos,ndev)
        dpage(i)=dpag(i,ndev)
        phome(i)=phom(i)
        savep(i)=phom(i)
```

```
81      continue
        dpgin=dpgi(ndev)
        hpgin=dpgin
        rasmin=1./dpage(2)
        if(npos.eq.3) goto 85
c-----not a quickplot-------------------------------------------------------
        call begin
        iqwc=0
        go to 500
c----- if qwickplot output points to a file to be spooled later---------
85      continue
        ierr=opnqwc()
             if(ierr.ge.0) goto 86
        kerror=2
        call gperr(kerror,"gpras","Cant create quickplot file",ierr)
        return
86      continue
c----- initialize trace counting---------------------------------------
        qlast=0.
        ntr=0
        iqwc=1
        goto 500
100     continue
c----- Set the character height-----------------------------------------
        if(point(1).lt.1.and.point(1).gt.0.) height=point(1)
        goto 500
c-*--SET THE LINE TYPE.-----------------------------------------------MODE 11
c       lmode = 1 move
c               2 vector line
c               3 point
c
c       ltype = 0 point
c               1 line
c               2 line of points
c               3 short dash
c               4 dot-dash
c               5 medium dash
c               6 long dash
c               >12     software dash patterns
c
110     continue
        ltype=point(1)+.5
        if(ltype.lt.i0) ltype=1
111     if(ltype.ne.i0) goto 113
c-----Point.-----------------------------------------------------------
        lmode=3
        go to 500
c-----Solid line------------------------------------------------------
113     if(ltype.ne.i1) go to 114
        lmode=2
        call linset(ltype-1)
        go to 500
114     if(ltype.gt.i11) goto 116
c-----Versatek known pattern: set proper code in versatek software--
        if(ltype.gt.i6) ltype=i6
        lmode=2
        call linset(ltype-2)
        go to 500
c-----Software dash(concatination of at least two integer codes ---
c     yielding dashes and spaces of desired lengths).  See 'gpcom'
c     for codes.
c-----Check if known versatek type. If so, use the versatek code---
116     do 117 i=1,4
             if(ltype.ne.ltypeh(i)) goto 117
```

```
            ltype=i+i2
            goto 114
117         continue
118     lmode=i2
        go to 500
c-*--set line shade-------------------------------------------------
120     continue
        pshade=point(1)
        goto 500
c-*--new page-------------------------------------------------------
130     continue
        if(iqwc.eq.1) goto 135
c----not a quickplot------------------------------------------------
        call end(mode)
        call begin
        go to 500
135     continue
c----plot is a quickplot--------------------------------------------
        call endqwc(numdev(1))
        ierr=opnqwc()
             if(ierr.ge.0) goto 136
        kerror=2
        call gperr(kerror,"gpras","Cant create quickplot file",ierr)
        return
136     continue
        goto 500
c-*--end the plot---------------------------------------------------
150     continue
        if(iqwc.eq.0) call end(mode)
        if(iqwc.ne.1) goto 500
        if(ibreak.eq.0) goto 155
c---- error in plot, restart quickplot by truncating file-----------
        ierr=opnqwc()
        if(ierr.ge.0) goto 500
        kerror=2
        call gperr(kerror,"gpras","Cant create quickplot file",ierr)
        goto 500
155     continue
        call endqwc(numdev(1))
500     continue
        if(mode.gt.4) go to 501
c----save the input point if valid----------------------------------
        savep(1)=point(1)
        savep(2)=point(2)
501     continue
        return
        end
```

```
/*----GPRECV----RECEIVE GEOPLOT COMMANDS FROM THE PIPE----------

      Programmers:  J W Herriot and P L Ward, USGS, Menlo Park.
                    Fall 1979

     This group of programs interfaces the geoplot child process to
     the pipes from the parent. Gprecv is called from a very small
     program called geoplt which is forked by the parent. Gprecv then
     takes command calling the subroutines in geoplot as requested
     by the command (cmd) coming down the pipe. Cmd contains in the
     tens digit the subroutine number (kmd) used in the switch and
     in the units digit the number of variables to be read from the
     pipe.
*/

#include "gpcomc.h"
#include <stdio.h>
/* xargv comes from main.c in /usr/src/lib/F77 */
char **xargv;
static int parent;
static int readint=0;
static int begin=1;
static float FAK=0.0;
static float ONE=1.0;
static int  DBUF=15;
static int  MONE= -1;
static char  NONE=' ';
static int async=0;

static int pdbug;

gprecv_()
{
    int i,kmd,nget,mode,int0,int1,int2,int4,int5;
    long cmd,vec[6];
    extern gpcom_,gpquit();

    parent=pipeopen(xargv[0],xargv[1]);
    common = &gpcom_;
    while( (gpcread(&cmd,4) == 4) && (cmd > 0) ) {
        kmd=cmd/10;
        nget=cmd%10;
        common->kerror=0;
        common->ibreak=0;
        if (common->idbug>=10)
            fprintf(stderr,"\ngprecv: command=%d get %d words\n",kmd,nget);
        if(nget>0)gpcread(vec,nget*4);
        switch(kmd) {
            case  1: signal(2,gpquit); async=0;
                     if(*vec >= 100) { *vec = *vec-100; async=1; }
                     gpdevs_(vec,vec+1,vec+2,vec+3,vec+4                    ); break;
            case  2: int0=vec[0]; int1=vec[1]; gpputc_(&int0,&int1          ); break;
            case  3: int0=vec[0]; int1=vec[1]; gpgetc_(&int0,&int1          ); break;
            case  4: mode=vec[2]; movdrw_(vec,vec+1,&mode);
                     int0 = (vec[2] < 0) ? -vec[2] : vec[2];
                     if(int0 >= 5 && int0 <= 7) {
                         vec[2] = mode; gpcwrite(vec,12);                   } break;
            case  5: int2=vec[2]; int4=vec[4];
                     letter_(vec,vec+1,&int2,&NONE,vec+3,&int4,&MONE        ); break;
            case  6: map_(                                                  ); break;
            case  7: frame_ (                                               ); break;
            case  8: int0=vec[0]; shade_ (&int0                             ); break;
            case  9: int0=vec[0]; int1=vec[1]; pltltr_(&int0,&int1,&FAK); break;
            case 10: int0=vec[0]; int1=vec[1]; pltltr_(&int0,&int1,&ONE); break;
```

```
            case 11: glplot_(                                          ); break;
            case 12: if(common->intact==1)movdrw_(&FAK,&FAK,&DBUF      ); break;
            case 13: int0=vec[0]; axis_   (&int0                       ); break;
            case 14: int2=vec[2]; int5=vec[5];
                     grid_  (vec,vec+1,&int2,vec+3,vec+4,&int5); break;
            case 15: int4=vec[4]; propor_(vec,vec+1,vec+2,vec+3,&int4  ); break;
            case 16: int0=vec[0]; label_(&int0,&NONE,&MONE             ); break;
            case 17: int0=vec[4]; disazm_(vec,vec+1,vec+2,vec+3,int0   );
                     gpcwrite(vec,12); break;
            default: fprintf(stderr,"geoplt: bad cmd thru pipe=%ld\n",cmd);
        }
        if(common->ibreak==-1) {
            common->ibreak= -2;
            movdrw_(&FAK,&FAK,&DBUF);
        }
        if(common->idbug>=10)
            fprintf(stderr,"gprecv: send kerror=%d ibreak=%d async=%d\n",
        common->kerror,common->ibreak,async);
        if(async == 0) gpcwrite(&common->kerror,2);
    }
    movdrw_(&FAK,&FAK,&DBUF);
    fflush(stdout); fflush(stderr);
    if(async == 0) gpcwrite(&common->kerror,2);
}

/*-----gpgpip-----geoplot get data from pipes-----------------------*/
gpgpip_(vec,nwords) char vec[]; int *nwords; {
    int nchar;
    nchar= *nwords*4;
    if(nchar<0)nchar= -(*nwords);
    gpcread(vec,nchar);
    if(*nwords<0)vec[nchar]='\0';
}

/* gpgstr  Change strings for letter.f coming down the pipe to
   integers that include all the escape characters and fonts.
*/

#define MAXLET 132
#define CMASK   0377

/* crossref is the integer value of the plot character in letter.f
   for the  character shown in the comment line when preceeded by
   a \.                                                          */
static char crossref[]={
/*   0,  1,  2,  3,  4,  5,  6,  7,  8,  9,  :,  ;,  <,  =,  >,  ?,  @,  A,  B,  C,   */
    42,  6,  7,  8,  9,10,42,42,42,42,42,42,42,11,42,42,42,19,20,14,

/*   D,  E,  F,  G,  H,  I,  J,  K,  L,  M,  N,  O,  P,  Q,  R,  S,  T,  U,  V,  W,   */
    21,22,23,24,42,16,42,42,25,26,27,28,29,42,17,18,30,42,42,42,

/*   X,  Y,  Z,  [,  \,  ],  ^,  _,  `,  a,  b,  c,  d,  e,  f,  g,  h,  i,  j,  k,  */
    42,31,42,42,92,42,42,15,42,42,  2,42,  0,32,42,42,42,127,42,42,

/*   l,  m,  n,  o,  p,  q,  r,  s,  t,  u,  v,  w,  x,  y,  z,  {,  |,  },  ~       */
     4,42,34,42,42,42,  3,13,42,  1,  5,42,42,42,42,42,42,42,42,12};

gpgstr_(vec,nwords)
    int vec[], *nwords;
{
    char  c='';
    int second=0;
    int font=0;
    int newfont=0;
```

```
   int i,io,readit=1;
   long lnwords;

   lnwords = *nwords;
   if(lnwords >= 0) readit=0;
   if(readit==1) { gpcread(&lnwords,4); *nwords = lnwords;}
   for(i=0,io=0;i<lnwords;i++) {
      if(readit==1) {
         gpcread(&c,1);
         c=c& CMASK;
      }
      else c=vec[i];
      switch(c){
         case '\\':   if(second !=1){second=1; break;}
                        else {second=0; goto others;}
         case '!' :   if(newfont!=1){newfont=1; break;}
                        else {newfont=0; goto others;}
         default:       ;
others: if(newfont==1) {
            font=(c-'0')*128;
            if(font<0 || font>1152) font=0;
            newfont=0;
            break;
         }
         else if(second==1) {
            if(c < '0' || c > '~') c=' ';
            c=crossref[c-'0'];
            second=0;
         }
         if(io < lnwords && io < MAXLET) {
            vec[io]=(int)c;
            io++;
            if (c == 34 && io<lnwords && io<MAXLET){
               vec[io-1]=crossref['r'-'0'];
               vec[io]=crossref['1'-'0'];
               io++;
            }
         }
      }
   }
   if(io < lnwords) *nwords=io;
}

/*-----gpppip------geoplot put data into pipe----------------------*/
gpppip_(vec,nwords) char vec[]; int *nwords; {
   gpcwrite(vec,(int)(*nwords*4));
}

/*-----gpquit------handle the break key in geoplt-------------*/
gpquit()
{
   extern gpcom_;

   signal(2,gpquit);
   common = &gpcom_;
   common->ibreak= -1;
   gperr_(&common->ibreak,&NONE,&NONE,&async);
}

/* Get a string from the pipe.    PLW USGS 4/4/80 */
gpstrg(lenask,strg,lenget)
   int *lenask, *lenget;
   char strg[];
{
```

```
    int i,j,dlen;
    char c;
    long llenget;

    gpcread(&llenget,4);
    *lenget = llenget;
    dlen = *lenget+1-*lenask;
    if(dlen>0) {
        i = *lenask;
        j=1;
        gperr_(&j,"gpstrg","Length of string too long. Truncate to",
            &i);
        *lenget = *lenask-1;
    }
    else dlen=0;
    gpcread(strg,*lenget);
    strg[*lenget]='\0';
    if(dlen!=0) for(i=0;i<dlen;i++) gpcread(c,1);
}

int gpcread(vec,length)
    char *vec;
    int length;
{
    int i;

    i=piread(parent,vec,length,&readint);
    if(i != length) {
        common.ibreak=2;
        gperr_(&common->ibreak,"gprecv_","Unable to read on pipes. piread returns",&i);
    }
    return(i);
}

int gpcwrite(vec,length)
    char *vec;
    int length;
{
    int i;

    i=piwrite(parent,vec,length);
    if(i != length) {
      common.ibreak=2;
      gperr_(&common->ibreak,"gprecv_","Unable to write on pipes. piwrite returns",&i);
    }
    return(i);
}
```

```
c-*--SCALE AND ROTATE PLOT ON DEVICE AND INVERSE THEREOF----------------
c
      subroutine gpscal(mode,point)
c
c-----Programmer: PLWard, USGS, Menlo Park, California 4/15/80
c
c       mode = 1 scale and rotate to device
c       mode = 2 do inverse for movdrw mode 6,7, and 19
c
c       kscale=0 scaling and rotation not in use.
c              1 in use but initialize first
c              2 in use
c
      save
      include 'gpcom.h'
c
c
      real factor(2),fsquar,point(2),temp
      integer mode
      parameter ( RAD=0.0174533 )
      data factor/1.0,0.0/
c
      if(idbug.ge.5) call printn("gpscal: %d %f %f kscale=%d\n",
     1 mode,point(1),point(2),kscale)
c
c-*--Initialize the scaling factors.-----------------------------------
      if(kscale.ne.i1) goto 100
      factor(1)=pgain(1)*cos(pangle*RAD)
      factor(2)=pgain(2)*sin(pangle*RAD)
      fsquar=factor(1)*factor(1)+factor(2)*factor(2)
      kscale=2
      if(pangle.eq.0.0.and.pgain(1).eq.1.0.and.offset(1).eq.0.0.and.
     & offset(2).eq.0.0.and.pgain(2).eq.1.0) kscale=0
      if(kscale.eq.0) return
c
c-*--Scale and rotate.-------------------------------------------------
100   if(mode.eq.i2) goto 200
      temp=factor(1)*point(1)-factor(2)*point(2)
      point(2)=factor(1)*point(2)+factor(2)*point(1)+offset(2)
      point(1)=temp+offset(1)
      return
c
c-*--Inverse.----------------------------------------------------------
200   temp=(factor(1)*(point(2)-offset(2))-factor(2)*(point(1)-offset(1)
     1))/fsquar
      point(1)=(point(2)-offset(2))/factor(2)-temp*factor(1)/factor(2)
      point(2)=temp
      end
```

```
c---GPSINC---GET SINE AND COSINE OF ANGLE THIS LINE FROM POINTL TO
c       POINT MAKES WITH THE X-AXIS.
c
      subroutine gpsinc(pointl,point,sinx,cosx,base)
c
      real pointl(2),point(2),sinx,cosx,base
c
      sinx=point(2)-pointl(2)
      cosx=point(1)-pointl(1)
      base=sqrt(sinx*sinx+cosx*cosx)
      if(base.lt.0.0000001) return
      sinx=sinx/base
      cosx=cosx/base
      return
      end
```

```
c-*--GPSYMB---DRAWS SYMBS OR ARROWS AT THE END OF EACH VECTOR.--------
c
      subroutine gpsymb(mode,pointt)
c
c-----Programmer  PLWard, USGS, Menlo Park, California, February, 1979.-
c
      save
c
      real point(2),points(2),pointm(2),pointz(2),
     1  pointt(2),base,sinx,cosx,asymsk,slenbg,angmax,ang,consz,
     2  part,dslen,dzlen,angsym,asymsn,asymcs,angbeg,xi,slen,angle,
     3  PI,RAD
      integer mode
c
      include 'gpcom.h'
c
      parameter ( PI=3.141593, RAD=0.0174533)
c
c
      if (idbug.ge.6)call printn("gpsymb: %d %f %f\n",
     1  mode,pointt(1),pointt(2))
c
      point(1)=pointt(1)
      point(2)=pointt(2)
      if(mode.ne.1) goto 20
c
c-*--Move to the point.-----------------------------------------------
 10   call gpdevo(i1,point)
c     If symbol is an arrow, do not put symbol at a point moved to
      if(symshp.le.2.0)return
c
c-*--Draw a vector, dashed vector, or point for ltype =0,>0,<0 respect.
 20   if(symshp.eq.0.0) goto 21
      call gpsinc(pointp,point,sinx,cosx,base)
      points(1)=point(1)
      points(2)=point(2)
      if(symspa.lt.0.0) goto 24
 21   if(ltype.gt.4) goto 25
      call gpdash(lmode,point)
      goto 30
 24   call gpdash(i1,point)
      goto 30
c-----For software generated dash pattern.-----------------------------
 25   call gpdash(lmode,point)
c
c-*--Draw symbols and arrow tips.-------------------------------------
c       symshp  = 0.0 do not draw symbols.
c                 <=2.0 draw arrow tips on vectors.
c                 > 2.0 draw symbols with symshp number of sides.
c                 < 0.0 draw symbols as above but fill them in.
c       symang  = angle of arrow sides to vector for arrows and angle
c                   of orientation of symbol about vertical for symbol.
c                 < 0 then rotate symbols to angle of vector + symang.
c                   or do not rotate arrow tips.
c       symlen = length of a side of a symbol for sides <4, otherwise.
c                   essentially the diameter of the symbol.
c       symspa  = spacing between concentric symbols in page units.
c                 < 0 used to turn lines connecting symbols on and off.
c       symfac = multiply length of vector by this and add to
c                   symlen to get vector length.
c
 30   if(symshp.eq.0.0) return
      if(nosym .ne.0) return
      if(linmod.eq.2) return
```

```
c
c-----Set the proper line type for symbols if necessary.-----------------
      if(ltype.eq.lintyp(3)) goto 331
      point(1)=lintyp(3)
      call gpdevo(i11,point)
      lintyp(3)=ltype
      linmod=3
c
c-----Set the line shade for symbols if necessary.-----------------------
331   if (pshade.eq.shade(3)) goto 31
      point(1)=shade(3)
      call gpdevo(i12,point)
c
 31   asymsk=abs(symshp)
      slenbg=(symlen+base*sign(symfac,symlen))*0.5
      if(asymsk.gt.2.0) goto 32
c-----For arrow tips.
      base=PI-2.0*abs(symang)
      if(symang.gt.0.78540) slenbg=slenbg/sin(base)
      angmax=base*2.0
      ang=PI-base
      consz=(1.0+cos(ang))*asymsk
      part=consz-1.0
      dslen=abs(symspa)/(cos(base*0.5)*consz)
      if(symlen.lt.0.0) dslen=-dslen
      dzlen=abs(symspa)/(sin(atan2(tan(symang),(1-asymsk)))*consz)
      if(symlen.lt.0.0) dzlen=-dzlen
      angsym=0.0
      if(symang.ge.0.0) angsym=atan2(cosx,sinx)
      asymsn=sin(angsym)
      asymcs=cos(angsym)
      pointm(1)=points(1)-slenbg*asymsn
      pointm(2)=points(2)-slenbg*asymcs
      goto 33
c-----For symbols.
 32   base=2.0*PI/asymsk
      angmax=aint(asymsk)*base
      ang=PI-angmax*0.5
      angsym=abs(symang)
      if(symang.lt.0.0) angsym=atan2(cosx,sinx)+angsym
      asymsn=sin(ang+angsym)
      asymcs=cos(ang+angsym)
      pointm(1)=points(1)
      pointm(2)=points(2)
      part=1.0
      dslen=abs(symspa)/cos(base/2.0)
c-----For arrow tips and symbols.
 33   angbeg=ang+angsym
      angmax=angmax+angbeg+base*0.5
      xi=-1.0
c-----Draw concentric symbols.
      do 38 slen=slenbg,dslen,-dslen
           pointz(1)=pointm(1)-asymsn*slen*part
           pointz(2)=pointm(2)-asymcs*slen*part
           call gpdash(i1,pointz)
c-----Draw each symbol.
           do 36 angle=angbeg,angmax,base
                point(1)=pointm(1)-slen*sin(angle)
                point(2)=pointm(2)-slen*cos(angle)
 36             call gpdash(lmode,point)
           call gpdash(lmode,pointz)
           if(symshp.gt.0.0) goto 39
           if(asymsk.gt.2.0) goto 38
           slen=slen-dzlen
```

```
            if(abs(slen).lt.abs(dslen)) goto 39
            xi=xi+consz
            pointm(1)=points(1)-(dslen*xi+slen)*asymsn
            pointm(2)=points(2)-(dslen*xi+slen)*asymcs
38          continue
39      call gpdash(i1,points)
        return
        end
```

```
c-*--GPTEK- --TEKTRONIX 4010,4012,4014 OUTPUT FOR GEOPLOT.----------------
c
      subroutine gptek(jmode,point)
c
c-----Programmer PLWard, USGS, Menlo Park, California, February, 1979.----
c
c-*--This subroutine formats the commands and outputs them into a
c       buffer for controlling Tektronix 4010,4012, and 4014 devices.
c
c       point is the x,y coordinates in page units.
c
c
c      point is the x,y coordinates in page units.
c
c      mode or jmode can be one of the following:
c        1 move to the point
c        2 draw a vector to the point
c        3 draw a point at the point
c        4 never called at this level
c        5 never called at this level
c        6 ask device for the present pen position
c        7 ask device for the present crosshair position
c        8 initialize this plot device
c        9 set the device for hardware lettering
c       10 set the character size
c       11 set the line type
c       12 set the line shade (intensity or color)
c       13 new page, erase, or start a new frame
c       14 make a hardcopy of this display
c       15 dump the buffer, end this plot
c       16 never called at this level
c
c-*--This routine supports the following device models (numdev(2)):
c        0   Tektronix 4014 output but sent to any ascii terminal
c                for debugging purposes
c        1   Tektronix 4014,  with Enhanced Graphics
c        2   Tektronix 4014
c        3   Tektronix 4012
c        4   Tektronix 4010
c        5   Retrographics 4010 simulator
c
c       Maximum number of models is MNUMD2
c
      save
c
c
      include 'gpcom.h'
      include 'gpbuf.h'
c
      integer jmode,ixl,iyl,mode,ix,iy,nsyns,i,kbaudr
      integer FF,SYN,ETB,SUB,ESC,FS,GS,US,MNUMD2
      integer kpchar(5),i32,i64,i119,i120,i164
      integer*4 ltypeh(4)
      real point(2),dheigh(4),dspacl(4),daspec(4),dspach(4),tens
      integer ibaudr
      external ibaudr
      parameter (MNUMD2 = 5)
      parameter (ENQ=5, FF=12, SYN=22, ETB=23, SUB=26, ESC=27, FS=28,
     1   GS=29, US=31, i32=32, i64=64, i119=119, i120=120, i164=164)
      data dheigh/0.01786,0.01603,0.01053,0.01007/
      data daspec/0.625,0.643,0.6,0.6/
      data dspacl/0.61,0.595,0.725,0.645/
      data dspach/0.58,0.635,0.61,0.525/
      data ltypeh/12,31212,32,54/,ixl,iyl/0,0/
```

```
c
      if (idbug.ge.10) call printn("gptek : %d %f %f lenbuf=%d\n",
     1  jmode,point(1),point(2),lenbuf)
c
      mode=jmode
c
c-*--Branch to the proper value of mode. Note that mode is not checked
c       for the proper value to save time since users are not supposed to
c       call this routine directly.
c
c       mode= 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 16
      goto ( 5, 5, 5, 4, 4,60,70,80,90,100,110,120,130,140,150,4)mode
c
c-*--Improper call, return.-----------------------------------------------
  4   return
c
c-*--Are enhanced graphics features  in use? If iegm <0 then device
c       does not support them, if =0 then they need to be set, if >0
c       then they are already set.
  5   if (iegm.ne.i0) goto 6
      ibuffr(lenbuf)=ESC
      ibuffr(lenbuf+1)=96+kzaxis*8+kline
      lenbuf=lenbuf+2
      iegm=1
  6   if(mode.eq.i2) goto 20
c
c-*--Set terminal for a move or for regular point mode.-------------------
      if(intens.ne.i120) goto 15
      if(kplt.ne.i3) goto 12
      ibuffr(lenbuf)=US
      lenbuf=lenbuf+1
 12   ibuffr(lenbuf)=GS
      lenbuf=lenbuf+1
      kplt=2
      goto  24
c
c-*--Set terminal for special point mode and output intensity.-----------
 15   if(kplt.eq.i3) goto 18
      ibuffr(lenbuf)=US
      ibuffr(lenbuf+1)=ESC
      ibuffr(lenbuf+2)=FS
      lenbuf=lenbuf+3
      kplt=3
 18   ibuffr(lenbuf)=intens
      lenbuf=lenbuf+1
      goto  24
c
c
c-*--Set terminal for drawing vectors.-----------------------------------
 20   goto ( 22, 24, 21) kplt
 21   ibuffr(lenbuf)=US
      lenbuf=lenbuf+1
 22   ibuffr(lenbuf)=GS
      ibuffr(lenbuf+1)=iplt(5)
      ibuffr(lenbuf+2)=SYN
      lenbuf=lenbuf+3
      kplt=2
c
c**************************************************************************
c
c     BEGIN FORMATTING AND OUTPUTTING PLOT CHARACTERS TO THE BUFFER
c
c**************************************************************************
c
```

```
 40    if(lenbuf.ge.lenmax) call gpbufo(i3)
c
c--*--Return unless this is the beginning point of hardware lettering.---
      if(mode.ne.i17) return
      goto 90
c
c*************************************************************************
c
c           END FORMATTING AND OUTPUTTING OF PLOT CHARACTERS
c
c*************************************************************************
c
c--*--INTEROGATE TEKTRONIX AND GET THE PRESENT PEN POSITION.-------MODE  6
 60    ibuffr(lenbuf)=SYN
       lenbuf=lenbuf+1
       iout(1)=ESC
       iout(2)=ENQ
       call gpbufo(i3)
       call tekin(iout,in)
 65    point(1)=in(2)/dpage(1)
       point(2)=in(3)/dpage(2)
       return
c
c--*--READ THE CROSSHAIR POSITION.-------------------------------MODE  7
 70    ibuffr(lenbuf)=SYN
       lenbuf=lenbuf+1
       iout(1)=ESC
       iout(2)=SUB
       call gpbufo(i3)
       call tekin(iout,in)
       jmode=in(1)
       goto 65
c
c--*--INITIALIZE THIS DEVICE.------------------------------------MODE  8
 80    if(kbmtyp.ne.2)kbintyp=1
       dpage(1)=3120.0
c-----Check the model number.
       if(numdev(2).le.MNUMD2) goto 81
       kerror=2
       call gperr(kerror,"gptek","Model not implemented yet",numdev(2))
       return
 81    if(numdev(2).ne.i1.and.numdev(2).ne.i5) dpage(1)=780.0
       dpage(2)=dpage(1)
       rasmin=1.0/dpage(1)
       devmax(1)=1.3125
       if(numdev(2).ne.i1.or.numdev(2).ne.i5) devmax(1)=1.311538
       devmax(2)=1.0
       dpgin=10.92
       if(numdev(2).gt.i2.and.numdev(2).lt.i5) dpgin=5.71
       if(numdev(2).eq.i5) dpgin=6.2
       hpgin=6.48
       intact=1
c-----Set constants for enhanced graphics.
       kzaxis=0
       kline=0
       iegm=-1
       kplt=1
       intens=120
c-----Set up buffer + delays for plot output.
c********LENMAX MUST BE 12 WORDS LESS THAN THE BUFFER SIZE**************
       lenmax=2030
       lenbuf=1
       kbaudr=ibaudr(2)
       nsyns=kbaudr/3080+1
```

```
c-*--Calculate the plot characters to arrive at ix,iy.--------------------
c       The order of characters is HIY, LSB X&Y, LOWY, HIX, LOWX.
 24    ix=point(1)*dpage(1)+0.5
       iy=point(2)*dpage(2)+0.5
       iplt(1)=mod(iy/128,32)+32
       iplt(2)=mod(iy,4)*4+mod(ix,4)+96
       iplt(3)=mod(iy/4,32)+96
       iplt(4)=mod(ix/128,32)+32
       iplt(5)=mod(ix/4,32)+64
c
c-*--Optimize the output by comparing it to last output.-----------------
       if(nsyns ne.i0) goto  31
c-----Do not optimize or calculate delays for high-speed interface.-------
       do  30 i=1,5
            ibuffr(lenbuf)=iplt(i)
 30         lenbuf=lenbuf+1
       goto  39
c-----Include high y if changed.
 31    if(kpchar(1).eq.iplt(1)) goto  32
       ibuffr(lenbuf)=iplt(1)
       lenbuf=lenbuf+1
       kpchar(1)=iplt(1)
c-----Include lsb of y and x if changed.
 32    if(numdev(2).ge.i2.and.numdev(2).lt.i5) goto  33
       if(kpchar(2).eq.iplt(2)) goto  33
       ibuffr(lenbuf)=iplt(2)
       lenbuf=lenbuf+1
       kpchar(2)=iplt(2)
       goto  34
c-----Include low y if changed.
 33    if(kpchar(3).ne.iplt(3)) goto  34
       if(kpchar(4).eq.iplt(4)) goto  35
 34    ibuffr(lenbuf)=iplt(3)
       lenbuf=lenbuf+1
       kpchar(3)=iplt(3)
c-----Include high x if changed.
       if(kpchar(4).eq.iplt(4)) goto  35
       ibuffr(lenbuf)=iplt(4)
       lenbuf=lenbuf+1
       kpchar(4)=iplt(4)
c-----Include low x.
 35    ibuffr(lenbuf)=iplt(5)
       lenbuf=lenbuf+1
       kpchar(5)=iplt(5)
c
c-*--Calculate the number of delay characters required after this vector
       if(numdev(2).lt.i3) goto  36
       kpad=nsyns-1
       goto  37
 36    kpad=((iabs(ixl-ix)+iabs(iyl-iy))*nsyns)/8192+1
 37    if(kpad.gt.i5)kpad=5
c-----Output the delays.
       do  38 i=1,kpad
            ibuffr(lenbuf)=SYN
 38         lenbuf=lenbuf+1
 39    ixl=ix
       iyl=iy
       if(mode.ne.i3.and.mode.ne.i5) goto 40
c
c-*--Output LOX again to write a point.----------------------------------
       ibuffr(lenbuf)=iplt(5)
       lenbuf=lenbuf+1
c
c-*--Check to see if the buffer needs dumping.---------------------------
```

```
      if(numdev(2).gt.i2.and.numdev(2).lt.i5) return
          if(kplt.eq.i1) goto 109
               ibuffr(lenbuf)=US
               lenbuf=lenbuf+1
               kplt=1
109       ibuffr(lenbuf)=ESC
          ibuffr(lenbuf+1)=55+kkchar
          lenbuf=lenbuf+2
          goto 150

c
c--*--SET THE LINE TYPE.-----------------------------------------MODE 11
c       lmode = 1 move
c               2 vector line
c               3 point
c               4 line of points
c
c       ltype  = 0 point
c                1 line
c                2 line of points
c                3 to 11 hardware line types
c                >12      software dash patterns
c
110   ltype=abs(point(1))+0.5
      lmode=2
      if(ltype.le.i0) lmode=3
      if(ltype.eq.i2) lmode=4
      kline=0
      if(iegm.eq.-i1) goto 111
      iegm=-1
      if(kzaxis.ne.i0)iegm=0
c-----Turn off the enhanced graphics features of dashed or defocussed
c       vectors.
      ibuffr(lenbuf)=ESC
      ibuffr(lenbuf+1)=96
      lenbuf=lenbuf+2
c----Point.
111   if(ltype.eq.i0) return
c----Solid line.
      if(ltype.gt.i1) goto 113
      intens=120
      return
c----Line of points
113   if(ltype.eq.i2) return
c----Hardware dash.
114   if(ltype.gt.i11) goto 116
      if(ltype.gt.i6) ltype=6
      if(numdev(2).gt.i2.and.numdev(2).lt.i5) goto 115
      kline=ltype-2
      iegm=0
      return
c----Software dash from hardware code.
115   ltype=ltypeh(ltype-2)
      return
c----Software dash.
c----If hardware type, use the hardware.
116   do 117 i=1,4
          if(ltype.ne.ltypeh(i)) goto 117
          ltype=i+2
          goto 114
117       continue
      return
c
c--*--SET THE LINE SHADE (INTENSITY OR COLOR).--------------------MODE 12
```

```
c----Initialize the last plot character.
      do 82 i=1,5
 82       kpchar(i)=-1
c     - Initialize the pen home position.
      phome(1)=0.0
      phome(2)=0.984295
      if(numdev(2).gt.i2) phome(2)=0.983333
c----Set up for hardware lettering.
      do 83 i=1,4
 83   chrhgt(i)=dheigh(i)
      nchar=4
      if(numdev(2).ge.i2.and.numdev(2).lt.i5) nchar=1
      if(kkchar.ge.1.and.kkchar.le.nchar) goto 108
      kkchar=nchar
      if(nchar.gt.1)kkchar=nchar-1
      kchar=kkchar
      goto 108
c
c--*--SET ALPHANUMERIC MODE FOR TEXT PRINTING.----------MODE  9
c     Move pen to beginning of alphanumeric text string to be printed
c       and return to get call to devprn in order to print the string.
 90   if(kplt.eq.i1) return
      if(iegm.gt.i0) iegm=0
      ibuffr(lenbuf)=US
      lenbuf=lenbuf+1
      kplt=1
      return
c
c--*--SET CHARACTER SIZE.------(SEND ESC 56 to 59)----------MODE 10
100   if(point(1).gt.0.99) goto 106
      if(point(1).ge.0.0 ) goto 101
           i=point(1)-0.5
           call gperr(kerror,"gptek","Negative letheight ignored.",i)
           return
101   height=abs(point(1))
102   if(width(2).gt.1.8*rasmin) kchar=0
      if(slopel.gt.0.01) kchar=0
      if(kchar.eq.0)return
      kchar=0
c----If any hardware size is close enough to software size specified,
c     use it.
      do 103 i=nchar,1,-1
           if(abs(height-dheigh(i)).lt.chrerr) goto 104
103        continue
      return
104   kchar=i
      if(abs(aspect-daspec(kchar)).gt.0.045) kchar=0
      if(abs(spacel-dspacl(kchar)).gt.0.15) kchar=0
      if(abs(spaceh-dspach(kchar)).gt.0.12) kchar=0
      if(kchar.eq.0)return
      goto 107
c----Set hardware character size.
c     kchar is hardware character size, = 0 if not hardware size
c     kkchar is hardware character size and never = 0. ie it is
c       the most recent hardware size set.
106   kchar=point(1)+0.5
107   if (kchar.lt.i1) kchar=1
      if (kchar.gt.nchar) kchar=nchar
      kkchar=kchar
108   slopel=0.0
      aspect=daspec(kkchar)
      spacel=dspacl(kkchar)
      spaceh=dspach(kkchar)
      height=dheigh(kkchar)
```

```
      return
c
      end

c---*--GPBUFO---OUTPUT THE GPBUFF OF PLOT CHARACTERS AND KEEP TRACK OF
c       THE BEAM POSITION ON THE TEKTRONIX
c
      subroutine gpbufo(mode)
c
c---  Programmer: PLWard, USGS, Menlo Park, California 94025, April,1979
c
c---*--Output the plot buffer (ibuffr(lenbuf)). Keep track of the beam
c       position so that after outputting plot characters the prompt
c       can be put on the line below where it was before plotting.
c         kbmflg = 0  save the beam position.
c                  1  beam position already saved.
c
c         kbmtyp = 1  restore beam to this position after plotting.
c                  2  restore beam to home after plotting.
c                  3  leave beam where it is but put terminal in
c                       alphanumeric mode.
c                  4  do nothing with the beam.
c
c         mode   = 1  dump the buffer and restore the beam position.
c                  2  reset the beam position for a new page.
c                  3  dump the buffer but expect more plotting so do
c                       not restore the beam position.
c
      save
c
      include 'gpcom.h'
      include 'gpbuf.h'
c
      integer*4 ITEMP,KSIZE(4)
      integer IGET(2),KHOME(7),ENQ,LF,SYN,ESC,FS,GS,US,
     1  mode,ksizel,kbeam,mbuf,num,i,j
      parameter ( ENQ=5, LF=10, SYN=22, ESC=27, FS=28, GS=29, US=31)
      data IGET /27,5/
      data KSIZE/35,38,58,64/
      data KHOME/108,127,32,64,22,22,22/
      data ksizel/3/,kbeam/0/,mbuf/0/
c
      if(idbug.lt.4) goto 1
      call printn("gpbufo:mode=%d lenbuf=%d mbuf=%d kbeam=%d kbmtyp=%d n
     1um=%d ibreak=%d\n",mode,lenbuf,mbuf,kbeam,kbmtyp,num,ibreak)
      call printn("gpbufo:kplt=%d ksizel=%d kkchar=%d iprnbf=%d\n",
     1  kplt,ksizel,kkchar,iprnbf)
      if(idbug.ge.12) call prnout(lenbuf-1,ibuffr)
c
c---*--Handle the BREAK key by deleting buffer each time it is to be out-
c       put and then the last time (ibreak=-2 set in gprecv) reset beam
c       position.
 1    if(ibreak.ne.-i1) goto 5
      lenbuf=1
      return
 5    if(ibreak.ne.-i2) goto 6
      ibreak=0
      lenbuf=1
      goto 41
 6    if(mode.eq.i2) goto 20
c
c****BEFORE OUTPUTTING BUFFER SAVE THE BEAM POSITION IF DESIRED.*******
c---*--Save the beam position.-----------------------------------------
10    if(kbeam.eq.i1) goto 30
```

```
120     pshade=point(1)
        if(point(1).lt.999.0) goto 121
c-----Set write-thru mode for tek 4014 with enhanced graphics.
        if(numdev(2).ne.i1.and.numdev(2).ne.i2) return
        kzaxis=2
        iegm=0
        return
c-----Set the intensity
121     tens=abs(point(1))
122     if(tens.lt.100.001) goto 123
        tens=tens-100.0
        goto 122
c-----Set for vector or point mode full intensity.
123     kzaxis=0
        iegm=0
        if(kline.eq.i0)iegm=-1
        if(tens.lt.99.5) goto 125
        intens=120
        if(lmode.ne i3)lmode=2
        if(point(1).gt.0.0)return
c-----Defocused beam.
        kzaxis=1
        iegm=0
        return
c-----Set intensity for special point mode.
c        This equation gives proper intensity character to <2%, usually<1%
125     if(tens.lt.0.000001) tens=0.000001
        intens=10.0**((0.117*alog10(tens)+1.845))
        if(intens.gt.i119) intens=119
        if(intens.lt.i64) intens=64
        if(point(1).gt.-0.0001) goto 126
c-----Defocused special point mode.
        intens=intens-64
        if(intens.lt.i32) intens=32
126     if(lmode.eq.i2)lmode=4
        return
c
c--*--PAGE: ERASE SCREEN AND RETURN CURSOR TO HOME.----------MODE 13
130     if(kplt.eq.1) goto 131
        ibuffr(lenbuf)=US
        lenbuf=lenbuf+1
        kplt=1
131     ibuffr(lenbuf)=ESC
        ibuffr(lenbuf+1)=FF
        lenbuf=lenbuf+2
        if(iegm.gt.i0) iegm=0
        call gpbufo(i2)
        call sleep(4)
        return
c
c--*--MAKE HARD COPY FROM THE HARDCOPY DEVICE.----------MODE 14
140     if(kplt.eq.1) goto 141
        ibuffr(lenbuf)=US
        lenbuf=lenbuf+1
        kplt=1
141     ibuffr(lenbuf)=ESC
        ibuffr(lenbuf+1)=ETB
        lenbuf=lenbuf+2
        call gpbufo(i1)
        call sleep(8)
        return
c
c--*--DUMP THE BUFFER.----------MODE 15
150     call gpbufo(i1)
```

```
c----If the buffer contains nothing new, return. Mbuf is set when a new
c      buffer is initialized at the end of this subroutine.
      if(lenbuf.le.mbuf) return
      kbeam=1
      num=0
      if (kbmtyp.gt.i1) goto 30
      call tekin(IGET,in)
c----See if margin 2 is selected.
      if(mod(in(1),i4).ge.i2) num=KSIZE(kkchar)+1
c ----Find the position in terms of the number of linefeeds.
c----The reason for sending linefeeds instead of simply moving the beam
c      to the right place is to be able to set margin 2 on the tek if
c      necessary.
      ITEMP=((3120-in(3))*KSIZE(kkchar))/3071.0 -0.5
      num=num+ITEMP
      if(ksizel.lt.kkchar) num=num+1
      if(num.gt.129)num=0
c----Save the character size.
      ksizel=kkchar
      if(idbug.ge.10)
     1   call printn("gpbufo: num=%d in=%d %d %d lenbuf=%d\n",
     2   num,in(1),in(2),in(3),lenbuf)
      goto 30
c
c****WHEN ERASING A PAGE................WHEN ERASING A PAGE..........
c--*--Reset beam position to home if page is called.---------------------
20    if(kbeam.eq.i0) ksizel=kkchar
      kbeam=1
      num=0
      goto 31
c
c****DUMP THE BUFFER............................DUMP THE BUFFER
30    if(mode.ne.i3.and.kbmtyp.ne.i4) goto 32
31    lenbuf=lenbuf-1
      if(kplt.ne.i3) goto 50
      lenbuf=lenbuf+1
      ibuffr(lenbuf)=US
      goto 50
c
c-*--Dump the buffer and restore to alphanumeric mode.------------------
32    kbeam=0
      if(kbmtyp.ne.i3)goto 35
      ibuffr(lenbuf)=US
      goto 50
c
c-*--Restore the pen position to where it was when plotting began.------
35    if(lenbuf+15+num.le.lenmax) goto 40
      if(iprnbf.ne.i1) call pltout(lenbuf-1,ibuffr)
      if(iprnbf.eq.i1) call prnout(lenbuf-1,ibuffr)
      if(idbug.ge.12) call prnout(lenbuf-1,ibuffr)
      lenbuf=1
40    if(kplt.ne.i3) goto 42
41    ibuffr(lenbuf)=US
      lenbuf=lenbuf+1
42    ibuffr(lenbuf)=GS
      lenbuf=lenbuf+1
      ibuffr(lenbuf)=55
      j=1
      if(numdev(2).ge.i2.and.numdev(2).lt.i5) j=2
      do 45 i=j,7
          lenbuf=lenbuf+1
          ibuffr(lenbuf)=KHOME(i)
45        continue
      lenbuf=lenbuf+1
```

```
      ibuffr(lenbuf)=US
      if(kbmtyp.eq.i2) goto 50
      if(num.eq.i0) goto 50
      if(ksizel.eq.kkchar) goto 46
          ibuffr(lenbuf+1)=ESC
          ibuffr(lenbuf+2)=55+ksizel
          lenbuf=lenbuf+2
46    if(lenbuf+num.le.lenmax) goto 47
          if(iprnbf.ne.i1) call pltout(lenbuf,ibuffr)
          if(iprnbf.eq.i1) call prnout(lenbuf,ibuffr)
          lenbuf=0
47    do 48 i=1,num
          lenbuf=lenbuf+1
48        ibuffr(lenbuf)=I.F
      if(ksizel.eq.kkchar)goto 50
          ksizel=kkchar
          ibuffr(lenbuf+1)=ESC
          ibuffr(lenbuf+2)=55+ksizel
          lenbuf=lenbuf+2
c
c---- Dump the buffer and initialize a new buffer by a move to the last
c       point plotted.
50    if(iprnbf.eq.i1) call prnout(lenbuf,ibuffr)
      if(iprnbf.ne.i1) call pltout(lenbuf,ibuffr)
49    lenbuf=1
      if(kplt.ne.3) goto 52
c-----Special point mode.
51    ibuffr(1)=ESC
      ibuffr(2)=FS
      ibuffr(3)=intens
      lenbuf=4
      goto 53
c-----Plot mode.
52    ibuffr(1)=GS
      lenbuf=2
53    ibuffr(lenbuf)=iplt(1)
      lenbuf=lenbuf+1
      if(numdev(2).gt.i2.and.numdev(2).lt.i5) goto 55
      ibuffr(lenbuf)=iplt(2)
      lenbuf=lenbuf+1
55    ibuffr(lenbuf)=iplt(3)
      ibuffr(lenbuf+1)=iplt(4)
      ibuffr(lenbuf+2)=iplt(5)
      ibuffr(lenbuf+3)=SYN
      ibuffr(lenbuf+4)=SYN
      ibuffr(lenbuf+5)=SYN
      lenbuf=lenbuf+6
      if(kplt.eq.1)kplt=2
      if(iegm.ne.i1) goto 58
      ibuffr(lenbuf)=ESC
      ibuffr(lenbuf+1)=96+kzaxis*8+kline
      lenbuf=lenbuf+2
58    if(mode.ne.i2) kbmflg=0
      mbuf=lenbuf
      return
      end
```

```
c-*--DRAW A TIC AND LABEL IT IF APPROPRIATE.-----------------------------
c
      subroutine gpticl(value,levlab,levtic,level)
c
c     Programmer: PLWard, U.S. Geological Survey, Menlo Park, Ca. 3/4/80
c
      save
c
      include 'gpcom.h'
      include 'gpaxis.h'
c
      integer len,one,two,mone,mtwo
      integer kposi(4),lenstr,iprec,levlab,levtic,level,jaxis,kpltsv
      real pone(2),ptwo(2),pdata(2),value,pdist,delta,sin1,cos1,val
      character*30 numstr,label
      real gpfunc
      external gpfunc
      data kposi/6,7,4,5/,one/1/,two/2/,mone/-1/,mtwo/-2/
c
      if(idbug.lt.2) goto 10
          call printn("gpticl: value=%f levlab=%d levtic=%d\n",
     1        value,levlab,levtic)
          call printn("gpticl: iaxis=%d iother=%d numax=%d\n",
     1        iaxis,iother,numax)
c
c-*--Return if tic and label are not to be drawn.-----------------------
 10   if(levtic.eq.0) return
c
c-*--Find the location of the tic mark and label.----------------------
      if(ltrans.gt.0)goto 350
c----For axis.
      pdata(iaxis )=value
      pdata(iother)=dother
      call gptran(i1,pdata)
      goto 375
c----For grid.
350   pdist=c1*gpfunc(ltrans,value)-c2
      pdata(1)=pdist*cosx+pbegin(1)
      pdata(2)=pdist*sinx+pbegin(2)
c
c-*--Draw tic marks on this axis.---------------------------------------
375   pold(1)=pdata(1)
      pold(2)=pdata(2)
      jaxis=5*(iaxis-1)+levtic
      if(ticfac(numax).eq.0.0) goto 520
      kpltsv=kpltsp
      kpltsp=1
      call movdrw(pdata(1)+ticbax*ticout(jaxis),pdata(2)+ticbay*
     1     ticout(jaxis),one)
      call movdrw(pdata(1),pdata(2),two)
      if(linspa.eq.1.and.itcpag.ne.1) goto 510
      call movdrw(pdata(1)+ticbax*ticin(jaxis),pdata(2)+ticbay*
     1     ticin(jaxis),two)
      kpltsp=kpltsv
      goto 520
c----Plot ticin in data units.
510   kpltsp=kpltsv
      pdata(iaxis)=value
      pdata(iother)=dother
      call movdrw(pdata(1),pdata(2),mone)
      pdata(iother)=dother+ticdat*ticin(jaxis)
      call movdrw(pdata(1),pdata(2),mtwo)
c
c-*--Print the label.---------------------------------------------------
```

```
520     if(levlab.eq.0.or.dislab(numax).eq.0.0) return
c
c--*--For curvilinear coordinates find instantaneous angle of line.-----
        if(linspa.ne.1) goto 530
        delta=differ(level)*0.1
        pone(iaxis)=value-delta
        pone(iother)=dother
        call gptran(i1,pone)
        ptwo(iaxis)=value+delta
        ptwo(iother)=dother
        call gptran(i1,ptwo)
        call gpsinc(pone,ptwo,sin1,cos1,delta)
        ang=atan2(sin1,cos1)*57.29578
        if(numax.eq.2.or.numax.eq.3) ang=ang+180.
c-----Find string for label according to data type.
530     val=value/faclab(numax)
        iprec=-alog10(differ(level)/faclab(numax))
        if(iprec.gt.0)iprec=iprec-1
        if(iprec.le.0)iprec=-1
        if(idbug.ge.4)call printn("gpticl: iprec is %d diff=%f\n",
       1 iprec,differ(level))
        label=numstr(val,iprec)
        len=lenstr(label)
        if(len.le.minlab(numax)) goto 550
        label=numstr(val,iprec+1000)
        len=lenstr(label)
550     call letter(pold(1)+ticbax*poslab(levlab),pold(2)+ticbay*
       1   poslab(levlab),one,label,ang+anglab(numax),labps,len)
        end
```

```
c--*--GPTICR---RECURSIVE SUBROUTINE TO DRAW TICS-------------------------
c
      subroutine gpticr(levold,levlab,levtic,base)
c
c--*--Programmer: PLWard, USGS, Menlo Park, California 4/80
c
c--*--This routine calls itself recursively. It determines how many
c     tics can be drawn on this level. If all tics can be drawn it
c     calls itself to descend one more level and try again.
c     Inputs are the level of the caller, the level of the label for
c     the next tic at the callers level, the level of the next tic at
c     the callers level, and the base or data value of the last tic at
c     the callers level.
c
      save
c
      automatic level,levla,levti,bas,k,klabsk
      include 'gpcom.h'
      include 'gpaxis.h'
      integer nums(2),levold,levlab,levtic,level,levla,levti,klabsk,num,
     1  k,j,klab,kticsk,levl,levt
      real pdata(2),polds(2),base,bas,valu,pdist,dist,value,temp
      real gpfunc
      external gpfunc
      integer iskip(12),MSKIP
c-----MSKIP is (length of iskip/2) -1
      parameter (MSKIP=5)
      data iskip/1,2,5,10,15,30,1,2,6,12,1000,1000/
c
c--*--Buffer inputs and store on the stack to keep each instance.--------
      level=levold+1
      levla=levlab
      levti=levtic
      bas=base
      klabsk=10000
      if(idbug.ge.2)call printn("gpticr: level=%d levla=%d levti=%d bas=
     1%f numtic=%d\n",level,levla,levti,bas,numtic(level))
c
c--*--Determine how many tics and labels fit on this level.--------------
c       If this is level 1 simple step through each tic indexed by i.
c
      num=0
      nums(1)=0
      nums(2)=0
      distic=0.0
      dislb=0.0
      polds(1)=pold(1)
      polds(2)=pold(2)
      do 400 k=1,numtic(level)
           valu=bas+k*differ(level)
           if(idbug.ge.2)call printn("gpticr: numtic=%d valu=%f\n",
     1numtic(level),valu)
           if((valu-drange(1))*axsign.lt.0.0) goto 390
           if((valu-drange(2))*axsign.gt.0.0) goto 410
           if(ltrans.gt.0)goto 350
c          <For axis.>
           pdata(iaxis )=valu
           pdata(iother)=dother
           call gptran(i1,pdata)
           if(kerror.ne.i0)return
           goto 375
c          <For grid.>
350        pdist=c1*gpfunc(ltrans,valu)-c2
           pdata(1)=pdist*cosx+pbegin(1)
```

```
            pdata(2)=pdist*sinx+pbegin(2)
c           <For axis and grid.>
375         dist=sqrt((pold(1)-pdata(1))**2+(pold(2)-pdata(2))**2)
            pold(1)=pdata(1)
            pold(2)=pdata(2)
            num=num+1
            distic=distic+dist
            if(idbug.ge.1)call printn("gpticr: k=%d distic=%f ticmin=%f\n"
     1        ,k,distic,ticmin(numax))
            if(num.gt.1.and.(distic.lt.ticmin(numax))) goto 390
            nums(1)=nums(1)+1
            distic=0.0
            if(levla.eq.0)goto 390
            dislb=dislb+dist
            if(dislb.lt.widlab) goto 390
            nums(2)=nums(2)+1
            dislb=0.0
390         if(idbug.ge.2)call printn("gpticr: lev=%d val=%f num=%d ntic=%
     1d nlab=%d\n",level,valu,num,nums(1),nums(2))
            if(idbug.ge.2)call printn("gpticr: dist=%f dlab=%f dtic=%f\n",
     1 dist,dislb,distic)
400         continue
c
410    pold(1)=polds(1)
       pold(2)=polds(2)
       klabsk=1
c
c--*--If all labels can not be drawn, decide on spacing for labels.-------
420    if(nums(2).ge.num.or.levla.eq.0) goto 450
       if(nums(2).gt.0) goto 425
            klabsk=10000
            goto 450
425    j=1
       if(numtic(level).eq.12.or.numtic(level).eq.24)j=6
       klabsk=num/nums(2)
       do 430 klab=j,j+MSKIP
            if(klabsk.le.iskip(klab)) goto 435
430         continue
435    klabsk=iskip(klab)
c
c--*--If all tics can be drawn then descend to the next lower level in
c        the tic hierarchy.
450    if(nums(1).lt.num.or.level.ge.maxpow(numax)) goto 500
       if(idbug.ge.2)call printn("gpticr 450: klabsk=%d kticsk=%d numtic=
     1 %d\n",klabsk,kticsk,numtic(level))
       do 490 k=1,numtic(level)
            levl=levla
            levt=level
            if(k.ne.((k/klabsk)*klabsk))levl=0
480         if(k.ne.numtic(level)) goto 485
            levt=levti
            levl=levla
485         call gpticr(level,levl,levt,bas)
            if(ibreak.le.-1)return
490         continue
       base=base+differ(level)*numtic(level)
       return
c
c--*--This is the bottom. Now decide on appropriate spacing for tics.-----
500    j=1
       kticsk=10000
       if(nums(1).eq.0) goto 600
       if(numtic(level).eq.12.or.numtic(level).eq.24)j=7
       temp=float(num)/float(nums(1))
```

```
      do 520 klab=j,j+MSKIP
           if(temp.le.iskip(klab))goto 530
520        continue
530   kticsk=iskip(klab)
c
c-*--Plot and label tics for this level.---------------------------------
600   if(idbug.ge.2)call printn("gpticr 600: klabsk=%d kticsk=%d\n",
     1 klabsk,kticsk)
      do 650 k=kticsk,numtic(level),kticsk
           value=bas+k*differ(level)
           if((value-drange(2))*axsign.le.0.0)goto 610
           ibreak=-3
           return
610        if((value-drange(1))*axsign.lt.0.0) goto 650
           levl=0
           levt=level
           levt=level
           if(levla.eq.0) goto 620
           if(k.eq.((k/klabsk)*klabsk).and.klabsk.ne.10000)levl=level
620        if(k.ne.numtic(level)) goto 625
           levt=levti
           levl=levla
           if(level.eq.1.and.nums(2).lt.1) levl=0
625        call gpticl(value,levl,levt,level)
           if(ibreak.le.-1) return
650        continue
      base=base+differ(level)*numtic(level)
c
      end
```

```
/*                          G P T R A N
 *
 *  Transformation routines for geoplot, qp, etc.
 *  Programmer  PLWard Spring 1983
 *  Definition of certain macros depend on application and ifdefs:
 *     ifdef GEOPLOT       uses geoplot common block variables
 *     ifdef QP            uses plot_glob global variable structure
 *     ifdef GENERIC       uses simple global variables
 *     (otherwise)         expects an include file "gptran.h"
 */

#include <stdio.h>
#include <math.h>

/* The following defines cover all the interface of gptran to geoplot.
     To interface to another package use different defines and supply
     a routine gperr_.                                              */
/*------------------------- G E O P L O T ------------------------------*/
#ifdef GEOPLOT

#include "gpcomc.h"                          /* common block for geoplot */
#define INITCOM   extern gpcom_; common = &gpcom_;
#define CTRANS    common->trans              /* transformation constants    */
#define NTRANS    common->ktrans             /* number of transformation in effect */
#define ERROR     common->kerror             /* error flag returns 2 for fatal error */
#define PS        (common->pspace)           /* page space    */
#define GS        (common->pspace+6)         /* grid space                  */
#define SS        (common->pspace+12)        /* subject space          */
#define DS        (common->pspace+18)        /* data space             */
#define TRANFL    common->ktrflg             /* flag true if need to initialize trans */
#define LINSPA    common->linspa             /* flag true if nonlinear transformation */
#define SPALIN    common->spalin             /* spacing in x and y data
                                                units for nonlinear transformation */
#define CURINC    common->curinc             /* spacing in page units for
                                                nonlinear transformations */
#define USYM      common->usym               /* missing data symbol */
#define DEBUG     common->idbug              /* debug flag set >= 5 for printout */
#define POLE      common->pole               /* pole of rotation for trans=30 */
#define SETSCA    common->setsca             /* scale to set map to*/
#define MAPSCA    common->mapsca             /* scale map is now set to*/
#define DPGIN     common->dpgin              /* inches per page unit on device*/

#endif

/*------------------------------ Q P ---------------------------------*/
#ifdef QP

#include "plot.h"                    /* common block for qp */

float trans[9];                     /* constants: cf CTRANS              */
int   settrans[9];                  /* gp_init: don't init trans[i] if 1 */
int   kerror=0;
int   ktrflg;
int   linspa;
float spalin[3];
int   idbug;
float pole[2],dpgin; float setsca=0.0; float mapsca=0.0;

#define INITCOM   qp_init();         /* extern gpcom_; common = &gpcom_;   */
#define CTRANS    trans              /* transformation constants          */
#define NTRANS    (P->p_trans)       /* number of transformation in effect */
```

```
#define ERROR    kerror          /* error flag returns 2 for fatal error */
#define PS       (& P->p_mxa)    /* page space  (float pspace[24])       */
#define GS       (& P->p_pxa)    /* grid space                           */
#define SS       (& P->p_pxa)    /* subject space                        */
#define DS       (& P->p_dxa)    /* data space                           */
#define TRANFL   ktrflg          /* flag true if need to init trans      */
#define LINSPA   linspa          /* flag true if nonlinear trans         */
#define SPALIN   spalin          /* spacing in x and y data units for    */
                                 /* nonlinear trans (float spalin[3])    */
#define CURINC   (P->p_resol)    /* spacing in page units for nl trans   */
#define USYM     (P->p_usym)     /* missing data symbol (float usym)     */
#define DEBUG    idbug           /* debug flag set >= 5 for printout     */
#define POLE     pole            /* pole of rotation for trans=30        */
#define SETSCA   setsca          /* scale to set map to*/
#define MAPSCA   mapsca          /* scale map is now set to*/
#define DPGIN    dpgin           /* inches per page unit on device*/

static qp_init() {
   static int once;
   int i;

   if(once)return;
   for(i=0; i<9; i++) if(!settrans[i]) trans[i] = USYM;
   once++;
}

gp_init() { static int three = 3; float d[2]; gptran_(&three, d); }

newpol_() { gperr_(0, "newpol_", "gptran routine not linked in."); }
refpt_()  { gperr_(0, "refpt_" , "gptran routine not linked in."); }

#endif

/*------------------------  G E N E R I C  -------------------------*/
#ifdef GENERIC

float ctrans[9];                 /* constants: cf CTRANS             */
int   ntrans;
int   kerror=0;
float ps[6];
float ss[6];
float ds[6];
int   ktrflg;
int   linspa;
float spalin[3];
float curinc;
float usym;
int   idbug;
float pole[2],dpgin; float setsca=0.0; float mapsca=0.0;

#define INITCOM                  /* extern gpcom_; common = &gpcom_;    */
#define CTRANS   trans           /* transformation constants            */
#define NTRANS   ntrans          /* number of transformation in effect  */
#define ERROR    kerror          /* error flag returns 2 for fatal error */
#define PS       ps              /* page space  (float pspace[24])       */
#define GS       ss              /* grid space                           */
#define SS       ss              /* subject space                        */
#define DS       ds              /* data space                           */
#define TRANFL   ktrflg          /* flag true if need to init trans      */
#define LINSPA   linspa          /* flag true if nonlinear trans         */
#define SPALIN   spalin          /* spacing in x and y data units for    */
```

```
                                    /* nonlinear trans (float spalin[3])  */
#define CURINC    curinc            /* spacing in page units for nl trans */
#define USYM      usym              /* missing data symbol (float usym)   */
#define DEBUG     idbug             /* debug flag set >= 5 for printout   */
#define POLE      pole              /* pole of rotation for trans=30      */
#define SETSCA    setsca            /* scale to set map to */
#define MAPSCA    mapsca            /* scale map is now set to */
#define DPGIN     dpgin             /* inches per page unit on device */

static qp_init() {
    static int once;
    int i;

    if(once)return;
    for(i=0; i<9; i++) if(!settrans[i]) trans[i] = USYM;
    once++;
}

gp_init() { static int three = 3; float d[2]; gptran_(&three, d); }

#endif
/*------------------------  I N C L U D E  ------------------------*/
#ifndef CTRANS

#include "gptran.h"

#endif
/*-------------------------------------------------------------------*/

/* CTRANS or transformation constants. Set to USYM if program is to use
     defaults which are given below in parentheses.
 0       mid-longitude of plot or central meridian          ( (DS[3]+DS[2])*0.5 )
 1       mid-latitude of plot                               ( (DS[1]+DS[0])*0.5 )
 2       lower standard parallel                            (1/6th way up map)
 3       upper standard parallel                            (5/6th way up map)
 4       major axis of earth, equatorial radius in meters(6378388.)
 5       minor axis of earth, polar radius in meters        (6356911.9462)
 6       conic projections, =1.0 for north polar plot, =-1.0 for south(1.0)
 7       perspective projection, ratio of elevation of viewer above
          earth to radius of earth                          (1.0)
 8       earth's radius in nautical miles                   (3437.9768)
*/

#define ITRANS   (*tr[NTRANS].init)
#define TRANS    (*tr[NTRANS].tran)

extern int user(),iuser(),linlin(),ilinlg(),linlog(),linln(),
    loglin(),loglog(),logln(),lnlin(),lnlog(),lnln(),polar(),
    ipolar(),npolar(),wulff(),schmid(),mercat(),imerca(),miller(),
    transv(),utm(),iutm(),areaeq(),iareae(),disteq(),
    gnomon(),ignomo(),orthog(),perspe(),
    stereo(),lamber(),ilambe(),ptolem(),iptole(),
    ikavra(),albers(),ialber(),polyco(),sinuso(),
    newpl() ,none();

struct func {
    int       tnum;
    char     *tname;
    int (*tran)();
    int (*init)();
};

struct func tr[]={
```

```
 0,"user",                user,    iuser,    /* user defined transformation          */
 1,"linlin",              linlin,  ilinlg,   /* linear x vs linear y                 */
 2,"linlog",              linlog,  ilinlg,   /* linear x vs log y                    */
 3,"linln",               linln,   ilinlg,   /* linear x vs ln y                     */
 4,"loglin",              loglin,  ilinlg,   /* log x    vs linear y                 */
 5,"loglog",              loglog,  ilinlg,   /* log x    vs log y                    */
 6,"logln",               logln,   ilinlg,   /* log x    vs ln y                     */
 7,"lnlin",               lnlin,   ilinlg,   /* ln x     vs linear y                 */
 8,"lnlog",               lnlog,   ilinlg,   /* ln x     vs log y                    */
 9,"lnln",                lnln,    ilinlg,   /* ln x     vs ln y                     */
10,"polar",               polar,   ipolar,   /* polar cc from east, x deg, y dist */
11,"npolar",              npolar,  ipolar,   /* polar from north,   x deg, y dist */
12,"wulff",               wulff,   ipolar,   /* wulff  equal angle net,x azm,y dip*/
13,"schmidt",             schmid,  ipolar,   /* schmidt equal area net,x azm,y dip*/
14,"mercator",            mercat,  imerca,   /* mercator cylindrical               */
15,"miller",              miller,  imerca,   /* miller   cylindrical               */
16,"transverse",          transv,  imerca,   /* transverse mercator cylindrical    */
17,"utm",                 utm,     iutm,     /* universal transverse mercator      */
18,"areaequal",           areaeq,  iareae,   /* equal area azimuthal               */
19,"distequal",           disteq,  iareae,   /* equal distance azimuthal           */
20,"gnomonic",            gnomon,  ignomo,   /* gnomonic azimuthal                 */
21,"orthographic",        orthog,  iareae,   /* orthographic azimuthal             */
22,"perspective",         perspe,  iareae,   /* perspective azimuthal              */
23,"stereographic",       stereo,  iareae,   /* stereographic azimuthal            */
24,"lambert",             lamber,  ilambe,   /* lambert conformal conic            */
25,"ptolemy",             ptolem,  iptole,   /* ptolemy equal interval conic       */
26,"kavraisky",           ptolem,  ikavra,   /* kavraisky equal interval conic     */
27,"albers",              albers,  ialber,   /* albers equal area conic            */
28,"polyconic",           polyco,  iutm,     /* polyconic                          */
29,"sinusoidal",          sinuso,  imerca,   /* sinusoidal                         */
30,"newpole",             newpl,   none,    /* rotate coordinates about a pole[2]*/
};

nameit_() {
    INITCOM;
    printf("Change to %s",tr[NTRANS].tname);
    if(NTRANS>13 && NTRANS<30)printf(" map");
    printf(" transformation.\n");
}
#define MAXTRAN sizeof(tr)/sizeof(struct func)

#define TOLERANCE 0.00001          /* inverse trans: iterate til change
                                          in data < this */
#define STEP      0.25             /* inverse trans: proportion of subjsp to
                                          move when point outside */

#define MAJOR     6378388.         /* Major axis of spheroidal earth in meters */
#define MINOR     6356911.9462     /* Minor axis of spheroidal earth in meters */
#define ERADIUS   3437.9768        /* Radius of earth in nautical miles        */
#define PI        3.141592654      /* pi */
#define PI2       1.570796327      /* pi/2 */
#define PI4       0.785398164      /* pi/4 */
#define RA        0.017453293      /* PI radians / 180.0 degrees */
#define INCHPKM   39370.07874      /* Inches per kilometer */

#define RI        register i
#define FORBOTH   for(i=0;i<2;i++)

#define I         (a,b,c) double a[],b[],c[];
#define T         (p) double p[];
#define X         p[0]
#define Y         p[1]
#define X0        cent[0]
#define Y0        cent[1]
```

```
#define LINEAR 0
#define POLAR  1
#define MAPS   2

extern double limit(),mscale(),noneg(),sign();

/* transformation constants */
static double cent[2],centp[2],ma,mb,ecc,stanpar[2],spole,elev,er;

/* scratch variables for any transformation */
static double ta,tb,tc,td,te,tf,tg,th,ti,tj;

static double fram[4];

iuser  I{return; };
user   T{return; };
ilinlg I{
    RI;
    mscale(LINEAR,a,b,c);
    a[0]=SS[0];
    a[1]=SS[2];
    FORBOTH {
        b[i]=DS[2*i];
        c[i]=DS[2*i+1];
    }
    TRANS(b);
    TRANS(c);
    FORBOTH c[i]= (SS[2*i+1]-SS[2*i])/(c[i]-b[i]);
}
linlin T{                                          };
linlog T{                          Y=log10(noneg(Y)); };
linln  T{                          Y=log  (noneg(Y)); };
loglin T{ X=log10(noneg(X));                          };
loglog T{ X=log10(noneg(X)); Y=log10(noneg(Y)); };
logln  T{ X=log10(noneg(X)); Y=log  (noneg(Y)); };
lnlin  T{ X=log  (noneg(X));                          };
lnlog  T{ X=log  (noneg(X)); Y=log10(noneg(Y)); };
lnln   T{ X=log  (noneg(X)); Y=log  (noneg(Y)); };

ipolar I{ mscale(POLAR,a,b,c); }
polar  T{ X=Y*cos(ta=RA*X); Y=Y*sin(ta); };
npolar T{ X=90.0-X; polar(p); };
wulff  T{ Y=limit(Y,0.,90.); Y=tan(RA*(45.0-Y*.5)); npolar(p); };
schmid T{ Y=limit(Y,0.,90.); Y=sin(RA*(45.0-Y*.5)); npolar(p); };

imerca I{mapcom(); mscale(MAPS,a,b,c); };
mercat T{
    X=(X-X0)*RA;
    Y=limit(Y,-89.9,89.9);
    ta=RA*fabs(Y);
    tb=ecc*sin(ta);
    Y=sign(Y)*(log(tan(PI4+.5*ta))+.5*ecc*log((1.0-tb)/(1.0+tb)));
}
miller T{
    X=(X-X0)*RA;
    Y=sign(Y)*1.25*log(tan(PI4+0.4*fabs(Y*RA)));
}
transv T{
    ta=(X-X0)*RA;
    tb=Y*RA;
    tc=cos(tb)*sin(ta);
    X=.5*log((1.0+tc)/(1.0-tc));
    Y=atan2(tan(tb),cos(fabs(ta)));
```

```
}

static double e2,ep2,ap,bp,cp,dp,ep,zkp,xX0;

iutm    I{
    RI;
    double an,an2,an3,an4,an5;
    mapcom();
    i=fabs(X0)/6.; xX0=sign(X0)*(6*i+3);
    ep2=(ma*ma-mb*mb)/(mb*mb);
    e2 =ecc*ecc;
    an =(ma-mb)/(ma+mb);
    an2=an*an;
    an3=an*an2;
    an4=an*an3;
    an5=an*an4;
    ap =ma*(1.-an+1.25*(an2-an3)+1.265625*(an4-an5));
    bp =1.5*ma*(an-an2+0.875*(an3-an4)+0.859375*an5);
    cp =0.9375*ma*(an2-an3+0.75*(an4-an5));
    dp =0.72916666667*ma*(an3-an4+0.6875*an5);
    ep =0.615234375*ma*(an4-an5);
    zkp=0.9996;
    FORBOTH centp[i]=0.;
    fram[0]=0.; fram[1]=cent[0];
    TRANS(fram);
    FORBOTH centp[i]=fram[i];
    mscale(MAPS,a,b,c);
};

utm     T{
    double siny,cosy,tany,siny2,cosy2,tany2,cosy5,ta2;

    ta=fabs((xX0-X)*RA); ta2=ta*ta;
    tb=fabs(Y*RA);
    siny=sin(tb); siny2=siny*siny;
    cosy=cos(tb); cosy2=cosy*cosy; cosy5=cosy*cosy2*cosy2;
    tany=tan(tb); tany2=tany*tany;
    tc=zkp*(ap*tb-bp*sin(2.*tb)+cp*sin(4.*tb)-dp*sin(6.*tb)+ep*sin(8.*tb));
    td=ma/sqrt(1.-e2*siny2);
    te=.5*td*siny*cosy*zkp;
    tf=td*siny*cosy2*cosy*(5.-tany2+9.*ep2*cosy2+4.*ep2*ep2*cosy2*cosy2)*zkp/24.;
    tg=td*cosy*zkp;
    th=(td/6.)*cosy2*cosy*(1.-tany2+ep2*cosy2)*zkp;
    ti=pow(ta,6.)*0.0013888889*td*siny*cosy5*(61.-58.*tany2+tany2*tany2+
                270.*ep2*cosy2-330.*ep2*siny2)*zkp;
    tj=pow(ta,5.)*0.0083333333*td*cosy5*(5.-18.*tany2+tany2*tany2+
                14.*ep2*cosy2-58.*ep2*siny2)*zkp;
    X=sign(X-xX0)*(tg*ta+th*ta2*ta+tj)-centp[0];
    Y=sign(Y)*(tc+te*ta2+tf*ta2*ta2+ti)-centp[1];
}

iareae I{
    mapcom();
    ti=PI2-Y0*RA;
    ta=cos(ti);
    tb=sin(ti);
    tj=acos(1.0/elev);
    mscale(MAPS,a,b,c);
}
azmcon T{
    if(X==X0 && Y==Y0) {te=tg=0.0; return;}
    X=(X-X0)*RA;
    th=fabs(X);
    tc=PI2-Y*RA;
```

```
    td=cos(tc);
    te=ta*td+tb*sin(tc)*cos(fabs(X));
    if(te>1.)te=1.; if(te<-1.)te= -1.; te=acos(te);
    if(th<0.00001){tg=tc>ti?PI:0.0; return;}
    if(fabs(ti-PI)<0.00003){tg=X; return;}
    if(sin(te)==0.) tg=PI-th;
    else {
        tg=(td-cos(te)*ta)/(sin(te)*tb);
        if(tg>1.)tg=1.; if(tg<-1.)tg= -1.; tg=acos(tg);
    }
    tg= X<0 ? 2.*PI-tg : tg;
}
areaeq T{
    azmcon(p);
    tf=2.*sin(0.5*te);
    X=tf*sin(tg);
    Y=tf*cos(tg);
}
disteq T{
    azmcon(p);
    X=te*sin(tg);
    Y=te*cos(tg);
}
orthog T{
    azmcon(p);
    tf=sin(limit( te,-PI2,PI2));
    X=tf*sin(tg);
    Y=tf*cos(tg);
}
perspe T{
    azmcon(p);
    if(te>tj)te=tj;
    th=er*sqrt(1.0+elev*elev-2.*elev*cos(te));
    tc=er*sin(te);
    th=atan(tc/sqrt(th*th-tc*tc));
    th=tan(th)*er*(elev+1.);
    X=th*sin(tg);
    Y=th*cos(tg);
}

ignomo I{
    mapcom();
    ta=cos(Y0*RA);
    tb=sin(Y0*RA);
    mscale(MAPS,a,b,c);
}
gnomon T{
    tc=cos(Y*RA);
    td=sin(Y*RA);
    te=(X-X0)*RA;
    tf=cos(te);
    tg=td*tb+tc*ta*tf;
    X=tc*sin(te)/tg;
    Y=(td*ta-tb*tc*tf)/tg;
}

stereo T{
    tc=Y*RA;
    td=(X-X0)*RA;
    te=ecc*sin(fabs(tc));
    tf=pow((1.-te)/(1.+te),.5*ecc);
    tg=1.0+sin(tc)*tb+cos(tc)*ta*cos(td);
    X=tf*cos(tc)*sin(td)/tg;
    Y=tf*(sin(tc)*ta-tb*cos(tc)*cos(td))/tg;
```

```
}
double eccsin(a) double a; { a=a*RA;
    return(pow((1.+ecc*sin(a))/(1.-ecc*sin(a)),.5*ecc)/tan(PI4+spole*a*.5));
}
ilambe I{
    mapcom();
    ta=eccsin(fabs(X0));
    tb=eccsin(fabs(stanpar[0]));
    tc=eccsin(fabs(stanpar[1]));
    td=ecc*sin(stanpar[0]*RA);
    td=er/sqrt(1.-td*td);
    te=ecc*sin(stanpar[1]*RA);
    te=er/sqrt(1.-te*te);
    tf=cos(stanpar[0]*RA);
    tg=cos(stanpar[1]*RA);
    th=fabs((log(tf)-log(tg)+log(td)-log(te))/(log(tb)-log(tc)));
    ti=.5*(td*tf/(th*pow(tb,th))+te*tg/(th*pow(tc,th)));
    ta=ti*pow(ta,th);
    mscale(MAPS,a,b,c);
}
lamber T{
    tb=eccsin(Y);
    tc=fabs(ti*pow(tb,th));
    td=RA*fabs(X-X0);
    if(td>PI)td= -(2.*PI-td)*sign(td);
    te=tc*sin(td*th);
    if(spole<0) {if(X>X0)te= -te; }
    else if(X<X0)te= -te;
    X=te;
    Y=ta-tc*cos(th*td);
}

iptole I{
    mapcom();
    ta=Y0*RA;
    th=sin(ta);
    ti=er*((1./tan(ta))+ta);
    ta=ti-er*Y0*RA;
    mscale(MAPS,a,b,c);
}
ptolem T{
    tc=ti-er*Y*RA;
    td=th*(X-X0)*RA;
    X=tc*sin(td);
    Y=ta-tc*cos(td);
}

ikavra I{
    mapcom();
    ta=stanpar[0]*RA;
    tb=stanpar[1]*RA;
    th=(cos(ta)-cos(tb))/(tb-ta);
    ti=.5*er*(cos(ta)/th+ta+cos(tb)/th+tb);
    ta=ti-er*cent[1]*RA;
    mscale(MAPS,a,b,c);
}

double series(a) double a; {
    a=a*a;
    return(1.+2.*a/3.+3.*a*a/5.+4.*a*a*a/7.);
}
double radcur(a) double a; {
    return(er*er*cos(a)*cos(a)/(1.-ecc*ecc*sin(a)*sin(a)));
}
```

```
ialber I{
    mapcom();
    ta=fabs(stanpar[0]*RA);
    tb=fabs(stanpar[1]*RA);
    tc=series(ecc);
    td=sin(ta);
    td=td*series(ecc*td)/tc;
    te=sin(tb);
    te=te*series(ecc*te)/tc;
    tf=1.-ecc*ecc;
    tf=er*er*tf*(.5/tf+0.25*log((1.+ecc)/(1.-ecc))/ecc);
    tg=(radcur(ta)-radcur(tb))*.5/(tf*(te-td));
    th=radcur(ta)/(tg*tg);
    ti=sin(fabs(cent[1]*RA));
    ti=ti*series(ti*ecc)/tc;
    te=sqrt(th+2.*tf*(td-ti)/tg);
    mscale(MAPS,a,b,c);
}
albers T{
    ta=sin(fabs(Y*RA));
    if((stanpar[1]>=0. && Y<0.) || (stanpar[1]<0. && Y>0.)) ta= -ta;
    ta=ta*series(ta*ecc)/tc;
    tb=sqrt(th+2.*tf*(td-ta)/tg);
    tj=(X-X0)*RA;
    if(fabs(tj)>PI)  tj=sign(tj)*(2.*PI-fabs(tj));
    tj=tg*fabs(tj);
    ta=tb*sin(tj);
    if(X<X0)  X=(stanpar[0]>=0.)?-ta:ta;
    else      X=(stanpar[0] <0.)?-ta:ta;
    Y=te-tb*cos(tj);
}
polyco T{
    ta=fabs((xX0-X)*RA);
    tb=fabs(Y*RA);
    tc=ap*tb-bp*sin(2.*tb)+cp*sin(4.*tb)-dp*sin(6.*tb)+ep*sin(8.*tb);
    td=ma/sqrt(1.-e2*sin(tb)*sin(tb));
    if(tb==0.0) { th=0.; tg=ta*ma; }
    else {
        te=ta*sin(tb); tf=td*cos(tb)/sin(tb);
        th=(tc+2.*tf*sin(.5*te)*sin(.5*te)); tg=tf*sin(te);
    }
    X=(xX0<X)? tg-centp[0]:-tg-centp[0]; X=X/100000.;
    Y=( Y<0)?-th-centp[1]: th-centp[1]; Y=Y/100000.;
};
sinuso T{X=(X-X0)*cos(Y*RA);};

newpl T { if(POLE[0]!=USYM && POLE[1]!=USYM) newpol_(&X,&Y);}
none  I {double r,s; a[0]=a[1]=b[0]=b[1]=0.0; c[0]=c[1]=1.0;
            r=POLE[0]*RA; s=POLE[1]*RA; refpt(r,s); }

double limit(a,b,c) double a,b,c; { if(a<b)a=b; if(a>c)a=c; return(a);}
double noneg(a) double a; {
    if(a>0) return(a);
    ERROR=2;
    gperr_(&ERROR,"gptran","Can not take log of number <= 0. trans=", &NTRANS);
    return(0.1);
}
double sign(a) double a; {return(a<0?-1.0:1.0);}

static int initialized,nonlinear;

minmax(ta,tb)
    float ta,tb;
{
```

```
    register i;
    double tp[2];

    if(ERROR!=0)return;
    tp[0]=ta; tp[1]=tb; TRANS(tp);
    if(!initialized) {
        fram[0]=fram[1]=tp[0];
        fram[2]=fram[3]=tp[1];
        initialized=1;
    }
    else {
        FORBOTH {
            if(tp[i] < fram[2*i  ]) fram[2*i  ]=tp[i];
            if(tp[i] > fram[2*i+1]) fram[2*i+1]=tp[i];
        }
    }
    if(DEBUG>=6)printf("minmax: %f %f %f %f fram=%f %f %f %f\n",
        ta,tb,tp[0],tp[1],fram[0],fram[1],fram[2],fram[3]);
}

mapcom(){
    register i;

    FORBOTH {
        cent[i] = (CTRANS[i]==USYM) ?
            (DS[2*i+1]+DS[2*i])*0.5 : CTRANS[i];
        stanpar[i]=(CTRANS[i+2]==USYM) ?
            (i==0 ? 1.0 : 5.0)*(DS[3]-DS[2])/6.0 +DS[2] : CTRANS[i+3];
    }
    ma    =(CTRANS[4]==USYM) ? MAJOR : CTRANS[2];
    mb    =(CTRANS[5]==USYM) ? MINOR : CTRANS[2];
    spole=(CTRANS[6]==USYM) ? 1.0 : -1.0;
    elev =(CTRANS[7]==USYM) ? 1.0 : CTRANS[7]; elev+=1.0;
    er    =(CTRANS[8]==USYM) ? ERADIUS : CTRANS[8];
    ecc   =sqrt((ma*ma-mb*mb)/(ma*ma));
}

/* find mapscale of this map */
static int FORW=1;
extern double earthdist();
float mpscale()
{
    float mp[4];
    double sca,scal;

    mp[0]=DS[0]; mp[1]=mp[3]=Y0; mp[2]=DS[1];
    if((mp[2]-mp[0])>180.) {
        mp[0]=(mp[2]+mp[0])*0.5-90.0; mp[2]=mp[0]+180.;
    }
    sca = INCHPKM * earthdist((double)mp[0],Y0,(double)mp[2],Y0);
    gptran_(&FORW,&mp[0]); gptran_(&FORW,&mp[2]);
    scal= DPGIN*hypot((double)(mp[2]-mp[0]),(double)(mp[3]-mp[1]));
    return(sca/scal);
}

/* find extreme page units for this transformation and dataspace */
int s[]={0,0,2,2,0,0,3,3,1,1,2,2,1,1,3,3,0,0,2,3,0,1,3,3,1,1,2,3,0,1,2,2};
static int mapdigc=0;
static int niter;
#define MAXNITER          30

pshrink(n)
    float n;
{
```

```
    float dx,dy;

    dx=(SS[1]-SS[0])*(1.0-n)*0.5;
    dy=(SS[3]-SS[2])*(1.0-n)*0.5;
    GS[0]=SS[0]=SS[0]+dx;
    GS[1]=SS[1]=SS[1]-dx;
    GS[2]=SS[2]=SS[2]+dy;
    GS[3]=SS[3]=SS[3]-dy;
    if(DEBUG>=5)printf("pshrink: ratio=%f subjsp= %f %f %f %f\n",n,SS[0],SS[1],SS[2],SS[3]);
}

dshrink(n)
    float n;
{
    float dx,dy;

    dx=(DS[1]-DS[0])*(1.0-n)*0.5;
    dy=(DS[3]-DS[2])*(1.0-n)*0.5;
    DS[0]=DS[0]+dx;
    DS[1]=DS[1]-dx;
    DS[2]=DS[2]+dy;
    DS[3]=DS[3]-dy;
    if(DEBUG>=5)printf("pshrink: ratio=%f datasp= %f %f %f %f\n",n,DS[0],DS[1],DS[2],DS[3]);
}

double mscale(type,a,b,c)
    int type;
    double a[],b[],c[];
{
    register i;
    double tta,ttb;

    initialized=0;
    switch (type) {
        case LINEAR: for(i=0;i<16;i+=4) minmax(DS[s[i]],DS[s[i+2]]);
                     break;
        case POLAR:  for(i=2;i<4;i++) {
                         minmax(DS[0],DS[i]);
                         tta=90.0*floor((double)DS[0]/90.0);
                         while( (tta+=90.0) < DS[1]) minmax(tta,DS[i]);
                         minmax(DS[1],DS[i]);
                     }
                     break;
        case MAPS:   for(i=0;i<32;i+=4) minmax((DS[s[i]]+DS[s[i+1]])*.5,
                         (DS[s[i+2]]+DS[s[i+3]])*.5 );
                     if(cent[0]>DS[0] && cent[0]<DS[1])
                         minmax(cent[0],DS[2]); minmax(cent[0],DS[3]);
                     break;
        default:    ERROR=2;
                     gperr_(&ERROR,"mscale","Illegal scale type",&type);
                     return;
    }
    if(fram[0]==fram[1] || fram[2]==fram[3]) {
        ERROR=2;
        gperr_(&ERROR,"mscale","Divide by zero in scaling type",&type);
        return(1.0);
    }
    if(mapdigc) {
        for(i=0;i<4;i++) SS[i]=fram[i];
        mapdigc=0;
    }
    MAPSCA=0.0;
    if(type==LINEAR) return;
    nonlinear=1;
```

```
    tta= (SS[1]-SS[0]) / (fram[1]-fram[0]);
    ttb= (SS[3]-SS[2]) / (fram[3]-fram[2]);
    tta=ttb<tta ? ttb:tta;
    FORBOTH {
        a[i]=0.5*(SS[2*i]+SS[2*i+1])-0.5*tta*(fram[2*i]+fram[2*i+1]);
        b[i]=0.0;
        c[i]=tta;
    }
    if(DEBUG >= 5) printf("mscale: fram=%f %f %f %f scale=%f\n",
        fram[0],fram[1],fram[2],fram[3],tta);
    if(type != MAPS) return;
    MAPSCA=mpscale();
    if(DEBUG >= 5)printf("mscale: mapscale=%f\n",MAPSCA);
    if(SETSCA==0.0 || SETSCA==MAPSCA)return;
    if(MAPSCA<SETSCA)pshrink(MAPSCA/SETSCA);
    else dshrink(SETSCA/MAPSCA);
    if(SETSCA!=MAPSCA && niter++ <= MAXNITER) mscale(MAPS,a,b,c);
    SETSCA=0.0;
}

double c0[2],c1[2],c2[2];              /* Transformation constants */

gptran_(forward,point)
    int *forward; /* forward = 1 forward, 2 inverse, 3 initialize, 4 initialize
                     subject space used in mapdig.c   */
    float point[2];
{
    register i;
    int j,k;
    double tmp[2],low[2],mid[2],hig[2],lowfcn[2],midfcn[2];
    double pointt[2];

    INITCOM;

    if(DEBUG >= 5) printf("gptran: %d %f %f ktrflg=%d ktran=%d \n",
        *forward,point[0],point[1],TRANFL,NTRANS);
    if(ERROR!=0)return;

    /* initialize transformations */
    if(TRANFL || *forward >= 3) {
        if(NTRANS<0 || NTRANS>=MAXTRAN) {
            ERROR=2;
            gperr_(&ERROR,"gptran","Illegal transformation of", &NTRANS);
            return;
        }
        mapdigc=(*forward<=3) ? 0:1;
        for(i=j=0;i<13;i+=6) {
            if(PS[i+1] < PS[i  ]) j++;
            if(PS[i+3] < PS[i+2]) j++;
        }
        if (j) {
            ERROR=2;
            gperr_(&ERROR,"gptran",
                "Plot space NOT initialized properly for this many elements",&j);
            return;
        }
        nonlinear=0;
        niter=0;
        TRANFL=0;
        ITRANS(c0,c1,c2);
        if(DEBUG > 5) printf("gptran: c0=%f %f c1=%f %f c2=%f %f\n",
            c0[0],c0[1],c1[0],c1[1],c2[0],c2[1]);
        if(nonlinear) {
            LINSPA=1;
```

```
        FORBOTH SPALIN[i]=CURINC*(DS[2*i+1]-DS[2*i])/(SS[2*i+1]-SS[2*i]);
    }
    else LINSPA=0;
}

/* forward transformations */
if(*forward==1) {
    FORBOTH pointt[i]=point[i];
    TRANS(pointt);
    FORBOTH point[i]=c0[i]+(pointt[i]-c1[i])*c2[i];
    return;
}

/* inverse transformations */
if(*forward==2) {
    FORBOTH {
        pointt[i]=((point[i]-c0[i])/c2[i])+c1[i];
        low[i]=DS[2*i]; hig[i]=DS[2*i+1];
    }
    if(DEBUG > 5) printf("gptran: pointt=%f %f c2=%f %f\n",pointt[0],pointt[1],c2[0],c2[1]);
    k=0;
    do {
        j=0; k++;
        FORBOTH { tmp[i]=low[i]; mid[i]=hig[i]; }
        TRANS(tmp); TRANS(mid);
        if(DEBUG > 5)
            printf("gptran: low=%f %f hig=%f %f\n  tmp pnt mid X[%f %f %f] Y[%f %f %f]\n",
            low[0],low[1],hig[0],hig[1],tmp[0],pointt[0],mid[0],tmp[1],pointt[1],mid[1]);
        FORBOTH {
            if(mid[i]>tmp[i]) {
                if(pointt[i]<tmp[i]) low[i]-=(hig[i]-low[i])*STEP; else j++;
                if(pointt[i]>mid[i]) hig[i]+=(hig[i]-low[i])*STEP; else j++;
            } else {
                if(pointt[i]>tmp[i]) low[i]-=(hig[i]-low[i])*STEP; else j++;
                if(pointt[i]<mid[i]) hig[i]+=(hig[i]-low[i])*STEP; else j++;
            }
        }
        if(k>50) {
            ERROR=2;
            gperr_(&ERROR,"gptran","Unable to get inverse of transformation ",&NTRANS);
            return;
        }
    } while(j<4);
    FORBOTH lowfcn[i]=low[i];
    TRANS(lowfcn);
    FORBOTH lowfcn[i]-=pointt[i];
    do { j=0;
        FORBOTH {
            if(fabs(hig[i]-low[i])<TOLERANCE) j++;
            else mid[i]=(hig[i]+low[i])*.5;
        }
        FORBOTH midfcn[i]=mid[i];
        TRANS(midfcn);
        FORBOTH {
            midfcn[i]-=pointt[i];
            if(lowfcn[i]*midfcn[i] > 0.0) {
                low[i]=mid[i];
                lowfcn[i]=midfcn[i];
            }
            else hig[i]=mid[i];
        }
    if(DEBUG > 5)
        printf("gptran: low=%f %f mid=%f %f hig=%f %f\n          lowf=%f %f midf=%f %f\n",
            low[0],low[1],mid[0],mid[1],hig[0],hig[1],lowfcn[0],lowfcn[1],
```

```
            midfcn[0],midfcn[1]);
        } while (j<2);
        FORBOTH point[i]=(hig[i]+low[i])*.5;
    }
}
```

```
/*---VPLOT-----formats plot data for versatek----------------------

Programmer: Barbara Bekins, February, 1980


These routines create an intermediate file in the directory /usr/tmp
called 'raster.suffix' where suffix is the id of the geoplot process.
    X,Y pairs are mapped into bits in this file by 'line' and 'point'.
This file is in turn converted into rows of rasterized data by the
routine putpic which outputs to a versatec, printronix or florida data
printer according to the plot device requested in geoplot.

-------------------------------------------------------------------*/
#include "stdio.h"
#include "gpcomc.h"
#include "gpbufc.h"
#define   NB        32
#define BSIZ        512
#define BLKNO(x,y) (((x&03700)>>6) + ((y&03700)>>1))
#define BITNO(x,y) (((y&077)<<3)+((x>>3)&07))

char      blocks   [NB][BSIZ];
struct    buf {
    int   bno;
    char *block;
};
struct    buf       bufs[NB];
int       in, out;                              /* plot file number */
char *picture="/usr/tmp/raster.XXXXXX"; /* plot file name */
int bnomax;                                     /*max block # in file */
int linmod= -1;                                 /* mask to create line pattern in line */
char *bf;  /* point to bufcom array of write flags: one bit per buffer */
char cmd1[50]="vpr -b -pg ";
char cmd2[50]="rm ";
char *devices="vlfu";

/* Open plot file and initialize io buffer */
begin_()
{
   register i,j;
   extern gpcom_;
   extern gpbuff_;

   /* set address of flag buffer from gpbuf.h */
   /* assign plot file unique name for this session */
   common= &gpcom_;
   bufcom= &gpbuff_;
   bf= bufcom->bfl;
   if(!out) cmd1[0]=devices[ common->numdev[0]-21];
   bnomax=0;
   for(i=16;i<22;i++) picture[i]='X';
   mktemp(picture);
   /* initialize buffers */
   for(i=0;i<NB;i++){
   for(j=0;j<BSIZ;j++) blocks[i][j]=0;
       bufs[i].bno= -1;
       bufs[i].block=blocks[i];
   }
   for(i=0;i<4096;i++) bf[i]=0;

   in=0;
tryagain:
   if(common->idbug>9) printf("creating %s\n",picture);
   out=creat(picture,0666);
```

```
    if(out==-1){
        in++;
        if(in!=3) goto tryagain;
        else{printf("cannot create %s\n",picture);return;}
    }
    close(out);
    in=out=open(picture,2);
}

/* End plot: output flags and blocks still in core and close file */

end_(mode)
int *mode;
{
    register i;
    int error;

    if(!out) {printf("error no plotfile\n");return;} /* if plotfile not open, return */
    /*  If error in geoplot restart the plotfile */
    if(common->ibreak){
        strcpy(&cmd2[3],picture);
        error=system(cmd2); /* remove plotfile */
        begin_();
        return;
    }
    /* write out blocks in core */
    for(i=0;i<NB;i++)
    if(bufs[i].bno!=-1) {
        zseek(out,bufs[i].bno);
        write(out,bufs[i].block,BSIZ);
    }

    /* set length of file to be a multiple of 32 blocks  */
    zseek(out,(bnomax/32+(bnomax%32>0))*32);
    write(out,cmd1,1);
    close(in);
    close(out);
    /* output plotfile */
    strcpy(&cmd1[11],picture);
    error=system(cmd1);
    if(error) printf("%s :error %d returned\n",cmd1,error);
    strcpy(&cmd2[3],picture);
    if(common->idbug>0)printf("vplot_:%s=cmd1 %s=cmd2\n",cmd1,cmd2);
    error=system(cmd2);
    if(error) printf("%s: error %d returned\n",cmd2,error);
}

/* Plot a line from (x0,y0) to (x1,y1) in the scratch file*/

line_(x0, y0, x1, y1)
register int *x0,*y0,*x1,*y1;
{
    int dx, dy;
    int xinc, yinc;
    register res1;
    int res2;
    int slope;

    xinc = 1;
    yinc = 1;
    if ((dx = *x1-*x0) < 0) {
        xinc = -1;
        dx = -dx;
    }
```

```
    if ((dy = *y1-*y0) < 0) {
        yinc = -1;
        dy = -dy;
    }
    slope = xinc*yinc;
    res1 = 0;
    res2 = 0;
    if (dx >= dy) while (*x0 != *x1) {
        if((*x0+slope*(*y0))&linmod)
            if (BLKNO(*x0,*y0) == bufs[0].bno)
                bufs[0].block[BITNO(*x0,*y0)] |= 1 << (7-(*x0&07));
            else
                ppoint_(x0, y0);
        if (res1 > res2) {
            res2 += dx - res1;
            res1 = 0;
            *y0 += yinc;
        }
        res1 += dy;
        *x0 += xinc;
    }
    else while (*y0 != *y1) {
        if((*x0+slope*(*y0))&linmod)
            if (BLKNO(*x0,*y0) == bufs[0].bno)
                bufs[0].block[BITNO(*x0,*y0)] |= 1 << (7-(*x0&07));
            else
                ppoint_(x0, y0);
        if (res1 > res2) {
            res2 += dy - res1;
            res1 = 0;
            *x0 += xinc;
        }
        res1 += dx;
        *y0 += yinc;
    }
    if((*x1+slope*(*y1))&linmod)
        if (BLKNO(*x1,*y1) == bufs[0].bno)
            bufs[0].block[BITNO(*x1,*y1)] |= 1 << (7-(*x1&07));
        else
            ppoint_(x1, y1);
}

/* Plot a point at (x,y) in the scratch file */

ppoint_(x, y)
register  *x, *y;
{
    register bno;
    bno = BLKNO(*x,*y);
    if (bno != bufs[0].bno) {
        getblk(bno);
    }
    bufs[0].block[BITNO(*x,*y)] |= 1 << (7-(*x&07));
    if(common->idbug>1) printf("getbit # %d\n",BITNO(*x,*y));
}

/* Put block b of the scratch file into the first block of the io buffer */

getblk(b)
register b;
{
    register i;
    register struct buf *bp1, *bp2;
    register char *tp;
```

```
    if(b>bnomax) bnomax=b; /* set bnomax to max block # plotted */
    if(common->idbug>1) printf("getblk # %d\n",b);
loop:
    for (bp1 = bufs; bp1 < &bufs[NB]; bp1++) {
        if (bp1->bno == b || bp1->bno == -1) {
            tp = bp1->block;
            for (bp2 = bp1; bp2>bufs; --bp2) {
                bp2->bno = (bp2-1)->bno;
                bp2->block = (bp2-1)->block;
            }
            bufs[0].bno = b;
            bufs[0].block = tp;
            bf[b/8]|= 1<<(7-(b&07));   /* set block written flag */
            return;
        }
    }
    zseek(out, bufs[NB-1].bno);
    write(out, bufs[NB-1].block, BSIZ);
        /* check if block has been written in this plot or not */
        if(bf[b/8]>>(7-(b&07))&1){
            zseek(in, b);
            read(in, bufs[NB-1].block, BSIZ);}
        else{
            for(i=0;i<512;i++) bufs[NB-1].block[i]=0;}
        bufs[NB-1].bno = b;
    goto loop;
}

/* Set file pointer in file #'a' to 'b'th block after 2 bytes count */


zseek(a, b)
int a,b;
{
    long fptr;
    fptr= ((long)b *512);
    return(lseek(a, fptr, 0));
}

/* Set the bit mask to be used in 'line' */

int modes[5]        {-1,014,054,034,074};

linset_(lintyp)
long *lintyp;
{
    linmod=modes[*lintyp];
}

char *qwcfil="/usr/tmp/quickXXXXXX";
extern gpcom_;
extern char *devices;
int qfd;

int opnqwc_()
/* previous plot must have been ended or it will be lost */
{
    common= &gpcom_;
    if(qfd) close(qfd);
    if(qwcfil[19]=='X') mktemp(qwcfil);
    if(common->idbug>9) printf("opnqwc: open %s\n",qwcfil);
    qfd=creat(qwcfil,0666);
    return(qfd);
```

```
}

putqwc_(mode,point)
int *mode; float point[2];
{
    struct {
        int mod;
        float poi[2];
    } put;
    put.mod= *mode;
    put.poi[0]=point[0];
    put.poi[1]=point[1];
    if(common->idbug>9) printf("putqwc: %d %f %f\n",put.mod,put.poi[0],
            put.poi[1]);
    write(qfd,&put,sizeof put);
}

endqwc_(nd)
int *nd;
{
int err;
    close(qfd);
    cmd1[0]=devices[*nd-21];
        cmd1[9]='q';
    strcpy(&cmd1[11],qwcfil);
    if(common->idbug>9) printf("endqwc: %s=cmd1 \n",cmd1);
    err=system(cmd1);
    if(err) printf("error on system call %s\n",cmd1);
    strcpy(&cmd2[3],qwcfil);
    err=system(cmd2);
    if(common->idbug>9) printf("endqwc: %s=system call\n",cmd2);
    if(err) printf("error on system call %s\n",cmd2);
    qfd=0;
    return;
}
```

```
c-*--GRID----DRAW AND LABEL A GRIDDED LINE TO X,Y----------------------
c
      subroutine grid(x,y,mode,datalo,datahi,itrans)
c
c-----Programmer: PLWard,USGS, Menlo Park, California 94025, January 1980
c
      integer mode,itrans,kpos,modes,linsa,ksubjs,linsav
      character*1 nostrg
      real datalo,datahi,x,y,xx,yy
      real gpfunc
      external gpfunc
c
      save
c
      include 'gpcom.h'
      include 'gpaxis.h'
      real gpfunc
      external gpfunc
c
      if(idbug.ge.3)call printn("grid  : %d %f %f label from %f to %f
     1 trans=%d\n",mode,x,y,datalo,datahi,itrans)
c
      ksubjs=ksubj
      ksubj=0
      drange(1)=datalo
      drange(2)=datahi
      ltrans=abs(itrans)
c
c-*--Check to be sure that the transformation is a legal type.----------
      if(ltrans.ge.1.and.ltrans.le.gpfunc(i0,pbegin)) goto 10
      kerror=1
      call gperr(kerror,"grid","Illegal transformation set to 1. Was",
     1 ltrans)
      ltrans=1
c
c-*--Set axis constants.-----------------------------------------------
 10   iax=2
      iaxis=1
      numax=1
c
c-*--Set data ranges.--------------------------------------------------
c
c-*--Change lintype to 1 for duration of grid.-------------------------
      linsa=lintyp(1)
      lintyp(1)=1
c
c-*--Draw a straight line and set page ranges. Set linspa if necessary
c     to force a straight line.
      pbegin(1)=pointp(1)
      pbegin(2)=pointp(2)
      linsav=linspa
      linspa=0
      modes=mode
      if(noaxis(numax).ne.1) goto 15
      if(mode.eq.-2.or.mode.eq.-4)modes=modes+1
      if(mode.eq. 2.or.mode.eq. 4)modes=modes-1
 15   call movdrw(x,y,mode)
      pend(1)   =pointp(1)
      pend(2)   =pointp(2)
c
c-*--Determine the angle of the line.----------------------------------
      call gpsinc(pbegin,pend,sinx,cosx,tdist)
      ang=atan2(sinx,cosx)*57.29578
c
```

```
c-*--Set label position
      labps=-6
      if(dislab(1).lt.0) labps=-7
c
c-*--Initialize the transformations.-----------------------------------
      c1=tdist/(gpfunc(ltrans,drange(2))-gpfunc(ltrans,drange(1)))
      c2=c1*gpfunc(ltrans,drange(1))
c
c-*--Tic and label the gridded line.-----------------------------------
      call gpaxis
      linspa=linsav
c
c-*--Print title for gridded line.-------------------------------------
      if(ibreak.eq.-1)goto 111
      xx=tdist*0.5*cosx+pbegin(1)
      yy=tdist*0.5*sinx+pbegin(2)
      if(itrans.lt.0) goto 100
c-*--spacing for label under axis
      kpos=7
      xx=xx+2.0*height*sinx
      yy=yy-2.0*height*cosx
      goto 110
c-*--spacing for label over axis
100   kpos=6
      dtic=0.0
      do 105 i=1,5
         dtic1=ticin(i)*ticfac(1)
         if(dtic1.gt.0.1)dtic1=0.0
         dtic2=-ticout(i)*ticfac(1)
         if(dtic2.gt.0.1)dtic2=0.0
         dtic=max(dtic,dtic1,dtic2)
105   continue
      dtic=dtic+0.5*height
      xx=xx-dtic*sinx
      yy=yy+dtic*cosx
110   call letter(xx,yy,1,nostrg,ang,kpos,-1)
111   lintyp(1)=linsa
      ksubj=ksubjs
      end
```

```
/* Find great cirle and rhumbline paths between two points
   Programmer  PLWard June 1983
*/
#include <stdio.h>
#include <math.h>
#define D (double)
#define RA 0.017453293
double cona,conb,conc;
int noinit;

int igtcir_(laspnt,pnt) float laspnt[],pnt[];
{
    conc=sin(D (pnt[0]-laspnt[0])*RA);
    if(conc==0.) {
        fprintf(stderr,"gtcir: cannot draw great circle between points with same longitude.");
        noinit=1;
        return(1);
    }
    noinit=0;
    cona=(tan(D pnt[1]*RA)*cos(D laspnt[0]*RA)-
          tan(D laspnt[1]*RA)*cos(D pnt[0]*RA)) /conc;
    conb=(tan(D pnt[1]*RA)*sin(D laspnt[0]*RA)-
          tan(D laspnt[1]*RA)*sin(D pnt[0]*RA)) /conc;
}

int gtcir_(pnt) float pnt[];
{
    if(noinit) return(1);
    pnt[1]=atan(cona*sin(D pnt[0]*RA)-conb*cos(D pnt[0]*RA))/RA;
}

double lntan(x) float x; {return(log(tan((45.+.5*x)*RA)));}

int irhumb_(laspnt,pnt) float laspnt[],pnt[];
{
    cona=lntan(laspnt[1]);
    conb=(lntan(pnt[1])-cona);
    if(conb==0.) {
        fprintf(stderr,"gtcir: cannot draw rhumbline between points with same latitude.");
        noinit=1;
        return(1);
    }
    noinit=0;
    conb=RA*(pnt[0]-laspnt[0])/conb;
    conc=laspnt[0];
}

int rhumb_(pnt) float pnt[];
{
    if(noinit)return(1);
    pnt[0]=conb*(lntan(pnt[1])-cona)/RA+conc;
}
```

```
/* IBAUDR fortran callable routine to return the baud rate
   of output from the computer to the current terminal.
   If in = 1 the number 0 to 15 given in tty(4) is returned.
   If in = 2 the baud rate is returned. However External A
     and External B are set to 65 and 66, the ASCII values
     of A and B.    */
#include <sgtty.h>
int rate [16] = {0,50,75,110,134.5,150,200,300,600,1200,
     1800,2400,4800,9600,65,66};
int ibaudr_(in)
   int *in;
{
   struct sgttyb sgtty; int b;
   ioctl(1,TIOCGETP,&sgtty);
   b=sgtty.sg_ospeed;
   if (*in == 2) b=rate[b];
   return(b);
}
```

```
c-*--LABEL---PLOT A LABEL ALONG AN AXIS---------------------------------
c
      subroutine label(ilab,jtext,nlen)
c
c     ilab > 0 plot label inside plot, < 0 plot label outside of plot.
c     ilab = 1,2,3,4 for lo x, hi x, lo y hi y respectively.
c
      include 'gpcom.h'
c
      integer ilab,jlab,jpos,i,j,jspace(9),nlen
      real x,y,sig,dtic1,dtic2,ang,dtic
      character*1 jtext(1)
      data jspace /2,1,4,3,0,9,10,7,8/
c
      if(idbug.ge.4)
     1  call printn("label : axis=%d length=%d\n",ilab,nlen)
      jlab=iabs(ilab)
c
c-*--Detect bad ilab and give error.-----------------------------------
      if(jlab.ge.i1.and.jlab.le.i4) goto 10
  5   kerror=1
      call gperr(kerror,"label","Illegal position of ",ilab)
      call letter(usym,usym,i1,jtext,0.0,i1,nlen)
      return
c
c-----Set position and angle.
 10   jpos=6
      if(jlab.eq.i2.or.jlab.eq.i3)jpos=7
      ang=0.0
      if(jlab.gt.i2)ang=90.0
      i=jspace(ilab+5)
c-----Set x and y.
      if(jlab.gt.i2) goto 50
          x=(pspace(7)+pspace(8))/2.0
          y=pspace(i)
          goto 60
 50   x=pspace(i)
      y=(pspace(9)+pspace(10))/2.0
 60   if(ilab.lt.0) goto 100
c-----If plotting inside plot, get tic lengths.
      jpos=-jpos
      dtic=0.0
      j=1
      if(ilab.gt.i2)j=6
      do 70 i=j,j+4
          dtic1=ticin(i)*ticfac(jlab)
          if(dtic1.gt.0.1)dtic1=0
          dtic2=-ticout(i)*ticfac(jlab)
          if(dtic2.gt.0.1)dtic2=0
          dtic=max(dtic,dtic1,dtic2)
 70       continue
      sig=1.0
      if(jlab.eq.i2.or.jlab.eq.i4)sig=-1.0
      if(jlab.le.i2)y=y+sig*dtic
      if(jlab.gt.i2)x=x+sig*dtic
100   call letter(x,y,i1,jtext,ang,jpos,nlen)
      return
      end
```

```
 0   5  15   \d subscript
 1   2  15   \u superscript
 2   1  15   \b backspace
 3   6  15   \r carriage return
 4   3  15   \l line feed down
 5   4  15   \v line feed up
 6   0   8   \1 arrow
-1   7   8
-1   6  10
-1  10   8
-1   6   6
-1   7   8
 7   5   3   \2 diamond
-1   0   8
-1   5  13
-1  10   8
-1   5   3
 8   0   5   \3 triangle
-1   5  14
-1  10   5
-1   0   5
 9   0   3   \4 square
-1   0  13
-1  10  13
-1  10   3
-1   0   3
10   2   3   \5 star
-1   5  13
-1   8   3
-1   0  10
-1  10  10
-1   2   3
11   3   5   \= not equal
-1   7  11
-1  15   0
-1   8   9
-1   2   9
-1  15   0
-1   2   7
-1   8   7
12   8   9   \~ approx equal
-1   7   8
-1   3  10
-1   2   9
-1  15   0
-1   2   7
-1   3   8
-1   7   6
-1   8   7
13   2   6   \S subset
-1   7   6
-1   8   7
-1   8   9
-1   7  10
-1   2  10
14   7   6   \C cents
-1   4   6
-1   3   7
-1   3   9
-1   4  10
-1   7  10
-1  15   0
-1   5  12
-1   5   4
```

```
15   2  14   \_ overline
-1  10  14
16   2   2   \I integral
 1   4   4
-1   6  12
-1   8  14
17   2   8   \R square root
-1   3   8
-1   5   3
-1   5  14
-1  10  14
18   8  13   \S sum, sigma
-1   2  13
-1   7   8
-1   2   3
-1   8   3
19   6   4   \A alpha
-1   6   8
-1   7   9
-1  15   0
-1   6   8
-1   5   9
-1   3   9
-1   2   8
-1   2   4
-1   3   3
-1   5   3
-1   6   4
-1   7   3
-1   8   4
20   3   0   \B beta
-1   3   9
-1  15   0
-1   3   8
-1   4   9
-1   6   9
-1   7   8
-1   7   7
-1   6   6
-1   3   6
-1  15   0
-1   6   6
-1   7   5
-1   7   4
-1   6   3
-1   4   3
-1   3   4
21   5   7   \D delta
-1   3   6
-1   3   4
-1   4   3
-1   6   3
-1   7   4
-1   7   6
-1   3   9
-1   3  10
-1   4  11
-1   5  11
-1   7   9
22   3   6   \E epsilon
-1   5   6
-1  15   0
-1   7   9
-1   5   9
```

```
-1   4   7
-1   4   5
-1   5   3
-1   7   3
23   3   1   \F phi
-1   6  10
-1  15   0
-1   7   8
-1   6   9
-1   4   9
-1   3   8
-1   2   4
-1   3   3
-1   5   3
-1   6   4
-1   7   8
24   3   8   \G gamma
-1   4   9
-1   6   6
-1  15   0
-1   7   9
-1   7   8
-1   4   3
25   3   3   \L lambda
-1   6   6
-1  15   0
-1   4  11
-1   7   3
26   2   0   \M mu
-1   4   9
-1  15   0
-1   3   5
-1   4   3
-1   5   3
-1   6   5
-1   7   9
27   3   8   \N eta
-1   4   9
-1   5   8
-1   4   5
-1  15   0
-1   5   8
-1   6   9
-1   7   8
-1   5   0
28   4   9   \O omega
-1   3   8
-1   2   5
-1   2   4
-1   3   3
-1   4   4
-1   5   6
-1  15   0
-1   4   4
-1   5   3
-1   6   4
-1   8   8
-1   8   9
29   3   3   \P pi
-1   4   9
-1  15   0
-1   3   9
-1   7   9
-1  15   0
```

```
-1   15    0
-1    6    9
-1    7    3
30    2    6    \T theta
-1    7    6
-1   15    0
-1    6    4
-1    7    8
-1    6    9
-1    4    9
-1    3    8
-1    2    4
-1    3    3
-1    5    3
-1    6    4
31    4    3    \Y psi
-1    6   10
-1   15    0
-1    3    9
-1    3    7
-1    4    6
-1    6    6
-1    7    7
-1    7    9
32    7   15    space
33    5   13    !
-1    5    7
-1   15    0
-1    5    4
-1    4    3
-1    6    3
-1    5    4
34    4    9    "
-1    4   13
-1   15    0
-1    6   13
-1    6    9
35    4    3    #
-1    4   13
-1   15    0
-1    6   13
-1    6    3
-1   15    0
-1    8   10
-1    2   10
-1   15    0
-1    2    6
-1    8    6
36    2    5    $
-1    3    4
-1    7    4
-1    8    5
-1    8    7
-1    7    8
-1    3    8
-1    2    9
-1    2   11
-1    3   12
-1    7   12
-1    8   11
-1   15    0
-1    4   13
-1    4    3
-1   15    0
```

```
-1   6  13
-1   6   3
37   2   3   %
-1   8  13
-1  15   0
 1   3  11
-1   4  12
-1   3  13
-1   2  12
-1   3  11
-1  15   0
-1   7   3
-1   6   4
-1   7   5
-1   8   4
-1   7   3
38   7   7   &
-1   7   4
-1   6   3
-1   4   3
-1   3   4
-1   3   6
-1   7  11
-1   7  12
-1   6  13
-1   4  13
-1   3  12
-1   3  11
-1   8   3
39   6  13   '
-1   4   9
40   6  13   (
-1   4  11
-1   4   5
-1   6   3
41   4  13   )
-1   6  11
-1   6   5
-1   4   3
42   8  12   *
-1   2   4
-1  15   0
-1   5   3
-1   5  13
-1  15   0
-1   2  12
-1   8   4
43   2   8   +
-1   8   8
-1  15   0
-1   5  11
-1   5   5
44   5   3   ,
-1   4   3
-1   4   4
-1   5   4
-1   5   3
-1   3   1
45   2   8   -
-1   8   8
46   5   3   .
-1   4   3
-1   4   4
-1   5   4
```

```
-1   5   3
47   2   3   /
 1   8  13
48   3   4   0
-1   7  12
-1   6  13
-1   4  13
-1   2  11
-1   2   5
-1   4   3
-1   6   3
-1   8   5
-1   8  11
-1   7  12
49   4  12   1
-1   5  13
-1   5   3
-1  15   0
-1   4   3
-1   6   3
50   2  12   2
-1   3  13
-1   7  13
-1   8  12
-1   8   9
-1   7   8
-1   3   8
-1   2   7
-1   2   3
-1   8   3
-1   8   4
51   2  12   3
-1   3  13
-1   7  13
-1   8  12
-1   8   9
-1   7   8
-1   4   8
-1  15   0
-1   7   8
-1   8   7
-1   8   4
-1   7   3
-1   3   3
-1   2   4
52   2  13   4
-1   2   8
-1   8   8
-1  15   0
-1   6  13
-1   6   3
-1  15   0
-1   5   3
-1   7   3
53   2   4   5
-1   3   3
-1   7   3
-1   8   4
-1   8   7
-1   7   8
-1   2   8
-1   2  13
-1   8  13
54   2   7   6
```

```
-1   3   8
-1   7   8
-1   8   7
-1   8   4
-1   7   3
-1   3   3
-1   2   4
-1   2  12
-1   3  13
-1   7  13
-1   8  12
55   2  13   7
-1   8  13
-1   3   5
-1   3   3
56   3   8   8
-1   2   9
-1   2  12
-1   3  13
-1   7  13
-1   8  12
-1   8   9
-1   7   8
-1   8   7
-1   8   4
-1   7   3
-1   3   3
-1   2   4
-1   2   7
-1   3   8
-1   7   8
57   2   4   9
-1   3   3
-1   7   3
-1   8   4
-1   8  12
-1   7  13
-1   3  13
-1   2  12
-1   2   9
-1   3   8
-1   7   8
-1   8   9
58   4   8   :
-1   4   9
-1   5   9
-1   5   8
-1   4   8
-1  15   0
-1   5   3
-1   4   3
-1   4   4
-1   5   4
-1   5   3
59   4   8   ;
-1   4   9
-1   5   9
-1   5   8
-1   4   8
-1  15   0
-1   5   3
-1   4   3
-1   4   4
-1   5   4
```

```
-1   5   3
-1   3   1
60   8  13   less than
-1   2   8
-1   8   3
61   8   9   =
-1   2   9
-1  15   0
-1   2   7
-1   8   7
62   2  13   greater than
-1   8   8
-1   2   3
63   2  10   ?
-1   2  12
-1   3  13
-1   7  13
-1   8  12
-1   8   9
-1   7   8
-1   5   8
-1   5   6
-1  15   0
-1   5   4
-1   4   3
-1   6   3
-1   5   4
64   6   7   @
-1   6  11
-1   5  11
-1   4  10
-1   4   7
-1   8   7
-1   8  12
-1   7  13
-1   4  13
-1   2  11
-1   2   5
-1   4   3
-1   6   3
-1   8   5
65   2   3   A
-1   2  10
-1   4  13
-1   6  13
-1   8  10
-1   8   3
-1  15   0
-1   2   8
-1   8   8
66   3   3   B
-1   3  13
-1  15   0
-1   2  13
-1   7  13
-1   8  12
-1   8   9
-1   7   8
-1   3   8
-1  15   0
-1   7   8
-1   8   7
-1   8   4
-1   7   3
```

```
-1   2   3
67   8  11   C
 1   6  13
 1   4  13
-1   2  11
 1   2   5
-1   4   3
-1   6   3
-1   8   5
68   3   3   D
-1   3  13
-1  15   0
-1   2  13
-1   6  13
-1   8  11
-1   8   5
-1   6   3
-1   2   3
69   8   3   E
-1   2   3
-1   2  13
-1   8  13
-1  15   0
-1   2   8
-1   6   8
70   2   3   F
-1   2  13
-1   8  13
-1  15   0
-1   2   8
-1   6   8
71   8  11   G
-1   6  13
-1   4  13
-1   2  11
-1   2   5
-1   4   3
-1   6   3
-1   8   5
-1   8   8
-1   6   8
72   2   3   H
-1   2  13
-1  15   0
-1   2   8
-1   8   8
-1  15   0
-1   8  13
-1   8   3
73   3  13   I
-1   7  13
-1  15   0
-1   5  13
-1   5   3
-1  15   0
-1   3   3
-1   7   3
74   2   5   J
-1   4   3
-1   5   3
-1   7   5
-1   7  13
-1  15   0
-1   6  13
```

```
 -1   8  13
 75   2   3   K
  1   2  13
  1  15   0
 -1   8  13
 -1   2   8
 -1   8   3
 76   2  13   L
 -1   2   3
 -1   8   3
 77   2   3   M
 -1   2  13
 -1   5   8
 -1   8  13
 -1   8   3
 78   2   3   N
 -1   2  13
 -1   8   3
 -1   8  13
 79   4   3   O
 -1   2   5
 -1   2  11
 -1   4  13
 -1   6  13
 -1   8  11
 -1   8   5
 -1   6   3
 -1   4   3
 80   2   3   P
 -1   2  13
 -1   6  13
 -1   8  11
 -1   8  10
 -1   6   8
 -1   2   8
 81   4   3   Q
 -1   2   5
 -1   2  11
 -1   4  13
 -1   6  13
 -1   8  11
 -1   8   5
 -1   6   3
 -1   4   3
 -1  15   0
 -1   6   6
 -1   8   3
 82   2   3   R
 -1   2  13
 -1   6  13
 -1   8  11
 -1   8  10
 -1   6   8
 -1   2   8
 -1  15   0
 -1   4   8
 -1   8   3
 83   2   4   S
 -1   3   3
 -1   6   3
 -1   8   5
 -1   8   7
 -1   7   8
 -1   3   8
```

```
-1   2   9
-1   2  11
-1   4  13
-1   7  13
-1   8  12
84   2  13   T
-1   8  13
-1  15   0
-1   5  13
-1   5   3
85   2  13   U
-1   2   5
-1   4   3
-1   6   3
-1   8   5
-1   8  13
86   2  13   V
-1   5   3
-1   8  13
87   2  13   W
-1   2   3
-1   5   8
-1   8   3
-1   8  13
88   2   3   X
-1   8  13
-1  15   0
-1   2  13
-1   8   3
89   2  13   Y
-1   2  11
-1   5   9
-1  15   0
-1   5   3
-1   5   9
-1   8  11
-1   8  13
90   2  13   Z
-1   8  13
-1   2   3
-1   8   3
91   8  13   [
-1   2  13
-1   2   3
-1   8   3
92   2  13   \
-1   8   3
93   2  13   ]
-1   8  13
-1   8   3
-1   2   3
94   2   3   ^
-1   5   9
-1   8   3
95   2   2   _
-1  10   2
96   3  13
-1   6   9
97   2   8   a
-1   3   9
-1   7   9
-1   8   8
-1   8   4
-1   7   3
```

```
 -1   3   3
 -1   2   4
  1   2   6
  1   3   7
 -1   7   7
 -1   8   6
 98   2  13   b
 -1   2   3
 -1   6   3
 -1   8   5
 -1   8   7
 -1   6   9
 -1   2   9
 99   8   8   c
  1   6   9
 -1   4   9
 -1   2   7
 -1   2   5
 -1   4   3
 -1   6   3
 -1   8   4
100   8  13   d
 -1   8   3
 -1   4   3
 -1   2   5
 -1   2   7
 -1   4   9
 -1   8   9
101   2   6   e
 -1   7   6
 -1   8   7
 -1   8   8
 -1   7   9
 -1   3   9
 -1   2   8
 -1   2   4
 -1   3   3
 -1   7   3
 -1   8   4
102   7  12   f
 -1   6  13
 -1   5  12
 -1   5   3
 -1  15   0
 -1   4   3
 -1   6   3
 -1  15   0
 -1   4   8
 -1   6   8
103   2   1   g
 -1   3   0
 -1   7   0
 -1   8   1
 -1   8   8
 -1   7   9
 -1   3   9
 -1   2   8
 -1   2   4
 -1   3   3
 -1   7   3
 -1   8   4
104   2   3   h
 -1   2  13
 -1  15   0
```

```
  -1    2    7
  -1    4    9
  -1    6    9
  -1    8    7
  -1    8    3
 105    5   10    i
  -1    4   11
  -1    6   11
  -1    5   10
  -1   15    0
  -1    5    8
  -1    5    3
  -1   15    0
  -1    4    3
  -1    6    3
 106    2    1    j
  -1    3    0
  -1    6    0
  -1    7    1
  -1    7    8
  -1   15    0
  -1    7   10
  -1    8   11
  -1    6   11
  -1    7   10
 107    2   13    k
  -1    2    3
  -1   15    0
  -1    2    6
  -1    8    9
  -1   15    0
  -1    4    7
  -1    8    3
 108    4   13    l
  -1    5   12
  -1    5    4
  -1    6    3
 109    2    3    m
  -1    2    9
  -1   15    0
  -1    2    8
  -1    3    9
  -1    4    9
  -1    5    8
  -1    5    3
  -1   15    0
  -1    5    8
  -1    6    9
  -1    7    9
  -1    8    8
  -1    8    3
 110    2    3    n
  -1    2    9
  -1   15    0
  -1    2    7
  -1    4    9
  -1    6    9
  -1    8    7
  -1    8    3
 110    4    3    o
  -1    2    5
  -1    2    7
  -1    4    9
  -1    6    9
```

```
 -1   8   7
 -1   8   5
 -1   6   3
 -1   4   3
112   2   0   p
 -1   2   9
 -1  15   0
 -1   2   8
 -1   3   9
 -1   6   9
 -1   8   7
 -1   8   5
 -1   6   3
 -1   3   3
 -1   2   4
113   8   0   q
 -1   8   9
 -1  15   0
 -1   8   8
 -1   7   9
 -1   4   9
 -1   2   7
 -1   2   5
 -1   4   3
 -1   7   3
 -1   8   4
114   3   3   r
 -1   3   9
 -1  15   0
 -1   3   7
 -1   5   8
 -1   7   8
115   2   4   s
 -1   3   3
 -1   7   3
 -1   8   4
 -1   8   5
 -1   7   6
 -1   3   6
 -1   2   7
 -1   2   8
 -1   3   9
 -1   7   9
 -1   8   8
116   3   9   t
 -1   7   9
 -1  15   0
 -1   5  13
 -1   5   4
 -1   6   3
 -1   7   4
117   2   9   u
 -1   2   5
 -1   4   3
 -1   6   3
 -1   8   5
 -1   8   9
118   2   9   v
 -1   5   3
 -1   8   9
119   2   9   w
 -1   2   5
 -1   4   3
 -1   5   5
```

```
 -1   6   3
 -1   8   5
 -1   8   9
120   2   3   x
 -1   8   9
 -1  15   0
  1   2   9
 -1   8   3
121   2   1   y
 -1   3   0
 -1   7   0
 -1   8   1
 -1   8   9
 -1  15   0
  1   8   5
 -1   7   4
 -1   3   4
 -1   2   5
 -1   2   9
122   2   8   z
 -1   3   9
 -1   8   9
 -1   2   3
 -1   7   3
 -1   8   4
123   6  13   {
 -1   5  12
 -1   5   9
 -1   4   8
 -1   5   7
 -1   5   4
 -1   6   3
124   5  13   |
 -1   5   9
 -1  15   0
 -1   5   7
 -1   5   3
125   4  13   }
 -1   5  12
 -1   5   9
 -1   6   8
 -1   5   7
 -1   5   4
 -1   4   3
126   2   8   ~
 -1   3   9
 -1   7   7
 -1   8   8
127   5   7   infinity
 -1   4   6
 -1   3   6
 -1   2   7
 -1   2   9
 -1   3  10
 -1   4  10
 -1   5   9
 -1   6  10
 -1   7  10
 -1   8   9
 -1   8   7
 -1   7   6
 -1   6   6
 -1   5   7
 -1   5   9
```

```
c-*--LETTER---PLOT ALPHANUMERIC CHARACTERS OR STRINGS OF CHARACTERS-----
c
      subroutine letter (x,y,mode,jtext,angle,kpos,nlen)
c
c-----Programmer  PLWard, USGS, Menlo Park, California, February, 1979.--
c
c-*--This subroutine writes out characters either using device hardware
c       generated characters or software generated characters depending
c       on the chosen angle and the device.-------------------------------
c
      include 'gpcom.h'
      include 'lmccom.h'
c
      save
c
      integer i,mode,klen,kpos,nlen,ONE,ix,iy,jkls,isup,isoflt,j,jsup,
     1  lchar,nnodes,itext(132),ichr,imode,ipos,
     2  i30,i32,i97,i122,i127,SUBS,SUPS,BS,CR,ixa(32),iya(32),iupos(8)
      real pointn(2),xg(16),yg(16),xgs(16),ygs(16),points(4),alow(4),
     1  ahigh(4),x,y,angle,oangle,oupdir,updir,sangwd,cangwd,sanghi,
     2  canghi,spalet,clen,a,b,spacew,dchar,chars,dir
      character*1 jtext(1)
c
      parameter( i30=30, i32=32, i97=97, i122=122, i127=127)
      parameter(SUBS=0, SUPS=1,    BS=2,     CR=3)
      data oangle/-9999.9/,oupdir/-2.0/
      data alow /90.0,90.0,0.0,180.0/
      data ahigh/270.0,270.0,180.0,360.0/
      data iupos/1,3,2,5,4,7,6,8/
      data ONE /1/
c
c
      if(idbug.ge.3)call printn("letter: %d %f %f %f %d %d\n",
     1  mode,x,y,angle,kpos,nlen)
c
c-*--If kpos le -100, as set in pltltr, then call movdrw not letter.-----
      if (kpos.gt.-100) goto 5
          call movdrw(x,y,mode)
          return
c
c-*--Get the text string from the pipe.---------------------------------
c     Note that itext is integer*2 rather than character to allow for
c     fonts that are decoded in gpgstr. If klen is < 0, gpgstr will get
c     string from pipe, otherwise gpgstr interprets itext for different
c     fonts and escape characters.
  5   if(nlen.lt.0) goto 7
      if(nlen.gt.132)nlen=132
          do 6 i=1,nlen
  6       itext(i)=ifix(jtext(i))
  7   klen=nlen
      call gpgstr(itext,klen)
      if(idbug.lt.3) goto 9
          call printn("letter: length=%d ",klen)
          do 8 i=1,klen
  8           call printn("%d ",itext(i))
          call printn("\n")
c
c     Also return if there is missing data.
  9   if (ibreak.eq.-1.or.kerror.ne.i0) return
c
c-*--Be sure angle is between 0 and 360.-------------------------------
      if(angle.ge.0.0.and.angle.lt.360.0) goto 10
          if(angle.ge.360.0)angle=angle-360.0
          if(angle.lt.0.0  )angle=angle+360.0
```

```
          goto 9
c
c--*--Check the mode to be sure that it is a legal value.----------------
c-----Mode is the same as used in movdrw.
 10   if(mode.gt.-i5.and.mode.lt.i5) goto 30
          kerror=1
          imode=mode
          call gperr(kerror,"letter","Illegal mode of",imode)
20        return
c
c
c--*--Scan the text and change any illegal characters to an asterisk.---
30    do 31 i=1,klen
          if(itext(i).lt.0.or.itext(i).gt.1279)itext(i)=42
31        continue
c
c--*--MOVE OR DRAW TO THE POINT.-----------------------------------------
      ix=kpltsp
      kpltsp=kltrsp
      call movdrw(x,y,mode)
      kpltsp=ix
      if (kerror.ne.i0) return
      if(x.eq.usym.or.y.eq.usym) return
      pointn(1)=point(1)
      pointn(2)=point(2)
c-----Save the present point to use after letter plotted.---------------
      do 32 i=1,4
32        points(i)=point(i)
      jkls=jkl
c
c--*--DECIDE IF LETTERING WILL BE RIGHT SIDE UP.-------------------------
      updir=1.0
      if(iup.le.0.or.iup.gt.4) goto 58
          if(iup.ne.2) goto 42
              if(angle.le.alow(2).or.angle.ge.ahigh(2)) goto 58
              goto 43
42        if(angle.ge.alow(iup).and.angle.le.ahigh(iup)) goto 58
43            updir=-1.0

c--*--FOR NEW THETA OR HEIGHT CALCULATE NEW OFFSETS FOR GRID USED FOR
c       SOFTWARE LETTERING.-----------------------------------------------
58    if (oangle.eq.angle.and.letchg.ne.1.and.oupdir.eq.updir)  goto 70
          oangle=angle
          letchg=0
          isup=1
          sangwd=sin(angle*0.0174533)*aspect*height*updir/6.0
          cangwd=cos(angle*0.0174533)*aspect*height*updir/6.0
          sanghi=sin((angle-slopel)*0.0174533)*height*updir/10.0
          canghi=cos((angle-slopel)*0.0174533)*height*updir/10.0
          do 60 i=1,16
              xg(i)=(i-4)*cangwd
              yg(i)=(i-4)*sangwd
              xgs(i)=(i-4)*canghi
60            ygs(i)=(i-4)*sanghi
c
c--*--DECIDE IF HARDWARE LETTERING CAN BE USED --------------------------
70    spalet=spacel+1.0
      isoflt=1
      if(kchar.eq.i0) goto 100
      if(pgain(1).ne.1.0.or.pgain(2).ne.1.0.or.pangle.ne.0.0) goto 100
      if(updir.ne.-1.0.and.angle.ne.0.0) goto 100
      if(updir.eq.-1.0.and.angle.ne.180.0) goto 100
          do 75  i=1,klen
75        if(itext(i).lt.32.or.itext(i).ge.127) goto 100
```

```
          isoflt=0
c
c
c--*--FIND NEW POINT TO BE AT LOWER LEFT CORNER OF THE FIRST CHARACTER
c      BASED ON THE VALUE OF KPOS.----------------------------------------
100   ipos=iabs(kpos)
      if(ipos.ge.i1.and.ipos.le.i8) goto 101
          imode=kpos
          call gperr(kerror,"letter","Position set to 1. Illegal value g
     1iven of",imode)
          kpos=1
          ipos=1
c-----If string is being flipped to be right side up, then get new ipos
c      and for ipos 2 and 3 move centering character to other end of
c      string.
101   if(updir.ne.-1.0) goto 105
          ipos=iupos(ipos)
          if(ipos.ne.2) goto 103
               j=itext(1)
               do 102 i=1,klen-1
102                 itext(i)=itext(i+1)
               itext(klen)=j
103       if(ipos.ne.3) goto 105
               j=itext(klen)
               do 104 i=klen,2,-1
104                 itext(i)=itext(i-1)
               itext(1)=j
c-----Find the length of the string omitting special characters and
c      correcting for sub and superscripts
105   clen=klen
      jsup=1
      do 108 i=1,klen
          if(mod(itext(i),128).gt.5) goto 107
               clen=clen-1.0
               if(itext(i).eq.BS)clen=clen-1.0/jsup
               if(itext(i).ne.SUPS.and.itext(i).ne.SUBS) goto 106
                    jsup=jsup+1
                    if(jsup.gt.i2)jsup=1
106            if(itext(i).ne.CR) goto 108
                    clen=float(klen)-clen-2.0+float(i)
                    goto 109
107       if(jsup.eq.i2)clen=clen-0.5
108       continue
c-----String centered around the point.-----------------------------------1-
109   a=5.0
      b=3.0*spalet*(clen-1.0)+3.0
c----- ipos= 1    2    3    4    5    6    7    8                          ipos
      goto(180,120,130,140,150,160,170,200) ipos
c-----String to left of point centered on last character.----------------2-
120   b=2.0*b-3.0
      goto 180
c-----String centered around left-most character.------------------------3-
c-----String to right of point centered on first character.--------------4-
130   b=3.0
      goto 180
c-----String centered to right of the point.-----------------------------4-
140   b=0.0
      if(kpos.lt.i0)b=b-4.0
      goto 180
c-----String centered to left of the point.------------------------------5-
150   b=2.0*b
      if(kpos.lt.i0)b=b+4.0
      goto 180
c-----String centered above the point.-----------------------------------6-
```

```
160     a=0.0
        if(kpos.lt.i0)a=-5.0
        goto 180
c-----String centered below the point.-----------------------------------7-
170     a=10.0
        if(kpos.lt.i0)a=15.0
180     pointn(1)=pointn(1)+a*sanghi-b*cangwd
        pointn(2)=pointn(2)-a*canghi-b*sangwd
c-----For kpos ne 6 correct centering for lower-case letters if necessary
        if(ipos.eq.i6) goto 200
            ichr=mod(itext(1),128)
            if(ipos.eq.i2) ichr=mod(itext(klen),128)
            if(ichr.gt.31.and.ichr.lt.97) goto 200
            if(ichr.lt.19.or.ichr.gt.122) goto 200
                a=2.0
                if(ipos.eq.i7)a=4.0
                pointn(1)=pointn(1)-a*sanghi
                pointn(2)=pointn(2)+a*canghi
200     if(isoflt.eq.i1) goto 300
        if(idbug.ge.3)call printn("letter: ipos=%d kpos=%d pointn=%f %f\n
       1        up=%f spacel=%f point=%f %f\n",ipos,kpos,pointn(1),
       2        pointn(2),updir,spacel,point(1),point(2))
c
c
c-*--IF HARDWARE LETTERING IS USED, OUTPUT POINT TO GPDEVO AND
c       CHARACTER STRING TO GPHLET.-----------------------------------------
c-*--First clip the string if necessary on one or both ends.-------------
        ix=kltrsp
        if(mode.lt.0) ix=kpltsp
        if(pointn(2).lt.pspace(ix+2)) return
        if((pointn(2)+height-0.00001).gt.pspace(ix+3)) return
        spacew=height*aspect*(1+spacel)
        do 202 i=0,klen
            if((pointn(1)+i*spacew).ge.pspace(ix)) goto 203
202         continue
203     do 204 j=klen,0,-1
            if((pointn(1)+j*spacew).le.pspace(ix+1)) goto 205
204         continue
205     if(i.eq.i0.and.j.eq.klen) goto 207
        do 206 ix=1,j
206         itext(ix)=itext(i+ix)
        pointn(1)=pointn(1)+i*spacew
        klen=j-i
207     ix=kpltsp
        kpltsp=kltrsp
        call movdrw(pointn(1),pointn(2),ONE)
        kpltsp=ix
        call gphlet(klen,itext)
        goto 600
c
c
c-*--PLOT SOFTWARE GENERATED LETTERS-------------------------------------
c       Coordinates for vectors for letters are passed from gpletr_.c
c-----Change line type if necessary.-------------------------------------
300     linmod=2
        if(ltype.eq.lintyp(2)) goto 301
            point(1)=lintyp(2)
            call gpdevo(i11,point)
            lintyp(2)=ltype
c
c-----Change line shade if necessary.------------------------------------
301     if (pshade.eq.shade(2)) goto 302
            point(1)=shade(2)
            call gpdevo(i12,point)
```

```
c
c-----Plot each character in the string which is klen characters long.---
302    dchar=1.0
       chars=0.0
       do 500 lchar=1,klen
          if(ibreak.eq.-1.or.kerror.ne.i0)goto 600
c-----Find the index to the nodes and the number of nodes in the letter.-
          nnodes=32
          call gpletr(itext(lchar),nnodes,ixa,iya)
          kmode=1
c
c
c-----Decode and plot each node.-----------------------------------------
          do 490 i=1,nnodes
             ix=ixa(i)+2
             iy=iya(i)+1
             if(ix.ne.i17) goto 320
                kmode=1
                goto 490
c-----Draw the vector to the new node.-----------------------------------
320          if(iy.eq.i16) goto 330
                point(1)=pointn(1)+xg(ix)-ygs(iy)
                point(2)=pointn(2)+yg(ix)+xgs(iy)
                call gpclip(kltrsp)
                kmode=2
                goto 490
c-----Handle special characters.-----------------------------------------
330          goto(410,420,430,440,450,460,470) ix-2
c-----Backspace.
410          dir=-1.0
             chars=chars-dchar*3.0
415          pointn(1)=pointn(1)+dir*(xg(10)*spalet-ygs(4))
             pointn(2)=pointn(2)+dir*(yg(10)*spalet+xgs(4))
             chars=chars+dchar
             goto 500
c-----Superscript.
420          dir=1.0
c-----Between sub or superscript commands, halve the letter size.
422          if(isup.eq.i2) goto 424
                isup=2
                goto 428
424          do 426 j=1,16
                xg(j)=xg(j)*2.0
                yg(j)=yg(j)*2.0
                xgs(j)=xgs(j)*2.0
426             ygs(j)=ygs(j)*2.0
             dchar=1.0
             isup=1
428          oangle=-9999.9
             pointn(1)=pointn(1)+dir*(xg(4)-yg(9))
             pointn(2)=pointn(2)+dir*(yg(4)+xg(9))
             if(isup.eq.i1) goto 500
             do 429 j=1,16
                xg(j)=xg(j)*0.5
                yg(j)=yg(j)*0.5
                xgs(j)=xgs(j)*0.5
429             ygs(j)=ygs(j)*0.5
             dchar=0.5
             isup=2
             goto 500
c-----Line feed.
430          dir=-1.0
433          pointn(1)=pointn(1)+dir*(xg(4)-ygs(14)*(1+spaceh))
             pointn(2)=pointn(2)+dir*(yg(4)+xgs(14)*(1+spaceh))
```

```
                goto 500
c-----Reverse line feed.
440             dir=1.0
                goto 433
c-----Subscript.
450             dir=-1.0
                goto 422
c-----Carriage return
460             pointn(1)=pointn(1)-chars*(xg(10)*spalet-ygs(4))
                pointn(2)=pointn(2)-chars*(yg(10)*spalet+xgs(4))
                chars=0.0
                goto 500
c-----Space.
470             dir=1.0
                goto 415
490             continue
c
c
c-----Set pointn for the lower left hand corner of next character.--------
            pointn(1)=pointn(1)+xg(10)*spalet-ygs(4)
            pointn(2)=pointn(2)+yg(10)*spalet+xgs(4)
            chars=chars+dchar
500         continue
c
c---MOVE PEN BACK TO INPUT POINT in case we need to draw to next point.
600     do 601 i=1,4
601         point(i)=points(i)
        jkl=jkls
        kmode=1
        call gpclip(kpltsp)
        return
        end
```

```
/* loadletr.c     Programmer: PLWard, USGS, 10/28/79    */

/* This program loads the data statements for arrays "indcx" and
   "nodes" used in the subroutine letter. Each letter is addressed
   as a code number which is the 8-bit integer
   representation of ASCII characters as extended with special
   characters in place of control codes. Letters are drawn by vectors
   between nodes. The nodes refer to a grid from 0 to 10 in the
   x-direction and 0 to 14 in the y-direction. The lower-left corner
   of most letters is point (2,3) on the grid. Most letters are
   centered around point (5,8) but lower case letters (codes 18 to
   30 and 97 to 122) are centered around point (5,6) not including
   tails, for example on letters like b and d or g. Space between
   letters is normally 8 to 10 in the x direction on the grid.
   Punctuation codes are not centered. The node addresses are
   encoded in the array called nodes where each byte is the 4 bit
   x address followed by the 4 bit y address. The array called index
   contains the byte in the node array of the first node for a given
   code (bits 5 to 15) and the number of nodes in the symbol (bits
   0 to 4). The array index is indexed by the symbol code + 1. Node
   (15,0) means move to the next node. Node (i,15) means this is a
   special character and is handled specially. A line feed is 15
   points in the y direction on the grid and a space is 10 points in
   the x direction.                    */

#include <stdio.h>
#define MAXCHARS 128
#define MAXNODES 1200
#define IFNAME "letter.codes"
#define OFNAME "gpletr.h"

main()
{
    long index[MAXCHARS];
    unsigned int nodes[MAXNODES];
    int inum,ix,iy,nnodes,ls,n2,nbyte,n,err;
    FILE *fopen(), *fp;
    FILE *fopen(), *fd;

    if((fp=fopen(IFNAME,"r")) == NULL){
        fprintf(stderr,"loadletr: can not open letter.codes.\n");
        exit(1);
    }
    for(nbyte=0,nnodes=0,n=0; inum < MAXCHARS; nbyte++){
        err=fscanf(fp,"%4d%4d%4d",&inum,&ix,&iy);
        if(err<=0){inum=128; continue;}
        while (getc(fp) != '\n');
        if (inum > 0){
              index[n]=((nbyte-nnodes)<< 5) + nnodes;
              nnodes=0; n++;
        }
        n2=nbyte/2; ls=8; nnodes++;
        if(n2*2 != nbyte) ls=0;
        nodes[n2]=nodes[n2]+(((ix<<4) +iy)<<ls);
    }
    if((fd=fopen(OFNAME,"w")) == NULL){
       fprintf(stderr,"loadletr: can not open OFNAME\n");
       exit(1);
    }
    fprintf(fd,"/* Include block for letter codes used by gpletr_.c */\n");
    fprintf(fd,"#define NCHARS %d\n",MAXCHARS);
    fprintf(fd,"long index[]={");
    for(inum=0; inum < n; inum++){
        if ((inum/10)*10 == inum) fprintf(fd,"\n ");
```

```
        fprintf(fd,"%ld,",index[inum]);
    }
    fprintf(fd,"0\n};\n");
    fprintf(fd,"\nunsigned int nodes[]={");
    for(inum=0; inum < n2; inum++){
        if  ((inum/10)*10 == inum) fprintf(fd,"\n ");
        fprintf(fd,"0x%x,",nodes[inum]);
    }
    fprintf(fd,"0x0\n};\n");
    printf("loadletr is finished\n");
    exit(0);
}
```

```
/* MAP - read map files and plot allowing for longitudinal wrap for
   example plotting a map with longitude in range -200 to 200 degrees.
   Programmer PLWard
*/
#define MAXSTRG  40
#define DS       (common->pspace+18)
int    LENSTRING=MAXSTRG;
char   strg[MAXSTRG];
char   strg1[MAXSTRG+4];
char   strg2[MAXSTRG+14];
int fd;

#include <stdio.h>
#include "gpcomc.h"
int THREE        =3;
float wrap[3]={ -360.0,0.0,360.0};
int jb,jl;

map_()
{
    register i;
    int save,lengot,nfile,lowlat,higlat;
    extern gpcom_;

    common = &gpcom_;
    gpstrg(&LENSTRING,strg,&lengot);
    if(strg[0]=='\0') strcpy(strg,"world.co");
    if (common->idbug >= 2) printf(" map: %s\n",strg);
    save = common->kpltsp;
    common->kpltsp = 19;
    jb=(DS[0] < -180.0) ? 0 : 1;
    jl=(DS[1] >  180.0) ? 2 : 1;
    lengot=strlen(strg);

    if((strg[lengot-2]=='.') && (strg[lengot-1]=='m')) {
        lowlat=((DS[2]<0)?DS[2]-10.:DS[2])/10.;
        higlat=((DS[3]<0)?DS[3]-10.:DS[3])/10.;
        nfile=higlat-lowlat+1;
        for(i=lowlat;i<=higlat;i++){
            sprintf(strg1,"%s.%d",strg,i*10);
            if((fd=open(strg1,0)) == -1) {
                strcpy(strg2,"/usr/maps/"); strcat(strg2,strg1);
                if((fd=open(strg2,0)) == -1 && --nfile==0){
                    common->kerror=2;
                    fprintf(stderr,"FATAL ERROR in map: Unable to open map files of type %s\n",
                    goto out;
                }
            }
            if(mapit()==1) goto out;
            close(fd);
        }
    }
    else {
        if((fd=open(strg,0)) == -1) {
            strcpy(strg2,"/usr/maps/"); strcat(strg2,strg);
            if((fd=open(strg2,0))== -1) {
                common->kerror=2;
                fprintf(stderr,"FATAL ERROR in map: Unable to open map file %s\n",strg);
                goto out;
            }
        }
        mapit();
    }
out:common->kpltsp = save;
```

```
    close(fd);
}

mapit() {
    register i,j;
    int num,mode;
    float coord[128],savelon[3],savelat[3];

    for(i=0;i<3;i++) savelon[i]=savelat[i]=9999.9;
    while ((common->ibreak != -1) && ((num=read(fd,coord,512)) > 0)) {
        for(j=jb; j<=jl; j++) {
            mode= -1;
            if(savelon[j]<9999.0) {
                movdrw_(&savelon[j],&savelat[j],&mode);
                mode= -2;
            }
            for(i=0; i<num/4; i+=2){
                if (coord[i+1] >= 990.0) {
                    mode = -1;
                    savelat[j] = savelon[j] = 9999.9;
                }
                else {
                    savelon[j]=coord[i]+wrap[j];
                    savelat[j]=coord[i+1];
                    movdrw_(&savelon[j],&savelat[j],&mode);
                    mode= -2;
                }
            }
        }
        if(common->ibreak != 0 || common->kerror != 0) return(1);
    }
}
```

```
/* MAPDIG - digitize a map using xy table-top digitizer.
   Programmer PLWard
*/
#define TRANSFORMATION    14
#define OUTPUTFILE    "mapdig.out"
int INTERACTIVE =    1;
int FILEOUT =        1;
int FILEMODE =        0666;
int DEBUG =        0;
int ERROROUT =        0;
#define PI    3.141592654
#define TWOPI    6.283185308

#include "gpcomc.h"
struct scom gpcom_;

#include <stdio.h>
#include <signal.h>
char outfile[40];
float yscale    =1.0;

main(argc,argv)
   int argc;
   char *argv[];
{
   int i;

   common= &gpcom_;
   common->ktrans = TRANSFORMATION;
   for(i=0;i<9;i++) common->trans[i] = 9999.0;
   strcpy(outfile,OUTPUTFILE);
   while(--argc>0 && **++argv == '-') {
      switch(argv[0][1]) {
         case 'c':
         case 'C': sscanf(&argv[0][2],"%d",&i);
                   sscanf(*++argv,"%f",&common->trans[i-1]);
                   if(INTERACTIVE)
                        printf("Set transcon[%d]=%f\n",i,common->trans[i-1]);
                   argc--; break;
         case 'd':
         case 'D': DEBUG = 1; break;
         case 'e':
         case 'E': ERROROUT=1; break;
         case 'o':
         case 'O':
         case 'f':
         case 'F': strcpy(outfile,*++argv);
                   argc--; break;
         case 'i':
         case 'I': INTERACTIVE=0; break;
         case 'n':
         case 'N': FILEOUT =0; break;
         case 'S':
         case 's': sscanf(*++argv,"%f",&yscale); break;
         case 't':
         case 'T': if(!sscanf(*++argv,"%d",&common->ktrans)) {
                   fprintf(stderr,"Illegal transformation of %s\n",argv[0]);
                   exit(1);
                   }
                   argc--; break;
         default: fprintf(stderr,"Unknown option %s.\n",argv[0]);
      }
   }
   if(INTERACTIVE)
```

```
        fprintf(stderr,"Map transformation=%d, output file named %s\n",
                common->ktrans,outfile);
    mapdig();
}

#include <sgtty.h>

setio(type)
    int type;
{
    struct sgttyb args;

    if(type) {
        gtty(2,&args);
        args.sg_flags |= CBREAK;
        args.sg_flags &= ~ECHO;
        stty(2,&args);
    }
    else {
        gtty(2,&args);
        args.sg_flags &= ~CBREAK;
        args.sg_flags |= ECHO;
        stty(2,&args);
    }
}

onbreak(sig)
    int sig;
{
    setio(0);
    kill(getpid(),sig);
}

int DATATOPAGE=1;
int PAGETODATA=2;
int INITIALIZE=3;
int MAPDIGINIT=4;
int index[] = {18,20,19,20,18,21,19,21};
#define DEGREES         57.29577951
#define ERRORANGLE      0.0175
#define ERRORSCALE      0.001
#include <math.h>
double exp(),pow(),sqrt(),sin(),cos(),tan(),atan2(),fabs();
int fd;
FILE *outputfile;

mapdig()
{
    register i;
    int j,k;
    float table[8],page[8],sintheta,costheta,offsetx,offsety;
    double theta[6],sum1,scale[6],sum2,avetheta,avescale;
    int fd;

    if(FILEOUT) {
        if((fd=creat(outfile,FILEMODE)) == -1) {
            fprintf(stderr,"Unable to create output file %s\n",outfile);
            exit(1);
        }
    outputfile = fdopen(fd,"w");
    }
    common->ktrflg=1;
    common->idbug=DEBUG%6;
    common->pole[0]=common->pole[1]=common->usym = 9999.0;
```

```
if(INTERACTIVE)
        fprintf(stderr,
                "Type westmost lon, eastmost lon, southmost lat, northmost lat. Type retu
for(i=0;i<4;i+=2) getpoint(&common->pspace[18+i],1.0);
if(INTERACTIVE) {
    for(i=0;i<15;i++) signal(i,onbreak);
    setio(1);
}
gptran_(&MAPDIGINIT,page);
for(i=0;i<8;i++) page[i] = common->pspace[index[i]];
for(i=0;i<8;i+=2) {
    if(INTERACTIVE)
        fprintf(stderr,"Now digitize corner with lon=%f lat=%f\n",page[i],page[i+1]);
    getpoint(&table[i],1000.0);
}
for(i=0;i<4;i++) gptran_(&DATATOPAGE,&page[i*2]);
if(DEBUG) { fprintf(stderr,"page=");
    for(i=0;i<8;i++) fprintf(stderr,"%f ",page[i]);
    fprintf(stderr,"\n");
}
sum1=sum2=0.0;
for(i=k=0;i<6;i+=2) {
    for(j=i+2;j<8;j+=2) {
        theta[k]=atan2((double)(table[j+1]-table[i+1]),
          (double)(table[j]-table[i]))
                -atan2((double)(page[j+1]-page[i+1]),
          (double)(page[j]-page[i]));
        while(theta[k]>PI) theta[k]=theta[k]-TWOPI;
        while(theta[k]<-PI) theta[k]=theta[k]+TWOPI;
        scale[k]=sqrt(pow((double)(page[j+1]-page[i+1]),
          2.0)+pow((double)(page[j]-page[i]),2.0))
                /sqrt(pow((double)(table[j+1]-table[i+1]),2.0)
          +pow((double)(table[j]-table[i]),2.0));
        if(DEBUG) fprintf(stderr,"angle and scale (%d) of leg %d %d = %f %f\n",
            k,i,j,theta[k]*DEGREES,scale[k]);
        sum1=sum1+theta[k];
        sum2=sum2+scale[k];
        k++;
    }
}
avetheta=sum1/6.0;
avescale=sum2/6.0;
if(DEBUG)fprintf(stderr,"Average angle =%f average scale =%f\n",
    avetheta*DEGREES,avescale);
for(i=0;i<6;i++) {
    if(fabs(theta[i]-avetheta) > ERRORANGLE)
        fprintf(stderr,"WARNING: angle(%d)=%f radians is too far from average angle=%f\n",
            i,theta[i],avetheta);
    if(fabs(scale[i]-avescale) > ERRORSCALE)
        fprintf(stderr,"WARNING: scale(%d)=%f is too far from average scale=%f\n",
            i,scale[i],avescale);
}
sintheta=sin(avetheta);
costheta=cos(avetheta);
offsetx = page[0]/avescale-table[0]*costheta-table[1]*sintheta;
offsety = page[1]/avescale-table[1]*costheta+table[0]*sintheta;
j=0;
if(DEBUG)fprintf(stderr,"cos=%f sin=%f scale=%f offset=%f %f\n",
    costheta,sintheta,avescale,offsetx,offsety);
if(INTERACTIVE)fprintf(stderr,"Initialization complete. Now start digitizing points.\n");
while(getpoint(table,1000.0)) {
    page[0]=(table[0]*costheta+table[1]*sintheta+offsetx)*avescale;
    page[1]=(table[1]*costheta-table[0]*sintheta+offsety)*avescale;
    if(DEBUG)fprintf(stderr,"page is %f %f\n",page[0],page[1]);
```

```
        gptran_(&PAGETODATA,page);
        putout(page,0);
    }
    if(FILEOUT) fclose(outputfile);
    if(INTERACTIVE) setio(0);
}

putout(point,backup)
    float *point;
    int backup;
{
    if(!backup) {
        fprintf(stdout,"%12.6f   %12.6f\n",point[0],point[1]);
        if(ERROROUT)
            fprintf(stderr,"%12.6f   %12.6f\n",point[0],point[1]);
        if(FILEOUT) fwrite(point,4,2,outputfile);
    }
    else {
        fprintf(stdout,"Delete one point.\n");
        if(ERROROUT)fprintf(stderr,"Delete one point.\n");
        if(FILEOUT)fseek(outputfile,-8L,1);
    }
}

int lastcard = -1;
float signn=1.0;
float savepoint[]= {
    9999.0,
    9999.0
};
float movepoint[]= {
    9999.0,
    9999.0
};
int move = -1;

getpoint(xy,denom)
    float xy[]; double denom;
{
    register i;
    int letter,nonum,cardnum,place,k;
    float mult,div;

    xy[0]=xy[1]=0.0;
    place=i=0;
    mult=10.0;
    div=1.0;
    cardnum=0;
    while(i<2 && letter!= -1) {
        nonum=1; move=0;
        if((letter=getc(stdin)) > 0) switch (letter) {
            case '0':
            case '1':
            case '2':
            case '3':
            case '4':
            case '5':
            case '6':
            case '7':
            case '8':
            case '9': nonum=0; place++;
                  xy[i]=xy[i]*mult+signn*div*(int)(letter-'0');
                  break;
            case '.': nonum=0; mult=1.0; div=div*0.1;    break;
```

```
        case '-': signn= -1.0; break;
        case '+': break;
        case '\n':break;
        case ' ': break;
        case '/': cardnum=(getc(stdin)-'0')*100;
              cardnum=(getc(stdin)-'0')*10+cardnum;
              cardnum=(getc(stdin)-'0')+cardnum;
           if(!INTERACTIVE)
             if(cardnum != lastcard + 1)
              fprintf(stderr,
        "Cards may be out of order. Last card was %d, this card is %d\n",
              lastcard,cardnum);
             lastcard=cardnum;
             break;
        case 'C':
        case 'c': move=2;
             if(place != 0) ungetc(letter,stdin);
             break;
        case 'D':
        case 'd': putout(xy,1); break;
        case 'e':
        case 'E': return(0); break;
        case 'f':
        case 'F':k= -1;
        letin:    letter=getc(stdin);
             if((k== -1) && (letter==' ')) goto letin;
             outfile[++k]=letter;
             if(letter != ' ' && letter != '\n') goto letin;
             outfile[k]='\0';
             if((fd=creat(outfile,FILEMODE)) == -1) {
               fprintf(stderr,"Unable to create output file %s\n",outfile);
             }
             else {
               if(FILEOUT) fclose(outputfile);
               FILEOUT=1;
               outputfile = fdopen(fd,"w");
               if(INTERACTIVE) fprintf(stderr,"New output file %s\n",outfile);
             }
             break;
        case 'M':
        case 'm': move=1;
              if(place != 0) ungetc(letter,stdin);
              break;
        case 'Q':
        case 'q': return(0); break;
        default: if(letter > ' ')
            fprintf(stderr,"Ignoring unrecognized letter %c on input.\n",(char)letter);
            else
            fprintf(stderr,"Ignoring ascii character %c number %d on input.\n",letter,letter);
      }
      if(nonum || place==5) {
        if(place > 0) {
          i++; mult=10.0;
          if(letter != '-') signn=1.0;
          div=1.0; place=0;
        }
        if(move > 0) {
          if(move == 2)putout(savepoint,0);
          putout(movepoint,0);
          move = -1;
        }
      }
    }
    xy[0]=xy[0]/denom;
```

```
    xy[1]=xy[1]/denom;
    if(denom > 10.)xy[1]=yscale*xy[1];
    if(move == -1) {
        savepoint[0] = xy[0];
        savepoint[1] = xy[1];
        move = 0;
    }
    if(DEBUG)fprintf(stderr,"Getpoint returns %f %f denom=%f\n",xy[0],xy[1],denom);
    if(letter== -1) return(0);
          putc((char)7,stderr);
    return(1);
}

gperr_(level,program,message,value)
    int *level, *value;
    char program[],message[];
{

    switch(*level){
        case 0:  fprintf(stderr," WARNING in geoplot on %s: %s %d.\n",
                  program,message,*value); break;
        case 1:  fprintf(stderr," ERROR in geoplot on %s: %s %d.\n",
              program,message,*value); break;
        case -1: fprintf(stderr,"\nBREAK hit in geoplot.\n"); break;
        default: fprintf(stderr," FATAL ERROR in geoplot on %s: %s %d.\n",
        program,message,*value);
    }
    return;
}
```

```
/* MAPUTIL - utility program for transforming, editing, rotating, and
   creating map files.
   Programmer PLWard
*/
#define LEN     20
#include <stdio.h>
#define BINARY    0
#define ASCII     1
#define TRUE      1
#define FALSE     0
int FILEMODE =    0666;
#define TMPFILE    "map.tmp.9999"
#define STDINPUT   0
#define STDOUTPUT 1

int EDIT = FALSE;
int FORCE = FALSE;
int RANGE = FALSE;
int minmax = FALSE;
int intype = BINARY;
int outtype= BINARY;
int saveout;
char fileout[40],cfile[40],filein[40];
int fdin  =    STDINPUT;
int fdout =    STDOUTPUT;
long firstdata = 0;
long lastdata  = 32768;
double pole[2 * LEN];
float move[] = {
   9999.0,
   9999.0
};
float limit[4],window[4];
int first=1;
int numpol= -1;
int thispole= -2;
float dellon[LEN];
float dellat[LEN];
FILE *infile, *outfile, *confile;

main(argc,argv)
   int argc;
   char *argv[];
{
   register i;

   for(i=0;i<LEN;i++) {
      pole[2*i] = 9999.0;
      pole[2*i+1]=9999.0;
      dellon[i] = 0.0;
      dellat[i] = 0.0;
   }
   fileout[0] = '\0';
   while(--argc > 0 && **++argv == '-') {
      switch(argv[0][1]) {
         case 'a':if(argv[0][2] == 'i') intype=ASCII;
                  if(argv[0][2] == 'o') outtype=ASCII;
                  break;
         case 'c':argc--;
                  if(!sscanf(*++argv,"%s",cfile)) {
                     fprintf(stderr,"Unable to read control file name.\n");
                     break;
                  }
                  confile=fopen(cfile,"r");
```

```
                if(confile==NULL) {
                    fprintf(stderr,"Unable to open control file %s.\n",cfile);
                    break;
                }
                while(++numpol<LEN && 4==fscanf(confile,"%f %f %f %f",
                    &pole[2*numpol],&pole[2*numpol+1],&dellon[numpol],
                    &dellat[numpol]));
                numpol--; break;
        case 'e':EDIT=TRUE; break;
        case 'f':if(!sscanf(*++argv,"%ld",&firstdata))
                    fprintf(stderr,"Unable to read first data.\n");
                argc--; break;
        case 'F':FORCE=TRUE; break;
        case 'l':if(!sscanf(*++argv,"%ld",&lastdata))
                    fprintf(stderr,"Unable to read last data.\n");
                argc--; break;
        case 'm':minmax=TRUE; break;
        case 'o':if(!sscanf(*++argv,"%s",fileout))
                    fprintf(stderr,"Unable to read output file name.\n");
                argc--; break;
        case 'p':numpol++;
                if(!sscanf(*++argv,"%f",&pole[2*numpol]))
                    fprintf(stderr,"Unable to read pole longitude.\n");
                argc--;
                if(!sscanf(*++argv,"%f",&pole[2*numpol+1]))
                    fprintf(stderr,"Unable to read pole latitude.\n");
                argc--; break;
        case 'r':if(!sscanf(*++argv,"%f",&window[0]))
                    fprintf(stderr,"Unable to read range.\n");
                argc--;
                if(!sscanf(*++argv,"%f",&window[1]))
                    fprintf(stderr,"Unable to read range.\n");
                argc--;
                if(!sscanf(*++argv,"%f",&window[2]))
                    fprintf(stderr,"Unable to read range.\n");
                argc--;
                if(!sscanf(*++argv,"%f",&window[3]))
                    fprintf(stderr,"Unable to read range.\n");
                argc--; RANGE=TRUE; break;
        case 'x':if(!sscanf(*++argv,"%f",&dellon[numpol]))
                    fprintf(stderr,"Unable to read change in longitude.\n");
                argc--; break;
        case 'y':if(!sscanf(*++argv,"%f",&dellat[numpol]))
                    fprintf(stderr,"Unable to read change in latitude.\n");
                argc--; break;
    }
}
if (EDIT) {
    if(createfile(TMPFILE)) exit(1);
    else {
        saveout=outtype;
        outtype=ASCII;
    }
}
else if (fileout[0] != '\0') createfile(fileout);
outfile = fdopen(fdout,"w");
if(argc == 0) {
    strcat(filein,"stdout");
    infile = fdopen (fdin, "r");
    mapconv(1);
    fclose(infile);
}
else while (--argc >= 0) {
    strcat(filein, *argv++);
```

```
        if((infile=fopen(filein,"r")) == NULL) {
            fprintf(stderr,"Unable to open input file %s\n", *(argv-1));
        }
        else {
            mapconv(1);
            fclose(infile);
        }
    }
    fclose(outfile);
    if(minmax)
        fprintf(stderr,"Min lon=%f Max lon=%f Min lat=%f Max lat=%f\n",
            limit[0],limit[1],limit[2],limit[3]);
    if (EDIT) {
        system(strcat("ex " , TMPFILE));
        intype = ASCII;
        outtype = saveout;
        if((infile=fopen(TMPFILE,"r")) == NULL) {
            fprintf(stderr,"Unable to open input file %s from editor.\n", TMPFILE);
            exit(1);
        }
        if(fileout[0] != '\0') createfile(fileout);
        else {FORCE=TRUE; createfile(filein);}
        outfile = fdopen(fdout,"w");
        first=1;
        mapconv(0);
        fclose(infile);
        fclose(outfile);
        system(strcat("rm " , TMPFILE));
    }
}

#include <sys/types.h>
#include <sys/stat.h>

int createfile(name)
    char *name;
{
    struct stat buf;

    if(!FORCE && stat(name,&buf)==0) {
        fprintf(stderr,"Output file %s already exists.\n", name);
        fdout = STDOUTPUT;
        return(1);
    }
    if((fdout=creat(name,FILEMODE)) == -1) {
        fprintf(stderr,"Unable to create output file %s\n", name);
        fdout = STDOUTPUT;
        return(1);
    }
    return(0);
}


mapconv(way)
    int way;
{
    register i;
    int got;
    float point[2];
    char filename[40];
    long thisdata=0;
    int move=0;

    while(1) {
```

```
        again:if(intype) {
            if((got=fscanf(infile,"%f %f",&point[0],&point[1]))==EOF) return;
            if(got == 0 && way == 0) {
                if((got=fscanf(infile,"%s",filename)) == 1 ) {
                    fclose(outfile);
                    createfile(filename);
                    outfile = fdopen(fdout,"w");
                    goto again;
                }
            }
        }
        else
            {if(fread(point,4,2,infile)==0) return;}
        if(++thisdata >= firstdata && thisdata <= lastdata) {
            if(way) {
                if(point[1] > 990.0) { move++; goto out; }
                if(RANGE) for(i=0; i<2; i++) {
                    if(point[i] < window[2*i] || point[i] > window[2*i+1]) {
                        point[0]=999.0; point[1]=999.0;
                        move++; goto out;
                    }
                }
                for(i=0; i<=numpol; i++) {
                    if(pole[2*i+1] < 900.0) {
                        if(i!=thispole){
                            thispole=i;
                            refpt(pole[2*i],pole[2*i+1]);
                        }
                        newpol_(&point[0],&point[1]);
                    }
                    point[0] = point[0] + dellon[i];
                    point[1] = point[1] + dellat[i];
                }
                move=0;
            }
            if (minmax && point[1] < 989.0) {
                if(first) {
                    limit[0]=limit[1]=point[0];
                    limit[2]=limit[3]=point[1];
                    first = 0;
                }
                else for(i=0; i<2; i++) {
                    if(point[i] < limit[i * 2])    limit[i * 2]=point[i];
                    if(point[i] > limit[i * 2 +1])limit[i * 2 +1]=point[i];
                }
            }
    out: if(move<=1) {
                if(outtype) fprintf(outfile,"%12.6f  %12.6f\n",point[0],point[1]);
                else fwrite(point,4,2,outfile);
            }
        }
    }
}
```

```
c-*--MOVDRW--MAIN USER INTERFACE TO GEOPLOT-----------------------------
c
      subroutine movdrw(x,y,amode)
c
c-----Programmer  PLWard, USGS, Menlo Park, California, January, 1979.--
c
c-*--This subroutine is the primary user called subroutine in GEOPLOT.
c       x and y are the coordinates of a point in page units if mode is
c       positive and in data units if mode is negative. The values of
c
c
c      amode or mode can be one of the following:
c        1 move to the point
c        2 draw a vector to the point
c        3 move to the relative point
c        4 draw to the relative point
c        5 transform the coordinates to or from page units
c        6 ask device for the present pen position
c        7 ask device for the present crosshair position
c        8 initialize this plot device
c        9 set the device for hardware lettering
c       10 set the character size
c       11 set the line type
c       12 set the line shade (intensity or color)
c       13 new page, erase, or start a new frame
c       14 make a hardcopy of this display
c       15 dump the buffer, end this plot
c       16 output plot to a plotfile ending in .PL
c      - 9 draw great circle to new point (for map projections)
c      -10 draw rhumbline     to new point (for map projections)
c
c      Only modes 1 to 8 may be negative.
c
       save
c
       include 'gpcom.h'
       include 'lmccom.h'
c
       real points(2),x,y,xx,yy,slope,b,savhgt,tichx,tichy,angla(6)
       integer amode,i,j,mode,lasmod,lusym,levlab,levgrd,k,
      1   numseg(2),iglst(6),gcircl
c
       data iglst/7,2,7,3,0,0/
       data lusym/0/,lasmod/0/
       data angla/0.,180.,90.,270.,0.,0./
c
c
       if(idbug.lt.4)goto 4
            call printn("movdrw: %d %f %f",amode,x,y)
            call printn(" ibrk=%d kerr=%d usym=%f\n",ibreak,kerror,usym)
            call printn("movdrw: idbug=%d \n",idbug)
c
c-*--If break key was hit return.--------------------------------------
  4    if (ibreak.eq.-1.or.kerror.ge.2) return
c
c-*--Handle missing data.----------------------------------------------
       if(x.ne.usym.and.y.ne.usym) goto 5
       if(amode.ge.6) goto 5
            lusym=1
            return
5      mode=amode
c
c-*--CHECK FOR THE PROPER VALUE OF MODE.-------------------------------
       if(mode.ge.-i10.and.mode.le.i18.and.mode.ne.i0) goto 15
```

```
 10         kerror=1
            call gperr(kerror,"movdrw","Illegal mode of",mode)
            return
c
c-*--Set line type if necessary.-------------------------------------
 15    linmod=1
       if (ltype.eq.lintyp(1)) goto 16
            point(1)=lintyp(1)
            call gpdevo(i11,point)
            lintyp(1)=ltype
c
c-*--Set line shade if necessary.------------------------------------
 16    if (pshade.eq.shade(1)) goto 20
            point(1)=shade(1)
            call gpdevo(i12,point)
c
c-*--BUFFER INPUTS---------------------------------------------------
 20    point(1)=x
       point(2)=y
       kmode=iabs(mode)
c
c-*--If last point was missing data, then move if this is a draw.-------
       if(lusym.eq.i0)goto 21
            if(kmode.gt.i4)goto 21
            if(kmode.ne.i2.and.kmode.ne.i4)goto 2100
            lasmod=kmode
            kmode=kmode-1
            mode=mode-1
            if(mode.lt.i0)mode=mode+2
2100        lusym=0
21     continue
c
c-*--BRANCH ACCORDING TO THE VALUE OF MODE.--------------------------
c       mode= -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0 , 1  2  3  4
       goto (  39, 38, 80, 70, 60, 50, 32, 32, 40, 47,10 , 49,49,30,30,
      &50,60,70,80,22,22,22,22,130,22,22,22,170,170)  mode+11
c       5, 6, 7, 8, 9,10,11,12, 13,14,15,16, 17, 18----------------------
c
c-*--PASS VALUES THROUGH TO GPDEVO.----------------------------------
 22    continue
       if(idbug.ge.4) call printn("movdrw 22: kmode=%d\n",kmode)
       call gpdevo(kmode,point)
       return
c
c-*--IF A RELATIVE POINT IS INPUT, ADD IT TO LAST POINT.--------------
c       pointl is the last point, unclipped. It is in data coordinates
c       if lastyp=-1 and page coordinates if lastyp=1.-------------------
 30    if(lastyp.eq.i1) goto 35
            call gptran(i1,pointl)
            if(kerror.ne.i0) return
            lastyp=1
            goto 35
 32         if(lastyp.eq.-i1) goto 35
            call gptran(i2,pointl)
            if(kerror.ne.i0) return
            lastyp=-1
 35    point(1)=pointl(1)+point(1)
       point(2)=pointl(2)+point(2)
       kmode=kmode-2
       if(mode.gt.i0) goto 49
       if(kmode.eq.i1) goto 47
c
c-*--DRAW IN DATA COORDINATES.---------------------------------------
c---For certain transformations (linspa = 1 as set in subroutine transf)
```

```
c       interpolate in data space to draw a curve to the point in page
c       space.
 38    gcircl=1
       goto 40
 39    gcircl=2
 40    if(linspa.ne.i1.or.curinc.le.0.00001) goto 47
           if(ktrflg.eq.i1) call gptran(i3,point)
           if(kerror.ne.i0) return
           if(lastyp.eq.-i1) goto 41
                call gptran(i2,pointl)
                if(kerror.ne.i0) return
                lastyp=-1
 41        do 42 i=1,2
                points(i)=point(i)
 42             numseg(i)=abs((pointl(i)-point(i))/spalin(i))+0.5
           if(numseg(1)+numseg(2).eq.i0) goto 47
           nosym=1
           xx=1.0
           j=1
           if(numseg(2).gt.numseg(1))j=2
           yy=spalin(j)
           if(point(j).lt.pointl(j))yy=-yy
           if(gcircl.ge.1)goto 43
                slope=(point(3-j)-pointl(3-j))/(point(j)-pointl(j))
                b=pointl(3-j)-pointl(j)*slope
                goto 44
 43        if(gcircl.eq.1)call igtcir(pointl,point)
           if(gcircl.eq.2)call irhumb(pointl,point)
           if(j.eq.gcircl)goto 44
               j=gcircl
               yy=(point(j)-pointl(j))/numseg(3-j)
               numseg(j)=numseg(3-j)
 44        if(xx.ge.numseg(j)) goto 46
                point(j)=pointl(j)+xx*yy
                xx=xx+1.0
                if(gcircl.eq.0)point(3-j)=slope*point(j)+b
                if(gcircl.eq.1)call gtcir(point)
                if(gcircl.eq.2)call rhumb(point)
                if (kpltsp.eq.i19) goto 45
                     call gptran(i1,point)
                     if(kerror.ne.i0) return
 45             call gpclip(kpltsp)
                if(lasmod.eq.0)goto 44
                     kmode=lasmod
                     lasmod=0
                     call gpclip(kpltsp)
                     if (kerror.ne.i0.or.ibreak.eq.-1) return
                     goto 44
 46        point(1)=points(1)
           point(2)=points(2)
           gcircl=0
           nosym=0
c
c---CLIP AND OUTPUT A MOVE OR DRAW IN DATA COORDINATES TO THE DEVICE.--
 47    lastyp=-1
       pointl(1)=point(1)
       pointl(2)=point(2)
       if(kpltsp.eq.i19) goto 949
            call gptran(i1,point)
            if (kerror.ne.i0) return
949    call gpclip(kpltsp)
c----If last point was mising data, move, then draw to this one.
       if(lasmod.eq.0)return
948         kmode=lasmod
```

```
          lasmod=0
          call gpclip(kpltsp)
          return
c
c--*--CLIP AND OUTPUT A MOVE OR DRAW IN PAGE COORDINATES TO THE DEVICE.--
 49   lastyp=1
      pointl(1)=point(1)
      pointl(2)=point(2)
      if(kpltsp.ne.i19) goto 950
          call gptran(i2,point)
          if(kerror.ne.i0) return
950   call gpclip(kpltsp)
      if(lasmod.eq.0)return
      goto 948
c
c--*--TRANSFORM DATA TO PAGE (neg) OR PAGE TO DATA (pos) COORD.---MODE  5
 50   if(mode.gt.i0) call gptran(i2,point)
      if(mode.lt.i0) call gptran(i1,point)
 56   if (kerror.ne.i0) return
      x=point(1)
      y=point(2)
      if(idbug.ge.4)call printn("movdrw: %d %f %f\n",amode,x,y)
      return
c
c--*--ASK DEVICE FOR THE PRESENT PEN POSITION.--------------------MODE  6
 60   call gpdevo(kmode,point)
      if(kscale.ne.i0) call gpscal(i2,point)
      if(mode.lt.i0) call gptran(i2,point)
      goto 56
c
c--*--LOCATE CROSSHAIR POSITION FROM DEVICE.----------------------MODE  7
 70   if(y.gt.600.0) goto 75
          call gpdevo(kmode,point)
          if(kscale.ne.i0) call gpscal(i2,point)
          if(mode.lt.i0) call gptran(i2,point)
          amode=kmode
          goto 56
c-----Use pick instead of fare.
 75   kmode=2
      call gppick(kmode,point)
      amode=kmode
      goto 56
c
c--*--INITIALIZE GPCOM BLOCK AND THIS DEVICE.--------------------MODE  8
 80   if(idbug.ge.4)call printn("movdrw: ifirst=%d\n",ifirst)
      gcircl=0
      savhgt=height
      if(ifirst.ne.i1) goto 87
          do 81 i=1,6
 81           lablsp(i)=iglst(i)
          lmode=2
          levlab=10
          levgrd=10
          linclp=1
          ktrans=1
          linspa=0
          gtrans=1
          letchg=1
          ktrflg=1
          lineid=-9999
          nfare=2
          ksubj=0
          kchar=0
          kkchar=0
```

```
      nosym=0
      do 82 i=1,3
           lintyp(i)=1
           kside(i)=1
           width(i)=0.0
           space(i)=0.0
           shade(i)=100.0
           clip(i)=1.0
82         kside(i)=1
      do 820 i=1,6
           anglab(i)=angla(i)
           faclab(i)=1.0
           dislab(i)=-1.0
           if(i.eq.2.or.i.eq.4) dislab(i)=0.0
           spalab(i)=0.04
           if(i.gt.2) spalab(i)=0.03
           ticfac(i)=1.0
           maxpow(i)=5
           minlab(i)=8
           idtype(i)=0
           noaxis(i)=0
820        ticmin(i)=0.02
      kltrsp=1
      kpltsp=13
      offset(1)=0.0
      offset(2)=0.0
      pangle=0.0
      pgain(1)=1.0
      pgain(2)=1.0
      iup=2
      tichx=0.25
      tichy=0.25
      chrerr=0.0002
      symshp=0.0
      symlen=0.02
      symspa=0.001
      symang=0.5236
      symfac=0.0
      ticbas=0.0
      julian=0
      do 83 i=1,6,2
           pspace(18+i)=0.0
           pspace(19+i)=1.0
           pspace(i)=0.0
83         pspace(i+1)=1.0
      do 84 i=1,9
84         trans(i)=usym
      pole(1)=usym
      pole(2)=usym
      mapsca=1.0
      setsca=0.0
      do 85 k=1,16
85         omitsp(k)=0.0
      do 86 i=1,10
      j=i
      if(j.gt.5)j=j-5
      ticin(i)=0.03*(1.2-.2*j)
86    ticout(i)=0.0
      nomit=0
      linflg=0
      aspect=0.6
      spacel=0.6
      spaceh=0.6
      slopel=0.0
```

```
            savhgt=0.0
            curinc=0.01
            filnam="plot.PL\0"
 87     intact=0
        nfield=0
c-----Initialize the device
        call gpdevo(kmode,point)
        if(y.gt.0.0.and.ifirst.ne.i1) goto 88
c-----Initialize letmdrclp common block.
            kpenl=13
            jkl=0
            penl(1)=phome(1)
            penl(2)=phome(2)
c-----Initialize constants.
 88     do 89 i=1,3
            if(space(i).lt.rasmin)space(i)=rasmin
            if(width(i).lt.rasmin)width(i)=rasmin
 89         continue
 90     linmod=1
        point(1)=lintyp(1)
        call gpdevo(i11,point)
        call gpdevo(i12,shade)
        if(kchar.eq.0.and.savhgt.le.0.0)savhgt=0.01053
        if(savhgt.le.0.0)goto 91
            point(1)=savhgt
            call gpdevo(i10,point)
91      do 92  i=1,2
            fullsp(i)=devmax(i)
            if(fullsp(i).gt.2.0)fullsp(i)=1.0
            if(ifirst.ne.i1.and.pspace(i*2).le.fullsp(i)) goto 92
                pspace(2*i)=fullsp(i)
                call gplink(2*i)
                call gplink(i*2-1)
92          continue
            ifirst=0
            return
c
c-*--PAGE-----------------------------------------------------------MO
130     do 131 i=1,2
            penl(i)=phome(i)
            pointl(i)=phome(i)
            pointp(i)=phome(i)
131         point(i)=phome(i)
        lastyp=2
        jkl=0
        nfield=0
        goto 22
c
c-*--Part operators. Divide picture space up into parts.----------------
170     i=mode-16
        point(2)=fullsp(i)/point(2)
        i=2*i-1
        if(i.eq.3) goto 174
        pspace(i)=point(2)*(point(1)-1.0)
        pspace(i+1)=point(1)*point(2)
        goto 176
174     pspace(i)=fullsp(2)-point(1)*point(2)
        pspace(i+1)=fullsp(2)-point(2)*(point(1)-1.0)
176     call gplink(i)
        call gplink(i+1)
        return
c
        end
```

```
/* MSCALE - print map scale. Programmer PLWard
*/
mscale_(lon1,lat1,lon2,lat2,dist)
    float *lon1, *lat1, *lon2, *lat2, *dist;
{
    printf("mscale: from %f %f to %f %f is %f inches.\n",
        *lon1, *lat1, *lon2, *lat2, *dist);
}
```

```
/*
 *  Geocentric distance algorithms
 *  Calculate geocentric postitions, distances, and azimuths.
 *      refpt(lon,lat)                  call first to set reference point
 *      delaz(lon,lat, dis, az0, az1)   compute distance, etc.
 *      dback(dis, az0, lon,lat)        inverse: compute lat & lon
 *      ggtogc(lat)                     convert geographic lat to geocentric
 *      gctogg(lat)                     convert geocentric lat to geographic
 *
 *  Above lat and lon in radians, delta in center of earth angle.
 *  For the following x,y points are in degrees and returns km dist.
 *      distance = earthdist(x0, y0, x1, y1)
 *      see disazm and newpol descriptions below
 *      Programmers Bruce Julian, Jim Herriot, PLWard, July 1983.
 */

#include <math.h>

#define  PI             3.14159265
#define  HALFPI         (0.5*PI)
#define  TWOPI          (2.0*PI)
#define  DEG            (180./PI)       /* Degrees/Radian       */
#define  RAD            (PI/180.)       /* Radians/Degree       */

#define  EQTRAD         6378.163        /* ~julian/include/earthconst.h */
#define  POLRAD         6356.177
#define  REARTH         6371.
#define  FLATTENING     ((EQTRAD-POLRAD)/EQTRAD)
#define  C1             pow(1.0-FLATTENING,2.0)
#define  C2             (HALFPI*(1.0/C1-1.0))
#define  THMAX          0.02

double signn(a,b)
    double a,b;
{
    return(b<0.0?-a:a);
}

/* Convert from geographic to geocentric latitude */
double ggtogc(lat)
    double lat;
{
    return( ((HALFPI-fabs(lat))<THMAX) ?
        lat/C1-signn(C2,lat) : atan(C1 * tan(lat)) );
}

/* Convert from geocentric to geographic latitude */
double gctogg(lat)
    double lat;
{
    return( ((HALFPI-fabs(lat))<THMAX) ?
        C1 * (lat+signn(C2,lat)) : atan(tan(lat)/C1) );
}

static double st0, ct0, phi0;

/* Store geocentric coordinates of reference point */

refpt(lon,lat)
    double lon,lat;
{
    st0 = cos(lat);
    ct0 = sin(lat);
    phi0 = lon;
```

```
}

/* Calculate the geocentric distance, azimuths */

delaz(lon,lat, delta, az0, az1)
    double    lon,lat;
    float  *delta, *az0, *az1;
{
    double st1,ct1,dlon,sdlon,cdlon;
    float sdelt,cdelt;

    ct1    = sin(lat);
    st1    = cos(lat);
    sdlon = sin(lon - phi0);
    cdlon = cos(lon - phi0);
    cdelt = st0*st1*cdlon + ct0*ct1;
    cvrtop (st0*ct1-st1*ct0*cdlon, st1*sdlon, &sdelt, az0);
    *delta = atan2((double)sdelt, (double)cdelt);
    cvrtop (st1*ct0-st0*ct1*cdlon, -sdlon*st0, &sdelt, az1);
}

/* Calculate geocentric coordinates of secondary point */

dback (delta, az0, lon,lat)
    double delta,az0, *lon, *lat;
{
    double sdelt,cdelt,cz0,ct1;
    float st1,dlon;

    sdelt = sin(delta);
    cdelt = cos(delta);
    cz0    = cos(az0);
    ct1    = st0*sdelt*cz0 + ct0*cdelt;
    cvrtop (st0*cdelt-ct0*sdelt*cz0, sdelt*sin(az0), &st1, &dlon);
    *lat   = atan2(ct1, (double) st1);
    *lon   = phi0 + dlon;
    if (fabs(*lon) > PI) *lon -= (*lon>0. ? TWOPI : -TWOPI);
}

/* cvrtop - convert from rectangular to polar coordinates */
/* copied from /we/julian/src/lib/math/cvrtop.c on the 70 */

cvrtop(x,y,r,theta)
    double x,y;
    float  *r, *theta;
{
    *r = hypot(x, y);
    *theta = atan2(y, x);
}

/* kilometers between two x,y points in degrees */

double earthdist(x0, y0, x1, y1)
    double x0, y0, x1, y1;
{
    double delta,az0,az1;

    az0 = x0 * RAD; az1 = ggtogc(y0 * RAD);
    refpt(az0,az1);
    delaz(x1*RAD,ggtogc(y1*RAD),&delta,&az0,&az1);

    return(delta * REARTH);
}
```

```
/* DISAZM -- driver routine used in geoplot -- PLWard, July,1983
 Type
  1   Input lon and lat of two points, return distance in km, azimuth
        from a to b and backazimuth from b to a. Calculates geocentric
        values.
  2   Same as 1 but calculates geographic values (spherical earth).
  3   Input lon and lat of a point and distance and azimuth to 2nd
        point. Return lon and lat of second point. Calculates
        geocentric values.
  4   Same as 3 but calculates geographic values (spherical earth).
*/

disazm_(arg1,arg2,arg3,arg4,type)
   float *arg1, *arg2, *arg3, *arg4;
   int type;
{
   double lo1,la1,lo2,la2;

   lo1 = *arg1 * RAD; la1 = *arg2 * RAD;
   lo2 = *arg3 * RAD; la2 = *arg4 * RAD;
   switch (type) {
      case  1: la1=ggtogc(la1); la2=ggtogc(la2);
      case  2: refpt(lo1,la1);
               delaz(lo2,la2,arg1,arg2,arg3);
               *arg1 = *arg1 * REARTH;
               *arg2 = *arg2 * DEG; *arg3 = *arg3 * DEG;
               break;
      case  3: la1=ggtogc(la1);
      case  4: refpt(lo1,la1);
               lo2 = *arg3/REARTH;
               dback(lo2,la2,&lo1,&la1);
               if(type==3) la1=gctogg(la1);
               *arg1 = lo1 * DEG; *arg2 = la1 * DEG; *arg3 = 0.0;
   }
}

/* NEWPOL - find new lon and lat of point rotated about a pole (refpt)*/

newpol_(longi,latit)
   float *longi, *latit;
{
   double lon,lat;
   float delt,azim,fake;

   lon= *longi * RAD; lat = *latit * RAD;
   delaz(lon,lat,&delt,&azim,&fake);
   *longi = (PI - azim) * DEG;
   *latit = (HALFPI - delt) * DEG;
}


#ifdef DEBUG

main(argc, argv) char **argv; {
   float x0,y0,x1,y1;
   int ty;

   if(argc != 6) {
      printf("Usage: geocen lon1 lat1 lon2 lat2 type\n"); exit(1);
   }

   x0 = atof(argv[1]);
   y0 = atof(argv[2]);
   x1 = atof(argv[3]);
```

```
    y1 = atof(argv[4]);
    ty = atoi(argv[5]);

    disazm_(&x0,&y0,&x1,&y1,ty);

    printf("%g %g ",x0,y0);
    if(ty<3)printf("%g",x1);
    printf("\n");
}

#endif
```

```
This directory contains source code and an archive in plot10.a for a
      FAKE TEKTRONIX PLOT 10 PACKAGE WHICH RUNS OFF GEOPLOT

      Programmer Barbara Bekins, PRI, April 1980

      This package contains entry points for all the subroutines in the
      tektronix plot10 package.  The routines simply call the
      appropriate routine in geoplot where possible.

      The following routines are not implememted:
        svstat and restat since the terminal control system is not used
        recovr
        ttblsz,settab,rsttab,tabhor,tabver,setmrg,tcslev
        toutst,toutpt
        tinstr,tinput,a1in,ainst,setbuf and seebuf
        leftio

As of 11/10/80 the routines bell,incplt,and czaxis do not work
because the geoplt routines they depend on dont.
       common/plot10/ page(2),kline,kcode,ntran,nterm,
      & pointp(2),pspace(24),dpgin
       subroutine a1out(nchar,iarray)
       entry aoutst(nchar,iarray)
       include 'plot10.h'
       character*81 iarray
c-------------------------------------------------------------------
c      Outputs a string of 'nchar' characters in iarray
c-------------------------------------------------------------------
       call letter(0.0,0.0,3,iarray,0.,8)
c----- update the position of the graphics beam--------------------
       jx=linwdt(nchar)
       jy=0
       call rtop(jx,jy,ax,ay)
       call movdrw(ax,ay,1)
       return
       end
       subroutine ancho(char)
       include 'plot10.h'
       character char(4)
c-------------------------------------------------------------------
c      Outputs a single alpha-numeric character to the screen
c-------------------------------------------------------------------
       jx=linwdt(1)
       jy=0
       call letter(0.0,0.0,3,char(3),0.,8)
c-----Update the graphics beam past the letter plotted
       call rtop(jx,jy,ax,ay)
       call movdrw(ax,ay,3)
       return
       end
       subroutine anmode
c-------------------------------------------------------------------
c      Switch to alpha-numeric mode by dumping the output buffer
c-------------------------------------------------------------------
       call movdrw(0.,0.,15)
       return
       end
       subroutine anstr(nstrng,num)
       integer nstrng(1)
       include 'plot10.h'
       character char(4),string(81)
       character*81 eqstr
       equivalence(string,eqstr),(jchar,char)
c-------------------------------------------------------------------
```

```
c      Outputs a string of ade characters
c-------------------------------------------------------------
       do 500 i=1,num
       jchar=nstrng(i)
       string(i)=char(3)
500    continue
       jx=linwdt(num)
       jy=0
       call letter(0.0,0.0,3,eqstr,0.,8)
c-----Update the graphics beam past the letter plotted
       call rtop(jx,jy,ax,ay)
       call movdrw(ax,ay,3)
       return
       end
       subroutine baksp
       include 'plot10.h'
c-------------------------------------------------------------
c      Generates a backspace.
c-------------------------------------------------------------
       jx=linwdt(-1)
       jy=0
c-----pick up last point plotted from common
       call getcom(16,2,pointp)
c-----Update the graphics beam past the letter plotted
       call rtop(jx,jy,ax,ay)
       call movdrw(ax,ay,3)
       return
       end
       subroutine bell
c-------------------------------------------------------------
c      Rings the bell on the terminal
c-------------------------------------------------------------
       call printn("^G")
       return
       end
       subroutine cartn
       include 'plot10.h'
c-------------------------------------------------------------
c      Generates a carriage return.
c-------------------------------------------------------------
c-----pick up last point plotted from common
       call getcom(16,2,pointp)
c-----Update the graphics beam past the letter plotted
       x=0.0
       y=pointp(2)
       call movdrw(x,y,1)
       return
       end
       subroutine chrsiz(ichar)
c-------------------------------------------------------------
c      Sets character size: from 4=normal to 1=74 per line
c-------------------------------------------------------------
       height=ichar
       call putcom(91,1,height)
       return
       end
       subroutine csize(ihorz,ivert)
c-------------------------------------------------------------
c      Returns width and height of current character size
c-------------------------------------------------------------
       real cspace(5)
       include 'plot10.h'
       call getcom(91,5,cspace)
       ihorz=(cspace(4)+1.)*cspace(1)*cspace(2)*page(1)+.5
```

```
      ivert=(cspace(5)+1.)*cspace(1)*page(2)+.5
      return
      end
      subroutine czaxis(icode)
      include 'plot10.h'
      real shades(3),shade(3)
      data shades/100.,-100.,1000./
c-----------------------------------------------------------------------
c     Sets brightness and storage properties of tube depending
c     on icode:          0=normal vectors
c                              1=defocused vectors
c                              2=enable write through mode
c-----------------------------------------------------------------------
      kcode=icode
      do 10 i=1,3
10    shade(i)=shades(icode+1)
      call putcom(87,3,shade)
      return
      end
      subroutine dasha(dx,dy,line)
      include 'plot10.h'
c-----------------------------------------------------------------------
c     Draws a dashed line from present point to the point
c     (dx,dy) in data space.
c-----------------------------------------------------------------------
      mode=-2
      x=dx
      y=dy
      kline=line
      typel=linset(line,mode)
      call putcom(181,1,typel)
      call movdrw(x,y,mode)
      return
      end
      subroutine dashr(dx,dy,line)
      include 'plot10.h'
c-----------------------------------------------------------------------
c     Draws a dashed line from present point to data values,
c     (dx,dy) relative to the present point.
c-----------------------------------------------------------------------
      mode=-4
      x=dx
      y=dy
      kline=line
      typel=linset(line,mode)
      call putcom(181,1,typel)
      call movdrw(x,y,mode)
      return
      end
      subroutine dashsa(dx,dy,line)
      include 'plot10.h'
c-----------------------------------------------------------------------
c     Draws a dashed arc to absolute virtual point (dx,dy)
c-----------------------------------------------------------------------
      mode=-2
      kline=line
      typel=linset(line,mode)
      call putcom(181,3,typel)
      call movdrw(dx,dy,mode)
      return
      end
      subroutine dashsr(dx,dy,line)
      include 'plot10.h'
c-----------------------------------------------------------------------
```

```
c     Draws a dashed arc to relative vitual coordinates (dx,dy)
c-------------------------------------------------------------------
      mode=-4
      kline=line
      typel=linset(line,mode)
      call putcom(181,3,typel)
      call movdrw(dx,dy,mode)
      return
      end
      subroutine drawa(dx,dy)
c-------------------------------------------------------------------
c    This routine draws a vector to the dataspace coordinates dx,dy
c-------------------------------------------------------------------
      typel=1
      mode=-2
      call putcom(181,1,typel)
      call movdrw(dx,dy,mode)
      return
      end
      subroutine drawr(dx,dy)
c-------------------------------------------------------------------
c    Draws a vector to relative dataspace coordinates
c-------------------------------------------------------------------
      typel=1
      mode=-4
      call putcom(181,1,typel)
      call movdrw(dx,dy,mode)
      return
      end
      subroutine drawsa(dx,dy)
c-------------------------------------------------------------------
c     Draws an arc in virtual space from present point to the
c     input point
c-------------------------------------------------------------------
      mode=-2
      typel=1
      call putcom(181,3,typel)
      call movdrw(dx,dy,mode)
      return
      end
      subroutine drawsr(dx,dy)
c-------------------------------------------------------------------
c     Draws an arc from present point to the relative value(dx,dy)
c-------------------------------------------------------------------
      mode=-4
      typel=1
      call putcom(181,3,typel)
      call movdrw(dx,dy,mode)
      return
      end
      subroutine drwabs(ix,iy)
c-------------------------------------------------------------------
c     Draw to absolute coordinates ix,iy from present position
c-------------------------------------------------------------------
      mode=2
      typel=1
c----convert raster coordinates to page units------------
      call rtop(ix,iy,x,y)
      call putcom(181,1,typel)
      call movdrw(x,y,mode)
      return
      end
      subroutine drwrel(ix,iy)
c-------------------------------------------------------------------
```

```
c     Draw to relative coordinates ix,iy
c-------------------------------------------------------------------
      typel=1
      mode=4
c-----convert raster coordinates to page units-------------
      call rtop(ix,iy,x,y)
      call putcom(181,1,typel)
      call movdrw(x,y,mode)
      return
      end
      subroutine dshabs(ix,iy,line)
      include 'plot10.h'
c-------------------------------------------------------------------
c     Draws a dashed line of type 'line' from the present
c     point to the point in raster coordinates(ix,iy)
c-------------------------------------------------------------------
      call rtop(ix,iy,x,y)
      mode=2
      kline=line
      typel=linset(line,mode)
      call putcom(181,1,typel)
      call movdrw(x,y,mode)
      return
      end
      subroutine dshrel(ix,iy,line)
      include 'plot10.h'
c-------------------------------------------------------------------
c     Draws a dashed line of type 'line' from the present point to
c     the raster values (ix,iy) relative to the present point.
c-------------------------------------------------------------------
      call rtop(ix,iy,x,y)
      mode=4
      kline=line
      typel=linset(line,mode)
      call putcom(181,1,typel)
      call movdrw(x,y,mode)
      return
      end
      subroutine dwindo(xmin,xmax,ymin,ymax)
      include 'plot10.h'
c-------------------------------------------------------------------
c    This routine also sets the values of dataspace,but accepts
c    different inputs.
c-------------------------------------------------------------------
      pspace(20)=xmax
      pspace(22)=ymax
      pspace(19)=xmin
      pspace(21)=ymin
      call putcom(39,4,pspace(19))
      return
      end
      subroutine erase
c-------------------------------------------------------------------
c     Erases the page
c-------------------------------------------------------------------
      call movdrw(0.,0.,13)
      return
      end
      subroutine finitt(ix,iy)
c-------------------------------------------------------------------
c     Moves the cursor to ix,iy and dumps the output buffer
c-------------------------------------------------------------------
      call rtop(ix,iy,x,y)
      call movdrw(x,y,1)
```

```
      call gpfin
      return
      end
      subroutine hdcopy
c-----------------------------------------------------------------------
c     Makes a hardcopy of the screen
c-----------------------------------------------------------------------
      call movdrw(0.,0.,14)
      return
      end
      subroutine home
      real phome(2)
c-------------------------------------------------------------
c     Moves the cursor to the upper left corner.
c-------------------------------------------------------------
      call getcom(5,2,phome)
      call movdrw(phome(1),phome(2),1)
      return
      end
      subroutine incplt(ionoff,idir,no)
      integer mdir(2,8)
      data mdir/0,1,1,1,1,0,1,-1,0,-1,-1,-1,-1,0,-1,1/
c-------------------------------------------------------------
c     Plots or moves depending on whether ionoff is 1 or 0,
c     'no' raster points in the direction given by 'idir'
c-------------------------------------------------------------
      mode=ionoff+3
      jx=no*mdir(1,idir+1)
      jy=no*mdir(2,idir+1)
c----plot these as relative coordinates
      call rtop(jx,jy,x,y)
      call movdrw(x,y,mode)
      return
      end
      subroutine initt(ibaud)
      real dum(2)
      character*2 terms(4)
      include 'plot10.h'
      data terms/"0 ","2 ","4 ","??"/,nterm/1/
c-----------------------------------------------------------------------
c     Initialize the terminal specified by term.  Default is tek4010
c-----------------------------------------------------------------------
      if(nterm.le.0.or.nterm.gt.4) nterm=1
      iplton=1
      tbase=0.0
      izone=-8
      isym=99999
      call ploton(iplton,"Rtek"//terms(nterm),tbase,izone,isym)
      call getcom(3,2,dum)
      page(1)=dum(1)
      page(2)=dum(2)
      call getcom(7,1,dpgin)
      if(ibaud.ne.-1) call movdrw(0.0,0.0,13)
      call home
      return
      end
      function kcm(rc)
      include 'plot10.h'
c-----------------------------------------------------------------------
c     Transforms centimeters to raster units
c-----------------------------------------------------------------------
      kcm=page(1)*rc/(dpgin*2.54) + .5
      return
      end
```

```
      function kin(ri)
      include 'plot10.h'
c-----------------------------------------------------------------------
c     Tranforms inches to raster units
c-----------------------------------------------------------------------
      kin=page(1)*ri/dpgin + .5
      return
      end
      function leftio(idum)
      return
      end
      subroutine linef
      include 'plot10.h'
c-------------------------------------------------------------
c     generates a linefeed
c-------------------------------------------------------------
      jx=0
      jy=linhgt(-1)
c-----Update the graphics beam past the letter plotted
      call rtop(jx,jy,ax,ay)
      call movdrw(ax,ay,3)
      return
      end
      function linhgt(num)
c-----------------------------------------------------------------------
c     Returns height in raster units of num lines of present size
c-----------------------------------------------------------------------
      call csize(ihorz,ivert)
      linhgt=ivert*num
      return
      end
      function linset(line,mode)
      include 'plot10.h'
c-----------------------------------------------------------------------
c     Relate requested line type in 'line' to geoplot known types
c-----------------------------------------------------------------------
      data ltyp/0/
c
      linset=1
      kline=line
      move=-1
      if(mode.lt.0) move=1
c-----set kline type in geoplot according to value of 'line'
      if(kline) 310,400,330
c-----move to the point
310   continue
      mode=mode+move
      goto 400
c-----various types of dashes
330   continue
      if(kline-9) 340,350,360
c-----geoplot known type: set 'typel' to 'kline'+2
340   if(kline.gt.4) kline=4
      linset=kline+2
      goto 400
c-----type 9: alternate moves and draws with each call
350   if(ltyp.ne.9) icount=0
      if(mod(icount,2).eq.1) mode=mode+move
      icount=icount+1
      linset=1
      goto 400
c-----'kline' is concatenated pattern of moves and draws
360   continue
      linset=kline
```

```
400   continue
      ltyp=kline
      return
      end
      subroutine lintrn
      include 'plot10.h'
c-----------------------------------------------------------------------
c     Sets linear transformation(default, but must be reset
c     log or polar has been set).
c-----------------------------------------------------------------------
      ntran=1
      trans=1
      call putcom(190,1,trans)
      return
      end
      function linwdt(num)
c-----------------------------------------------------------------------
c     Returns width in raster units of num characters of
c     present size
c-----------------------------------------------------------------------
      call csize(ihorz,ivert)
      linwdt=ihorz*num
      return
      end
      subroutine logtrn(key)
      include 'plot10.h'
      integer lgtran(3)
      data lgtran/4,2,5/
c-----------------------------------------------------------------------
c     Sets one or both axes logrithmic according to 'key'
c               1=x axis logrithmic, y linear
c                       2=y axis logrithmic, y linear
c                       3=both x and y logrithmic
c-----------------------------------------------------------------------
      if(key.gt.3.or.key.lt.1) return
      trans=lgtran(key)
      ntran=2
      call putcom(190,1,trans)
      return
      end
      subroutine movabs(ix,iy)
c-----------------------------------------------------------------------
c     Move to absolute raster coordinates ix,iy
c-----------------------------------------------------------------------
      mode=1
c-----convert raster coordinates to page units------------
      call rtop(ix,iy,x,y)
      call movdrw(x,y,mode)
      return
      end
      subroutine movea(dx,dy)
c-----------------------------------------------------------------------
c    This routine moves to the dataspace coordinates dx,dy
c-----------------------------------------------------------------------
      mode=-1
      call movdrw(dx,dy,mode)
      return
      end
      subroutine mover(dx,dy)
c-----------------------------------------------------------------------
c    Moves to relative dataspace coordinates
c-----------------------------------------------------------------------
      mode=-3
      call movdrw(dx,dy,mode)
```

```
      return
      end
      subroutine movrel(ix,iy)
c-----------------------------------------------------------------------
c     Move to relative coordinates ix,iy from present position
c-----------------------------------------------------------------------
      mode=3
c----convert raster coordinates to page units-----------
      call rtop(ix,iy,x,y)
      call movdrw(x,y,mode)
      return
      end
      subroutine newlin
      include 'plot10.h'
c-------------------------------------------------------------
c     Starts a newline
c-------------------------------------------------------------
      jy=linhgt(-1)
c----pick up last point plotted from common
      call getcom(16,2,pointp)
c----Update the graphics beam past the letter plotted
      call rtop(jx,jy,ax,ay)
      x=0.0
      y=pointp(2)+ay
      call movdrw(x,y,1)
      return
      end
      subroutine newpag
c-----------------------------------------------------------
c     Generates a new page
c-----------------------------------------------------------
      call movdrw(0.,0.,13)
      return
      end
      subroutine pntabs(ix,iy)
c-----------------------------------------------------------------------
c     Plot a point at ix,iy
c-----------------------------------------------------------------------
      typel=0
      mode=2
c----convert raster coordinates to page units-----------
      call rtop(ix,iy,x,y)
      call putcom(181,1,typel)
      call movdrw(x,y,mode)
      return
      end
      subroutine pntrel(ix,iy)
c-----------------------------------------------------------------------
c     Plot a point at the relative coordinates ix,iy
c-----------------------------------------------------------------------
      typel=0
      mode=4
c----convert raster coordinates to page units-----------
      call rtop(ix,iy,x,y)
      call putcom(181,1,typel)
      call movdrw(x,y,mode)
      return
      end
      subroutine pointa(dx,dy)
c-----------------------------------------------------------------------
c    This routine plots a point at the dataspace coordinates dx,dy
c-----------------------------------------------------------------------
      typel=0
      mode=-2
```

```
      call putcom(181,1,typel)
      call movdrw(dx,dy,mode)
      return
      end
      subroutine pointr(dx,dy)
c-----------------------------------------------------------------
c     Plots a point at relative dataspace coordinates
c-----------------------------------------------------------------
      typel=0
      mode=-4
      call putcom(181,1,typel)
      call movdrw(dx,dy,mode)
      return
      end
      subroutine poltrn(angmin,angmax,idum)
      include 'plot10.h'
c-----------------------------------------------------------------
c     Sets x axis as radius, y axis as angle so that all
c     vitual points are transformed. Radius suppression
c     is not implemented. Limits of the virtual window
c     are set to angmin and angmax
c-----------------------------------------------------------------
      pspace(19)=angmin
      pspace(20)=angmax
      call putcom(39,2,pspace(19))
      trans=10.
      ntran=3
      call putcom(190,1,trans)
      return
      end
      subroutine recovr
      return
      end
      subroutine reset
      real scale(3)
      data scale/0.0,1.0,1.0/
c-----------------------------------------------------------------
c     Resets plot to no gain or rotation
c-----------------------------------------------------------------
      call putcom(72,3,scale)
      call initt(-1)
      return
      end
      subroutine rrotat(deg)
c-----------------------------------------------------------------
c     Sets factor to rotate plot 'deg' degrees (default=0.)
c-----------------------------------------------------------------
      pangle=deg
      call putcom(72,1,pangle)
      return
      end
      subroutine rscale(fac)
      real pgain(2)
c-----------------------------------------------------------------
c     Sets scaling factor for plot-(default=1)
c-----------------------------------------------------------------
      pgain(1)=fac
      pgain(2)=fac
      call putcom(73,2,pgain)
      return
      end
      subroutine rtop(ix,iy,x,y)
      include 'plot10.h'
c-----------------------------------------------------------------
```

```
c     Convert raster coordinates to page coordinates
c-------------------------------------------------------------
c      call printn("rtop:%n %n %f %f",ix,iy,x,y)
      x=ix/page(1)
      y=iy/page(2)
      return
      end
      subroutine scursr(ichar,ix,iy)
      entry dcursr(ichar,ix,iy)
      include 'plot10.h'
c-------------------------------------------------------------
c     Displays crosshairs and waits for typed character. Upon
c     receiving, puts coordinates of crosshairs in ix and iy
c     and the ade vlue of the typed character in ichar
c-------------------------------------------------------------
      mode=7
      call movdrw(x,y,mode)
      ix=page(1)*x+.5
      iy=page(2)*y+.5
      ichar=mode
      return
      end
      subroutine seedw(xmin,xmax,ymin,ymax)
      include 'plot10.h'
c-------------------------------------------------------------------
c     Returns the current values of the virtual window limits
c-------------------------------------------------------------------
      call getcom(39,4,pspace(19))
      xmin=pspace(19)
      xmax=pspace(20)
      ymin=pspace(21)
      ymax=pspace(22)
      return
      end
      subroutine seeloc(ix,iy)
      include 'plot10.h'
c-----------------------------------------------------------------
c     Locate last position of the graphics beam
c-----------------------------------------------------------------
      call getcom(16,2,pointp)
      ix=pointp(1)*page(1)+.5
      iy=pointp(2)*page(2)+.5
      return
      end
      subroutine seemod(line,izaxis,idum)
      include 'plot10.h'
c-----------------------------------------------------------------
c     Returns current linetype and z-axis mode(brightness),
c     but not software mode
c-----------------------------------------------------------------
      line=kline
      izaxis=kcode
      return
      end
      subroutine seerel(rcos,rsin,rscale)
c-----------------------------------------------------------------
c     Returns values of the common variables used to rotate
c     and scale plots
c-----------------------------------------------------------------
      parameter (RAD=0.0174533334)
      call getcom(72,1,pangle)
      rcos=cos(pangle*RAD)
      rsin=sin(pangle*RAD)
      call getcom(73,1,rscale)
```

```
      return
      end
      subroutine seetrm(idum,iterm,nchar,maxsr)
      include 'plot10.h'
      real devmax(2)
c-----------------------------------------------------------------------
c     Returns terminal type as set by term,character size,
c     and maximum addressable raster value. Doesn't return
c     baud rate.
c-----------------------------------------------------------------------
      iterm=nterm
      call getcom(265,1,height)
      nchar=height
      call getcom(1,1,devmax)
      maxsr=page(1)*devmax(1)
      return
      end
      subroutine seetrn(xfac,yfac,key)
      include 'plot10.h'
c-----------------------------------------------------------------------
c     Returns factors used by the transforming routines. These
c     depend on the screen window, and the data space values.
c     Also returns the current tranformation number.
c-----------------------------------------------------------------------
      call getcom(21,24,pspace)
      xfac=(pspace(2)-pspace(1))/(pspace(20)-pspace(19))*page(1)
      yfac=(pspace(4)-pspace(3))/(pspace(22)-pspace(21))*page(2)
      key=ntran
      return
      end
      subroutine seetw(minx,maxx,miny,maxy)
      include 'plot10.h'
c-----------------------------------------------------------------------
c     Returns limits of screen window in raster units
c-----------------------------------------------------------------------
      call getcom(21,4,pspace)
      minx=pspace(1)*page(1)+.5
      maxx=pspace(2)*page(1)+.5
      miny=pspace(3)*page(2)+.5
      maxy=pspace(4)*page(2)+.5
      return
      end
      subroutine svstat(dum)
      entry restat(dum)
      return
      end
      subroutine swindo(minx,lenx,miny,leny)
      include 'plot10.h'
c-----------------------------------------------------------------------
c     This routine sets the picture space, which is the space
c     on the screen that the plot goes in.
c-----------------------------------------------------------------------
      call rtop(minx+lenx,miny+leny,pspace(2),pspace(4))
      call rtop(minx,miny,pspace(1),pspace(3))
      call putcom(21,4,pspace)
      return
      end
      subroutine term(iterm,iscale)
      include 'plot10.h'
c-----------------------------------------------------------------------
c    Sets terminal type indicated by value of iterm:
c                0       4010    2       4014
c                        1       4012    3       4014 with enhanced graphics
c    'iscale' is ignored because geoplot sets it automatically
```

```
c-------------------------------------------------------------
      nterm=iterm+1
      return
      end
      subroutine tinstr(idum,jdum)
      entry tinput(idum)
      entry alin(idum,jdum)
      entry ainst(idum,jdum)
      entry setbuf(idum)
      entry seebuf(idum)
      return
      end
      subroutine toutst(idum,jdum)
      entry toutpt(idum)
      return
      end
      subroutine tsend
c-------------------------------------------------------------
c     Dump the output buffer
c-------------------------------------------------------------
      call movdrw(0.,0.,15)
      return
      end
      subroutine twindo(minx,maxx,miny,maxy)
      include 'plot10.h'
c-------------------------------------------------------------
c     This routine is the same as 'swindo' except that the limits
c     of the window are specified differently
c-------------------------------------------------------------
      call rtop(maxx,maxy,pspace(2),pspace(4))
      call rtop(minx,miny,pspace(1),pspace(3))
      call putcom(21,4,pspace)
      return
      end
      subroutine vcursr(ichar,dx,dy)
c-------------------------------------------------------------
c     Same as above but returned coordinates are converted to
c     dataspace.
c-------------------------------------------------------------
      mode=-7
      call movdrw(dx,dy,mode)
      ichar=mode
      return
      end
      subroutine vwindo(xmin,xrange,ymin,yrange)
      include 'plot10.h'
c-------------------------------------------------------------
c     This routine sets the values of dataspace in geoplot
c     Data values are scaled using data space to fit in the
c     designated picture space on the screen.
c-------------------------------------------------------------
c
      pspace(20)=xmin+xrange
      pspace(22)=ymin+yrange
      pspace(19)=xmin
      pspace(21)=ymin
      call putcom(39,4,pspace(19))
      return
      end
```

```
c-----PLOTS---Emulates versatec plot software using geoplot calls.
c------Programmer Dave Oppenheimer
c
c
      subroutine axis (x,y,label,nchar,axlen,angle,fval,dv)
c
c axis - plot an annotated axis.
c
c call axis (x,y,label,nchar,axlen,angle,fval,dv)
c
c       (x,y)   = starting coordinates for axis generation
c       label   = alphanumeric text string for labelling the axis
c       nchar   = number of characters in the axis label
c                   + = annotations are generated above the axis
c                   - = annotations are generated below the axis
c       axlen   = axis length in inches
c       angle   = angle in degrees at which axis is to be drawn
c       fval    = first annotation value
c       dv       = delta annotation value
c
c called by:  user program
c
c calls:  number, plot, symbol
c
c commons used:  none
c
c-
c
      character*(*) label
      data  radn/0.01745329/
c
c...  locate which side of the axis to annotate and label
      side = +1.0
      nc = nchar
c
c...  non-negative nchar?
      if (nc .ge. 0) goto 10
      nc = -nc
      side = -1.0
c
c..   determine value of dv exponent
   10 exp = 0.0
      adv = abs(dv)
c
c...  zero delta annotation value?
      if (adv .eq. 0.0) goto 50
c
c...  dv exponent calculation completed?
   20 if (adv .lt. 99.0) goto 40
      adv = adv / 10.0
      exp = exp + 1.0
      goto 20
c
   30 adv = adv * 10.0
      exp = exp - 1.0
c
c...  dv exponent calculation completed?
   40 if (adv .lt. 0.01) goto 30
c
c...  compute normalized fval and dv
   50 val = fval * (10.0**(-exp))
      adv = dv * (10.0**(-exp))
c
c...  set up angular orientation variables
```

```
      t2 = angle * radn
      sina = sin(t2)
      cosa = cos(t2)
c
      dx = -0.1
      dy = 0.15*side - 0.05
      xx = x + dx*cosa - dy*sina
      yy = y + dy*cosa + dx*sina
c
c...  annotate axis
      ntic = axlen + 1.0
      do 60 i = 1,ntic
         call number(xx,yy,0.105,val,angle,2)
         val = val + adv
         xx = xx + cosa
   60    yy = yy + sina
c
c...  label axis
      t2 = nc
c
c...  does dv exponent exist?
      if (exp .ne. 0.0) t2 = nc + 6
c
      dx = -0.07*t2 + 0.5*axlen
      dy = 0.325*side - 0.075
      xx = x + dx*cosa - dy*sina
      yy = y + dy*cosa + dx*sina
      call symbol(xx,yy,0.14,label,angle,nc)
c
c...  no dv exponent to plot?
      if (exp .eq. 0.0) goto 70
c
c...  plot exponent
c       call symbol(999.,999.,0.14,5h   *10,angle,5)
      t2 = nc + 5
      xx = xx + (t2*cosa - 0.8*sina)*0.14
      yy = yy + (t2*sina + 0.8*cosa)*0.14
      call number(xx,yy,0.07,exp,angle,-1)
c
c...  draw axis and tic marks
   70 dx = -0.07 * side * sina
      dy = +0.07 * side * cosa
      xx = x
      yy = y
      call plot(xx,yy,3)
      do 80 i = 1,ntic
         call plot(xx,yy,2)
         call plot(xx+dx,yy+dy,2)
         call plot(xx,yy,3)
         xx = xx + cosa
   80    yy = yy + sina
c
      return
c
      end
         subroutine cursor(x,y)
c
c  subroutine cursor
c  places crosshairs on screen, returns x,y position in inches using
c  geoplot software
c
c  programmer DOppenheimer X2923 2/10/80
c
         common/vrstec/ sclfac,xpgmin,ypgmin,xpgmax,ypgmax,ierr,
```

```
&symtab(128),xoff,yoff
        save /vrstec/
        character*2 symtab
        mode=7
        call movdrw(xpos,ypos,mode)
        x=xpos/sclfac-xoff
        y=ypos/sclfac-yoff
        return
        end
        subroutine factor(fact)
c
c  subroutine factor
c  emulates standard versatec FACTOR subroutine call using
c  geoplot software
c
c  programmer DOppenheimer X2923 2/10/80
c
        common/vrstec/ sclfac,xpgmin,ypgmin,xpgmax,ypgmax,ierr,
&symtab(128),xoff,yoff
        save /vrstec/
        sclfac=sclfac*fact
        return
        end
        subroutine line (xarray,yarray,npts,inc,lintyp,inteq)
c
c  subroutine line
c  emulates standard versatec LINE subroutine call using
c  geoplot software
c
c  programmer DOppenheimer X2923 2/10/80
c
        common/vrstec/ sclfac,xpgmin,ypgmin,xpgmax,ypgmax,ierr,
&symtab(128),xoff,yoff
        save /vrstec/
        character*2 symtab
        dimension xarray(*),yarray(*)
c
c  functions valx,valy compute x,y positions in picture space
c
        valx(a)=((a-fvalx)/dx+xoff)*sclfac
        valy(a)=((a-fvaly)/dy+yoff)*sclfac
c
c  error conditions
c
        if (inc.eq.0) then
                write(6,10)
10              format('INC = 0, SUBROUTINE LINE')
                return
        end if
        call putcom(91,1,.075*sclfac)
        ipts=(npts-1)*abs(inc)+1
        fvalx=xarray((npts+1)*abs(inc))
        dx=xarray((npts+2)*abs(inc))
        fvaly=yarray((npts+1)*abs(inc))
        dy=yarray((npts+2)*abs(inc))
        i=1
        k=i+abs(inc)
c
c  move to first point
c
        x=valx(xarray(1))
        y=valy(yarray(1))
        call movdrw(x,y,1)
        itype=0
```

```
c
c  plot points
c
        do 110 i=k,ipts,abs(inc)
        itype=itype+1
        x=valx(xarray(i))
        y=valy(yarray(i))
c
c  connect points
c
        if (lintyp.ge.0) call movdrw(x,y,2)
c
c  plot integer symbol numbers at lintyp frequency
c
        if (itype.eq.abs(lintyp)) then
                call letter(x,y,1,symtab(inteq+1),0.,1)
                call movdrw(x,y,1)
                itype=0
        end if
110     continue
        return
        end
        subroutine newpen(inp)
c
c  subroutine newpen
c  emulates standard versatec NEWPEN subroutine call using
c  geoplot software
c
c  programmer DOppenheimer X2923 2/10/80
c
        common/vrstec/ sclfac,xpgmin,ypgmin,xpgmax,ypgmax,ierr,
&symtab(128),xoff,yoff
        save /vrstec/
        character*2 symtab
        dimension values(5)
        data values /.0001,.0005,.00075,.001,.002/
        call clrbuf
        inp1=inp
        if (inp.lt.1) inp1=1
        if (inp.gt.5) inp1=5
        do 10 i=1,3
        call putcom(83+1,3,values(inp1))
10      continue
        return
        end
        subroutine nplots(i,j,k)
c
c  subroutine nplots
c  emulates standard versatec PLOTS subroutine call using
c  geoplot software
c
c  programmer DOppenheimer X2923 2/10/80
c
        common/vrstec/ sclfac,xpgmin,ypgmin,xpgmax,ypgmax,ierr,
&symtab(128),xoff,yoff
        save /vrstec/
        character*2 symtab
        character*20 dev
        real gridsp(4)
c
c  open device
c
        write(6,'("Input device string:")')
        read(5,'(a20)') dev
```

```
                call ploton(1,dev//'\0',0.,-8,99999.)
c
c  entry point for new plot (reinitialize parameters)
c
        entry nplot1(i,j,k)
c
c  set offsets to zero
c
        call getcom(33,1,xoff)
        call getcom(35,1,yoff)
c  set gridsp same as pictsp
        call getcom(21,4,gridsp)
        call putcom(27,4,gridsp)
c
c  determine scale factor to inches
c
        call getcom(7,1,sclfac)
        sclfac=1/sclfac
        ierr=1
        if(dev(1:1).eq.'t') call movdrw(0.,0.,13)
        return
        end
        subroutine number(x,y,height,fpn,angle,ndec)
c
c  subroutine number
c  emulates standard versatec NUMBER subroutine call using
c  geoplot software
c
c  programmer DOppenheimer X2923 2/10/80
c
        common/vrstec/ sclfac,xpgmin,ypgmin,xpgmax,ypgmax,ierr,
&symtab(128),xoff,yoff
        save /vrstec/
        real x,y,height,fpn,angle,xpos,ypos,hite
        integer ndec
        character*2 symtab
        character*10 numstr
        xpos=(x+xoff)*sclfac
        ypos=(y+yoff)*sclfac
        hite=height*sclfac
        call putcom(91,1,hite)
        call letter(xpos,ypos,1,numstr(fpn,ndec),angle,8)
        return
        end
        subroutine plot(x,y,ipen)
c
c  subroutine plot
c  emulates standard versatec PLOT subroutine call using
c  geoplot software
c
c  programmer DOppenheimer X2923 2/10/80
c
        common/vrstec/ sclfac,xpgmin,ypgmin,xpgmax,ypgmax,ierr,
&symtab(128),xoff,yoff
        save /vrstec/
        character*2 symtab
        ipena=iabs(ipen)
c
c  terminate plotting
c
        if (ipen.eq.999) then
                call clrbuf
                call gpfin
                goto 1000
```

```
c
c  new plot
c
        else if (ipen.eq.-999.or.ipena.eq.22.or.ipena.eq.23) then
                call clrbuf
                call plots1(0,0,0)
                goto 1000
c
c  offset calls
c
        else if (ipena.eq.13.or.ipena.eq.12) then
                write(6,10)
10              format('IPEN =12,13 NOT SUPPORTED YET, SUBROUTINE PLOT')
                goto 1000
c
c  move, draw calls
c
        else if (ipena.eq.3.or.ipena.eq.2) then
                mode=1
                if (ipena.eq.2) mode=2
                call movdrw((x+xoff)*sclfac,(y+yoff)*sclfac,mode)
c
c  re-origin
c
                if (ipen.lt.0) then
                        xoff=x+xoff
                        yoff=y+yoff
                end if
c
c  bad call
c
        else
                write(6,20) ipen
20              format('IPEN = ',i10,' ILLEGAL-CALL IGNORED, SUBROUTINE
& PLOT')
        end if
1000    return
        end
        subroutine plots(i,j,k)
c
c  subroutine plots
c  emulates standard versatec PLOTS subroutine call using
c  geoplot software
c
c  programmer DOppenheimer X2923 2/10/80
c
        common /vrstec/ sclfac,xpgmin,ypgmin,xpgmax,ypgmax,ierr,
&symtab(128),xoff,yoff
        save /vrstec/
        character*2 symtab
c
c  open device
c
        if (k.eq.20) then
                call ploton(1,'versR',0.,-8,99999.)
        else
                call ploton(1,'tekR',0.,-8,99999.)
        end if
c
c  entry point for new plot (reinitialize parameters)
c
        entry plots1(i,j,k)
c
c  set offsets to zero
```

```
c
          xoff=0
          yoff=0
c
c  determine scale factor to inches
c
          call getcom(7,1,sclfac)
          sclfac=1/sclfac
          ierr=1
          call movdrw(0.,0.,13)
          return
          end
c
c+
c
       subroutine scale (array,axlen,npts,inc)
c
c scale - scale data values for line and axis.
c
c call scale (array,axlen,npts,inc)
c
c       array  = array of unscaled data points
c       axlen  = length of scaled axis in inches
c       npts   = number of data points to be scaled
c       inc    = increment between points in array
c
c called by:  user program
c
c calls:  none
c
c commons used:  none
c
c-
c
       dimension  array(1), units(7)
c
       data units(1)/1./,units(2)/2./,units(3)/4./,units(4)/5./
       data units(5)/8./,units(6)/10./,units(7)/20./
c
c...  determine min and max values of array
       k = iabs(inc)
       j = npts * k
       amin = array(1)
       amax = amin
       do 10 i = 1,j,k
          a = array(i)
c
c...  new min found?
          if (a .lt. amin) amin = a
c
c...  new max found?
          if (a .gt. amax) amax = a
   10     continue
c
c...  compute delta value for unscaled unit interval
       dv = (amax-amin) / axlen
c
c...  delta ok?
       if (dv .gt. 0.0) goto 20
c
c...  error - amax and amin not in correct range
       dv = abs((amin+amin)/axlen) + 1.0
c
c...  compute tens power of dv value (watch out for negative logs!)
```

```
   20 adv = alog10(dv)
      logdv = ifix(adv)
      if (float(logdv) .gt. adv) logdv = logdv - 1
      a = 10.0**logdv
c
c...  compute normalized dv value (1. < dv < 10.)
      dv = dv/a
c
c...  locate closest "unit" dv value (normalized)
      do 30 i = 1,6
c
c...  value found?
         if (units(i) .ge. dv) goto 50
   30    continue
c
c...  expand unit dv to floating
   50 dv = units(i) * a
c
c...  compute "unit" minimum based on "unit" dv
      adv = aint(amin/dv)
      if (adv .gt. amin/dv) adv = adv - 1.
      tmin = dv * adv
c
c...  does adjusted "unit" scale fit axis length?
      if ((tmin+(axlen+0.01)*dv) .ge. amax) goto 60
      adv = aint(amin/a)
      if (adv .gt. amin/a) adv = adv - 1.
      tmin = a * adv
c
c...  does adjusted "unit" scale fit axis length?
      if ((tmin+(axlen+0.01)*dv) .ge. amax) goto 60
      i = i + 1
      goto 50
c
c...  compute final adjusted minimum
   60 tmin = tmin - dv*aint((axlen+(tmin-amax)/dv)/2.0)
c
c...  does tmin need correction?
      if (amin*tmin .le. 0.0) tmin = 0.0
c
c...  scale direction ok?
      if (inc .gt. 0) goto 70
c
c...  reverse direction of scale
      tmin = tmin + dv*aint(axlen+0.5)
      dv = -dv
c
c...  set scale factors into user's array
   70 j = j + 1
      array(j) = tmin
      k = j + k
      array(k) = dv
c
      return
c
      end
         subroutine symbol(x,y,height,itext,angle,nc)
c
c  subroutine symbol
c  emulates standard versatec SYMBOL subroutine call using
c  geoplot software
c
c  programmer DOppenheimer X2923 2/10/80
c
```

```
      common/vrstec/ sclfac,xpgmin,ypgmin,xpgmax,ypgmax,ierr,
&symtab(128),xoff,yoff
      save /vrstec/
      character*(*) itext
      character*1 symone
      character*2 symtab
      hite=height*sclfac
      call putcom(91,1,hite)
      xpos=(x+xoff)*sclfac
      ypos=(y+yoff)*sclfac
c
c  plot text string
c
      if (nc.gt.0) then
            mode=1
            ipos=8
            call letter(xpos,ypos,mode,itext,angle,ipos)
c
c  plot single right-justified character
c
      else if (nc.eq.0) then
            mode=1
            ipos=8
            length=len(itext)
            symone=itext(length:length)
            call letter(xpos,ypos,mode,symone,angle,ipos)
c
c  plot integer symbol number
c
      else if (nc.le.-1) then
            if (nc.eq.-1) then
                  mode=1
            else
                  mode=2
            end if
c
c  find index of symbol
c
            read(itext,"(i3)") index
            if (index.lt.0.or.index.gt.127) then
                  write(6,20) index
20                format('INTEGER SYMBOL ',i13,' NOT IN RANGE 0-127,
& SUBROUTINE SYMBOL')
                  return
            else
                  index=index+1
            end if
c
c  center symbol
c
            if (index.ge.0.and.index.le.13) then
                  ipos=1
            else
                  ipos=8
            end if
            call letter(xpos,ypos,mode,symtab(index),angle,ipos)
      end if
      return
      end
      block data
      common /vrstec/ sclfac,xpgmin,ypgmin,xpgmax,ypgmax,ierr,symtab(128),xoff,yoff
      save /vrstec/
      character*2 symtab
c
```

```
c  integer symbol table
c
      data symtab /'\\4',' ','\\3',' ',' ','\\2',' ',' ',' ',
&' ',' ',' ',' ',' ',' ','\\5','_',' ','\\b','}',' ',' ','\\1','\\r',
&'\\=',' ','{',' ',' ','\\_','\\I','\\s','v',' ','~',' ','\\~',' ',' ','!',' ',
&'#',' ','$',' ','%',' ','&',' ','"',' ',' ',' ','\\-',' ','+',' ',' ','-',' ',' ','\\/',
&'0',' ','1',' ','2',' ','3',' ','4',' ','5',' ','6',' ','7',' ','8',' ','9',' ',':',' ',';',' ','<',
&'=',' ','>',' ','?',' ','@',' ','A',' ','B',' ','C',' ','D',' ','E',' ','F',' ','G',' ','H',' ','I',
&'J',' ','K',' ','L',' ','M',' ','N',' ','O',' ','P',' ','Q',' ','R',' ','S',' ','T',' ','U',' ','V',' ','W',
&'X',' ','Y',' ','Z',' ','[',' ','\',' ',']',' ','^',' ','_',' ','\\i',' ',' ','\\M','\\P','\\F',
&'\\T',' ',' ',' ','\\O','\\L','\\A','\\D','\\E','\\N','\\u','\\d','\\S',
&' ',' ',' ',' ','\\3','\\C','-',' ',' ',' ','\\R',' ',' ',' ',' ',' ',
&' ',' '/
      end
```

```
c-*--PLTLTR--MAIN GEOPLOT SUBROUTINE FOR PLOTTING ARRAYS OF DATA--------
c
      subroutine pltltr(length,kptype,iflag)
c
c-----Programmer: PLWard, US Geological Survey, Menlo Park, California,
c       June and November, 1979.
c
c-*--This program plots x-y data passed in arrays in a number of ways.
c       a.) kptype designates the plot type as follows:
c                0 plot no lines
c                1 plot lines
c                2 plot segmented lines, alternate move and draw
c                3 plot tic marks along the x axis
c                4 plot tic marks along the y axis
c                5 plot triangles along the x axis
c                6 plot triangles along the y axis
c                7 plot bar graph along the x axis
c                8 plot bar graph along the y axis
c       b.) kptype is the sum of the above numbers plus 100*itype
c            where itype may be:
c                0 plot in absolute coordinates in data space
c                1 plot in relative coordinates in page space
c                2 plot in absolute coordinates in data space
c                3 plot in relative coordinates in page space
c       c. ) iflag = 0 for plt and = 1 for pltltr
c
      save
c
c
      include 'gpcom.h'
      real relx,rely,angle,xxx,yyy,xx,x,yy,y,oldy,oldx,posn,
     1  center
      integer length,kptype,kmode,mmode,iposn,mode,klen,iflag,
     1  i,itype,misdat,misold,jtype
      data klen /-1/
c
c
      if (idbug.ge.2) call printn("pltltr: %d %d %f\n",length,kptype,
     1  iflag)
c
c-*--Decode kptype.-----------------------------------------------------
      if(kptype.gt.400)kptype=1
      itype=kptype/100
      kptype=kptype-100*itype
      if (itype.ge.i0.and.itype.le.i3) goto 70
      call gperr(kerror,"pltltr","itype set to 0. Was",itype)
      itype=0
 70   if(kptype.ge.0.and.kptype.le.8) goto 75
      i=kptype
      call gperr(kerror,"pltltr","Plot type set to 1. Was",i)
      kptype=1
 75   mode=-2
      if(itype.eq.i1.or.jtype.eq.i1) mode=-4
      if(itype.eq.i2.or.jtype.eq.i2) mode=2
      if(itype.eq.i3.or.jtype.eq.i3) mode=4
      relx=0.0
      rely=0.0
      if(itype.ne.i1.and.itype.ne.i3) goto 100
      relx=1.0
      rely=1.0
100   mmode=mode-isign(1,mode)
      kmode=mmode
      iposn=-100
      angle=0.0
```

```
      xxx=0.0
      yyy=0.0
      misdat=0
c
c--*--Main plot loop.-------------------------------------------------------
3000  do 5200 i=1,length
c-----Input x from pipe and check for missing data.
          call gpgpip(xx,i1)
          if(xx.ne.usym) goto 3020
          misdat=1
          x=xx
          goto 3100
3020      x=xx+relx*xxx
          xxx=x
          if(i.eq.i1) oldx=x
c-----Input y from pipe and check for missing data.
3100      call gpgpip(yy,i1)
          if(yy.ne.usym) goto 3120
          misdat=misdat+2
          y=yy
          goto 3130
3120      y=yy+rely*yyy
          yyy=y
          if(i.eq.i1) oldy=y
          if(i.eq.i1)misold=misdat
          if(misold.ne.i0.and.misdat.eq.i0)kmode=mode
c
c--*--Get the angle and position from the pipe.-----------------------------
3130      if (iflag.lt.1) goto 5000
          call gpgpip(angle,i1)
          call gpgpip(posn ,i1)
          iposn=posn+0.5
          if(posn.lt.0)iposn=posn-0.5
          if(idbug.ge.2)call printn("pltltr: %f %d\n",angle,iposn)
c
c--*--Plot different line types where
c     kptype= 0 just move to each point and plot string
c             1 plot lines
c             2 alternate move, draw(segmented lines)
c             3 tic marks along the x axis
c             4 tic marks along the y axis
c             5 triangles along the x axis
c             6 triangles along the y axis
c             7 bar graph along the x axis
c             8 bar graph along the y axis
c
c--*--Plot all points other than the first and last.-----------------------
5000      goto (5005,5010,5020,5030,5040,5050,5060,5070,5080) kptype+1
c-----Just go to the point.
5005      kmode=mmode
          goto 5015
c-----Draw a line to the point.
5010      if(kmode.eq.mmode) goto 5015
          if(misold.ne.i0.and.misdat.eq.i0) call movdrw(x,y,mmode)
          goto 5015
5014      kmode=mode
5015      call letter(x,y,kmode,"fake",angle,iposn,klen)
          kmode=mode
          misold=0
          if(misdat.eq.0)goto 5197
               misold=misdat
               kmode=mmode
               if(x.ne.usym)oldx=x
               if(y.ne.usym)oldy=y
```

```
                goto 5198
c----Segmented line.
5020        if(i.ne.((i/2)*2)) kmode=mmode
            goto 5015
c----Tic marks along the x axis.
5030        call movdrw(x,ticbas,mmode)
            goto 5014
c----Tic marks along the y axis.
5040        call movdrw(ticbas,y,mmode)
            goto 5014
c----Triangle along the x axis.
5050        if(x.eq.usym)goto 5055
                center=(x+oldx)*0.5
                if(misold.ne.i0) goto 5053
                    call movdrw(center,ticbas,kmode)
                    goto 5014
5053            if(misold.ne.i2)goto 5054
                    call movdrw(center,ticbas,mmode)
                    goto 5014
5054            call movdrw(x,ticbas,mmode)
                goto 5014
5055        if(misold.eq.i0)call movdrw(oldx,ticbas,kmode)
            goto 5014
c----Triangle along the y axis.
5060        if(y.eq.usym)goto 5065
                center=(y+oldy)*0.5
                if(misold.ne.i0) goto 5063
                    call movdrw(ticbas,center,kmode)
                    goto 5014
5063            if(misold.ne.i1)goto 5064
                    call movdrw(ticbas,center,mmode)
                    goto 5014
5064            call movdrw(ticbas,y,mmode)
                goto 5014
5065        if(misold.eq.i0)call movdrw(ticbas,oldy,kmode)
            goto 5014
c----Bar graph along the x axis.
5070        if(x.eq.usym)goto 5075
                center=(x+oldx)*0.5
                if(misold.ne.i0) goto 5073
                    call movdrw(center,oldy,kmode)
                    call movdrw(center,ticbas,kmode)
                    call movdrw(center,y,mode)
                    goto 5014
5073            if(misold.ne.i2)goto 5074
                    call movdrw(center,ticbas,mmode)
                    call movdrw(center,y,mode)
                    goto 5014
5074            call movdrw(x,ticbas,mmode)
                goto 5014
5075        if(misold.eq.i0)call movdrw(oldx,ticbas,kmode)
            goto 5014
c----Bar graph along the y axis.
5080        if(y.eq.usym)goto 5085
                center=(y+oldy)*0.5
                if(misold.ne.i0) goto 5083
                    call movdrw(oldx,center,kmode)
                    call movdrw(ticbas,center,kmode)
                    call movdrw(x,center,mode)
                    goto 5014
5083            if(misold.ne.i1)goto 5084
                    call movdrw(ticbas,center,mmode)
                    call movdrw(x,center,mode)
                    goto 5014
```

```
5084            call movdrw(ticbas,y,mmode)
                goto 5014
5085        if(misold.eq.i0)call movdrw(ticbas,oldy,kmode)
            goto 5014
c
5197        oldx=x
            oldy=y
5198        misdat=0
5200        continue
c
c--*--End of plot.-------------------------------------------------------
      goto(6010,6010,6010,6010,6010,6002,6004,6002,6004)kptype+1
6002  call movdrw(x,ticbas,kmode)
      goto 6010
6004  call movdrw(ticbas,y,kmode)
c
6010  return
      end
```

```
c-*--PROPOR--PROPORTION GRIDSPACE SETTING SUBJECT AND DATA SPACES-------
c
      subroutine propor(center,total,datlo,dathi,ktype)
c
c-*--Set subject space to be a proportion of gridspace. Divide
c       grid space up into "total" units in the x direction if ktype=1
c       or the y direction if ktype=2. Center this new subject space
c       around "center" if this is a non-integer value or if this is
c       and integer value count this many subject spaces from the top
c       and work on that one. Clip is a factor by which you can then
c       expand or decrease  subject space to control clipping as a
c       factor of the width of the proportion assigned to this subject
c       space. dat is the low and high data space mapped into the
c       original subject space and then modified to conform with
c       clip.
c
      save
      include 'gpcom.h'
c
      real dat(2),center,total,dathi,datlo,zero,ptgrid,cent,sign
      integer ktype,i,j,k
c
      if(idbug.ge.3)call printn("propor: cent=%f tot=%f dat=%f %f type=%
     1d\n",center,total,datlo,dathi,ktype)
      if(center.le.total) goto 10
          kerror=2
          i=center
          call gperr(kerror,"propor","Center larger than total in propor. Is
     1    ",i)
          return
c
 10   if(ktype.ne.1.and.ktype.ne.2) ktype=1
      j=5+2*ktype
      dat(1)=datlo
      dat(2)=dathi
      zero=(dat(1)+dat(2))*0.5
      ptgrid=0.5*(pspace(j+1)-pspace(j))/total
      if(center.gt.pspace(j+1).or.center.eq.1.0) go to 15
          cent=pspace(j)+center*(pspace(j+1)-pspace(j))
          go to 20
 15   continue
           if(ktype.eq.1)cent=pspace(j)+(2.0*center-1.0)*ptgrid
           if(ktype.eq.2)cent=pspace(j+1)-(2.0*center-1.0)*ptgrid
 20   sign=-1.0
      k=1
      do 100 i=j,j+1
           pspace(i+6)=cent+sign*clip(k)*ptgrid
           pspace(i+12)=zero+clip(k)*(dat(k)-zero)
           if((pspace(i+6)-pspace(i))*sign.le.0.0)goto 90
           pspace(i+12)=zero+(pspace(i+12)-zero)*(pspace(i)-cent)/
     1         (pspace(i+6)-cent)
           pspace(i+6)=pspace(i)
 90        sign=1.0
100        k=2
      ktrflg=1
      end
```

```
c-*--SCALE----SET PLOT DATA SPACE ACCORDING TO SIZES OF X AND Y ARRAYS--
c
      subroutine scale(xarr,xlen,yarr,ylen)
c
c-*--Programmer: PLWard, USGS, Menlo Park, California  12/13/79
c
c-----Find the maximum and minimum values of the x and y arrays and
c       use them to set datasp in geoplot. If xlen or ylen is 0 then
c       that array is assumed to be counting numbers from arr(1) to
c       arr(1)+len other array -1.
c
      real xarr(*),yarr(*),dspace(4)
      integer xlen,ylen,i
c
      include 'GPCOM.h'
c
      if(xlen.ge.2) goto 80
      dspace(1)=xarr(1)
      dspace(2)=xarr(1)+ylen-1
      goto 105
 80   dspace(1)=xarr(1)
      dspace(2)=xarr(1)
      do 100 i=2,xlen
      if(xarr(i).lt.dspace(1)) dspace(1)=xarr(i)
      if(xarr(i).gt.dspace(2)) dspace(2)=xarr(i)
100   continue
105   if(ylen.ge.2) goto 106
      dspace(3)=yarr(1)
      dspace(4)=yarr(1)+xlen-1
      goto 115
106   dspace(3)=yarr(1)
      dspace(4)=yarr(1)
      do 110 i=2,ylen
      if(yarr(i).lt.dspace(3)) dspace(3)=yarr(i)
      if(yarr(i).gt.dspace(4)) dspace(4)=yarr(i)
110   continue
115   continue
c      write(6,999)dspace
c999   format("scale:",4f10.3)
      call putcom(DATASP,4,dspace)
      end
```

```
c------shade----shade any polygonal area-------------------------------
c-------Not implemented yet.  Programmer PLWard.
      subroutine shade(len)
c
      integer len,i,i2
      real xy(2)
      data i2/2/
c
      call printn("shade: %d\n",len)
      do 2 i=1,len
         call gpgpip(xy,i2)
    2    call printn("shade: %d %f %f\n",i,xy(1),xy(2))
      return
      end
```

```
/*      modifications for menlo park tekhi_speed calls
 *              by RS Dollar     12-23-80                        */
#include <sgtty.h>
#include <sys/ioctl.h>  /* menlo highspeed   -rsd  */
#include <fasttek.h>
/*
#define FASTON (('x'<<8)+0)
#define FASTOFF (('x'<<8)+1)
 *                              UW highspeed --out -rsd */
faston_()
{
/*
        ioctl(1, FASTON,0);
 *                  UW highspeed call commented out ---rsd */
        ioctl(1, FTHISPD,0);  /* menlo highspeed  -rsd */
}

fastoff_()
{
/*
        ioctl(1, FASTOFF,0);
 *                  UW highspeed call commented out --- rsd */
        ioctl(1, FTNORML,0);  /* menlo normalspeed -rsd */
}
```

```
/* TEKIN interrogate Tektronix and pass back integer
   values of last NUMIN characters then received from
   device.     Fortran callable.       PLWard 10/13/79

   Tekin looks for NUMIN characters terminated by
   a carriage newline or EOF.

   When an ESC ENQ (27 5) is sent, an alarm is set and if
   an appropriate response is not made in WAIT seconds,
   beamtype is changed to 2 which means the gpbufo.f
   will not look for the beam position before dumping the
   buffer in the future.
*/

#include "gpcomc.h"
#include <stdio.h>
#include <signal.h>
#define ENQ      5
#define NUMTRY  10
#define NUMOUT   2
#define NUMIN    5
#define MSK5    31
#define WAIT     3
int termread,termwrite;    /* file descriptors opened in gpdevs_.c */
static int j;
static char c;


tekin_(out,in)
    int out[],in[];
{
    int i,k;
    char bufin[NUMIN+2];
    extern gpcom_,tekhung(),alarm();
    char bufout[NUMOUT];

    for(i=0; i<NUMOUT;i++) bufout[i]=out[i];
    stty(stdout,"raw");
    signal(SIGALRM,tekhung);
    if(out[1] == ENQ) alarm(WAIT);
    for(j=0;j < NUMTRY;j++){
        write(termwrite,bufout,NUMOUT);
        if( (i=read(termread,bufin,NUMIN+2)) >= NUMIN ) goto leave;
    }
leave:stty(stdout,"-raw"); alarm(0);
    if(i < NUMIN) {
        common = &gpcom_;
        common->kbmtyp=2;
        fprintf(stderr,
"\n*WARNING in geoplot on tekin: Cannot read cursor position from terminal.\n      Unable
        in[0]=0;
        in[1]=0;
        in[2]=common->devmax[1] * common->dpage[1];
        return;
    }
    in[0]=(int)bufin[0];
    in[1]=(((int)bufin[1]&MSK5)<<7)+(((int)bufin[2]&MSK5)<<2);
    in[2]=(((int)bufin[3]&MSK5)<<7)+(((int)bufin[4]&MSK5)<<2);
    return;
}

tekhung(sig) int sig; {
     signal(SIGALRM,tekhung);
     j=NUMTRY;
```

```
    c='\n';
}

/*PLTOUT output *len integers in buf as characters. PLWard 10/13/79*/
#include <sgtty.h>
struct sgttyb args;
static char *cp;
static int *ip;
static int olen,ret;

pltout_(len,buf)
    int *len,buf[];
{   olen= *len;
    /* Change integers to characters within buf.*/
    for(cp=ip=buf;(*len)-- >0;cp++,ip++)
        *cp= (char)*ip;
    /* Output in raw mode.*/
    gtty(2,&args);
    args.sg_flags |= RAW;
    stty(2,&args);
    ret=write(termwrite,buf,olen);
    args.sg_flags &= ~RAW;
    stty(2,&args);
    if (ret <= 0) printf("pltout: ERROR write returns %d fd=%d len=%d buf=%d\n",ret,termwri
}

/* Print data in plot buffer as integers for dbugging
   PLWard, USGS Menlo Park, Calif 3/22/80
*/
prnout_(length,data)
    int *length,data[];
{
    int i;

    printf("prnout: len=%d buf=%d\n",*length,data);
    for (i=0;i<*length;i++) {
        if(i%15==0)printf("\n");
        printf("%4d ",data[i]);
    }
    printf("\n");
}
```