DEFENSE INFORMATION INFRASTRUCTURE (DII)

COMMON OPERATING ENVIRONMENT (COE)

PROGRAMMER'S REFERENCE MANUAL SET

VOLUME III

JMTK Version 3.0 Developers Manual - Part 1.
(Source Code)

28 June 1996

Prepared for:

Defense Information Systems Agency

TABLE OF CONTENTS

TITLE PAGE

INTRODUCTION 1

IMPORTING MAP DATA 14

SUN Workstation Map Data Importing Procedure 15

HP Workstation Map Data Importing Procedure 18

APPENDIX A 21

TEST BED / DEMO SOFTWARE 21

This page intentionally left blank.

INTRODUCTION

This segment provides the JMTK mapping services. The segment is an software segment consisting of the parent (JMTK) and 3 pseudo children. Each child represents one of the functional pieces of JMTK. Only the spatial database (labeled JMS) and the visual (labeled JMV) pieces are included with this delivery. The analysis (labeled JMA) will be added in the future.

This segment has been verified with the GCCS 2.1 Common Operating Environment (COE) tools for this delivery. The VerifySeg and MakeInstall tools for DII 2.0 have not been developed at the time of this delivery. Once these tools are available, this segment will be converted to the DII 2.0 segmentation, and if necessary, redelivered.

This segment includes JMTK demo software (/h/JMTK/DemoSource) and executables, also. To run the JMTK demo, follow the steps below as the user:

1. Run JMTK_demo script which is located in /h/JMTK/bin/JMTK_DEMO_bin.

   This program starts the following processes.

   a. Cartographer,
   b. DrawVector,
   c. DrawCountries, and
   d. Demo.

2. When you exit from the demo, you must kill three of the above processes (a, b, and c) which will still be active. To kill the three processes, at the shell prompt, type in the following:

   `ps` - gives you the PID's

   `kill PID #` - kills a process (repeat it for all three)

If the processes are not killed, typing `demo` will start the demo again.

The processes you need to run the JMTK demo are attached in the following pages.

Following the installation of JMTK 3.0/DII Version 2.0, make sure that you are in cshell. Source the file cshrc.JMTK in the /h/JMTK/Scripts directory to set the environment correctly.

Following is a list of all directories and titles under /h/JMTK:

```
Resources   Scripts     SegDescrip  bin         data
               include
lib         man         source

/h/JMTK/Resources:
Jms

/h/JMTK/Resources/Jms:
Excl     SDBM     view.mf

/h/JMTK/Scripts:

/h/JMTK/SegDescrip:
DEINSTALL      Installed.DII  ModName        PostInstall
               ReleaseNotes
SegInfo        SegName        SegType        VERSION
               Validated
/h/JMTK/bin:
Jmd  Jms  Jmv

/h/JMTK/bin/Jmd:
Demo              JMTKDemoPrint.sh

/h/JMTK/bin/Jms:
APITEST        JMSErrors.def  SDBM

/h/JMTK/bin/Jmv:
CDMaps         CR1            Cartographer   CenterWidth
               ChartDemo
ChartInset     Countries      DMADraw        DMALoad
               DMASpatial
DestroyChart   DrawCAC        DrawCountries  DrawDTED
               DrawRaster
DrawVector     ExportChart    Features       LoadADRG
               LoadCAC
LoadCD         LoadDTED       LoadRaster     MapColorCtl
               MapControl
MapIntensity   MapList        MapManager     Monitor
               MountCD
Overlays       Pimtracks      RPFDraw        RPFEdit
               RPFLoad
ReloadPath     SymPlot        UBChart        VEDEdit
               mkmapdir
setup

/h/JMTK/data:
Jmd  Jms  Jmv

/h/JMTK/data/Jmd:
menu.sys

/h/JMTK/data/Jms:
DB   Map
```

```
/h/JMTK/data/Jms/DB:
CIB_data   DBFile     DTED_data  cib_data

/h/JMTK/data/Jms/DB/CIB_data:
CIB_data.md  RPF

/h/JMTK/data/Jms/DB/CIB_data/RPF:
CIB

/h/JMTK/data/Jms/DB/CIB_data/RPF/CIB:
CIB.md                   CIB_19960408_0000247810

/h/JMTK/data/Jms/DB/CIB_data/RPF/CIB/CIB_19960408_0000247810:
CIB_19960408_0000247810.md

/h/JMTK/data/Jms/DB/DTED_data:
DTED          DTED_data.md

/h/JMTK/data/Jms/DB/DTED_data/DTED:
DTED

/h/JMTK/data/Jms/DB/DTED_data/DTED/DTED:
DTED.md                    DTED_19960408_0000157707
DTED_19960408_0000159837
/h/JMTK/data/Jms/DB/DTED_data/DTED/DTED/DTED_19960408_0000157707:
DTED_19960408_0000157707.md
/h/JMTK/data/Jms/DB/DTED_data/DTED/DTED/DTED_19960408_0000159837:
DTED_19960408_0000159837.md
/h/JMTK/data/Jms/DB/cib_data:
RPF          cib_data.md

/h/JMTK/data/Jms/DB/cib_data/RPF:
CIB

/h/JMTK/data/Jms/DB/cib_data/RPF/CIB:
CIB.md                   CIB_19960405_0000231439

/h/JMTK/data/Jms/DB/cib_data/RPF/CIB/CIB_19960405_0000231439:
CIB_19960405_0000231439.md

/h/JMTK/data/Jms/Map:

/h/JMTK/data/Jmv:
JUnits        MapData       Menus         VectorData    app-
defaults
fonts

/h/JMTK/data/Jmv/JUnits:
JUnitColors  JUnitTypes   ModTypes

/h/JMTK/data/Jmv/MapData:
DBDB        Features    M           RasterMaps  VectorMaps
maps.dir
/h/JMTK/data/Jmv/MapData/DBDB:
DBDB      DBDB.mfh

/h/JMTK/data/Jmv/MapData/Features:
```

```
Borders    Bottoms    Countries  GridLines  Rivers     Roads
/h/JMTK/data/Jmv/MapData/Features/Borders:
political        political.index

/h/JMTK/data/Jmv/MapData/Features/Bottoms:
Level-0050        Level-0050.index  Level-0100        Level-
0100.index
Level-0200        Level-0200.index  Level-0300        Level-
0300.index
Level-0400        Level-0400.index  Level-0500        Level-
0500.index
Level-0600        Level-0600.index  Level-0700        Level-
0700.index
Level-0800        Level-0800.index  Level-0900        Level-
0900.index
Level-1000        Level-1000.index  Level-2000        Level-
2000.index
Level-3000        Level-3000.index  Level-4000        Level-
4000.index
Level-5000        Level-5000.index  Level-6000        Level-
6000.index
Level-7000        Level-7000.index  Level-8000        Level-
8000.index

/h/JMTK/data/Jmv/MapData/Features/Countries:
AC.dat          AC.index        AF.dat          AF.index AG.dat
AG.index        AL.dat          AL.index        AN.dat
                AN.index
AO.dat          AO.index        AQ.dat          AQ.index AR.dat
AR.index        AS.dat          AS.index        ASKA.dat
                ASKA.index
AU.dat          AU.index        AV.dat          AV.index AY.dat
AY.index        BA.dat          BA.index        BB.dat
                BB.index
BC.dat          BC.index        BD.dat          BD.index BE.dat
BE.index        BG.dat          BG.index        BH.dat
                BH.index
BL.dat          BL.index        BM.dat          BM.index BP.dat
BP.index        BR.dat          BR.index        BT.dat
                BT.index
BU.dat          BU.index        BV.dat          BV.index BX.dat
BX.index        BY.dat          BY.index        CA.dat
                CA.index
CB.dat          CB.index        CD.dat          CD.index CE.dat
CE.index        CF.dat          CF.index        CG.dat
                CG.index
CH.dat          CH.index        CI.dat          CI.indx
                CJ.dat
CJ.index        CK.dat          CK.index        CM.dat
                CM.indx
CN.dat          CN.index        CO.dat          CO.index
                CS.dat
CS.indx         CT.dat          CT.index        CU.dat
                CU.indx
CV.datCV.index  CW.dat          CW.index        CY.dat
CY.indx         CZ.dat          CZ.index        CountryColors
DA.dat
```

DA.index	DJ.dat	DJ.index	DO.dat
	DO.indx		
DR.dat	DR.index	EC.dat	EC.index
	EG.dat		
EG.indx	EI.dat	EI.index	EK.dat
	EK.indx		
ES.dat	ES.index	ET.dat	ET.index
	FA.dat		
FA.index	FG.dat	FG.index	FI.dat
	FI.index		
FJ.dat	FJ.index	FO.dat	FO.index
	FP.dat		
FP.index	FR.dat	FR.index	FR_CLP.dat
	FR_CLP.index		
FR_IN.dat	FR_IN.index	FR_clp.dat	FR_clp.index
	FR_is.dat		
FR_is.index	GA.dat	GA.index	GB.dat
	GB.index		
GE.dat	GE.index	GH.dat	GH.index
	GI.dat		
GI.index	GJ.dat	GJ.index	GL.dat
	GL.index		
GP.dat	GP.index	GQ.dat	GQ.index
	GR.dat		
GR.index	GT.dat	GT.index	GV.dat
	GV.index		
GY.dat	GY.index	GZ.dat	GZ.index
	HA.dat		
HA.index	HAW.dat	HAW.index	HK.dat
	HK.index		
HM.dat	HM.index	HO.dat	HO.index
	HU.dat		
HU.index	IC.dat	IC.index	ID.dat
	ID.index		
IN.dat	IN.index	IO.dat	IO.index
	IR.dat		
IR.index	IS.dat	IS.index	IT.dat
	IT.index		
IV.dat	IV.index	IY.dat	IY.index
	IZ.dat		
IZ.index	JA.dat	JA.index	JM.dat
	JM.index		
JO.dat	JO.index	JQ.dat	JQ.index
	KE.dat		
KE.index	KN.dat	KN.index	KR.dat
	KR.index		
KS.dat	KS.index	KU.dat	KU.index
	LA.dat		
LA.index	LE.dat	LE.index	LI.dat
	LI.index		
LS.dat	LS.index	LT.dat	LT.index
	LU.dat		
LU.index	LY.dat	LY.index	MA.dat
	MA.index		
MB.dat	MB.index	MC.dat	MC.index
	MG.dat		
MG.index	MH.dat	MH.index	MI.dat

	MI.index		
ML.dat	ML.index	MN.dat	MN.index
	MO.dat		
MO.index	MP.dat	MP.index	MQ.dat
	MQ.index		
MR.dat	MR.index	MT.dat	MT.index
	MU.dat		
MU.index	MV.dat	MV.index	MX.dat
	MX.index		
MY.dat	MY.index	MZ.dat	MZ.index
	NA.dat		
NA.index	NC.dat	NC.index	NE.dat
	NE.index		
NF.dat	NF.index	NG.dat	NG.index
	NH.dat		
NH.index	NI.dat	NI.index	NL.dat
	NL.index		
NO.dat	NO.index	NP.dat	NP.index
	NR.dat		
NR.index	NS.dat	NS.index	NU.dat
	NU.index		
NZ.dat	NZ.index	PA.dat	PA.index
	PC.dat		
PC.index	PE.dat	PE.index	PF.dat
	PF.index		
PG.dat	PG.index	PK.dat	PK.index
	PL.dat		
PL.index	PM.dat	PM.index	PO.dat
	PO.index		
PP.dat	PP.index	PU.dat	PU.index
	QA.dat		
QA.index	RE.dat	RE.index	RO.dat
	RO.index		
RP.dat	RP.index	RQ.dat	RQ.index
	RW.dat		
RW.index	SA.dat	SA.index	SB.dat
	SB.index		
SC.dat	SC.index	SE.dat	SE.index
	SF.dat		
SF.index	SG.dat	SG.index	SH.dat
	SH.index		
SL.dat	SL.index	SM.dat	SM.index
	SN.dat		
SN.index	SO.dat	SO.index	SP.dat
	SP.index		
ST.dat	ST.index	SU.dat	SU.index
	SW.dat		
SW.index	SY.dat	SY.index	SZ.dat
	SZ.index		
SoPac.dat	SoPac.index	TC.dat	TC.index
	TD.dat		
TD.index	TH.dat	TH.index	TK.dat
	TK.index		
TL.dat	TL.index	TN.dat	TN.index
	TO.dat		
TO.index	TP.dat	TP.index	TS.dat
	TS.index		

```
TU.dat          TU.index        TV.dat          TV.index
                TW.dat
TW.index        TZ.dat          TZ.index        UG.dat
                UG.index
UK.dat          UK.index        UK_is.dat       UK_is.index
                UR.dat
UR.index        US.dat          US.index        US_AK.dat
                US_AK.index
US_HA.dat       US_HA.index     US_is.dat       US_is.index
                UV.dat
UV.index        UY.dat          UY.index        VC.dat
                VC.index
VE.dat          VE.index        VI.dat          VI.index
                VM.dat
VM.index        VQ.dat          VQ.index        VT.dat
                VT.index
WA.dat          WA.index        WF.dat          WF.index
                WQ.dat
WQ.index        WS.dat          WS.index        WZ.dat
                WZ.index
YE.dat          YE.index        YO.dat          YO.index
                YS.dat
YS.index        ZA.dat          ZA.index        ZI.dat
                ZI.index

/h/JMTK/data/Jmv/MapData/Features/GridLines:
grid.index

/h/JMTK/data/Jmv/MapData/Features/Rivers:
rivers        rivers.index

/h/JMTK/data/Jmv/MapData/Features/Roads:
roads        roads.index

/h/JMTK/data/Jmv/MapData/M:
FeatureLaunch       MapFeatures.xrf     MapProjection.xrf
                MapSubFeatures.xrf
MapSubTypes.xrf     MapTypes.xrf

/h/JMTK/data/Jmv/MapData/RasterMaps:

/h/JMTK/data/Jmv/MapData/VectorMaps:
WDBII  WVS

/h/JMTK/data/Jmv/MapData/VectorMaps/WDBII:
wdbii10         wdbii10.index   wdbii180       wdbii180.index

/h/JMTK/data/Jmv/MapData/VectorMaps/WVS:
wvs1        wvs1.index

/h/JMTK/data/Jmv/Menus:
ChartBar        ChartInset      SystemBar.JMTK

/h/JMTK/data/Jmv/VectorData:
defaults  profiles  tables

/h/JMTK/data/Jmv/VectorData/defaults:
```

```
Colors  Default.DFL   Default.DNC   Default.DTED  Default.DTP
Default.UVM   Default.VMP   Default.VTD   Default.WVP  scales

/h/JMTK/data/Jmv/VectorData/profiles:
DTED_00S_048E-12N_108E_16Color   DTED_00S_048E-12N_108E_180Color
DTED_00S_048E-12N_108E_64Color

/h/JMTK/data/Jmv/VectorData/tables:
AttValues.DTED     FeatureAtts.DTED   FeatureNames.DTED

/h/JMTK/data/Jmv/app-defaults:
CDMaps           MapEditor        MapSelect        Wizard
WizardXm         WizardXm.wizard

/h/JMTK/data/Jmv/fonts:
vpf_pt_sym.pcf

/h/JMTK/include:
Jms  Jmv

/h/JMTK/include/Jms:
JMS.h                     JMS_.h                  JMS_AFileIO.h
JMS_AFileIO_.h            JMS_ApiDefines.h        JMS_ApiGlobals.h
JMS_DataAccessDefs.h      JMS_DataAccessGlbls.h   JMS_Defines.h
JMS_DtedDataAccess.h      JMS_DtedDataAccess_.h   JMS_ErrorMsg.h
JMS_Errors.h              JMS_Globals.h           JMS_Main_.h
JMS_MetaData.h            JMS_MetaDataGlbls.h
JMS_RPFDataAccess.h
JMS_RPFDataAccess_.h      JMS_ResourceDefs.h
                JMS_ResourceGlbls.h
JMS_ScreensDefs.h         JMS_ScreensGlbls.h
                JMS_ScreensMacros.h
JMS_Screens_.h            JMS_VPFDataAccess.h
                JMS_VPFDataAccessGlbls.h
JMS_VPFDataAccess_.h      JMU.h                    JMU_AFileIO.h
JMU_AFileIO_.h            JMU_TT.h
                JMU_ThreadTrack.h
JMU_ThreadTrackGlbls.h    JMU_ThreadTrack_.h

/h/JMTK/include/Jmv:
JMTK_1_1.h  JMU         JMV         M           Z

/h/JMTK/include/Jmv/JMU:
NormDefs.h  Types.h

/h/JMTK/include/Jmv/JMV:
Load.h

/h/JMTK/include/Jmv/M:
Alloc.h          ChartCxt.h       GenInit.h        Genconst.h
Generror.h       Genlib.h         Gentypes.h       MapData.h
MapRecall.h      MapRecallCol.h   MapRecallCxt.h   MapRecallRec.h
MapRecallRow.h   MapUtils.h       Mconst.h         Merror.h
Mlib.h           Mtypes.h         Mu.h             MuError.h
MuInit.h         MuUnits.h        Qualifiers.h     Symbols.h
Types.h          VectorProduct.h  extern.h
```

```
/h/JMTK/include/Jmv/Z:
ChannelIO.h      ChannelIOP.h     Math.h       MathP.h
                 ModemP.h
Msgid.h          Prt.h            Services.h   ServicesP.h
                 Tty.h
TtyP.h           Ttydefs.h        Ttys.h       TtysP.hZ.h
Zlib.h           ZlibErrors.h     ZlibP.h      ZxBuffer.h
                 ZxDeviceIO.h
ZxDirectory.h    ZxFiles.h        ZxGetMapPath.h    ZxIPC.h
                 ZxMath.h
ZxMemory.h       ZxMntentP.h      ZxProcesses.h     ZxSignals.h
                 ZxStrings.h
ZxSysTables.h    ZxTime.h         assert.h          ctype.h
                 dirent.h
errno.h          fcntl.h          file.h            filio.h
                 if.h
in.h             inet.h           ioccom.h          ioctl.h
                 ipc.h
limits.h         locale.h         malloc.h          math.h
                 memory.h
mman.h           mntent.h         modem.h           msg.h
                 netdb.h
param.h          poll.h           pwd.h             regex.h
                 resource.h
rtprio.h         search.h         select.h          sem.h
                 setjmp.h
shm.h            signal.h         socket.h          stat.h
                 stdarg.h
stdio.h          stdlib.h         string.h          strings.h
                 strredir.h
swap.h           symlink.h        syslog.h          termio.h
                 termios.h
termiox.h        time.h           times.h           types.h
                 uio.h
un.h             unistd.h         utime.h           utsname.h
                 values.h
vfs.h            wait.h

/h/JMTK/lib:
Jms  Jmv

/h/JMTK/lib/Jms:
libDa.a         libJms.a        libcallbacks.a  libsupport.a

/h/JMTK/lib/Jmv:
libJMTK.a

/h/JMTK/man:
Jmv

/h/JMTK/man/Jmv:
man1       man2       man3       man4       man5       man6
man7
man8       man9       man_macro  mann

/h/JMTK/man/Jmv/man1:
```

/h/JMTK/man/Jmv/man2:

/h/JMTK/man/Jmv/man3:

GenAddFeatures.3	GenAddProducts.3	GenAttach.3
GenChangedFeature.3	GenChangedMap.3	GenClip.3
GenCoord.3	GenDetach.3	
GenDrawingFeature.3		
GenDrawingMap.3	GenError.3	
GenFlushAllRequests.3		
GenGetDisplay.3	GenInit.3	GenNextRequest.3
GenPending.3	GenRemoveFeatures.3	
GenRemoveProducts.3		
GenReserveFeature.3	GenReserveMap.3	GenSendError.3
GenServerToSocket.3	JMV_LoadMap.3	JMV_UnLoadMap.3
MAbortAnimation.3	MAbortMap.3	MAddFeature.3
MAddFeatures.3	MAddInput.3	MAddObject.3
MAddPoint.3	MAddProduct.3	MAddProducts.3
MAddTimeOut.3	MAddVolume.3	
MApplyAttributes.3		
MApplyColor.3	MApplyData.3	
MApplyFillOffset.3		
MApplyFillType.3	MApplyFillWeight.3	MApplyFont.3
MApplyHiLite.3	MApplyLineStyle.3	MApplyLineType.3
MApplyLineWidth.3	MApplyPickability.3	MApplyPixel.3
MApplyTemplate.3	MApplyVisibility.3	MChangeMap.3
MChangeSymbol.3	MChangeText.3	
MChannelToSocket.3		
MCloseChannel.3	MCopyTemplate.3	MCreateClass.3
MCreateList.3	MCreateObject.3	MCreatePoly.3
MCreateTemplate.3	MCreateText.3	MCreateWindow.3
MDebug.3	MDestroyList.3	MDestroyObject.3
MDestroyWindow.3	MDrawArc.3	MDrawBitmap.3
MDrawBox.3	MDrawChar.3	MDrawChar16.3
MDrawCircle.3	MDrawEllipse.3	MDrawLine.3
MDrawPolyLine.3	MDrawPolygon.3	MDrawRectangle.3
MDrawSector.3	MDrawSegment.3	MDrawSlash.3
MDrawSymbol.3	MDrawText.3	MDrawWeather.3
MDrawWorld.3	MError.3	MExchangeObject.3
MFlush.3	MFlushAllEvents.3	
MGetProjectionData.3		
MGetSearchPath.3	MGetServiceContext.3	MGetXWindow.3
MKillServer.3	MListFeatures.3	MListMaps.3
MListObjects.3	MMainLoop.3	MMapWindow.3
MMemory.3	MModifyFeature.3	MModifyFeatures.3
MModifyObject.3	MMoveObject.3	MNextEvent.3
MNoOp.3	MOpenChannel.3	MPending.3
MPixelsToPosition.3	MPositionToPixels.3	MPutBackEvent.3
MQueryChannel.3	MQueryFeatures.3	MQueryMap.3
MQueryObject.3	MQueryObjectBBox.3	MQueryWindow.3
MQuickZoom.3	MRecenterMap.3	MReleaseFocus.3
MReleaseWindow.3	MReloadSearchPath.3	MRemoveFeature.3
MRemoveFeatures.3	MRemoveInput.3	MRemoveObject.3
MRemoveProduct.3	MRemoveProducts.3	MRemoveTimeOut.3
MRemoveVolume.3	MReorderMaps.3	MRequestFocus.3
MRestoreCursor.3	MScaleMap.3	MSendEvent.3
MSetAnimateKeys.3	MSetAttributes.3	MSetColor.3
MSetCursorAnnotation.3	MSetCursorMode.3	MSetData.3

```
MSetEventHandler.3      MSetEventMask.3         MSetFillOffset.3
MSetFillType.3          MSetFillWeight.3        MSetFont.3
MSetHiLite.3            MSetHiLiteColor.3       MSetIntensity.3
MSetLineStyle.3         MSetLineType.3          MSetLineWidth.3
MSetMapBounds.3         MSetMapColors.3
MSetMapColorsByRGB.3
MSetMapWidth.3          MSetObjectData.3        MSetOffset.3
MSetPickability.3       MSetPixel.3             MSetPriority.3
MSetSegment.3           MSetSymbolSize.3        MSetTemplate.3
MSetVisibility.3        MSync.3                 MUnMapWindow.3
MUpdateClass.3          MUseNamedWindow.3       MUseWindow.3
MuAltitude.3            MuConvert.3             MuDistance.3
MuError.3               MuFont.3                MuGeoPosn.3
MuInit.3                MuMgrPosn.3             MuOption.3
MuPosition.3            MuReference.3           MuUnits.3
MuUtmPosn.3

/h/JMTK/man/Jmv/man4:
/h/JMTK/man/Jmv/man5:
/h/JMTK/man/Jmv/man6:
/h/JMTK/man/Jmv/man7:
/h/JMTK/man/Jmv/man8:
/h/JMTK/man/Jmv/man9:
/h/JMTK/man/Jmv/mann:
/h/JMTK/source:
JMD.h           _colors.c       _features.c     _layers.c
                _projection.c
connect.c       country.c       features.c      getwinid.c
main.c
makefile        mappath.c       menu.c          menu.h
                menuCB.c
motif.c         motif.h         myutil.c        myutil.h
                private.h
```

```
projection.c    template.c      xcolors.c       xcountry.c
                xfeatures.c
xintensity.c    xmenu.c         xmenu.h         xprojection.c
```

Importing Map Data

The Spatial Database Manager (SDBM) is used to import and register map data from a CDROM. This data is used by other JMTK components, so importing the data correctly is extremely important. In order to do this, the proper operating environment must be set up. Once this is accomplished, the user selects a new or existing database name, specifies the path and type of data to be imported from the CD, and then directs the SDBM manager to import the data. The SDBM manager validates the data type and structure of the input data, checks for available disk space, imports the data to an available hard-drive partition, and creates metadata entries for each import of data from a CDROM in its tables. The following pages describe how to import CDROM-based DMA map data to the local system using either SUN or HP workstations.

SUN Workstation Map Data Importing Procedure

The following is a step-by-step procedure for importing CDROM-based DMA map data to the local machine using the Spatial Database Manager on the SUN workstation:

A. Operating System Prerequisites:

- Solaris 2.4 (SunOS 5.4)
  - Common Desktop Environment (CDE) running TriTeal Corporation TED Version 4.0
  - X window system version X11R5
  - Motif Version 1.2.2

B. Necessary Environment Variables:

XAPPLRESDIR	This variable identifies the path to the resource directory. The resource files located there define the look and feel of the application and support error message and map display.
JMS_DBHOME	This variable identifies the path to the database home directory where the databases are created and maintained.
JMS_DATA_DIRS*	This variable is comprised of a colon-separated list of pathnames. Each path identified should be a pre-existing directory available for data import.
JMS_ERROR_PATH	This variable identifies the pathname of the error messages file used by the application.
JMS_CDROM	This variable is comprised of a colon-separated list of pathnames identifying possible mount points for CDROMS.
DISPLAY	This variable identifies the screen that the application will present the graphical user interface on.

* Note: The pathname to the directory must explicitly list the absolute path to the actual physical partition the directory resides on. This enables the application to properly determine the available space on that partition.

C. Operator steps

1. Login using the prescribed operating system environment.

2. Change directory to the pathname of the JMTK/Scripts directory.

3. Type **source .cshrc.JMTK <enter>** to set up the environmental variables, which will also place the SDBM executable in your path.

4. Insert the CDROM in the drive. The volume manager should automatically mount the CD in a few moments, and you should see the root directory of the CD under /cdrom if you type **ls -l /cdrom <enter>**.

5. Check the JMS_DATA_DIRS environmental variable by typing at the shell prompt: **echo $JMS_DATA_DIRS <enter>** and make a preliminary check that it contains a path with enough disk space, by typing **df -k <enter>** and comparing the space on the CD with the space on the partition holding a path in question. If necessary, add another path to the variable, or have your system administrator make another partition available for storage of the data.

6. Start the SDBM by typing **SDBM & <enter>**.

7. When the SDBM window appears, there should be a list of existing databases in a pane with a scrollbar to the right of the map pane. You may double click on one of the database names to see the type of data stored within.

8. To create a new database, select the "New Database" option under the Files Menu on the main menu bar of the SDBM window. Enter the database name into the "Name" text field, and approve by selecting the "Okay" button. The new name will appear I the right pane of the SDBM window.

9. To import data from the CD into a database listed in the right pane of the SDBM window, select the "Import" option under the Files Menu on the main menu bar of the SDBM window.

10. The "Import Parameters" window should appear.

11. For VPF and RPF data, enter the path to the CDROM up to and including the **vpf** or **rpf** directory in the Source Path text field. For DTED data, enter the path up to, but not including the **dted** directory. The root directory of the CD can be determined as in step 4, above.

12. Use the Database Name popup list to pick the database to import into.

13. Use the Data Type popup to pick the data type of the import media.

14. Click on the DISK radio button for the Destination.

15. Click on the "Import" button to begin the Import process.

    ***** Important note: *****

Once you click on the "Import" button, several background processes are initiated to copy the data from the CD to your hard drive partition. In this version, there is no way to stop this activity from the SDBM user interface. The processes have to be killed from the UNIX command line by using the ps -aefl <enter> command to check the process table and the kill -15 [pid] <enter> command to kill the process id (pid) of each of the processes. <u>LOGGING OUT OF THE CDE WILL NOT STOP THE IMPORT.</u>

A successful import will first display a "Validating Product Data" message, then an "Importing Data" message, and finally, a "Building Metadata" message. Other messages that may occur are "Disk Space Error", "Invalid Data Type", and "Source Path Does Not Exist". In this version of the SDBM, you must first close the Input Parameters window from its Files menu and then Exit the SDBM window from its Files menu (and confirm the popup), and go back to step 6, above, to continue with another import after either a successful or unsuccessful attempt.

16. To eject the cd, type **eject <enter>**.

HP Workstation Map Data Importing Procedure

The following is a step-by-step procedure for importing CDROM-based DMA map data to the local machine using the Spatial Database Manager on the SUN workstation:

A. Operating System Prerequisites:

- HP-UX 9.07
- HP Vue
- X window system version X11R5
- Motif Version 1.2.2

B. Necessary Environment Variables:

XAPPLRESDIR	This variable identifies the path to the resource directory. The resource files located there define the look and feel of the application and support error message and map display.
JMS_DBHOME	This variable identifies the path to the database home directory where the databases are created and maintained.
JMS_DATA_DIRS*	This variable is comprised of a colon-separated list of pathnames. Each path identified should be a pre-existing directory available data import.
JMS_ERROR_PATH	This variable identifies the pathname of the error messages file used by the application.
JMS_CDROM	This variable is comprised of a colon-separated list of pathnames identifying possible mount points for CDROMS.
DISPLAY	This variable identifies the screen that the application will present the graphical user interface on.

* Note: The pathname to the directory must explicitly list the absolute path to the actual physical partition the directory resides on. This enables the application to properly determine the available space on that partition.

C. Operator steps

1. Login using the prescribed operating system environment.

2. Change directory to the pathname of the JMTK/Scripts directory.

3. Type **source .cshrc.JMTK <enter>** to set up the environmental variables, which will also place the SDBM executable in your path.

4. Insert the CDROM in the drive. Then as superuser, type **/etc/mount /dev/dsk//c201d4s0 /cdrom <enter>**. Then type **/etc/mount <enter>** to verify the CD has been mounted. Leave superuser mode. You should then be able to see the root directory of the CD if you type **ls -l /cdrom <enter>**.

5. Check the JMS_DATA_DIRS environmental variable by typing at the shell prompt: **echo $JMS_DATA_DIRS <enter>** and make a preliminary check that it contains a path with enough disk space, by typing **bdf <enter>** and comparing the space on the CD with the space on the partition holding a path in question. If necessary, add another path to the variable, or have your system administrator make another partition available for storage of the data.

6. Start the SDBM by typing **SDBM & <enter>**.

7. When the SDBM window appears, there should be a list of existing databases in a pane with a scrollbar to the right of the map pane. You may double click on one of the database names to see the type of data stored within.

8. To create a new database, select the "New Database" option under the Files Menu on the main menu bar of the SDBM window. Enter the database name into the "Name" text field, and approve by selecting the "Okay" button. The new name will appear I the right pane of the SDBM window.

9. To import data from the CD into a database listed in the right pane of the SDBM window, select the "Import" option under the Files Menu on the main menu bar of the SDBM window.

10. The "Import Parameters" window should appear.

11. For VPF and RPF data, enter the path to the CDROM up to and including the **VPF** or **RPF** directory in the Source Path text field. For DTED data, enter the path up to, but not including the **DTED** directory. The root directory of the CD can be determined as in step 4, above.

12. Use the Database Name popup list to pick the database to import into.

13. Use the Data Type popup to pick the data type of the import media.

14. Click on the DISK radio button for the Destination.

15. Click on the “Import” button to begin the Import process.

***** Important note: *****

Once you click on the “Import” button several background processes are initiated to copy the data from the CD to your hard drive partition. In this version, there is no way to stop this activity from the SDBM user interface. The processes have to be killed from the UNIX command line by using the ps -aefl <enter> command to check the process table and the kill -15 [pid] <enter> command to kill the process id (pid) of each of the processes.
<u>LOGGING OUT OF THE CDE WILL NOT STOP THE IMPORT.</u>

A successful import will first display a “Validating Product Data” message, then an “Importing Data” message, and finally a “Building Metadata” message. Other messages that may occur are “Disk Space Error”, “Invalid Data Type”, and “Source Path Does Not Exist”. In this version of the SDBM, you must first close the Input Parameters window from its Files menu and then Exit the SDBM window from its Files menu (and confirm the popup), and go back to step 6, above, to continue with another import after either a successful or unsuccessful attempt.

16. To unmount the CD, make sure all processes have left the directory tree of the CD. Then as superuser type **/etc/umount /cdrom <enter>**. You can verify that the CD is no longer mounted by typing **/etc/mount <enter>**. It is now safe to eject the CD by depressing the eject button on the drive.

Appendix A
Test Bed / Demo Software

GCCS JMTK 3.0/DII Version 2.0 is a toolkit. The object libraries are provided in the segment on the tape. As a toolkit, there is nothing to run or see directly. Accordingly, some source code is provided which tests the APIs and demonstrates JMTK capabilities to the Mission Application Developer. This source code also provides examples of uses of the APIs.

The following list indicates the source files in this appendix:

```
JMD.h
_colors.c
_features.c
_layers.c
_projection.c
connect.c
country.c
features.c
getwinid.c
main.c
makefile
mappath.c
menu.c
menu.h
menuCB.c
motif.c
motif.h
myutil.c
myutil.h
private.h
projection.c
projection.h
template.c
xcolors.c
xcountry.c
xfeatures.c
xintensity.c
xmenu.c
xmenu.h
xprojection.c
```

```
/*!!=====================================================================
     Disclaimer:             This software was developed for the
                             Department of Defense (DoD). All US
                             Government restrictions on software source
                             code distribution apply.

     Attribution:   This software is considered part of the GCCS Joint
                             Mapping Toolkit (JMTK).

     Organization:  Computer Sciences Corporation
                             6565 Arlington Boulevard
                             Falls Church, VA  22042 703.538.7276 Date:
                             19960308
                             Copyright (1996)        All Rights Reserved
     Source File:   JMD.h
     Domain:                 PUBLIC
     Scope:                  JMTK Demo
     Version:                1.1
     Last Delta:             04/08/96 18:52:30
     Checked Out:   04/08/96 18:58:56
     Software Category:

     Revision History:

     Version        Person                   Date         Reason
     =========      ==================       ========     =============
     sw 001         Brad Bunton              08 MAR 1996  Initial Draft
==============================================================================

     Purpose:       This include file defines all of the public structures,
                    constanst, and API's for creating the demonstration.
                    All Demo API source files should include this file.

     Notes:

=========================================================================!!*/
```

```
#include <M/Mlib.h>
#include <M/Mu.h>

#ifndef _JMD_H
#define _JMD_H

#define MAX_MAP_ATTRIBUTES_NUM  10

typedef char * tJmdPointer;

typedef struct {
          Channel  channel;
          WindowId wid;
} tJmdConnectionId;

typedef struct {
          Bool          status;
          MapColor     *pColor;
          MapLineStyle  linestyle;  /* Note: field not used for country
feature */ } tJmdFeatureAttributes;

typedef struct {
          char *topic;
          char *sub_topic;
          char *detail_topic;
          char *txt;
} tJmdHelpMsg;

typedef enum {
          JmdHelpTopic,
          JmdHelpSubTopic
} tJmdHelpType;

typedef enum {
          JmdArc,
          JmdBackground,
          JmdBottomContours,
          JmdCenter,
          JmdCountry,
          JmdForeground,
          JmdGridlines,
          JmdIntensity,
          JmdName,
          JmdPoliticalBoundaries,
          JmdPolyline,
          JmdProjection,
          JmdRectangle,
          JmdRivers,
          JmdRoads,
          JmdSector,
          JmdSymbol,
          JmdText,
          JmdWidth
} tJmdMapAttributesType;

typedef enum {
          JmdFeatureDraw,
          JmdFeatureModify,
          JmdFeatureErase
} tJmdFeatureAction;

typedef struct {
          char *pArea;
          char *UpperLeft;
          char *LowerRight;
} tJmdRectangleData;
typedef struct {
```

```
          char *pArea;
          char *pCenter;
          char *pStartBrg;
          char *pEndBrg;
          char *pRange;
} tJmdArcData;

typedef struct {
          char *pCenter;
          char *pText;
} tJmdTextData;

typedef struct {
          char *pArea;
          char *pCenter;
          char *pStartBrg;
          char *pEndBrg;
          char *pOuterRange;
          char *pInnerRange;
} tJmdSectorData;

typedef struct {
          char *pLabel;
          char *pCenter;
          char *pRotation;
          char *pSymbol;
} tJmdSymbolData;

typedef struct {
          char  *pArea;
          char  *pPosition;
          int    nPoints;
          char **pPoint;
} tJmdPolylineData;

/*typedef union {*/
typedef struct {
          int                     type;
          tJmdArcData             *arc;
          tJmdPolylineData        *polyline;
                      tJmdRectangleData  *rectangle;
          tJmdSectorData          *sector;
          tJmdSymbolData          *symbol;
          tJmdTextData            *text;
} tJmdLayerData;

typedef struct _tJmdLayerAttributes {
          char                              *pName;
          Bool                               status;
          char                              *color;
          MapLineStyle                       linestyle;
          char                              *pFont;
          MapLineType                        linetype;
          tJmdLayerData                     *data;
                      struct _tJmdLayerAttributes *pNext;
} tJmdLayerAttributes;

typedef struct {
                        tJmdLayerAttributes     *pArc;
          char                          *pName;
          MapColor                      *pBackground;
          tJmdFeatureAttributes   *pBottomContours; MapPoint    *pCenter;
                      tJmdFeatureAttributes  **pCountry;
          MapColor                      *pForeground;
          tJmdFeatureAttributes    *pGridlines;
          int                           intensity;
          int                           ncountries;
          tJmdLayerAttributes      *pRectangle;
```

```
          tJmdFeatureAttributes    *pPoliticalBoundaries;
          tJmdLayerAttributes      *pPolyline;
          ProjectionType               pProjection;
          tJmdFeatureAttributes    *pRivers;
          tJmdFeatureAttributes    *pRoads;
          tJmdLayerAttributes      *pSector;
          tJmdLayerAttributes      *pSymbol;
          FLOAT                        pWidth;
          tJmdLayerAttributes      *pText;
} tJmdMapAttributes;

typedef enum {
          JmdOkButton,
          JmdHelpButton,
          JmdCancelButton,
} tJmdButtonTag;

typedef enum {
          JmdDefaultMapSelectionBox = 0, JmdDefaultMapWidthText         = 3,
          JmdDefaultMapCenterText   = 5, JmdEditLayerData
                         = 4,
          JmdRedScale                      = 2,
          JmdGreenScale,
          JmdBlueScale,
          JmdForegroundToggle         = 0,
          JmdBackgroundToggle,
          JmdIntensityScale           = 0,
          JmdFeatureRadioBox          = 0,
          JmdStatusRadioBox,
          JmdColorRadioBox,
          JmdLinestyleRadioBox,
          JmdMapListSelectionBox      = 0,
          JmdMapListRadioBox          = 2,
          JmdRPF                         = 0,
          JmdVPF,
          JmdDTED
} tJmdWidgetChildNumber;


/*****************************************************************//*
   Chart Service function prototypes                              */
/* ************************************************************** */
ObjectId JMD_TemplateCreate( tJmdConnectionId *pId, char *pLinecolor,

                                MapLineType linetype, MapLineStyle linestyle,
                                MapFillType filltype, char *pFont );

WindowId JMD_WindowCreate( Channel channel, char *pName, char *pTitle,
                                int height, int width, Window xwid );

Channel JMD_ChannelOpen( void );

void JMD_FeatureSet( tJmdConnectionId *pId, tJmdMapAttributesType type,
                         tJmdFeatureAction action, MapLineStyle linestyle,
                         unsigned short red, unsigned short green,
                         unsigned short blue );

void JMD_ProjectionSet( tJmdConnectionId *pId, char *name, char *width, char
                             *center);


/* ************************************************************** */ /*
         X-window/Motif function prototypes      */ /*
************************************************************** */ void
JMD_DefaultMapWindowCreate( Widget parent, tJmdConnectionId *pId );

void JMD_FeatureWindowCreate( Widget parent, tJmdConnectionId *pId,
```

```
                                    tJmdMapAttributesType type );
void JMD_MapColorsWindowCreate( Widget parent, tJmdConnectionId *pId );
void JMD_MapIntensityWindowCreate( Widget parent, tJmdConnectionId *pId );

#endif  /* _JMD_H */
```

```
/*   File:  xcolors.c   */
#include "JMD.h"
#include "motif.h"

Widget gUSER_DEFINED_COLORS_WINDOW = NULL;

void UserDefinedColorsButtonCB ( Widget w, tJmdButtonTag button_id, XtPointer
                                        call_data );

void UserDefinedColorsChangeCB ( Widget w, MapColor *color, XtPointer
                                        call_data );

void CreateUserDefinedColorsWindow ( Widget parent, MapColor *color, XtPointer
                                             call_data )
{
  Widget bboard,
             gparent,
             scale_red, scale_green, scale_blue,
             sep,
             child,
             btn_ok, btn_cancel;
  Cardinal  n,
             numChildren;
  Arg        args [15];
  Position  x, y;
  Dimension h, w;
  int        nitems;
  char    **items;

            /* determine if the window exists but is unmanaged */
  gparent = XtParent(parent);
  gparent = XtParent(parent);
  if (gUSER_DEFINED_COLORS_WINDOW) {
                   bboard = gUSER_DEFINED_COLORS_WINDOW;

         child = XtNameToWidget (bboard, "Ok"); XtSetArg (args[0],
         XmNuserData, color); XtSetValues (child, args, 1);

         child = XtNameToWidget (bboard, "scale_red"); XmScaleSetValue
         (child, color -> red);

         child = XtNameToWidget (bboard, "scale_green");
         XmScaleSetValue (child, color -> green);

         child = XtNameToWidget (bboard, "scale_blue");
         XmScaleSetValue (child, color -> blue);

         XtManageChild (bboard);

         return;
  }

  /* create bulletin board */
  bboard = MyCreateBulletinBoardDialog (gparent, "bboard", 500, 500,
                                             "User Defined Color",
XmRESIZE_ANY);
  scale_red = MyCreateScale (bboard, "scale_red", 15, 10, 255, 30, 255, 0);
  XtSetArg (args[0], XmNbackground, GetPixel (bboard, "red3"));
  XtSetArg (args[1], XmNuserData, 0);
  XtSetArg (args[2], XmNvalue, color -> red);
  XtSetValues (scale_red, args, 3);

  scale_green = MyCreateScale (bboard, "scale_green", 100, 10, 255, 30, 255,
  0); XtSetArg (args[0], XmNbackground, GetPixel (bboard, "forest green"));
  XtSetArg (args[1], XmNuserData, 1);
  XtSetArg (args[2], XmNvalue, color -> green);
  XtSetValues (scale_green, args, 3);
```

```
   scale_blue  = MyCreateScale (bboard, "scale_blue", 175, 10, 255, 30, 255,
   0); XtSetArg (args[0], XmNbackground, GetPixel (bboard, "royal blue"));
   XtSetArg (args[1], XmNuserData, 2);
   XtSetArg (args[2], XmNvalue, color -> blue);
   XtSetValues (scale_blue, args, 3);

   MyWidgetGetDimension (bboard, &h, &w);
     sep = MyCreateSeparator (bboard, "sep1", 10, NULL, XmHORIZONTAL, 230);

   btn_ok = MyCreatePushButton (bboard, "Ok", 10, NULL,
                                      (XtCallbackProc)UserDefinedColorsButtonC
                                      B, (XtPointer)JmdOkButton);
   XtSetArg (args[0], XmNuserData, color);
   XtSetValues (btn_ok, args, 1);
   MyWidgetGetPosition (btn_ok, &x, &y);

   btn_cancel = MyCreatePushButton (bboard, "Cancel", x+125, y,
                                            (XtCallbackProc)UserDefinedColorsBut
                                            tonCB, (XtPointer)JmdCancelButton);

   MyBulletinBoardSetDefaultButton (bboard, btn_ok);

   /* add callbacks */
   XtAddCallback (scale_red, XmNvalueChangedCallback,
                      (XtCallbackProc) UserDefinedColorsChangeCB, color);
   XtAddCallback (scale_red, XmNdragCallback,
                      (XtCallbackProc) UserDefinedColorsChangeCB, color);
   XtAddCallback (scale_green, XmNvalueChangedCallback,
                      (XtCallbackProc) UserDefinedColorsChangeCB, color);
   XtAddCallback (scale_green, XmNdragCallback,
                      (XtCallbackProc) UserDefinedColorsChangeCB, color);
   XtAddCallback (scale_blue, XmNvalueChangedCallback,
                      (XtCallbackProc) UserDefinedColorsChangeCB, color);
   XtAddCallback (scale_blue, XmNdragCallback,
                      (XtCallbackProc) UserDefinedColorsChangeCB, color);

   gUSER_DEFINED_COLORS_WINDOW = bboard;
}

void UserDefinedColorsButtonCB ( Widget w, tJmdButtonTag button_id, XtPointer
                                          call_data )
{
   Widget    parent, child;
   Arg        args[2];
   MapColor *color;
   int        red_val, green_val, blue_val;

   parent = XtParent(w);
   if (button_id == JmdOkButton) {
          child = XtNameToWidget (parent, "scale_red"); XmScaleGetValue
          (child, &red_val);

          child = XtNameToWidget (parent, "scale_green"); XmScaleGetValue
          (child, &green_val);

          child = XtNameToWidget (parent, "scale_blue"); XmScaleGetValue
          (child, &blue_val);

                    XtSetArg (args[0], XmNuserData, &color);
          XtGetValues (w, args, 1);

          color -> red   = red_val;
          color -> green = green_val; color -> blue  = blue_val;
   }
   XtUnmanageChild (parent);
}
```

```
void UserDefinedColorsChangeCB ( Widget w, MapColor *color, XtPointer
                                          call_data )
{
  int          scale_num,   /* 0 = red, 1 = green, 2 = blue */
               value,       /* value of scale */
               side;        /* 0 = background, 1 = foreground */ Widget
  parent,
             *children;
  Arg          args[2];
  MapColor **mapcolorptr,
             *mcptr1,
             *mcptr2;
  Pixel pixel_value;

  XmScaleGetValue (w, &value);
  XtSetArg (args[0], XmNuserData, &scale_num);
  XtGetValues (w, args, 1);

  parent = XtParent(w);
  XtSetArg (args[0], XmNchildren, &children);
  XtGetValues (parent, args, 1);

  switch (scale_num) {
    case 0: color -> red   = value; break;   /* red */
    case 1: color -> green = value; break;   /* green */ case 2: color ->
    blue  = value; break;   /* blue */
  }
  pixel_value = (color -> blue + (color->green *256) + color -> red *
                   (256*256));
                printf ("pixel_value = %u\n", pixel_value);
  XtSetArg (args[0], XmNbackground, pixel_value); XtSetValues (children[4],
  args, 1);
}
```

```
/*   File:  _features.c   */
Widget gFEATURES_WINDOW = NULL;
void FeaturesToggleButtonCB();
void CreateFeaturesWindow ( Widget parent )
{
   Widget      bboard, lbl, feats_box, status_box, color_box,
               linestyle_box, sep, btn_ok, btn_cancel, *children; char
   **items;
   int         nitems,
               width,
               i;
   Position  x, y;

   /* create bulletin board */
   bboard = MyCreateBulletinBoardDialog (parent, "bboard", 500, 300,
                                                  "Features Edit Window",
   XmRESIZE_ANY); XtUnmanageChild (bboard);
   /* create feature items box */
   lbl = MyCreateLabel (bboard, 10, NULL, "Feature:");
   nitems = 6;
   items  = MyVaCreateCharList( nitems, "Bottom Contours", "Country Colors",
                                 "Gridlines", "Boundaries", "Rivers",
                                 "Roads" );
   feats_box = MyCreateBox( bboard, "features", MyRADIO_BOX, 20, NULL,
                              nitems, items, 2 );
   MyDestroyCharList (nitems, items);
   width = MyBulletinBoardGetWidthFromX( bboard, 10 );
    sep = MyCreateSeparator (bboard, "sep1", 10, NULL, XmHORIZONTAL, width);


   /* create status box */
   lbl = MyCreateLabel (bboard, 10, NULL, "Status:");
   nitems = 2;
   items  = MyVaCreateCharList (nitems, "Off", "On");
   status_box = MyCreateBox (bboard, "radio_status", MyRADIO_BOX, 20, NULL,
                                 nitems, items, 2);
   MyDestroyCharList (nitems, items);
    sep = MyCreateSeparator (bboard, "sep1", 10, NULL, XmHORIZONTAL, width);

   /* create color box */
   lbl = MyCreateLabel (bboard, 10, NULL, "Color:");
   nitems = 9;
   items  = MyVaCreateCharList (nitems, "Black", "Blue", "Green",
                                            "Cyan", "Red", "Magenta",
                                            "Yellow", "White", "Other
                                            ...");
   color_box = MyCreateBox (bboard, "radio_color", MyRADIO_BOX, 20, NULL,
                               nitems, items, 3);
   MyDestroyCharList (nitems, items);
    sep = MyCreateSeparator (bboard, "sep2", 10, NULL, XmHORIZONTAL, width);

   /* create linestyle box */
   lbl = MyCreateLabel (bboard, 10, NULL, "LineStyle:");
   nitems = 6;
   items  = MyVaCreateCharList (nitems, "Dashed", "DoubleDashed",
                                     "LongDashed", "Dotted",
                                     "DotDashed", "Solid");
   linestyle_box = MyCreateBox (bboard, "radio_linestyle", MyRADIO_BOX, 20,
                                     NULL, nitems, items, 2);
   MyDestroyCharList (nitems, items);
    sep = MyCreateSeparator (bboard, "sep3", 10, NULL, XmHORIZONTAL, width);
   btn_ok = MyCreatePushButton (bboard, "Ok", 10, NULL, NULL, NULL);

   MyBulletinBoardSetDefaultButton (bboard, btn_ok);
   MyWidgetGetPosition (btn_ok, &x, &y);
        btn_cancel = MyCreatePushButton (bboard, "Cancel", width-100, y,
                                              NULL, NULL);
   children = (Widget *) MyWidgetGetChildren( feats_box ); XtAddCallback(
```

```
   children[0], XmNarmCallback,
                    (XtCallbackProc) FeaturesToggleButtonCB, (XtPointer)
                    JmdBottomContours );
   XtAddCallback( children[1], XmNarmCallback,
                    (XtCallbackProc) FeaturesToggleButtonCB, (XtPointer)
                    JmdCountry );
   XtAddCallback( children[2], XmNarmCallback,
                    (XtCallbackProc) FeaturesToggleButtonCB, (XtPointer)
                    JmdGridlines );
   XtAddCallback( children[3], XmNarmCallback,
                    (XtCallbackProc) FeaturesToggleButtonCB, (XtPointer)
                    JmdPoliticalBoundaries );
   XtAddCallback( children[4], XmNarmCallback,
                    (XtCallbackProc) FeaturesToggleButtonCB, (XtPointer)
                    JmdRivers );
   XtAddCallback( children[5], XmNarmCallback,
                    (XtCallbackProc) FeaturesToggleButtonCB,
                    (XtPointer) JmdRoads );

   gFEATURES_WINDOW = bboard;
}

void FeaturesButtonCB ( Widget w, tJmdButtonTag button_id, XtPointer call_data
) {
   Widget            parent,
                          child;
   int                    item;
   Arg                    args[2];

   /* get parent */
   parent = XtParent(w);
   XtUnmanageChild (parent);

   if (button_id == JmdOkButton) {
          child = XtNameToWidget (parent, "radio_status"); item  =
          MyGetSelectedRadioBoxItem (child);

          child = XtNameToWidget (parent, "radio_color"); item  =
          MyGetSelectedRadioBoxItem (child);

          child = XtNameToWidget (parent, "radio_linestyle"); item  =
          MyGetSelectedRadioBoxItem (child);

                   /* determine what feature action to do */
   }
   else { /* button_id = JmdCancelButton */
   }
   XtUnmanageChild (parent);
}

void FeaturesToggleButtonCB (Widget w, tJmdMapAttributesType type, XtPointer
                                     call_data )
{
     Widget                    *children,
                                 parent;
     int                         feature,
                                 status,
                                 color,
                                   linestyle;
                         tJmdFeatureAttributes  featatts;
                          tJmdMapAttributesType  maptype;

     parent    = XtParent( w );
     parent    = XtParent( parent );
children  = (Widget *) MyWidgetGetChildren( parent );
     feature   = MyGetSelectedRadioBoxItem( children[JmdFeatureRadioBox] );
     status    = MyGetSelectedRadioBoxItem( children[JmdStatusRadioBox] );
```

```
     color       = MyGetSelectedRadioBoxItem( children[JmdColorRadioBox] );
     linestyle = MyGetSelectedRadioBoxItem( children[JmdLinestyleRadioBox] );

     switch( feature )
     {
          case 0:
               maptype = JmdBottomContours; break;
          case 1:
               maptype = JmdCountry; break;
          case 2:
               maptype = JmdGridlines; break;
          case 3:
               maptype = JmdPoliticalBoundaries; break;
          case 4:
               maptype = JmdRivers; break;
          case 5:
               maptype = JmdRoads; break;
     }
     featatts.status    = (Bool) status;
     featatts.pColor = (MapColor *) malloc (sizeof (MapColor));
     GetRGBColorByButtonNumber( featatts.pColor, color ); featatts.linestyle =
     (MapLineStyle) linestyle;

     /* call vimala's code */

     free( featatts.pColor );
}

void
GetButtonNumberByRGBColor ( MapColor *color, int *button_num )
{
  if (color -> red == 0 && color -> green == 0 && color -> blue == 0)
         *button_num = 0;
  else
    if (color -> red == 0 && color -> green == 0 && color -> blue == 255)
         *button_num = 1;
  else
    if (color -> red == 0 && color -> green == 255 && color -> blue == 0)
         *button_num = 2;
  else
   if (color -> red == 0 && color -> green == 255 && color -> blue == 255)
         *button_num = 3;
  else
    if (color -> red == 255 && color -> green == 0 && color -> blue == 0)
         *button_num = 4;
  else
   if (color -> red == 255 && color -> green == 0 && color -> blue == 255)
         *button_num = 5;
  else
   if (color -> red == 255 && color -> green == 255 && color -> blue == 0)
         *button_num = 6;
  else
  if (color -> red == 255 && color -> green == 255 && color -> blue == 255)
         *button_num = 7;
  else
         *button_num = 8;
}

void
GetRGBColorByButtonNumber ( MapColor *color, int button_num ) {
           if (button_num == 8) return;  /* if user defined color */

  color -> red   = 0;
  color -> green = 0;
  color -> blue  = 0;

  switch (button_num) {
         case 0: break;                                  /* black */
```

```
            case 1: color -> blue  = 255;                    /* blue */
                      break;
            case 2: color -> green = 255;                    /* green */
                      break;
            case 3: color -> green = 255;                    /* cyan */
                      color -> blue  = 255;
                      break;
            case 4: color -> red       = 255;                /* red */
                      break;
            case 5:                                          /* magenta */
                      color -> red      = 255;
                      color -> blue  = 255;
                      break;
            case 6:                                          /* yellow */
                      color -> red      = 255;
                      color -> green = 255;
                      break;
            case 7:                                          /* white */
                      color -> red      = 255;
                      color -> green = 255;
                      color -> blue  = 255;
                      break;
    }
}
```

```
#include "JMD.h"
#include "motif.h"

void LayerWindowCreate( Widget parent )
{
     Widget sbox;                /* selectionbox */
     sbox = MyCreateSelectionBoxDialog( parent, "sbox", 300, 300, "Layers" );
     MySelectionBoxSetLabel( sbox, XmDIALOG_LIST_LABEL, "Layers:" );
     MyVaSelectionBoxUnmanageChild( sbox, 2, XmDIALOG_APPLY_BUTTON,
                                             XmDIALOG_HELP_BUTTON );
}

Widget LayerDefaultDataCreate( Widget parent )
{
  Widget       bboard,
              *labeled_txt,
               label,
               optionmenu;
  int          offset,
               cols,
               xint,
               yint,
               i,
               nitems;
  char     **items;
  Arg          args[10];
  Cardinal     n;
  XmString  *list;
  Position     x,
               y;

    bboard = MyCreateBulletinBoard( parent, "DefaultDataBboard", 0, NULL,
                                    400, 300 );
  offset  = 90;
  cols    = 10;
  yint    = 10;

   labeled_txt = MyCreateLabeledTextField( bboard, "name", 10, 10, "Name:",
                                                   offset, cols, "" );
  yint += 35;

  /* create status label  and option menu */
  label        = MyCreateLabel( bboard, 10, yint, "Status:" );
  nitems       = 2;
  items        = MyVaCreateCharList( nitems, "Off", "On" );
  MyWidgetGetPosition( label, &x, &y );
  optionmenu = MyCreateOptionMenu( bboard, "Status:", offset, y-15, 20, 100,
                                             items, nitems );
  MyDestroyCharList( nitems, items );
  yint += 35;

  /* create Color label  and option menu */
  label        = MyCreateLabel( bboard, 10, yint, "Color:" );
  nitems       = 8;
  items        = MyVaCreateCharList( nitems, "Black", "Blue", "Green", "Red",
                                             "White", "Yellow", "Cyan", "Magenta"
  ); MyWidgetGetPosition( label, &x, &y );
  optionmenu = MyCreateOptionMenu( bboard, "Color:", offset, y-15, 20, 100,
                                             items, nitems );
  MyDestroyCharList( nitems, items );
  yint += 35;

  /* create LineStyle and option menu */
  label        = MyCreateLabel( bboard, 10, yint, "LineStyle:" );
  nitems       = 3;
  items        = MyVaCreateCharList( nitems, "GreatCircle", "Rhumbline",
                                             "GeoDesic" );
  MyWidgetGetPosition( label, &x, &y );
```

```
   optionmenu = MyCreateOptionMenu( bboard, "LineStyle:", offset, y-15, 20,
                                          100, items, nitems );
   MyDestroyCharList( nitems, items );
   yint += 35;

   /* create Font and option menu */
   label        = MyCreateLabel( bboard, 10, yint, "Font:" );
   nitems      = 3;
   items        = MyVaCreateCharList( nitems, "Small", "Medium", "Large" );
   MyWidgetGetPosition( label, &x, &y );
    optionmenu = MyCreateOptionMenu( bboard, "Font:", offset, y-15, 20, 100,
                                          items, nitems );
   MyDestroyCharList( nitems, items );
   yint += 35;

   /* create Linetype and option menu */
   label        = MyCreateLabel( bboard, 10, yint, "LineType:" );
   nitems      = 6;
   items        = MyVaCreateCharList( nitems, "Dashed", "LongDashed","Dotted",
                                          "DotDashed", "DoubleDashed", "Solid"
   ); MyWidgetGetPosition( label, &x, &y );
   optionmenu = MyCreateOptionMenu( bboard, "LineType:", offset, y-15, 20,
                                          100, items, nitems );
   MyDestroyCharList( nitems, items );

   return( bboard );
}

Widget * EditLayerDataCreate( Widget parent, tJmdMapAttributesType type ) {
     Widget  *textfields;
     char   **items;
     int        nitems,
                offset,
                cols;


     offset = 100;
     cols   = 20;
     switch( type )
     {
          case JmdArc:
               nitems = 5;
               items  = MyVaCreateCharList( nitems, "Area", "Center", "Start
                                                Brg", "End Brg", "Range" );
               break;
          case JmdPolyline:
               nitems = 2;
               items  = MyVaCreateCharList( nitems, "Area", "Position" );
               break;
          case JmdRectangle:
               nitems = 3;
               items  = MyVaCreateCharList( nitems, "Area", "Upper Left",
                                                "Lower Right" );
               break;
          case JmdSector:
               nitems = 6;
               items  = MyVaCreateCharList( nitems, "Area", "Center",
                                          "Start  Brg", "End Brg",
                                          "Outer Range","Inner Range" );
               break;
          case JmdSymbol:
               nitems = 4;
               items  = MyVaCreateCharList( nitems, "Label", "Center",
                                                "Rotation", "Symbol" );
               break;
          case JmdText:
               nitems = 2;
            items  = MyVaCreateCharList( nitems, "Center", "Text" );
```

```
                break;
     }
      textfields = MyCreateLabeledTextFieldGroup( parent, 10, 10, items,
                                                   nitems, offset,cols);
     MyDestroyCharList( nitems, items );
     return( textfields );
}


void EditLayerDataChange( Widget w, tJmdMapAttributesType type, XtPointer
                                call_data )
{
     Widget     *textfields,
                 bboard_data,
                *children,
                 parent;
     Cardinal   num,
                 n;
     Arg         args[10];
     int         i;

     parent        = XtParent( w );
     parent        = XtParent( parent );
     children     = ( Widget * ) MyWidgetGetChildren( parent );
     bboard_data = children[JmdEditLayerData];

     n = 0;
     XtSetArg( args[n], XmNchildren, &children ); n++; XtSetArg( args[n],
     XmNnumChildren, &num ); n++; XtGetValues( bboard_data, args, n );

     for(i = 0; i < num; i++ )
          XtUnmanageChild( children[i] );

     EditLayerDataCreate( bboard_data, type );

}

void EditLayerWindowCreate( Widget parent )
{
     Widget       bboard,
                  bboard_2,
                  bboard_3,
                  rbox,              /* radio box */
                  sep,
                  pdm,
                  optionmenu,
                 *children,
                  scrollbar;
     char       **items;
     int          nitems;
     Arg          args[10];

     /* create bulletin board */
       bboard = MyCreateBulletinBoardDialog( parent, "bboard", 500, 375,
                                "Edit Layers", XmRESIZE_ANY );

     /* create radio buttons */
     nitems = 6;
     items  = MyVaCreateCharList( nitems, "Arc", "Polyline", "Rectangle",
                                       "Sector", "Symbol", "Text" );
     rbox   = MyCreateBox( bboard, "rbox", MyRADIO_BOX, 10,NULL,nitems,items,3
                                );
     MyDestroyCharList( nitems, items );

     /* create separator */
    sep = MyCreateSeparator( bboard, "sep1", 10, NULL, XmHORIZONTAL, 320 );
```

```
  /* create default data section */
  bboard_2 = LayerDefaultDataCreate( bboard );

  /* create separator */
sep = MyCreateSeparator( bboard, "sep1", 10, NULL, XmHORIZONTAL, 320 );
```

```
     /* create sub bboard */
     bboard_3 = MyCreateBulletinBoard( bboard, "bboard_3", 10, NULL, 500,375);


     /* create sub bboard */
     EditLayerDataCreate( bboard_3, JmdArc );

     /* add scrollbar */
     XtSetArg( args[0], XmNwidth, 30 ); XtSetArg( args[1], XmNheight, 400 );
     XtSetArg( args[2], XmNx, 360 ); XtSetArg( args[3], XmNy, 0 );
scrollbar = XmCreateScrollBar( bboard_2, "scrollbar", args, 3 );
XtManageChild( scrollbar );

     /* add callbacks */
     children = ( Widget * ) MyWidgetGetChildren( rbox ); XtAddCallback(
     children[0], XmNarmCallback,
                       ( XtCallbackProc ) EditLayerDataChange,
                       ( XtPointer) JmdArc );
     XtAddCallback( children[1], XmNarmCallback,
                       ( XtCallbackProc ) EditLayerDataChange,
                         ( XtPointer) JmdPolyline );
     XtAddCallback( children[2], XmNarmCallback,
                       ( XtCallbackProc ) EditLayerDataChange,
                        ( XtPointer) JmdRectangle );
     XtAddCallback( children[3], XmNarmCallback,
                       ( XtCallbackProc ) EditLayerDataChange,
                       ( XtPointer) JmdSector );
     XtAddCallback( children[4], XmNarmCallback,
                       ( XtCallbackProc ) EditLayerDataChange,
                       ( XtPointer) JmdSymbol );
     XtAddCallback( children[5], XmNarmCallback,
                       ( XtCallbackProc ) EditLayerDataChange,
                       ( XtPointer) JmdText );
}
```

```
/*   File:  _projection.c   */
void DefaultMapButtonCB( Widget w, tJmdButtonTag button_id, XtPointer
                                call_data )
{
  Widget                 parent,
                        *pChildren,
                         child;
  Arg                    args[5];
  char                  *pTxtProj,
                        *pTxtCenter,
                        *pTxtWidth;
  tJmdConnectionId  *pId;

  parent = XtParent( w );
  switch( button_id )
  {
       case JmdOkButton:
            XtSetArg( args[0], XmNchildren, &pChildren ); XtSetArg( args[1],
            XmNuserData, &pId ); XtGetValues( parent, args, 2 );
            child = XmSelectionBoxGetChild
                        (pChildren[JmdDefaultMapSelectionBox], XmDIALOG_TEXT);
            pTxtProj   = XmTextGetString( child );
            pTxtWidth  = XmTextGetString( pChildren[JmdDefaultMapWidthText] );
            pTxtCenter = XmTextGetString( pChildren[JmdDefaultMapCenterText]);
            JMD_ProjectionSet( pId, pTxtProj, pTxtWidth, pTxtCenter );
            XtFree( pTxtProj );
            XtFree( pTxtWidth );
            XtFree( pTxtCenter );
            XtUnmanageChild( parent );
            break;
       case JmdHelpButton:
            break;
       case JmdCancelButton:
            XtUnmanageChild( parent );
            break;
  }
}
```

```
/*!!=============================================================================
==
     Disclaimer:             This software was developed for the Department of
                             Defense (DoD). All US  Government restrictions on
                             software source code distribution apply.

     Attribution:  This software is considered part of the GCCS Joint
                             Mapping Toolkit (JMTK).

     Organization: Computer Sciences Corporation
                             6565 Arlington Boulevard
                             Falls Church, VA  22042 703.538.7276    Date:
                             19960000
Copyright (1996)        All Rights Reserved
     Source File:  connect.c
     Domain:                 PUBLIC
     Scope:                  JMTK Demo
     Version:                1.1
     Last Delta:             04/08/96 18:52:31
     Checked Out:  04/08/96 18:58:56
     Software Category:

     Revision History:

     Version       Person                Date          Reason
     =========     ===================   ========      =============
     sw 001        Brad Bunton           15 MAR 96     Initial Draft

=====================================================================

     Purpose:      JMD_ChannelOpen create a connection to the Chart Manager.

     Synopsis:     #include "JMD.h"
                   Channel JMD_OpenChannel( void )
     Description:  This function creates a connection to the Chart
                             Manager.

     Return:       A valid channel is returned on success, the value
                    InvalidChannel is returned on failure.

     Errors:       Errors will be posted in an X-window.

     Notes:

=====================================================================!!*/
```

```
#include "JMD.h"

Channel JMD_ChannelOpen( void )
{
   Channel channel;
   char    node[40];

                        /* connect to the chart manager */
   gethostname( node, 40 );
   channel = MOpenChannel( node );
   if (channel == InvalidChannel)
   {
        fprintf( stdout, "Error connecting to the Chart Manager\n" ); return(
        channel );
   }
   return( channel );
}
```

```
/*!!=========================================================================
     Disclaimer:             This software was developed for the Deprtment of
                             Defense (DoD). All US Government restrictions on
                             software source code distribution apply.

     Attribution:   This software is considered part of the GCCS Joint
                            Mapping Toolkit (JMTK).

     Organization:  Computer Sciences Corporation
                             1301 Piccard Drive
                             Rockville,  Maryland  20850
                             Phone (301.670.2000)    Date: 19951220 Copyright
                             (1995)        All Rights Reseved
     Source File:   country.c
     Domain:                 JMTK Visual
     Scope:                  PUBLIC
     Version:                %001%
     Last Delta:             04/08/96 18:41:35
     Checked Out:   04/08/96 18:58:56
     Software Category:

     Revision History:

     Version        Person                Date          Reason
     =========      ==================    =======       ====================
     sw 001         Wadiah Mikhail        20 DEC 1995   Demonstration for JMTK


==============================================================================

     Purpose:       The main purpose of this function is to display a list of
                    countries contained in the system. The name of the
                    country, the two-character country code, and the current
                    on/off state for each country is listed. There is  and
                    UPPER WIDTH field below the scroll list.

     Synopsis:
                 void CreateCountryListWindow (Widget parent)


     Description:
                    Select the COUNTRY COLORS option from the MAP OPTIONS
                    pull-down menu to display the LIST COUNTRIES window.

     Return:        Widget parent

     Errors:        Only one LIST COUNTRIES window to be displayed at any
                    given time. Notes:

                    List of Country Name, Country Code, and STATE are hard
                    coded for now.


==============================================================================!!*/


extern char **COUNTRY_LIST;
extern int    COUNTRY_LIST_CNT;
Widget        COUNTRY_LIST_TEXT;

void
CountryListBtnExitCB (
    Widget    w,
    XtPointer client_data,
    XtPointer call_data
);
```

```
void
CountryListBtnOkCB (
    Widget    w,
    XtPointer client_data,
    XtPointer call_data
);

void
CreateCountryListWindow (
   Widget parent
);

void CreateCountryListWindow (
   Widget parent
)
{
   int n, i;
   Arg args [15];
   XmString lbl, *cl, btn_on, btn_off, btn_tog, btn_exit, title;
   Widget   lbl_UW, sb, child;

/* Check if country list window already on display                        */
   if (wCOUNTRY_LIST) return;

  /* create the scrolled bar after loading hardcoded country names        */

   n = 0;
   cl = MyCreateXmStringList (COUNTRY_LIST_CNT, COUNTRY_LIST);

  lbl = XmStringCreate ("COUNTRY NAME                  COUNTRY CODE  STATE",
                          XmFONTLIST_DEFAULT_TAG);
   btn_on = XmStringCreate ("ON", XmFONTLIST_DEFAULT_TAG);
           btn_off  = XmStringCreate ("OFF", XmFONTLIST_DEFAULT_TAG);
   btn_tog  = XmStringCreate ("TOGGLE", XmFONTLIST_DEFAULT_TAG); btn_exit =
   XmStringCreate ("EXIT", XmFONTLIST_DEFAULT_TAG); title= XmStringCreate
   ("COUNTRY LIST", XmFONTLIST_DEFAULT_TAG); XtSetArg (args [n],
   XmNdialogTitle, title); n++;
              XtSetArg (args [n], XmNokLabelString, btn_on); n++;
   XtSetArg (args [n], XmNapplyLabelString, btn_off); n++; XtSetArg (args [n],
   XmNcancelLabelString, btn_tog); n++; XtSetArg (args [n],
   XmNhelpLabelString, btn_exit); n++; XtSetArg (args [n], XmNlistLabelString,
   lbl); n++; XtSetArg (args[n], XmNy, 250); n++;
   XtSetArg (args [n], XmNautoUnmanage, False); n++; XtSetArg (args[n],
   XmNlistItems, cl); n++;
         XtSetArg (args[n], XmNlistItemCount, COUNTRY_LIST_CNT); n++;
   XtSetArg (args[n], XmNx, 10); n++;
            sb = XmCreateSelectionDialog (parent, "text", args, n);
   XtManageChild (sb);
   XmStringFree(lbl);

  /* Output "UPPER WIDTH" label                                            */
   lbl = XmStringCreate ("UPPER WIDTH: ", XmFONTLIST_DEFAULT_TAG);
   n = 0;
   XtSetArg (args [n], XmNlabelString, lbl); n++;
   XtSetArg (args [n], XmNy, 500); n++;
   XtSetArg (args [n], XmNx, 10); n++;
                  lbl_UW = XmCreateLabel (sb, "lbl", args, n);
   XtManageChild (lbl_UW);
   XmStringFree (lbl);

/* Disable selection box                                                  */
   child = XmSelectionBoxGetChild (sb, XmDIALOG_TEXT);
   XtUnmanageChild (child);

/* Disable selection title                                                */
   child = XmSelectionBoxGetChild (sb, XmDIALOG_SELECTION_LABEL);
   XtUnmanageChild (child);
```

```
   child = XmSelectionBoxGetChild (sb, XmDIALOG_HELP_BUTTON);
    XtAddCallback (child, XmNactivateCallback, CountryListBtnExitCB, NULL);

   child = XmSelectionBoxGetChild (sb, XmDIALOG_OK_BUTTON); XtAddCallback
   (child, XmNactivateCallback, CountryListBtnOkCB, NULL);

   child = XmSelectionBoxGetChild (sb, XmDIALOG_APPLY_BUTTON); XtUnmanageChild
   (child);
   child = XmSelectionBoxGetChild (sb, XmDIALOG_CANCEL_BUTTON);
   XtUnmanageChild (child);

   child = XmSelectionBoxGetChild (sb, XmDIALOG_TEXT);
   COUNTRY_LIST_TEXT = child;

   wCOUNTRY_LIST = sb;
   MyDestroyXmStringList (COUNTRY_LIST_CNT, cl);

}

/* Country list exit button handler                                        */
void
CountryListBtnExitCB (
   Widget    w,
   XtPointer client_data,
   XtPointer call_data
)
{
   XtDestroyWidget (wCOUNTRY_LIST);
   wCOUNTRY_LIST = NULL;
}

void
CountryListBtnOkCB (
   Widget    w,
   XtPointer client_data,
   XtPointer call_data
)
{
   char *txt;

   txt = XmTextGetString (COUNTRY_LIST_TEXT);
   dDrawCountry(txt);
   XtFree (txt);
}
```

```
/*!!==============================================================================
     Disclaimer:             This software was developed for the Department of
                             Defense (DoD). All US  Government restrictions on
                             software source code distribution apply.

     Attribution:       This software is considered part of the GCCS Joint
                             Mapping Toolkit (JMTK).

     Organization:      Computer Sciences Corporation
                             6565 Arlington Boulevard
                             Falls Church, VA  22042 703.538.7276    Date:
                             19960000
Copyright (1996)        All Rights Reserved
     Source File:       features.c
     Domain:                 PUBLIC
     Scope:                  JMTK Demo
     Version:                1.1
     Last Delta:             04/08/96 18:52:31
     Checked Out:       04/08/96 18:58:56
     Software Category:

     Revision History:

     Version        Person                   Date            Reason
     =========      =====================    ========       ===================
     sw 001         Brad Bunton              15 MAR 96       Initial Draft

==============================================================================
     Purpose:
     Synopsis:      #include "JMD.h"
                    void JMD_FeatureSet(
                         tJmdConnectionId *pId, tJmdFeatureType type,
                         tJmdFeatureAction action, MapLineStyle linestyle,
                   unsigned short red, unsigned short green,
                         unsigned short blue )
     Description:
     Return:        Nothing.
     Errors:        Errors will posted in an X-window.
     Notes:
==============================================================================
===!!*/
```

```
#include "JMD.h"

void JMD_FeatureSet( tJmdConnectionId *pId, tJmdMapAttributesType type,
                         tJmdFeatureAction action, MapLineStyle linestyle,
                         unsigned short red, unsigned short green, unsigned
                         short blue )
{
  FeatureType                 feature_type;
  FeatureProduct              featprod;
  int                         nfeatures;
  FeatureListAttributes *pList;
  MapFeatureAttributes   featatts;

  featprod.sub_type = AnyFeature;
  switch( type )
  {
       case JmdBottomContours:
                 featprod.feature_type = BottomContourFeature;
            break;
       case JmdCountry:
            /* void for now */
            break;
       case JmdGridlines:
            featprod.feature_type = GridLines;
            break;
       case JmdPoliticalBoundaries:
                 featprod.feature_type = PoliticalBoundaries;
            break;
       case JmdRivers:
            featprod.feature_type = RiverFeature;
            break;
       case JmdRoads:
            featprod.feature_type = RoadFeature;
            break;
  }
  if( action == JmdFeatureErase )
  {
       MRemoveFeature( pId -> channel, pId -> wid, &featprod ); MFlush( pId -
       > channel );
       return;
  }
  pList = MListFeatures( pId -> channel, featprod.feature_type,
                              featprod.sub_type, &nfeatures );
  if( pList )
  {
       featatts.mask                 = FEPixelColor | FELineStyle;
       featatts.type                 = pList -> type;
       featatts.atts.line_style  = linestyle; featatts.atts.color.red   =
       red; featatts.atts.color.green = green; featatts.atts.color.blue  =
       blue;
  }
  else
  {
        fprintf( stderr, "JMD_FeatureSet: MListFeatures failed!\n" );
       return;
  }
  switch( action )
  {
       case JmdFeatureDraw:
             MAddFeature( pId -> channel, pId -> wid, &featatts );
            break;
       case JmdFeatureModify:
            MModifyFeature( pId -> channel, pId -> wid, &featatts, ModifySet
            ); MUpdateFeatures( pId -> channel, pId -> wid );
            break;
  }
  MFlush( pId -> channel );
  MFree( (char **) &pList );
```

}

```
/*!!==============================================================================
     Disclaimer:             This software was developed for the Department of
                             Defense (DoD). All US  Government restrictions on
                             software source code distribution apply.

     Attribution:       This software is considered part of the GCCS Joint
                             Mapping Toolkit (JMTK).

     Organization:      Computer Sciences Corporation
                             6565 Arlington Boulevard
                             Falls Church, VA  22042 703.538.7276    Date:
                             19960315
Copyright (1996)        All Rights Reserved
     Source File:       getwinid.c
     Domain:                 JMTK Demo
     Scope:                  PUBLIC
     Version:                1.1
     Last Delta:             04/08/96 18:52:32
     Checked Out:       04/08/96 18:58:56
     Software Category:

     Revision History:

     Version        Person                 Date               Reason
     =========      ====================   ========           ===========
     sw 001         Brad Bunton            08 MAR 96          Initial Draft
==============================================================================

     Purpose:

     Synopsis:      #include "JMD.h"

                    WindowId JMD_WindowCreate(
                         Channel channel, char *pName, char *pTitle,
                      int height, int width, Window xwid )

     Description:

     Return:        On success, a valid window id is returned.  On failure,
                    the value InvalidWindowId is returned.

     Errors:        Errors will posted in an X-window.

     Notes:

==============================================================================!!*/
```

```
#include "JMD.h"

extern Widget gROOT;

WindowId JMD_WindowCreate( Channel channel, char *pName, char *pTitle, int
                                 height, int width, Window xwid )
{
  WindowAttributes        atts;          /* window attributes structure */
  WindowId                wid;                /* chart window id */
  MapValueMask            atts_mask,        /* attribute mask */
                          map_mask;         /* map attribute mask */
  MapChangeAttributes mapatts;       /* map change attributes structure */
  /* create a window and display it */
  atts_mask       = WIName | WIEventMask | WITitle | WIGeometry |
                       WIPixmapExtents;

   /* if xwid > -1, there is request to reparent the default map window */
  if( xwid != -1 )
  {
         atts_mask     = atts_mask | WISpecial | WIParent | WILocation;
         atts.x        = 0;
         atts.y        = 50;
         atts.parent = xwid;
  }
  map_mask                          = CMSetBoundary;
  atts.event_mask                   = KeyPressedMask | BoxMask |
ObjectSelectMask;
  strcpy( atts.name, pName );
  strcpy( atts.title, pTitle ); atts.pixmap_height               = 1024;
  atts.pixmap_width                 = 1280;
  atts.height                       = height;
  atts.width                        = width;
  mapatts.boundary.width.width = WorldWidth;
wid = MCreateWindow( channel, &atts, atts_mask, NULL, map_mask ); if( wid ==
InvalidWindowId )
  {
             fprintf(stderr, "Unable to create map window\n" );
       return( InvalidWindowId );
  }
  else
  {
            fprintf( stdout, "Window created, id = %d\n", wid );
  }
  MMapWindow( channel, wid );
  MFlush( channel );
  return( wid );
}
```

```
#include "JMD.h"
#include "menuCB.c"
#include <JMS.h>

/* global variables */
Widget   ROOT           = NULL;
WindowId MAIN_WINDOW_ID = InvalidWindowId; Channel  CHANNEL        =
InvalidChannel; ObjectId TEMPLATE_ID    = InvalidObjectId;

tJmdConnectionId *CID;

void main (argc, argv)
  int argc;
  char ** argv;
{
 Widget       root, submenu, form,  menu_bar, pdm[20], *children, *children2,
                                   mb, child;
  XtAppContext app_context;
  Arg          args [20];
  int          n, button_cnt, pdm_cnt, cnt;
                   Cardinal     numChildren, numChildren2;
  char * txt;
  String name;

  CID = (tJmdConnectionId *) malloc (sizeof (tJmdConnectionId));
  n    = 0;
  root = XtAppInitialize (&app_context, "Main", NULL, 0, &argc, argv, NULL,
                                 args, n);
  ROOT = root;
  form=MyCreateBulletinBoardDialog( root, "bboard", 550, 500, "JMTK Testbed",
                                          XmRESIZE_NONE );
                                          /*
  XtSetArg( args[0], XmNresizePolicy, XmRESIZE_ANY );
  XtSetValues( form, args, 1 );
  */
  ROOT =form;
menu_bar = (Widget) XMenuCreateMenuBar (form, "MainWindow", MainMenuCallback);

  CHANNEL = JMD_ChannelOpen ();
  CID -> channel = CHANNEL;

  MainMenuCallback( NULL, 112, NULL );

/*
  XtRealizeWidget (root);
  */
  XtAppMainLoop (app_context);
}
```

```
#
JMVINC  =  -I/h/inri/JMV/include JMVLIB  =  -L/h/inri/JMV/lib
XINC    =  -I/usr/include/X11R5
XLIB    =  -L/usr/lib/X11R5
XMINC  =  -I/usr/include/Motif1.2 XMLIB  =  -L/usr/lib/Motif1.2
ALLLIB =  -lXm -lXt -lX11 -lJMTK_1_1 -lm

demo: demo.o dengine.o Xmenu.o Xconv_units.o myutil.o motif.o xlayers.o
          cc -g demo.o dengine.o Xmenu.o Xconv_units.o myutil.o motif.o
          xlayers.o $(JMVLIB) $(XMLIB) $(XLIB) $(ALLLIB) -o demo
dengine.o: dengine_filter.h dengine_filter.c dengine.h dengine.c
          cc -Aa -c -g $(JMVINC) $(XINC) dengine.c

demo.o: demo.h projection.h box.h projection.c storedmap.c country.c box.c
menuCB.c intensity.c layers_update.c xlayers_update.c demo.c cc -Aa -c -g
          $(JMVINC) $(XINC) $(XMINC) demo.c

Xmenu.o: menu.h menu.c Xmenu.h Xmenu.c
          cc -Aa -c -g $(XINC) $(XMINC) Xmenu.c

Xconv_units.o: Xconv_units.c
          cc -Aa -c -g $(XINC) $(XMINC) Xconv_units.c

myutil.o: myutil.h myutil.c
          cc -c -g -Aa $(XMINC) $(XINC) myutil.c

motif.o: motif.h motif.c
          cc -c -g -Aa $(XMINC) $(XINC) motif.c

xlayers.o: picklist.h picklist.c layers.h layers.c xlayers.h xlayers.c
          cc -c -g -Aa $(XMINC) $(XINC) xlayers.c
```

```
#include "JMD.h"
#include <JMS.h>
#include <JMS_Errors.h>
#include "motif.h"

void MapListButtonCB( Widget w,tJmdButtonTag button_id, XtPointer call_data );

void JMD_MapListCreate( Widget parent, tJmdConnectionId *pId )
{
     tJmsDbList     *list;
     tJmsPathStruct *path;
     Widget             bboard,
                        sbox,
                        sep,
                        rbox,
                        btn_ok,
                        btn_cancel;
     int                width,
                        nitems,
                        i;
     Position           x, y;
     Arg                args[5];
     char          **items;

     bboard = MyCreateBulletinBoardDialog( parent, "bboard", 300, 300, "Map
                                                   List", XmRESIZE_ANY );
     XtSetArg( args[0], XmNuserData, pId );
     XtSetValues( bboard, args, 1 );

     /* create topic selection box */
     sbox = MyCreateSelectionBox( bboard, "sbox", 10, 10, 300, 330 );
     MySelectionBoxSetLabel( sbox, XmDIALOG_LIST_LABEL,
                                 "Maps:              Data Types:" );
     MyVaSelectionBoxUnmanageChild( sbox, 8, XmDIALOG_APPLY_BUTTON,
          XmDIALOG_CANCEL_BUTTON,XmDIALOG_DEFAULT_BUTTON,XmDIALOG_HELP_BUTTON,
     XmDIALOG_OK_BUTTON, XmDIALOG_SELECTION_LABEL, XmDIALOG_SEPARATOR,
     XmDIALOG_TEXT );

     JMS_DbListGet( &list );
     MySelectionBoxSetListItems( sbox, list -> items, list -> pDbNames );

     width = MyBulletinBoardGetWidthFromX( bboard, 0 );
   sep = MyCreateSeparator(bboard, "sep1", 10, NULL, XmHORIZONTAL, width );

     /* create radio box entry */
     nitems = 3;
     items=MyVaCreateCharList( nitems, "RPF", "VPF", "DTED" );
     rbox = MyCreateBox( bboard, "rbox", MyRADIO_BOX, 10, NULL, nitems, items,
1 );

   sep = MyCreateSeparator(bboard, "sep1", 10, NULL, XmHORIZONTAL, width );
     btn_ok = MyCreatePushButton( bboard, "Ok", 10, NULL,(XtCallbackProc)
                                       MapListButtonCB,
                                       (XtPointer) JmdOkButton );
     MyWidgetGetPosition( btn_ok, &x, &y );
     btn_cancel = MyCreatePushButton( bboard, "Cancel", width-100, y,
                                            (XtCallbackProc) MapListButtonCB,
                                            (XtPointer) JmdCancelButton );

     /* free up list */
     for(i = 0; i < list -> items; i++ )
          free( list -> pDbNames[i] );
     free( list -> pDbNames );
     free( list );

}
```

```
void MapListButtonCB( Widget w, tJmdButtonTag button_id, XtPointer call_data )
{
     tJmsConnection        jms_id;
     tJmsPathStruct       *jms_path;
     tJmsDbInventory  *jms_inventory;
     tJmdConnectionId *jmd_id;
     char                 *txt,
                          *jmv_path,
                           buf[256];
     Widget                parent,
                           child,
                          *children;
     Arg                   args[5];
     int                   i,
                           item;

     parent = XtParent( w );
     switch( button_id )
     {
          case JmdOkButton:
               /* get children of bboard */
              children = (Widget *) MyWidgetGetChildren( parent );

               /* get user data (jmv connection id) from parent */
               XtSetArg( args[0], XmNuserData, &jmd_id );
               XtGetValues( parent, args, 1 );

               /* get text field widget id */
               child     =
XmSelectionBoxGetChild(children[JmdMapListSelectionBox],
                                                        XmDIALOG_TEXT );
               txt       = XmTextGetString( child );

               item      = MyGetSelectedRadioBoxItem(
children[JmdMapListRadioBox] );

               /* connect to the database */
                         JMS_DbConnect( txt, &jms_id );

               /* find format type(s) of database */ JMS_InventoryGet( jms_id,
               &jms_inventory);

               /* get the path to the database */ JMS_DataPathnameGet( jms_id,

jms_inventory -> pVolList[0] -> volumnName, &jms_path );
               fprintf(stdout, "%s\n", jms_path -> pPathNames[0] );
               fprintf(stdout, "items = %d\n", jms_path -> items );

               /* create index from inri code */
                     jmv_path = getenv( "MapSearchPath" );
               switch( item )
               {
                    case JmdDTED:
/*
                         JMV_LoadMap( jms_path -> pPathNames[0], "/tmp",
                                        JMV_ATOM_DTED );
*/
                         sprintf(buf, "LoadDTED -s %s -t /tmp", jms_path ->
                                   pPathNames[0] );
                         system(buf);
                         break;
                    case JmdRPF:
/*
                         JMV_LoadMap( jms_path -> pPathNames[0], "/tmp",
                                        JMV_ATOM_RPF );
*/
                         sprintf(buf, "RPFLoad -dev %s -dir /tmp",
                                   jms_path -> pPathNames[0] );
```

```
                    system( buf );
                    break;
            }

            /* add path to chart server */
            for( i = 0; i < jms_path -> items; i++ )
            {
                 MAddVolume( jmd_id -> channel, jms_path -> pPathNames[i]
                 ); MFlush( jmd_id -> channel );
            }
            break;
        case JmdCancelButton:
            break;
    }
    XtUnmanageChild( parent );
}
```

```
/**********************************/
/*  File:  menu.c                 */ /**********************************/

#include "menu.h"

long MenuFindStartText (fptr, menubar_name) FILE * fptr;
   char * menubar_name;
{
   char buf [80],
         name [40];
   long filepos;

   filepos = ftell (fptr);
                     while (fgets (buf, 80, fptr) != NULL) {
      if (buf [0] == '*') {
         /*filepos++;*/
         sscanf (&buf[1], "%[^:]%*c", name);      /* extract name from file */
         if (strcmp (menubar_name, name) == 0) {  /* compare names */
            return (filepos);
         }
      }
      filepos = ftell (fptr);
   }
   return (-1);
}

long MenuFindEndText (fptr)
   FILE * fptr;
{
   char buf [80];
   long filepos;

   filepos = ftell (fptr);
                     while (fgets (buf, 80, fptr) != NULL) {
      if (buf [0] == '*') {
         break;
      }
      filepos = ftell (fptr);
   }
   return (--filepos);
}

char * MenuGetText (menubar_name)
   char * menubar_name;
{
   FILE * fptr;
   char * txt;
   int     startpos, endpos, length;

   fptr = fopen (SYSTEM_MENU_FILE, "r");
               startpos = MenuFindStartText (fptr, menubar_name);
   endpos   = MenuFindEndText (fptr);
   length   = endpos - startpos;
   txt       = (char *) malloc (length +1);
   fseek (fptr, startpos, SEEK_SET); fread (txt, length, 1, fptr);
   fclose (fptr);
   txt[length] = '\0';
   return (txt);
}

int MenuParseText (txt, pos, pbval1, pbval2, btn_id, pbval3) char * txt, *
   pbval1, * pbval2;
   int  * pos,
         * btn_id,
         * pbval3;
{
   while (txt[*pos] != '\0') {
               if (txt[*pos] == '\t') {     /* skip over tabs */
```

```
      (*pos)++;
      continue;
    }
    else
    if (txt[*pos] == '*') {
                sscanf (&txt[*pos], "%*[^:]%*c%d", pbval3);
      while (txt[(*pos)++] != '\n');
      return (tokMENUBAR);
    }
    else
    if (txt[*pos] == '-') {
      sscanf (&txt[++(*pos)], "%[^:]%*c%[^:]%*c%d%*c%d", pbval1, pbval2,
                pbval3, btn_id);
      while (txt[(*pos)++] != '\n');
      return (tokCASCADE_MENU_ITEM);
    }
    else {
      sscanf (&txt[*pos], "%[^:]%*c%[^:]%*c%d", pbval1, pbval2, btn_id);
      while (txt[(*pos)++] != '\n');
      return (tokMENU_ITEM);
    }
  }
  return (tokMENU_END);
}
```

```
#include <stdio.h>
#define SYSTEM_MENU_FILE     "menu.sys"
enum MENU_TOKENS { tokMENUBAR, tokCASCADE_MENU_ITEM, tokMENU_ITEM, tokMENU_END
                        };
long     MenuFindStartText           ();
long     MenuFindEndText             ();
char * MenuGetText                   ();
int      MenuParseText  ();
```

```
void
MainMenuCallback (
  Widget    w,
  int        btn_id,
  XtPointer call_data
)
{
  Layer *layer;
  layer = (Layer *) malloc (sizeof (Layer));
  switch (btn_id) {

          /*   Options under "File"   */
          case 111:  /*                                    Add Product   */
                     dChangeTheMap ();
                     break;
          case 112:  /*                                    Default Map   */
                     dGetWindow ();
                     break;
          case 113:  /*                                    Set Default Map   */
                    CreateDistanceConversionWindow (wROOT);
                     break;
          case 114:  /*                                    Map List   */
                     break;
          case 115:  /*                                    Previous Map/Layer
*/
                     break;
          case 116:  /*                                    Stored Map/Layer
*/
                     CreateStoredMapLayerWindow (wROOT);
                     break;
          case 12:                                         /*    Save   */
                     break;
          case 13:                                         /*    Save As   */
                     break;
          case 14:                                         /*    Print   */
                     break;
          case 151:  /*                                    SDBM Importer   */
                     break;
          case 16:                                         /*    Quit   */
                     dExitAll ();
                     break;

          /*   Options under "Edit"   */
          case 211:  /*                                    Add Layers   */
                     break;
          case 212:  /*                                    Polyline Layers   */
                     layer->xlayer = CreatePolyLineLayerWindow (wROOT);
                     layer->layer  = CreateLayerAttributes ();
InitPolylineData
                     (layer->layer);
                     CURRENT_LAYER = layer;
                    dCreateTemplate ("white", 2, 5, 0, NULL);
                     dCreateEventMask (4);
                     break;
          case 213:  /*                                    Arc Layers   */
                     layer->xlayer = CreateArcLayerWindow (wROOT);
                     break;
          case 214:  /*                                    Sector Layers   */
                     layer->xlayer = CreateSectorLayerWindow (wROOT); break;
          case 215:  /*                                    Rectangle Layers
*/
                     layer->xlayer = CreateRectangleLayerWindow (wROOT);
                     layer->layer  = CreateLayerAttributes ();
                     CURRENT_LAYER = layer;
                     dCreateTemplate ("white", 2, 5, 0, NULL);
                     dCreateEventMask (5);
                     break;
          case 216:  /*                                    Text Layers   */
                     layer->xlayer = CreateTextLayerWindow (wROOT);
```

```
            break;
case 217:   /*                                      Symbol Layers   */
            break;
case 221:   /*                                      Map Colors   */
            CreateMapColorsWindow (wROOT);
            break;
case 222:   /*                                      Country List   */
            CreateCountryListWindow (wROOT);
            break;
case 223:   /*                                      Map Intensity   */
            CreateMapIntensityWindow (wROOT);
            break;
case 23:                                            /*    Symbols   */
            break;
case 24:                                            /*    Distance Conversion   */
            CreateDistanceConversionWindow (wROOT);
            break;

/*   Options under "View"   */
case 31:                                            /*    Zoom   */
            dSetCursorMode (5);
            break;
case 32:                                            /*    Double Width  */
            dScaleMap (2.0);
            break;
case 33:                                            /*    Half Width   */
            dScaleMap (0.5);
            break;
case 34:                                            /*    Pan   */
            dSetCursorMode (3);
            break;
case 35:                                            /*    Center   */
            dSetCursorMode (4);
            break;
case 36:                                            /*    Whole World   */
            dWholeWorld ();
case 37:                                            /*    Insets   */
            break;
case 38:                                            /*    Map Monitor */
            break;

/*   Options under "Mapping"   */
case 411:   /*                                      Bottom Contours */
            break;
case 412:   /*                                      Country Colors   */
            break;
case 413:   /*                                      Grid Lines   */
            dDrawGridLines ();
            break;
case 414:   /*                                      Political Boundaries   */
            dDrawPoliticalBoundaries ();
            break;
case 415:   /*                                      Rivers   */
            break;
case 416:   /*                                      Roads   */
            break;
case 42:                                            /*    Line of Sight   */
            break;
case 43:                                            /*    Area Coverage   */
            break;
case 44:                                            /* Coordinate Transformation
*/
            break;
case 45:                                            /*    Area of Interest   */
            break;
case 46:                                            /*    Map Manager   */
            break;

   }
```

```
}
```

```
/****************************/
/*  File:  motif.c          */ /****************************/


#include "motif.h"

Widget
MyCreateBulletinBoardDialog (

  Widget  parent,
  int       height,
  int       width,
  char   *title

)
{
  Widget   bboard;
  Cardinal n;
  Arg       args [10];
  XmString lbl;

  n = 0;
  if (title) {
    lbl     = XmStringCreate (title, XmFONTLIST_DEFAULT_TAG);
    XtSetArg (args[n], XmNdialogTitle, lbl); n++;
  }
  else {
    lbl = NULL;
  }
  if (width) {
    XtSetArg (args[n], XmNwidth, width); n++;
  }
  if (height) {
    XtSetArg (args[n], XmNheight, height); n++;
  }
  XtSetArg (args[n], XmNautoUnmanage, False); n++; XtSetArg (args[n],
  XmNresizePolicy, XmRESIZE_ANY); n++;
bboard = XmCreateBulletinBoardDialog (parent, "bboard", args, n);
XtManageChild (bboard);
  XmStringFree (lbl);
  return (bboard);
}

Widget
MyCreateBulletinBoard (

  Widget  parent,
  int       x,
  int       y,
  int       height,
  int       width

)
{
  Widget   bboard;
  Cardinal n;
  Arg       args [10];

  n = 0;
  MySetXYArgs (parent, args, &n, x, y);
  if (width) {
    XtSetArg (args[n], XmNwidth, width); n++;
  }
  if (height) {
    XtSetArg (args[n], XmNheight, height); n++;
  }
  XtSetArg (args[n], XmNautoUnmanage, False); n++; XtSetArg (args[n],
```

```
  XmNresizePolicy, XmRESIZE_ANY); n++;
bboard = XmCreateBulletinBoard (parent, "bboard", args, n); XtManageChild
(bboard);
  return (bboard);
}

Widget
MyCreatePushButton (

  Widget          parent,
  char           *title,
  int             x,
  int             y,
  XtCallbackProc  cb,
  XtPointer       cb_val

)
{
  Widget    btn;
  Cardinal  n;
  Arg       args[10];

  n = 0;
  MySetXYArgs (parent, args, &n, x, y);
  XtSetArg (args[n], XmNheight, MyBUTTON_HEIGHT); n++; XtSetArg (args[n],
  XmNwidth, MyBUTTON_WIDTH); n++;
  btn = XmCreatePushButton (parent, title, args, n); XtManageChild (btn);

  /* add callback */
  if (cb) {
            XtAddCallback (btn, XmNactivateCallback, cb, cb_val);
  }
  return (btn);
}

Widget
MyCreateSeparator (

  Widget        parent,
  int           x,
  int           y,
        unsigned char orientation,     /* XmHORIZONTAL, XmVERTICAL */
  int           length

)
{
  Widget   sep;
  Cardinal n;
  Arg      args[10];

  n = 0;
  MySetXYArgs (parent, args, &n, x, y);
  XtSetArg (args[n], XmNorientation, orientation); n++;
  if (orientation == XmVERTICAL) {
    XtSetArg (args[n], XmNheight, length); n++;
  }
  else {
    XtSetArg (args[n], XmNwidth, length); n++;
  }
  sep = XmCreateSeparator (parent, "sep", args, n); XtManageChild (sep);
  return (sep);
}

Widget
MyCreateLabel (

  Widget  parent,
```

```
   int       x,
   int       y,
   char   *txt

)
{
   Widget     label;
   Cardinal n;
   Arg        args[10];

   n = 0;
   MySetXYArgs (parent, args, &n, x, y);
   label = XmCreateLabel (parent, txt, args, n); XtManageChild (label);
   return (label);
}

Widget *
MyCreateLabeledBox (

   Widget  parent,
   int       x,
   int       y,
   int       height,
   int       width,
   char   *txt

)
{
   Widget     child[5],
              sep;
   Cardinal n;
   int        hh,         /* default font char half height */
              hw;         /* default font char half width  */ Arg
   args[10];

   hh = MyCHAR_HEIGHT / 2;
   hw = MyCHAR_WIDTH / 2;

   /* create box counter clockwise starting from top-left */

   /* left */
   child[0] = MyCreateSeparator (parent, x+hw, y+hh, XmVERTICAL, height-hh);

   /* top */
  child[1] = MyCreateSeparator (parent, x+hw, y+hh, XmHORIZONTAL, width-hw);

   /* create label, stick in top-left corner */
   child[2] = MyCreateLabel (parent, x, y, txt);

   /* right */
 child[3] = MyCreateSeparator (parent, x+width, y+hh, XmVERTICAL, height-hh);

   /* bottom */
child[4] = MyCreateSeparator (parent, x+hw, y+height, XmHORIZONTAL, width-hw);

   return (child);
}

Widget
MyCreateBox (

   Widget        parent,
   MyBoxType   type,
   int           x,
   int           y,
   int           nitems,
```

```
   char        **items,
   int           numcols

)
{
   Widget      box;
   Cardinal  n;
   Arg         args[15];
   XmString *Items;

   if (!nitems) {
     return (NULL);
   }

                 Items = MyCreateXmStringList (nitems, items);

   /* create box */
   n = 0;
              XtSetArg (args[n], XmNpacking, XmPACK_COLUMN); n++;
   XtSetArg (args[n], XmNbuttonSet, 0); n++; XtSetArg (args[n],
   XmNresizeWidth, True); n++; XtSetArg (args[n], XmNresizeHeight, True); n++;
   MySetXYArgs (parent, args, &n, x, y);
   XtSetArg (args[n], XmNbuttonCount, nitems); n++; XtSetArg (args[n],
   XmNbuttons, Items); n++; XtSetArg (args[n], XmNnumColumns, numcols); n++;

   if (type == MyRADIO_BOX) {
     box = XmCreateSimpleRadioBox (parent, "rbox", args, n); }
   else
   if (type == MyCHECK_BOX) {
     box = XmCreateSimpleCheckBox (parent, "cbox", args, n); }
   else {
     MyDestroyXmStringList (nitems, Items);
     return (NULL);
   }
   XtManageChild (box);
   MyDestroyXmStringList (nitems, Items);
   return (box);
}

int
MyGetSelectedRadioBoxItem (

   Widget      box

)
{
   Cardinal  n;
   int         i,
               buttonCount;
   Arg         args[10];
   Widget   *children;

   /* get children of box */
   n = 0;
   XtSetArg (args[n], XmNchildren, &children); n++; XtSetArg (args[n],
   XmNbuttonCount, &buttonCount); n++;
   XtGetValues (box, args, n);
   /* if no children, return immediately */
   if (!buttonCount) {
     return (-1);
   }

   /* find selected radio box */
   for (i = 0; i < buttonCount; i++) {
               if (XmToggleButtonGadgetGetState (children[i])) {
       return (i);
     }
```

```
  }
  return (-1);
}

int *
MyGetSelectedCheckBoxItems (

  Widget      box
)
{
  int      *items,
              i,
              cnt,
              buttonCount;       /* number of children */ Cardinal  n;
  Arg         args[10];
  Widget   *children;

  /* get children of box */
  n = 0;
  XtSetArg (args[n], XmNchildren, &children); n++; XtSetArg (args[n],
  XmNbuttonCount, &buttonCount); n++; XtGetValues (box, args, n);

  /* if no children, return immediately */
  if (!buttonCount) {
    return (NULL);
  }

            items = (int *) calloc (sizeof (int) * buttonCount);
  cnt   = 0;

  /* find selected check boxes */
  for (i = 0; i < buttonCount; i++) {
              if (XmToggleButtonGadgetGetState (children[i])) {
       items[cnt++] = i;
    }
  }
  return (items);
}

Widget
MyCreateSelectionBox (

  Widget parent,
  int    x,
  int    y,
  int    height,
  int    width

)
{
  Widget         selectionbox,
                  child;
  Cardinal        n;
  Arg             args[10];
  XmFontList      fontlist;
  XFontStruct *font;

  n = 0;
  MySetXYArgs (parent, args, &n, x, y); XtSetArg (args[n], XmNheight,
  height); n++; XtSetArg (args[n], XmNwidth, width); n++;
selectionbox = XmCreateSelectionBox (parent, "sbox", args, n); XtManageChild
(selectionbox);

  /* create non-proportional font for formatting purposes */
  font     = XLoadQueryFont (XtDisplay (parent), "lucidasans-18"); fontlist =
  XmFontListCreate (font, XmSTRING_DEFAULT_CHARSET);
```

```
   /* add font to list label string */
   child = XmSelectionBoxGetChild (selectionbox, XmDIALOG_LIST_LABEL);
   n     = 0;
   XtSetArg (args[n], XmNfontList, fontlist); n++;
   XtSetValues (child, args, n);

   /* add font to list items */
      child = XmSelectionBoxGetChild (selectionbox, XmDIALOG_LIST);
   XtSetValues (child, args, n);

   return (selectionbox);
}

Widget
MyCreateSelectionBoxDialog (
   Widget  parent,
   int     height,
   int     width,
   char   *title
)
{
   Widget          selectionbox,
                   child;
   Cardinal        n;
   Arg             args[10];
   XmFontList      fontlist;
   XFontStruct *font;
   XmString        xmtitle;

   n = 0;
   xmtitle = XmStringCreate (title, XmFONTLIST_DEFAULT_TAG); XtSetArg
   (args[n], XmNheight, height); n++;
   XtSetArg (args[n], XmNwidth, width); n++; XtSetArg (args[n],
   XmNdialogTitle, xmtitle); n++;
selectionbox = XmCreateSelectionDialog (parent, "sbox", args, n);
XtManageChild (selectionbox);
   XmStringFree (xmtitle);
```

```
   /* create non-proportional font for formatting purposes */
   font     = XLoadQueryFont (XtDisplay (parent), "lucidasans-18"); fontlist =
   XmFontListCreate (font, XmSTRING_DEFAULT_CHARSET);

   /* add font to list label string */
      child = XmSelectionBoxGetChild (selectionbox, XmDIALOG_LIST_LABEL);
   n     = 0;
   XtSetArg (args[n], XmNfontList, fontlist); n++;
   XtSetValues (child, args, n);

   /* add font to list items */
         child = XmSelectionBoxGetChild (selectionbox, XmDIALOG_LIST);
   XtSetValues (child, args, n);
   return (selectionbox);
}

void
MySelectionBoxSetLabel (

   Widget            box,
   unsigned char  child,
   char          *txt

)
{
   Widget            wchild;
   Arg               args[1];
   XmString          label;

   wchild = XmSelectionBoxGetChild (box, child);
             label = XmStringCreate (txt, XmFONTLIST_DEFAULT_TAG);
   XtSetArg (args[0], XmNlabelString, label); XtSetValues (wchild, args, 1);
   XmStringFree (label);
}

void
MyVaSelectionBoxSetLabel (

   Widget            box,
   int               nitems,
   unsigned char  child1,
   char          *txt1,
                     ...
)
{
   Widget            wchild;
   Arg               args[1];
   va_list           argp;
   unsigned char  child;
   char          *txt;
   XmString          label;
```

```
   if (!nitems) {
     return;
   }

   child = child1;
   txt   = txt1;
   va_start (argp, txt1);

   while (nitems--) {
                 wchild = XmSelectionBoxGetChild (box, child);
     label = XmStringCreate (txt, XmFONTLIST_DEFAULT_TAG); XtSetArg (args[0],
     XmNlabelString, label); XtSetValues (wchild, args, 1);
     XmStringFree (label);
     child = va_arg (argp, unsigned char); txt  = va_arg (argp, char *);
   }
}

void MySelectionBoxUnmanageChild (

   Widget        box,
   unsigned char child
)
{
   Widget wchild;

                 wchild = XmSelectionBoxGetChild (box, child);
   XtUnmanageChild (wchild);
}

void MyVaSelectionBoxUnmanageChild (

   Widget           box,
   int              nitems,
   unsigned char child1,
                    ...
)
{
   Widget           wchild;
   unsigned char child;
   va_list          argp;

   if (!nitems) {
     return;
   }

   child = child1;
   va_start (argp, child1);
   while (nitems--) {
     wchild = XmSelectionBoxGetChild (box, child); XtUnmanageChild (wchild);
                      child = va_arg (argp, unsigned char);
   }
}

void MySelectionBoxManageChild (

   Widget           box,
   unsigned char child

)
{
   Widget wchild;

                 wchild = XmSelectionBoxGetChild (box, child);
   XtManageChild (wchild);
}
```

```
void MyVaSelectionBoxManageChild (

   Widget           box,
   int              nitems,
   unsigned char child1,
                    ...
)
{
   Widget           wchild;
   unsigned char child;
   va_list          argp;

   if (!nitems) {
      return;
   }

   child = child1;
   va_start (argp, child1);
   while (nitems--) {
      wchild = XmSelectionBoxGetChild (box, child); XtManageChild (wchild);
                      child = va_arg (argp, unsigned char);
   }
}

void
MySelectionBoxButtonAddCallback (

   Widget         box,
   unsigned char  child,
   XtCallbackProc cb,
   XtPointer      value

)
{
   Widget wchild;
   wchild = XmSelectionBoxGetChild (box, child);

   XtAddCallback (wchild, XmNactivateCallback, cb, value);
}

char *
MySelectionBoxGetText (

   Widget box

)
{
   Widget  wchild;
       char   *txt;          /* use XtFree to dispose of this variable */

   wchild = XmSelectionBoxGetChild (box, XmDIALOG_TEXT);
   txt    = XmTextGetString (wchild);

   return (txt);
}

void
MySelectionBoxSetListItems (

   Widget   box,
   int      nitems,
   char   **items

)
{
   Widget    wchild;
```

```
  Arg       args[2];
  Cardinal  n;
  XmString *Items;

  Items = MyCreateXmStringList (nitems, items);
  n     = 0;
  XtSetArg (args[n], XmNlistItems, Items); n++;
              XtSetArg (args[n], XmNlistItemCount, nitems); n++;
  XtSetValues (box, args, n); MyDestroyXmStringList (nitems, Items);
}

Widget
MyCreateText (

  Widget parent,
  int    x,
  int    y,
  int    rows,
  int    cols

)
{
  Widget   text;
  Arg       args[10];
  Cardinal n;

  n = 0;
  MySetXYArgs (parent, args, &n, x, y); XtSetArg (args[n], XmNrows, rows);
  n++; XtSetArg (args[n], XmNcolumns, cols); n++; if (rows > 1) {
    XtSetArg (args[n], XmNeditMode, XmMULTI_LINE_EDIT); n++; }
  else {
    XtSetArg (args[n], XmNeditMode, XmSINGLE_LINE_EDIT); n++; }
  text = XmCreateText (parent, "text", args, n); XtManageChild (text);

  return (text);
}

Widget
MyCreateTextField (

  Widget  parent,
  int      x,
  int      y,
  int      cols

)
{
  Widget   textfield;
  Arg       args[10];
  Cardinal n;

  n = 0;
  MySetXYArgs (parent, args, &n, x, y);
                  XtSetArg (args[n], XmNcolumns, cols); n++;
  XtSetArg (args[n], XmNheight, 20); n++; XtSetArg (args[n],
  XmNshadowThickness, 0); n++; XtSetArg (args[n], XmNmarginHeight, 0); n++;
textfield = XmCreateTextField (parent, "textfield", args, n); XtManageChild
(textfield);

  return (textfield);
}

Widget *
MyCreateLabeledTextField (

  Widget  parent,
  int      x,
```

```
   int       y,
   char   *label,
   int       offset,
   int       cols,
   char   *val

)
{
   Widget *w;

   w = (Widget *) malloc (sizeof (Widget) * 2);

   w[0]      = MyCreateLabel (parent, x, y, label);
   w[1]      = MyCreateTextField (parent, x + offset, y, cols);
   if (val) {
      XmTextFieldSetString (w[1], val);
   }
   return (w);
}

Widget *
MyWidgetGetChildren (
   Widget parent
)
{
   Widget *children;
   Arg       args[1];

   XtSetArg (args[0], XmNchildren, &children);
   XtGetValues (parent, args, 1);
   return (children);
}
```

```
/******************************/
/*  File:   motif.h           */ /******************************/


#include <stdio.h>
#include <Xm/BulletinB.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>
#include <Xm/Separator.h>
#include <Xm/Label.h>
#include <Xm/SelectioB.h>
#include <Xm/Text.h>
#include <Xm/TextF.h>
#include "myutil.h"

Widget
MyCreateBulletinBoardDialog (

   Widget  parent,
   int       height,
   int       width,
   char   *title

);

Widget
MyCreateBulletinBoard (

   Widget  parent,
   int       x,
   int       y,
   int       height,
   int       width

);


#define MyCHAR_HEIGHT        15 #define MyCHAR_WIDTH         11 #define
MyBUTTON_HEIGHT      35 #define MyBUTTON_WIDTH      100

Widget
MyCreatePushButton (

   Widget          parent,
   char             *title,
   int               x,
   int               y,
   XtCallbackProc  cb,
   XtPointer       cb_val

);

Widget
MyCreateSeparator (

   Widget        parent,
   int             x,
   int             y,
         unsigned char orientation,     /* XmHORIZONTAL, XmVERTICAL */
   int             length

);

Widget
MyCreateLabel (
```

```
   Widget  parent,
   int       x,
   int       y,
   char   *txt

);

Widget *
MyCreateLabeledBox (

   Widget  parent,
   int       x,
   int       y,
   int       height,
   int       width,
   char   *txt

);

typedef enum _MyBoxType {

   MyRADIO_BOX,
   MyCHECK_BOX

} MyBoxType;

Widget
MyCreateBox (

   Widget    parent,
   MyBoxType type,
   int         x,
   int         y,
   int         nitems,
   char      **items,
   int         numcols

);

int
MyGetSelectedRadioBoxItem (

   Widget    radiobox

);

int *
MyGetSelectedCheckBoxItems (

   Widget checkbox

);

Widget
MyCreateSelectionBox (

   Widget parent,
   int    x,
   int    y,
   int    height,
   int    width

);

Widget
```

```
MyCreateSelectionBoxDialog (

   Widget  parent,
   int     height,
   int     width,
   char   *title

);

void
MySelectionBoxSetLabel (

   Widget            box,
   unsigned char  child,
   char             *txt

);

void
MyVaSelectionBoxSetLabel (

   Widget            box,
   int               nitems,
   unsigned char  child1,
   char             *txt1,
                     ...
);
void
MySelectionBoxUnmanageChild (

   Widget           box,
   unsigned char child

);

void
MyVaSelectionBoxUnmanageChild (

   Widget           box,
   int              nitems,
   unsigned char child1,
                    ...
);
void
MySelectionBoxManageChild (

   Widget           box,
   unsigned char child

);

void
MyVaSelectionBoxManageChild (

   Widget           box,
   int              nitems,
   unsigned char child1,
                    ...
);

void
MySelectionBoxButtonAddCallback (

   Widget          box,
   unsigned char   child, XtCallbackProc  cb,
   XtPointer       value
```

```
);

char *
MySelectionBoxGetText (

  Widget box

);

void
MySelectionBoxSetListItems (

  Widget   box,
  int       nitems,
  char    **items

);

Widget
MyCreateText (

  Widget parent,
  int     x,
  int     y,
  int     rows,
  int     cols

);

Widget
MyCreateTextField (

  Widget parent,
  int     x,
  int     y,
  int     cols

);

Widget *
MyCreateLabeledTextField (

  Widget  parent,
  int      x,
  int      y,
  char    *label,
  int      offset,
  int      cols,
  char    *val

);
```

```
/*******************************/
/**                           **/
/**  File:  util.c            **/
/**                           **/
/*******************************/

#include "myutil.h"

int
MyGetNextYPosition (

  Widget w
)
{
  Widget    *children;
  Cardinal     numChildren,
               n;
  Dimension  height;
  Position     y;
  Arg          args [10];
  int          nexty;

  /* get children of widget */
  n = 0;
  XtSetArg (args[n], XmNchildren, &children); n++; XtSetArg (args[n],
  XmNnumChildren, &numChildren); n++; XtGetValues (w, args, n);

  /* if no children exist */
  if (!numChildren) {
    return (0);
  }

  /* get values of last child under parent */
  n = 0;
  XtSetArg (args[n], XmNheight, &height); n++; XtSetArg (args[n], XmNy, &y);
  n++;
  XtGetValues (children[numChildren-1], args, n);

  /* calculate next y position */
  nexty = (int) ((int)height + (int)y);
  return (nexty);
}

XmString *
MyCreateXmStringList (

  int    nitems,
  char **items

)
{
  XmString *newitems;
  int       i;

  if (!nitems) {
    return (NULL);
  }
        newitems = (XmString *) malloc (sizeof (XmString) * nitems);

  for (i = 0; i < nitems; i++) {
    newitems[i] = XmStringCreate (items[i], XmFONTLIST_DEFAULT_TAG); }
  return (newitems);
}

XmString *
MyVaCreateXmStringList (
```

```
   int    nitems,
   char *item1,
          ...
)
{
   XmString *newitems;
   int        i;
   va_list   argp;
   char     *txt;

   if (!nitems) {
     return (NULL);
   }
         newitems = (XmString *) malloc (sizeof (XmString) * nitems);

   va_start (argp, item1);

   txt = item1;
   i   = 0;
   while (nitems--) {
     newitems[i++] = XmStringCreate (txt, XmFONTLIST_DEFAULT_TAG); txt =
     va_arg (argp, char *);
   }
   return (newitems);
}

char **
MyVaCreateCharList (

   int    nitems,
   char *item1,
          ...
)
{
   char    **newitems;
   int        i;
   va_list   argp;
   char     *txt;

   if (!nitems) {
     return (NULL);
   }
            newitems = (char **) malloc (sizeof (char *) * nitems);

   va_start (argp, item1);

   txt = item1;
   i   = 0;
   while (nitems--) {
     newitems[i++] = (char *)strdup (txt);
     txt = va_arg (argp, char *);
   }
   return (newitems);
}

void
MyDestroyXmStringList (

   int        nitems,
   XmString *items

)
{
   int i;

   for (i = 0; i < nitems; i++) { XmStringFree (items[i]);
   }
```

```
   free (items);
   items = NULL;
}

void
MyDestroyCharList (

   int       nitems,
   char     **items

)
{
   int i;

   for (i = 0; i < nitems; i++) {
      free (items[i]);
   }
   free (items);
   items = NULL;
}

void
MySetXYArgs (

   Widget    parent,
   Arg      *args,
   Cardinal *n,
   int       x,
   int       y

)
{
   if (x) {
      XtSetArg (args[*n], XmNx, x); (*n)++; }
   if (y) {
      XtSetArg (args[*n], XmNy, y); (*n)++; }
   else {
                        y = MyGetNextYPosition (parent);
      XtSetArg (args[*n], XmNy, y + MyWIDGET_OFFSET); (*n)++; }
}

void
MyBulletinBoardSetDefaultButton (

   Widget  parent,
   Widget  button

)
{
   Arg args[1];

   XtSetArg (args[0], XmNdefaultButton, button); XtSetValues (parent, args,
   1);
}

void
MyWidgetGetPosition (

   Widget      w,
   Position *x,
   Position *y

)
{
   Arg args[2];
```

```
   XtSetArg (args[0], XmNx, x); XtSetArg (args[1], XmNy, y); XtGetValues (w,
   args, 2);
}

void
MyWidgetGetDimension (

   Widget       w,
   Dimension *height,
   Dimension *width

)
{
   Arg args[2];

   XtSetArg (args[0], XmNheight, height); XtSetArg (args[1], XmNwidth, width);
   XtGetValues (w, args, 2);
}

void
MyLabeledBoxGetCoords (

   Widget  top,
   Widget  right,
   Widget  bottom,
   Widget  left,
   int    *x,
   int    *y,
   int    *height,
   int    *width

)
{
   Position  bx,
               by;
   Dimension bheight,
               bwidth;

   /* get x value */
                    MyWidgetGetPosition (left, &bx, &by);
                   *x = (int) ((int) bx - MyWIDGET_OFFSET);

   /* get y value */
                     MyWidgetGetPosition (top, &bx, &by);
                   *y = (int) ((int) by - MyWIDGET_OFFSET);

   /* get width value */ MyWidgetGetPosition (right, &bx, &by);
MyWidgetGetDimension (right, &bheight, &bwidth);
*width = (int) ((int) bx + (int) bwidth + MyWIDGET_OFFSET);

   /* get height value */
                    MyWidgetGetPosition (bottom, &bx, &by);
   MyWidgetGetDimension (bottom, &bheight, &bwidth);
       *height = (int) ((int) by + (int) bheight + MyWIDGET_OFFSET - *y);
}

int
MyWidgetGetNextXPosition (
   Widget sibling
)
{
   int         nextx;
   Position  x,
               y;
   Dimension height,
               width;
```

```
   MyWidgetGetPosition (sibling, &x, &y);
                MyWidgetGetDimension (sibling, &height, &width);
             nextx = (int) ((int)x + (int)width + MyWIDGET_OFFSET);
   return (nextx);
}

int
MyBulletinBoardGetWidthFromX (
   Widget bboard,
   int    x
)
{
   Dimension width;
   int         w;
   Arg         args[1];

   XtSetArg (args[0], XmNwidth, &width);
   XtGetValues (bboard, args, 1);
   w = (int) ((int)width - x);
   return (w);
}
```

```
/*********************************/ /**                              **/
/**  File:  util.h              **/ /**                              **/
/*********************************/

#include <stdio.h>
#include <stdarg.h>
#include <Xm/Xm.h>

int
MyGetNextYPosition (

  Widget w
);
XmString *
MyCreateXmStringList (

  int     nitems,
  char **items

);

XmString *
MyVaCreateXmStringList (

  int    nitems,
  char *item1,
         ...
);
char **
MyVaCreateCharList (

  int    nitems,
  char *item1,
         ...
);
void
MyDestroyXmStringList (

  int        nitems,
  XmString *items

);

void
MyDestroyCharList (

  int        nitems,
  char    **items

);

#define MyWIDGET_OFFSET     20
#define MyBUTTON_OFFSET     10

void
MySetXYArgs (

  Widget    parent,
  Arg        *args,
  Cardinal *n,
  int         x,
  int         y

);

void
```

```
MyBulletinBoardSetDefaultButton (

   Widget  parent,
   Widget  button

);

void
MyWidgetGetPosition (

   Widget    w,
   Position *x,
   Position *y

);

void
MyWidgetGetDimension (

   Widget     w,
   Dimension *height,
   Dimension *width

);

void
MyLabeledBoxGetCoords (

   Widget  top,
   Widget  right,
   Widget  bottom,
   Widget  left,
   int    *x,
   int    *y,
   int    *height,
   int    *width

);

int
MyWidgetGetNextXPosition (
   Widget sibling
);

int
MyBulletinBoardGetWidthFromX (
   Widget bboard,
   int    x
);
```

```
/*  File:  private.h  */
void
CreateUserDefinedColorsWindow ( Widget parent, MapColor *color );

void
FeaturesButtonCB ( Widget w, tJmdButtonTag button_id, XtPointer call_data );

void
GetRGBColorByButtonNumber ( MapColor *color, int button_num );

void
UserDefinedColorsButtonCB ( Widget w, tJmdButtonTag button_id,
                            XtPointer call_data );

void
DefaultMapButtonCB ( Widget w, tJmdButtonTag button_id, XtPointer call_data );

void
DefaultMapChangeCB ( Widget w, tJmdConnectionId *id, XtPointer call_data );

void
DefaultMapDraw ( Widget parent, tJmdConnectionId *id );

void
DefaultMapGetAttributes ( ProjectionType *type, FLOAT *width,
                                MapPoint *center );

Widget MyCreateRadioBox (parent, txt, nitems, colnum) Widget      parent;
  char   ** txt;
  int         nitems,
              colnum;
{
  Widget       radio_box;
  int          i,  n, y;
  Arg          args [10];
  XmString * lbl;

  lbl = (XmString *) malloc (sizeof (XmString) * nitems);
  for (i = 0; i < nitems; i++) {
    lbl[i] = XmStringCreate (txt[i], XmFONTLIST_DEFAULT_TAG); n++; }

  n = 0;
  XtSetArg (args [n], XmNbuttonCount, nitems); n++;
  XtSetArg (args [n], XmNbuttons, lbl); n++; XtSetArg (args [n],
  XmNnumColumns, colnum); n++; XtSetArg (args [n], XmNpacking,
  XmPACK_COLUMN); n++; XtSetArg (args [n], XmNbuttonSet, 0); n++;
radio_box = XmCreateSimpleRadioBox (parent, "rbox", args, n); XtManageChild
(radio_box);

  for (i = 0; i < nitems; i++)
    XmStringFree (lbl[i]);

  return (radio_box);
}

void CreateDistanceConversionWindow (parent)
  Widget parent;
{
             Widget   bboard, radio_box, btn_ok, btn_cancel, sep1;
  int      n, i, x, y;
  Arg      args [10];
  XmString lbl;

  if (wPROJECTION) return;

   /* create bulletin board */
```

```
   lbl    = XmStringCreate ("Set Default Map", XmFONTLIST_DEFAULT_TAG);
n      = 0;
XtSetArg (args [n], XmNautoUnmanage, FALSE); n++;
XtSetArg (args [n], XmNdialogTitle, lbl); n++; XtSetArg (args [n],
XmNresizePolicy, XmRESIZE_ANY); n++; XtSetArg (args [n], XmNwidth, 900);
n++;
XtSetArg (args [n], XmNheight, 600); n++;
bboard = XmCreateBulletinBoardDialog (parent, "bboard", args, n);
XtManageChild (bboard);
XmStringFree (lbl);

radio_box =  MyCreateRadioBox (bboard, PROJECTION_TXT, PROJECTION_CNT, 2);

/* create separator 1*/
n = 0;
XtSetArg (args [n], XmNorientation, XmHORIZONTAL); n++;
XtSetArg (args [n], XmNwidth, 600); n++;
XtSetArg (args [n], XmNy, 400); n++;
            sep1 = XmCreateSeparator (bboard, "sep", args, n);
XtManageChild (sep1);
```

```
   /* create "Ok" button */
   n = 0;
   XtSetArg (args [n], XmNy, 410); n++; XtSetArg (args [n], XmNx, 10); n++;
   XtSetArg (args [n], XmNheight, 40); n++; XtSetArg (args [n], XmNwidth,
   100); n++;
btn_ok = XmCreatePushButton (bboard, "Ok", args, n); XtManageChild (btn_ok);

         /* set ok btn as default button to allow for keyboard input */
   n = 0;
   XtSetArg (args [n], XmNdefaultButton, btn_ok); n++; XtSetValues (bboard,
   args, n);

   /* create "Cancel" button */
   n = 0;
   XtSetArg (args [n], XmNy, 410); n++;
                     XtSetArg (args [n], XmNx, 500); n++;
   XtSetArg (args [n], XmNheight, 40); n++; XtSetArg (args [n], XmNwidth,
   100); n++;
btn_cancel = XmCreatePushButton (bboard, "Cancel", args, n);
   XtManageChild (btn_cancel);

   /* add callbacks */
   XtAddCallback (btn_ok, XmNactivateCallback, ProjectionBtnOkCB, radio_box);
   XtAddCallback (btn_cancel, XmNactivateCallback, ProjectionBtnCancelCB,
   NULL);

   wPROJECTION = bboard;
}

void ProjectionBtnOkCB (w, radio_box, call_data)
   Widget    w, radio_box;
   XtPointer call_data;
{
   Widget *children;
   int     n, i;
   Arg     args [10];

   n = 0;
                XtSetArg (args [n], XmNchildren, &children); n++;
   XtGetValues (radio_box, args, n);

   for (i = 0; i < PROJECTION_CNT; i++) {
                 if (XmToggleButtonGadgetGetState (children [i]))
        break;
   }
   dSetProjection (i);
}

void ProjectionBtnCancelCB (w, client_data, call_data) Widget w;
   XtPointer client_data, call_data;
{
   ProjectionDelete ();
}

void ProjectionDelete () {
   XtDestroyWidget (wPROJECTION);
   wPROJECTION = NULL;
}
```

```
/*!!==========================================================================
==
     Disclaimer:             This software was developed for the Department of
                             Defense (DoD). All US  Government restrictions on
                             software source code distribution apply.

     Attribution:       This software is considered part of the GCCS Joint
                             Mapping Toolkit (JMTK).

     Organization:      Computer Sciences Corporation
                             6565 Arlington Boulevard
                             Falls Church, VA  22042 703.538.7276    Date:
                             19960000
Copyright (1996)        All Rights Reserved
     Source File:       template.c
     Domain:                 PUBLIC
     Scope:                  JMTK Demo
     Version:                1.1
     Last Delta:             04/08/96 18:52:33
     Checked Out:       04/08/96 18:58:56
     Software Category:

     Revision History:

     Version        Person                  Date            Reason
     =========      =====================   ========
========================
     sw 001         Brad Bunton             15 MAR 96       Initial Draft


==========================================================================
==

     Purpose:       JMD_TemplateCreate allows the creation of templates with
                    specific attributes which will apply on the next object
                    drawn on the map.

     Synopsis:      #include "JMD.h"

                    ObjectId JMD_TemplateCreate(
                         tJmdConnectionId *pId, char *pLinecolor,
                         MapLineType linetype, MapLineStyle linestyle,
                         MapFillType filltype, char *pFont )

     Description:

     Return:        void

     Errors:        If the template cannot be created, an error message will
be
                    displayed in X-window explaining why.

     Notes:

==========================================================================!!
*/
```

```
#include "JMD.h"

ObjectId JMD_TemplateCreate( tJmdConnectionId *pId, char *pLinecolor,
                                  MapLineType linetype, MapLineStyle
                                  linestyle, MapFillType filltype, char
                                  *pFont )
{
   ObjectAttributes atts;
   MapValueMask     atts_mask;
   ObjectId         template;

   atts_mask = OBColor | OBLineType | OBLineStyle | OBFillType | OBFontName;
   strcpy( atts.color, pLinecolor );
   atts.line_type  = linetype;
   atts.line_style = linestyle;
   atts.fill_type  = filltype;
   strcpy( atts.font_name, "9x15" );

  template = MCreateTemplate( pId -> channel, pId -> wid, &atts, atts_mask );
   MFlush( pId -> channel );
   if (template == InvalidObjectId)
   {
        fprintf( stdout, "Unable to create template\n" );
        return( InvalidObjectId );
   }
   return( template );
}
```

```
/*   File:  xcolors.c   */
#include "JMD.h"
#include "motif.h"

/* private function prototypes */
void MapColorsButtonCancelCB( Widget w, tJmdConnectionId *id,
                                        XtPointer call_data );
void MapColorsButtonOkCB( Widget w, tJmdConnectionId *id, XtPointer call_data
);
void MapColorsChangeCB( Widget w, tJmdConnectionId *id, XtPointer call_data );
void MapColorsToggleButtonCB( Widget w, tJmdConnectionId *id,
                            XtPointer call_data );

/* global variables */
Widget gMAP_COLORS_WINDOW = NULL;

void JMD_MapColorsWindowCreate( Widget parent, tJmdConnectionId *id )
{
  Widget      bboard,
             *children,
              scale_red,
              scale_green,
              scale_blue,
              box_side,
              sep,
              btn_cancel,
              btn_ok,
              label;
  Cardinal  n;
  Arg         args [15];
  Position  x, y;
  int         nitems,
              width;
  char    **items;

  if( gMAP_COLORS_WINDOW == NULL ) {
  }
  else {
    XtManageChild( gMAP_COLORS_WINDOW );
    return;
  }

  /* create bulletin board */
  bboard      = MyCreateBulletinBoardDialog( parent, "bboard", 500, 500,
                                                        "Map Colors", XmRESIZE_ANY
);
  label = MyCreateLabel( bboard, 10, 10, "100%" );
  label = MyCreateLabel( bboard, 10, 255, "0%" );

  scale_red = MyCreateScale( bboard, "scale_red", 55, 10, 255, 30, 255, NULL
                                    );
  XtSetArg( args[0], XmNuserData, JmdRedScale );
  XtSetArg( args[1], XmNshowValue, False );
  XtSetArg( args[2], XmNbackground, GetPixel( parent, "red3" ) );
  XtSetValues( scale_red, args, 3 );

  scale_green = MyCreateScale( bboard, "scale_green", 140, 10, 255, 30, 255,
                                        NULL );
  XtSetArg( args[0], XmNuserData, JmdGreenScale );
  XtSetArg( args[1], XmNshowValue, False );
   XtSetArg( args[2], XmNbackground, GetPixel( parent, "forest green" ) );
  XtSetValues( scale_green, args, 3 );

  scale_blue  = MyCreateScale( bboard, "scale_blue", 215, 10, 255, 30, 255,
                                        NULL );
  XtSetArg( args[0], XmNuserData, JmdBlueScale );
  XtSetArg( args[1], XmNshowValue, False );
    XtSetArg( args[2], XmNbackground, GetPixel( parent, "royal blue" ) );
```

```
   XtSetValues( scale_blue, args, 3 );

   nitems = 2;
   items = MyVaCreateCharList( nitems, "Background", "Foreground" ); box_side
   = MyCreateBox( bboard, "radio_side", MyRADIO_BOX, 10, NULL,
                                  nitems, items, 2 );
   MyDestroyCharList( nitems, items );

   width = MyBulletinBoardGetWidthFromX( bboard, 0 );
    sep = MyCreateSeparator( bboard, "sep", 10, NULL, XmHORIZONTAL, width );

   btn_ok = MyCreatePushButton( bboard, "Ok", 10, NULL,
                                       ( XtCallbackProc ) MapColorsButtonOkCB, id
   ); MyBulletinBoardSetDefaultButton( bboard, btn_ok );
   MyWidgetGetPosition( btn_ok, &x, &y );
   btn_cancel = MyCreatePushButton( bboard, "Cancel", width-110, y,
                                            ( XtCallbackProc )
                                             MapColorsButtonCancelCB, id );


   XtSetArg( args[0], XmNchildren, &children );
   XtGetValues( box_side, args, 1 );
   XtSetArg( args[0], XmNuserData, JmdBackgroundToggle );
   XtSetValues( children[JmdBackgroundToggle], args, 1 ); XtSetArg( args[0],
   XmNuserData, JmdForegroundToggle ); XtSetValues(
   children[JmdForegroundToggle], args, 1 );

   /* add callbacks */
   XtAddCallback( scale_red, XmNvalueChangedCallback,
                    ( XtCallbackProc ) MapColorsChangeCB, id ); XtAddCallback(
   scale_red, XmNdragCallback,
                    ( XtCallbackProc ) MapColorsChangeCB, id ); XtAddCallback(
   scale_green, XmNvalueChangedCallback,
                    ( XtCallbackProc ) MapColorsChangeCB, id ); XtAddCallback(
   scale_green, XmNdragCallback,
                    ( XtCallbackProc ) MapColorsChangeCB, id ); XtAddCallback(
   scale_blue, XmNvalueChangedCallback,
                    ( XtCallbackProc ) MapColorsChangeCB, id ); XtAddCallback(
   scale_blue, XmNdragCallback,
                    ( XtCallbackProc ) MapColorsChangeCB, id ); XtAddCallback(
   children[0], XmNarmCallback,
                    ( XtCallbackProc ) MapColorsToggleButtonCB, id );
   XtAddCallback( children[1], XmNarmCallback,
                    ( XtCallbackProc ) MapColorsToggleButtonCB, id );
   XtAddEventHandler( bboard, StructureNotifyMask, False,
                         ( XtEventHandler )MyBulletinBoardDoubleClickCB, NULL
);

   gMAP_COLORS_WINDOW = bboard;
}

void
MapColorsButtonCancelCB( Widget w, tJmdConnectionId *pId, XtPointer call_data
) {
   MapColor *foreground,
              *background;

   /* call vimala's code */

   MSetMapColorsByRGB( pId -> channel, pId -> wid,
                          background -> red, background -> green,
                          background -> blue, foreground -> red, foreground -
                          > green, foreground -> blue );
   MFlush( pId -> channel );
}

void MapColorsButtonOkCB( Widget w, tJmdConnectionId *id, XtPointer call_data
```

```
) {
}

void MapColorsChangeCB( Widget w, tJmdConnectionId *id, XtPointer call_data )
{
}

void MapColorsToggleButtonCB( Widget w, tJmdConnectionId *id,
                              XtPointer call_data )
{
}
```

```
/*   File:  xcountry.c   */
#include "JMD.h"
#include "motif.h"

Widget                gCOUNTRY_WINDOW           = NULL,
                      gCOUNTRY_COLOR_WINDOW = NULL;
void
CountryColorButtonOkCB (Widget w, MapColor *mc, XtPointer call_data);

void
CountryColorButtonCancelCB (Widget w, XtPointer client_data,
                            XtPointer call_data);
void
CountryButtonColorCB ( Widget btn_color, MapColor *mc, XtPointer call_data) {
  Widget     parent,
             bboard,
             lbl,
             radiobox,
             sep,
             btn_ok,
             btn_cancel,
            *children;
  char   **items;
  int        nitems,
             button_num;
  Arg        args[2];
  Dimension height, width;
  MapColor *prev_mc;
  Position  x, y;

  button_num = GetColorButtonNumberMask (mc);
  if (gCOUNTRY_COLOR_WINDOW) {
    XtSetArg (args[0], XmNchildren, &children); XtGetValues
    (gCOUNTRY_COLOR_WINDOW, args, 1); MyBoxSetButton (children[0],
    button_num); XtManageChild (gCOUNTRY_COLOR_WINDOW);
    return;
  }
  else {
    parent = XtParent(btn_color);
    prev_mc = (MapColor *) malloc (sizeof (MapColor)); }
      bboard = MyCreateBulletinBoardDialog (parent, "bboard", 200, 200,
                                                  "Country Colors",
  XmRESIZE_ANY); nitems = 6;
  items  = MyVaCreateCharList (nitems,
                                   "Red", "Green", "Blue",
                                   "White", "Black", "Yellow");
  radiobox = MyCreateBox (bboard, "radio_color",MyRADIO_BOX, 10, NULL,
                              nitems, items, 2);
  MyBoxSetButton (radiobox, button_num); MyWidgetGetDimension (bboard,
  &height, &width);
sep = MyCreateSeparator (bboard, "sep1", 10, NULL, XmHORIZONTAL, (int)width);
btn_ok = MyCreatePushButton (bboard, "Ok", 10, NULL,
                                   (XtCallbackProc) CountryColorButtonOkCB,
  mc); MyBulletinBoardSetDefaultButton (bboard, btn_ok);
  XtSetArg (args[0], XmNuserData, mc);
  XtSetValues (btn_ok, args, 1);
  MyWidgetGetPosition (btn_ok, &x, &y);
  btn_cancel = MyCreatePushButton (bboard, "Cancel", x+110, y,
                                         (XtCallbackProc)
                                         CountryColorButtonCancelCB, NULL);
  XtAddEventHandler (bboard, StructureNotifyMask, False,
                         (XtEventHandler) MyBulletinBoardDoubleClickCB,
  NULL); gCOUNTRY_COLOR_WINDOW = bboard;
}

void
CreateCountryWindow ( Widget parent, DConnectionId  *id )
{
```

```
   Widget     bboard,
              selectionbox,
              btn_on, btn_off, btn_color,
              sep,
              btn_ok, btn_cancel,
              child;
   Arg        args[10];
   char   **items;
   int        nitems,
              i;
   Position x, y;
   Dimension h, w;
   MapCountryAttributes *mcatts;
   CountryAttributes   **catts;

   if (gCOUNTRY_WINDOW) {
      XtManageChild (gCOUNTRY_WINDOW);
      return;
   }
   else {
      items  = CountryGetList (id->channel, &nitems);
      mcatts = (MapCountryAttributes *) malloc (sizeof (MapCountryAttributes));
      catts  = (CountryAttributes **) malloc (sizeof(CountryAttributes
      *)*nitems); for (i = 0; i < nitems; i++) {
     catts[i] = (CountryAttributes *) malloc (sizeof (CountryAttributes));
         catts[i] -> status = False;
         catts[i] -> name   = (char *) strdup (items[i]);
         catts[i] -> color  = (MapColor *) malloc (sizeof (MapColor));
     catts[i] -> color -> red   = 255;          /* default color = white */
         catts[i] -> color -> green = 255;
         catts[i] -> color -> blue  = 255;
      }
      mcatts -> id        = id;
      mcatts -> catts  = catts;
      mcatts -> ncatts = nitems;
   }
   bboard = MyCreateBulletinBoardDialog (parent, 200, 550, "Countries",
                                                      XmRESIZE_ANY);
   XtSetArg (args[0], XmNuserData, mcatts);
   XtSetValues (bboard, args, 1);
   selectionbox = MyCreateSelectionBox (bboard, 10, NULL, 200, 330);
   MyVaSelectionBoxUnmanageChild (selectionbox, 8,
                                        XmDIALOG_APPLY_BUTTON,
                                        XmDIALOG_CANCEL_BUTTON,
                                        XmDIALOG_DEFAULT_BUTTON,
                                        XmDIALOG_HELP_BUTTON,
                                        XmDIALOG_OK_BUTTON,
                                        XmDIALOG_SELECTION_LABEL,
                                        XmDIALOG_SEPARATOR, XmDIALOG_TEXT);
   child = XmSelectionBoxGetChild (selectionbox, XmDIALOG_TEXT);
   XtAddCallback (child, XmNvalueChangedCallback,
                     (XtCallbackProc) CountryChangeCB, mcatts);
   MySelectionBoxSetListItems (selectionbox, nitems, items);
   MyDestroyCharList (nitems, items);
   MySelectionBoxSetLabel (selectionbox, XmDIALOG_LIST_LABEL,
                                 "Country:
Status:");
   btn_on = MyCreatePushButton (bboard, "On", 130, NULL,
                                       (XtCallbackProc) CountryButtonOnCB, id);
   MyBulletinBoardSetDefaultButton (bboard, btn_on);
   MyWidgetGetPosition (btn_on, &x, &y);
   btn_off = MyCreatePushButton (bboard, "Off", 250, y, (XtCallbackProc)
                                      CountryButtonOffCB, id);
   btn_color = MyCreatePushButton (bboard, "Color ...", 370, y,
                                       (XtCallbackProc) CountryButtonColorCB,
                                       id);
   MyWidgetGetDimension (bboard, &h, &w);
   sep = MyCreateSeparator (bboard, 10, NULL, XmHORIZONTAL, (int)w); btn_ok =
   MyCreatePushButton (bboard, "Ok", 10, NULL,
```

```
                                        (XtCallbackProc) CountryButtonOkCB, id);
   MyWidgetGetPosition (btn_ok, &x, &y);
   btn_cancel = MyCreatePushButton (bboard, "Cancel", (int)w-100, y,
                                        (XtCallbackProc) CountryButtonCancelCB,
                                        id);

   XtAddEventHandler (bboard, StructureNotifyMask, False,
                           (XtEventHandler) MyBulletinBoardDoubleClickCB,
   NULL); gCOUNTRY_WINDOW = bboard;
}

void
CountryButtonOnCB (Widget w, DConnectionId *id, XtPointer call_data)
{
   Arg         args[2];
   Widget      parent,
              *children,
               child;
   int         index;
   MapCountryAttributes *mcatts;

   parent = XtParent(w);
   XtSetArg (args[0], XmNchildren, &children);
   XtSetArg (args[1], XmNuserData, &mcatts);
   XtGetValues (parent, args, 2);

         child = XmSelectionBoxGetChild (children[0], XmDIALOG_TEXT);
   XtSetArg (args[0], XmNuserData, &index);
   XtGetValues (child, args, 1);

   JMD_DrawCountry (id->channel, id->wid, index, mcatts->catts[index]->color-
                           >red, mcatts->catts[index]->color->green, mcatts-
                           >catts[index]->color->blue);

}

void
CountryButtonOffCB (Widget w, DConnectionId *id, XtPointer call_data) {
   Widget      parent;


   parent = XtParent (w);
}
void
CountryButtonOkCB (Widget w, DConnectionId *id, XtPointer call_data) {
   Widget      parent;

   parent = XtParent (w);
   XtUnmanageChild (parent);
}
void
CountryButtonCancelCB (Widget w, DConnectionId *id, XtPointer call_data) {
   Widget      parent;

   parent = XtParent (w);
   XtUnmanageChild (parent);
}

void
CountryChangeCB ( Widget w, MapCountryAttributes *mcatts, XtPointer call_data
) {
   int   i;
   char *txt;
   Arg   args[2];
```

```
   txt = XmTextGetString (w);
   i   = 0;
   while (i < mcatts -> ncatts) {
      if (strcmp (txt, mcatts->catts[i]->name) == 0)
         break;
      i++;
   }
   fprintf (stdout, "index = %d\n", i);
   XtSetArg (args[0], XmNuserData, i);
   XtSetValues (w, args, 1);
   XtFree (txt);
}

void
CountryColorButtonOkCB (Widget w, MapColor *mc, XtPointer call_data) {
   Widget      parent,
              *children;
   Arg         args[2];
   int         button_num;

   parent = XtParent (w);
   XtSetArg (args[0], XmNchildren, &children);
   XtGetValues (parent, args, 1);

   button_num = MyGetSelectedRadioBoxItem (children[0]);
   SetColorByButtonNumberMask (mc, button_num);
   JMD_DrawCountry (
   XtUnmanageChild (parent);
}

void
CountryColorButtonCancelCB (Widget w, XtPointer client_data, XtPointer
                                    call_data)
{
   Widget      parent;

   parent = XtParent (w);
   XtUnmanageChild (parent);
}


char **
CountryGetList ( Channel channel, int *nitems )
{
   int                         nfeatures;
   FeatureListAttributes *list;
   int                         cnt;
   char                        tmp[80],
                             **items;

    list  = MListFeatures (channel, CountryFeature, AnyFeature, &nfeatures);
   cnt   = 0;
   items = (char **) malloc (sizeof (char *) * nfeatures);
   while (cnt < nfeatures) {
      sprintf (tmp, "%-40s Off", list[cnt].label.name);
      items[cnt] = (char *) strdup (tmp);
      cnt++;
   }
   *nitems = cnt;
   MFree ((char **) &list);
   return (items);
}
```

```
/*   File:  xfeatures.c   */
#include "JMD.h"
#include "motif.h"
#include "private.h"
#include "_features.c"

extern Widget gFEATURES_WINDOW;

void
PrintFeatures () {

   MapReference *subfeats, *feats;
   int size,
        i;
   char *tmp;

   subfeats = MuListSubFeatures ( BottomContourFeature, &size );
   for (i = 0; i < size; i++) {
          tmp = MuSubFeatureToString ( subfeats[i].typevalue, AnyFeature,
                                            subfeats, size);
          printf ("%s\n", tmp);
          free (tmp);
   }

   feats = MuListFeatures ( &size );
   for (i = 0; i < size; i++) {
          tmp = MuFeatureToString ( feats[i].typevalue, feats, size);
          printf ("%s\n", tmp);
          free (tmp);
   }

   subfeats = MuListSubTypes ( AnyFeature, &size );
   for (i = 0; i < size; i++) {
          tmp = MuSubTypeToString ( subfeats[i].typevalue,
                                          BottomContourFeature, subfeats,
                                          size);
          printf ("%s\n", tmp);
          free (tmp);
   }

   feats = MuListTypes ( &size );
   for (i = 0; i < size; i++) {
          tmp = MuTypeToString ( feats[i].typevalue, feats, size); printf
          ("%s\n", tmp);
          free (tmp);
   }
}

void JMD_FeatureEdit ( Widget parent, tJmdConnectionId *id,
                            tJmdMapAttributesType type )
{
   Arg                       args[5];
   char                      txt[80];
   XmString                  dialogTitle;
   Widget                   bboard,
                             child,
                            *children;
   int                       i,
                             numFeatures,
                             item;
   Cardinal                  numChildren;

   bboard      = gFEATURES_WINDOW;
   numFeatures = 6;                        /* excluding country features */
/*
   PrintFeatures ();
*/
```

```
   strcpy (txt, "Edit Features");

   if (bboard == NULL) {
          CreateFeaturesWindow (parent);
          bboard = gFEATURES_WINDOW;
   }
   else {
   }

   dialogTitle = XmStringCreate (txt, XmFONTLIST_DEFAULT_TAG); XtSetArg
   (args[0], XmNdialogTitle, dialogTitle); XtSetValues (bboard, args, 1);
   XmStringFree (dialogTitle);

   /* update widgets to values */

   /* save values in case of a cancel */
   XtManageChild (bboard);
}
```

```
/*   File:  xintensity.c   */
#include "JMD.h"
#include "motif.h"

/* private function prototypes */

void IntensityButtonCB( Widget w, tJmdButtonTag button_id,
                                XtPointer call_data );
void IntensityChangeCB( Widget w, tJmdConnectionId *pId, XtPointer call_data
);


/* global variables */
Widget gMAP_INTENSITY_WINDOW = NULL;

void JMD_MapIntensityWindowCreate( Widget parent, tJmdConnectionId *pId )
{
  Widget     bboard,
            *pChildren,
             scale,
             btn_cancel,
             btn_ok;
  int        level;
  Cardinal n;
  Arg        args [15];

  if(gMAP_INTENSITY_WINDOW == NULL)
  {
       level = DefaultIntensity;
  }
  else
  {
       XtManageChild( gMAP_INTENSITY_WINDOW );
       return;
  }

  /* create bulletin board */
  bboard      = MyCreateBulletinBoardDialog( parent, "bboard", 250, 120,
                                                  "Map Intensity",
  XmRESIZE_GROW ); scale              = MyCreateScale( bboard, "scale1", 35,
  NULL, 192, 30, 255, level );
  btn_cancel = MyCreatePushButton( bboard, "Cancel", NULL, NULL,
                                        (XtCallbackProc) IntensityButtonCB,
                                        (XtPointer) JmdCancelButton );

  btn_ok = MyCreatePushButton( bboard, "Ok", NULL, NULL,
                                    (XtCallbackProc ) IntensityButtonCB,
                                    (XtPointer) JmdOkButton );
  MyBulletinBoardSetDefaultButton( bboard, btn_ok );

  /* add callbacks */
  XtAddCallback( scale, XmNvalueChangedCallback,
                 ( XtCallbackProc ) IntensityChangeCB, NULL );
  XtAddCallback( scale, XmNdragCallback,
                 ( XtCallbackProc ) IntensityChangeCB, NULL );
  XtAddEventHandler( bboard, StructureNotifyMask, False,
                       ( XtEventHandler )MyBulletinBoardDoubleClickCB, NULL
);

  gMAP_INTENSITY_WINDOW = bboard;
}

void IntensityButtonCB( Widget w, tJmdButtonTag button_id, XtPointer call_data
) {
   int       val;
   Widget *children;
```

```
    children = (Widget*) MyWidgetGetChildren( XtParent( w ) );
    XmScaleGetValue( children[JmdIntensityScale], &val );
    /* call vimala's code */
    XtUnmanageChild( XtParent(w) );
}

void IntensityChangeCB( Widget w, tJmdConnectionId *pId, XtPointer call_data )
{
    int level;

    XmScaleGetValue( w, &level );
    MSetIntensity( pId -> channel, pId -> wid, level );
    MFlush( pId -> channel );
}
```

```
/*!!==========================================================================
==
     Disclaimer:             This software was developed for the Department of
                             Defense (DoD). All US  Government restrictions on
                             software source code distribution apply.

     Attribution:   This software is considered part of the GCCS Joint
                             Mapping Toolkit (JMTK).

     Organization:  Computer Sciences Corporation
                             6565 Arlington Boulevard
                             Falls Church, VA  22042 703.538.7276   Date:
                             19960319
                             Copyright (1996)        All Rights Reserved
     Source File:   xmenu.c
     Domain:                 JMTK Demo
     Scope:                  Private
     Version:                1.1
     Last Delta:             04/08/96 18:52:34
     Checked Out:   04/08/96 18:58:56
     Software Category:

     Revision History:

     Version        Person                      Date               Reason
     =========      =====================       ========
========================
     sw 001         Brad Bunton                 30 NOV 95          Initial
Draft
     sw 002         Brad Bunton                 18 MAR 96          Added
separator,

accelerator
==============================================================================!!
*/
```

```
#include "xmenu.h"


     /*!!=====================================================================
     ======= Purpose:                 Main function to build a menu bar.
     Synopsis:   #include "xmenu.h"
                   XMenuCreateCascadeButton( Widget parent,
                                                 XtCallbackProc menuCBname,
                                                 char *txt, int *pos,
                                                 char *btn_label, char
*mnemonic,
                                                 int *nitems,
                                                 char *accel_text, char
     *accel ); Description:  XMenuCreateCascadeButton is a recursive calling
     function
                             which creates the cascade menu button and all of
                             its children.  A child can be a cascade menu
                             button, a menu button, or a separator.
     Return:       Nothing is returned.  The children can be obtained from
the
                   parent which is known from the calling process.
     Errors:       There is not error checking.
     Notes:      See also MenuParseText for more details.
=========================================================================!!
*/ void XMenuCreateCascadeButton( Widget parent, XtCallbackProc menuCBname,
                             char *txt, int *pos,
                                     char *btn_label, char *mnemonic, int
                                     *nitems, char *accel_txt, char *accel )
{
  Widget   btn, subbtn, pdm, sep;
  Arg        args [10];
  int        retval, cnitems, btn_id;
  Cardinal n;
  XmString xmstring;

  /* create pulldown menu pane */
  n = 0;
  pdm = XmCreatePulldownMenu( parent, "pdm", args, n );
  XtManageChild( pdm );

  /* create cascade button and attach it to "pdm" */
  n = 0;
  XtSetArg( args [n], XmNmnemonic, mnemonic[0] ); n++;
  XtSetArg( args [n], XmNsubMenuId, pdm ); n++;
          btn = XmCreateCascadeButton( parent, btn_label, args, n );
  XtManageChild( btn );

  /* create children of the cascade button */
  cnitems = *nitems;
  while( cnitems-- )
  {
       retval   = MenuParseText( txt, pos, btn_label, mnemonic, nitems,
                                   accel_txt, accel, &btn_id );
       n          = 0;
       xmstring = NULL;
       switch( retval )
       {
            case tokCASCADE_MENU_ITEM:
                 XMenuCreateCascadeButton( pdm, menuCBname, txt, pos,
                                                btn_label, mnemonic, nitems,
                                                accel_txt, accel );
                 break;
            case tokMENU_SEPARATOR:
                sep = XmCreateSeparator( pdm, "sep", args, n );
                 XtManageChild( sep );
                 break;
            case tokMENU_ACCELERATOR_ITEM:
```

```
                xmstring = XmStringCreate( accel_txt, XmFONTLIST_DEFAULT_TAG
                ); XtSetArg( args[n], XmNacceleratorText, xmstring ); n++;
                XtSetArg( args[n], XmNaccelerator, accel ); n++;
                /* do not break */
            case tokMENU_ITEM:
                XtSetArg( args [n], XmNmnemonic, mnemonic[0] ); n++; subbtn =
                XmCreatePushButton( pdm, btn_label, args, n ); if( xmstring )
                XmStringFree( xmstring ); XtManageChild( subbtn );

                 /* add callback for button if one exists */
                if( menuCBname )
                {
                     XtAddCallback( subbtn, XmNactivateCallback, menuCBname,
                     ( XtPointer ) btn_id );
                }
                break;
            case tokMENU_END:
                break;
        }
    }
}


     /*!!=====================================================================
     ======= Purpose:        Creates menubar, cascade buttons, and menu
     buttons.
     Synopsis:   #include "xmenu.h"
                   Widget XMenuCreateMenuBar( Widget parent, char
                                                  *menubar_name,
                                                  XtCallbackProc menuCBname
                                                  );
     Description:        XMenuCreateMenuBar creates the parent widget and all
of
                             the children for a menubar.
     Return:     The menubar widget id is returned.
     Errors:     There is no error checking done in this function.
     Notes:      See also MenuParseText, XMenuCreateCascadeButton.
=========================================================================!!
*/ Widget XMenuCreateMenuBar( Widget parent, char *menubar_name,
                          XtCallbackProc menuCBname )
{
  Widget     menubar;
  char       btn_label[80],
             mnemonic[2],
             accel_txt[80],
             accel[80],
            *txt;
  int        pos,
             nitems,
             cnitems,
             retval,
             btn_id;

  /* create and manage menu bar */
  menubar = XmCreateMenuBar( parent, "menubar", NULL, 0 );
  XtManageChild( menubar );

  /* get menubar description text from file */
  txt       = MenuGetText( menubar_name );
  pos       = 0;

  /* parse the first line in the text */
  /* no errors assumed, the first line will be the name of menubar */
      retval  = MenuParseText( txt, &pos, btn_label, mnemonic, &nitems,
                                  accel_txt, accel, &btn_id );
  cnitems = nitems;
  while( cnitems-- )
  {
```

```
        /* parse the next line in the text */
        /* no errors assumed, this line must be a cascade menu button */
        MenuParseText( txt, &pos, btn_label, mnemonic, &nitems,
                           accel_txt, accel, &btn_id );

        /* create the cascade menu button and all of its children */
        XMenuCreateCascadeButton( menubar, menuCBname,
                                       txt, &pos, btn_label, mnemonic,
                                       &nitems, accel_txt, accel );
    }
    free( txt );
    return( menubar );
}
```

```
/*!!==========================================================================
==
     Disclaimer:             This software was developed for the Department of
                             Defense (DoD). All US  Government restrictions on
                             software source code distribution apply.

     Attribution:   This software is considered part of the GCCS Joint
                             Mapping Toolkit (JMTK).

     Organization:  Computer Sciences Corporation
                             6565 Arlington Boulevard
                             Falls Church, VA  22042 703.538.7276 Date:
                             19960319
                             Copyright (1996)        All Rights Reserved
     Source File:   xmenu.h
     Domain:                 JMTK Demo
     Scope:                  PRIVATE
     Version:                1.1
     Last Delta:             04/08/96 18:52:34
     Checked Out:   04/08/96 18:58:56
     Software Category:

     Revision History:

     Version        Person                      Date            Reason
     =========      =====================       ========
============================
     sw 001         Brad Bunton                 30 NOV 95       Initial Draft
     sw 002         Brad Bunton                 18 MAR 96       Added
separator,
                                                                accelerator
==========================================================================
==

     Purpose:       This file defines the function prototypes for xmenu
functions.

     Notes:
===========================================================================!!
*/
```

```
#include "menu.c"
#include <Xm/MenuShell.h>
#include <Xm/PushB.h>
#include <Xm/CascadeB.h>
#include <Xm/RowColumn.h>
#include <Xm/Separator.h>

void XMenuCreateCascadeButton( Widget parent, XtCallbackProc menuCBname, char
                                      *txt, int *pos, char *btn_label, char
                                      *mnemonic, int *nitems,
                                      char *accel_txt, char *accel ); Widget
XMenuCreateMenuBar( Widget parent, char *menubar_name,
                                 XtCallbackProc menuCBname );
```

```
/*   File:  xprojection.c   */
#include "JMD.h"
#include "motif.h"
#include "_projection.c"

Widget gDEFAULT_MAP_WINDOW = NULL;

void JMD_DefaultMapWindowCreate( Widget parent, tJmdConnectionId  *pId ) {
  Widget       bboard,
               selectionbox,
               sep,
              *pLabeledtxt,
               btn_ok,
               btn_cancel,
               btn_help;
  char       **pItems;
  int          nitems;
  Cardinal     n;
  Arg          args[10];
  Dimension  h,
               w;
  Position     x,
               y;

  if( gDEFAULT_MAP_WINDOW )
  {
       bboard = gDEFAULT_MAP_WINDOW;
       XtManageChild( bboard );
       return;
  }

      bboard = MyCreateBulletinBoardDialog( parent, "bboard", 370, 350,
                                                "Default Map", XmRESIZE_GROW
  ); XtSetArg( args[0], XmNuserData, pId );
  XtSetValues( bboard, args, 1 );

  selectionbox = MyCreateSelectionBox( bboard, "sbox", 10, 10, 300, 330 );
  MyVaSelectionBoxUnmanageChild( selectionbox, 8, XmDIALOG_APPLY_BUTTON,
       XmDIALOG_CANCEL_BUTTON, XmDIALOG_DEFAULT_BUTTON, XmDIALOG_HELP_BUTTON,
       XmDIALOG_OK_BUTTON, XmDIALOG_SELECTION_LABEL, XmDIALOG_SEPARATOR,
       XmDIALOG_TEXT );
 MySelectionBoxSetLabel( selectionbox, XmDIALOG_LIST_LABEL, "Projections:" );
  nitems = 22;
  pItems = MyVaCreateCharList( nitems, "Mercator", "LambertConformalConic",
                 "Orthographic", "Stereographic", "Sinusoidal",
                 "TransverseMercator", "Gnomonic", "AzimuthalEqualArea",
                 "CylindricalEqualArea", "EquidistantCylindrical",
                 "VanDerGritten", "Linear", "MillerCylindrical", "Cassini",
                 "Mollweide", "EckertIV", "EckertVI",
                 "TransverseCylindricalEqualArea", "AlbersEqualAreaConic",
                 "EquidistantConic", "PolyConic", "Bonne" );
  qsort( (void*) pItems, (size_t) nitems,
          (size_t) sizeof(char*), MySortCharList );
  MySelectionBoxSetListItems( selectionbox, nitems, pItems );
  MyDestroyCharList( nitems, pItems );

   sep = MyCreateSeparator( bboard, "sep1", 10, NULL, XmHORIZONTAL, 330 );

  MyWidgetGetPosition( sep, &x, &y );
  y += MyWIDGET_OFFSET;
       pLabeledtxt = MyCreateLabeledTextField( bboard, "width", 80, y,
                                                   "Width:", 80, 20, "" );
  y += 20;
  pLabeledtxt = MyCreateLabeledTextField( bboard, "center", 80, y, "Center:",
                                                   80, 20, "" );
   sep = MyCreateSeparator( bboard, "sep2", 10, NULL, XmHORIZONTAL, 330 );
  btn_ok = MyCreatePushButton( bboard, "Ok", 10, NULL,
                                  (XtCallbackProc) DefaultMapButtonCB,
```

```
                                        JmdOkButton );
   MyBulletinBoardSetDefaultButton(bboard, btn_ok);

   MyWidgetGetPosition(btn_ok, &x, &y);
   btn_help = MyCreatePushButton( bboard, "Help", x+110, y, (XtCallbackProc)
                                          DefaultMapButtonCB, JmdHelpButton );

   btn_cancel = MyCreatePushButton( bboard, "Cancel", x+220, y,
                                            (XtCallbackProc) DefaultMapButtonCB,
                                            JmdCancelButton );

   XtAddEventHandler( bboard, StructureNotifyMask, False, (XtEventHandler)
                          MyBulletinBoardDoubleClickCB, NULL);

   gDEFAULT_MAP_WINDOW = bboard;
}
```