Subject: [SUMMARY & Software] Tcl/Tk with IDL/Pvwave or Matlab
Posted by atae on Fri, 18 Feb 1994 17:59:30 GMT
View Forum Message <> Reply to Message

G'Day

I wrote:
> Is anyone using Tcl/Tk from within IDL or Matlab ? What I was
> interested in what a direct interface between Tcl/Tk and these
> packages as opposed to say via C programs. Any help, tips etc..
> is greatly appreciated. I will make a summary of responses,
> whether by Email or on the Usenet, and post it since this info.
> is not in the FAQ.

Here is the only answer I received. Thanks Joe.

best regards
Ata <(|)>.

Date: Mon, 14 Feb 94 16:26:44 MST
From: vanandel@EDU.ucar.atd.rsf (Joe VanAndel)
Message-Id: <9402142326.AA19797@rsf.atd.ucar.EDU>
Subject: Re: Tcl/Tk with IDL or Matlab ?

I've used Tcl/Tk with Visual Numerics PV-WAVE. I basically registered some C routines as TCL commands so I could access WAVE variables and issue WAVE commands from TCL. It worked quite nicely, since I could use all the power of PV-WAVE for graphing, but could use the familiar TK commands for my user interface.

(I really didn't want to learn yet another set of widgets for the user interface, even though the widgets provided by Wave or IDL are perfectly adequate.)

I don't know how close the C language interfaces are for WAVE and IDL (although they share a common source "ancestor"), but you could certainly have my TCL interface to WAVE, if you wanted it.

--
Joe VanAndel Internet: vanandel@ncar.ucar.edu
National Center for Web http://www.atd.ucar.edu/jva/home.html
Atmospheric Research

Date: Tue, 15 Feb 1994 15:43:33 -0700
From: Joe Van Andel <vanandel@EDU.ucar.atd.stout>

Here's a shar file containing my TCL interfaces to PV-WAVE. You'll have to adapt the internals for IDL vs WAVE, but at least you'll see the framework. I

had to do some work to build a custom "wave" executable that included wave
along with TCL/TK, so you'll have to do this same thing for IDL.
shar wave_tk.tar tcl_wave.c tkAppInit.c wave_util.c wave.h

```
#! /bin/sh
# This is a shell archive.  Remove anything before this line, then unpack
# it by saving it into a file and typing "sh file".  To overwrite existing
# files, type "sh file -c".  You can also feed this as standard input via
# unshar, or by typing "sh <file", e.g..  If this archive is complete, you
# will see the following message at the end:
#  "End of shell archive."
# Contents:  wave_tk.tar tcl_wave.c tkAppInit.c wave_util.c wave.h
# Wrapped by vanandel@wabbit4 on Tue Feb 15 15:40:37 1994
PATH=/bin:/usr/bin:/usr/ucb ; export PATH
if test -f 'wave_tk.tar' -a "${1}" != "-c" ; then
  echo shar: Will not clobber existing file \"'wave_tk.tar'\"
else
echo shar: Extracting \"'wave_tk.tar'\" \(24576 characters\)
sed "s/^X//" >'wave_tk.tar' <<'END_OF_FILE'
Xtcl_wave.cX
X#include <tcl.h>
X
X#include "wave.h"
X
X
 X/********************************************************** ********************
X*                                                          *
X* Copyright (c) 1993,            *
X*  National Center for Atmospheric Research                          *
X*                                                          *
 X********************************************************** ********************/
X
X#if !defined(lint) && !defined(VX)
Xstatic char rcsid[]=
X"$Header: /scr/rsf/rsg_cvs/rsg/radar/src/rds/diag/fft/tcl_wave.c,v 1.3 1993/12/29 22:54:13
vanandel Exp $";
X#endif
X
 X/********************************************************** ********************
X* Revision history:                                                 *
X* $Log: tcl_wave.c,v $
X * Revision 1.3  1993/12/29  22:54:13  vanandel
X * :  Makefile tcl_wave.c tkAppInit.c ts.pro wave.h wave_util.c
X * : Removed Files:
X * :  main.c
X * :  ---------------------------------------------------------- ----------
X * Cleanup to use Tk's main().  Added support for accessing WAVE's short
X * variables.
```

```
X *
X * Revision 1.2  1993/10/20  15:08:14  vanandel
X * *** empty log message ***
X *
X * Revision 1.1  1993/09/21  18:05:33  vanandel
X * Initial revision
X *
X*                                                              *
X*********************************END OF RCS INFO*****************************/
X
X/*
X * utility routine to allow controlling VISUAL NUMERICS WAVE CL via TCL
X *
X */
X
Xextern int cwavec(int action, int argc, char *argv[]);
Xenum WAVE_CALL {WAVE_INTERACT=1,WAVE_CALL=2, WAVE_CLOSE=3};
X
Xint
XTcl_CallWave(ClientData *clientData, Tcl_Interp *interp, int argc,
X char *argv [ ])
X{
X    int action = (int) clientData;
X    char *cmd = Tcl_Concat(argc-1, argv+1);
X    int stat;
X
X    if ((action < 1) || (action > 3))
X    {
X Tcl_AppendResult (interp, argv[0], "ERROR: action must be 1, 2, or 3",
X  (char *) NULL);
X return TCL_ERROR;
X    }
X    /* call PVWAVE to invoke the command */
X    stat = cwavec(action, 1, &cmd);
X
X    ckfree(cmd);
X    return TCL_OK;
X}
X#define MAX_NAME 80
X
Xint
XTcl_WaveFloatVar(ClientData *clientData, Tcl_Interp *interp, int argc,
X char *argv [ ])
X{
X    char *common;
X    char *varName;
X    char buf[MAX_NAME];
X    float *valuePtr;
```

```
X    double newVal;
X
X    if (argc == 3 || argc == 4) {
X if (argv[1][0] == '\0') {
X     strcpy(buf, argv[2]);
X } else {
X     sprintf(buf, "%s (%s)", argv[2], argv[1]);
X }
X valuePtr = WaveGetFloatVar(buf);
X if (!valuePtr )
X {
X     Tcl_AppendResult (interp, argv[0],
X  "ERROR: can't find  floating point value with name ",
X  buf, (char *) NULL);
X     return TCL_ERROR;
X }
X if (argc == 3)
X {
X     /* retrieving */
X     sprintf(buf, "%f", *valuePtr);
X     Tcl_SetResult(interp, buf, TCL_VOLATILE);
X     return TCL_OK;
X } else {
X     if (Tcl_GetDouble(interp, argv[3], &newVal) != TCL_OK)
X     {
X  return TCL_ERROR;
X     }
X     /* set the WAVE variable */
X     *valuePtr = (float) newVal;
X }
X
X    } else  {
X Tcl_AppendResult (interp, "usage: ", argv[0],
X  "common_tag var_name ?value?" ,  (char *) NULL);
X return TCL_ERROR;
X    }
X}
X
Xint
XTcl_WaveLongVar(ClientData *clientData, Tcl_Interp *interp, int argc,
X char *argv [ ])
X{
X    char *common;
X    char *varName;
X    char buf[MAX_NAME];
X    long *valuePtr;
X    long newVal;
X
```

```
X    if (argc == 3 || argc == 4) {
X if (argv[1][0] == '\0') {
X     strcpy(buf, argv[2]);
X } else {
X     sprintf(buf, "%s (%s)", argv[2], argv[1]);
X }
X valuePtr = WaveGetLongVar(buf);
X if (!valuePtr )
X {
X     Tcl_AppendResult (interp, argv[0],
X  "ERROR: can't find  long value with name ",
X  buf, (char *) NULL);
X     return TCL_ERROR;
X }
X if (argc == 3)
X {
X     /* retrieving */
X     sprintf(buf, "%d", *valuePtr);
X     Tcl_SetResult(interp, buf, TCL_VOLATILE);
X     return TCL_OK;
X } else {
X     if (Tcl_GetInt(interp, argv[3], (int *) &newVal) != TCL_OK)
X     {
X  return TCL_ERROR;
X     }
X     /* set the WAVE variable */
X     *valuePtr =  newVal;
X }
X
X    } else  {
X Tcl_AppendResult (interp, "usage: ", argv[0],
X  "common_tag var_name ?value?" ,  (char *) NULL);
X return TCL_ERROR;
X    }
X}
Xint
XTcl_WaveShortVar(ClientData *clientData, Tcl_Interp *interp, int argc,
X char *argv [ ])
X{
X    char *common;
X    char *varName;
X    char buf[MAX_NAME];
X    short *valuePtr;
X    long newVal;
X
X    if (argc == 3 || argc == 4) {
X if (argv[1][0] == '\0') {
X     strcpy(buf, argv[2]);
```

```
X } else {
X     sprintf(buf, "%s (%s)", argv[2], argv[1]);
X }
X valuePtr = WaveGetShortVar(buf);
X if (!valuePtr )
X {
X     Tcl_AppendResult (interp, argv[0],
X  "ERROR: can't find  long value with name ",
X  buf, (char *) NULL);
X     return TCL_ERROR;
X }
X if (argc == 3)
X {
X     /* retrieving */
X     sprintf(buf, "%d", *valuePtr);
X     Tcl_SetResult(interp, buf, TCL_VOLATILE);
X     return TCL_OK;
X } else {
X     if (Tcl_GetInt(interp, argv[3], (int *) &newVal) != TCL_OK)
X     {
X  return TCL_ERROR;
X     }
X     /* set the WAVE variable */
X     *valuePtr =  newVal;
X }
X
X    } else  {
X Tcl_AppendResult (interp, "usage: ", argv[0],
X  "common_tag var_name ?value?" ,  (char *) NULL);
X return TCL_ERROR;
X    }
X}
X/* initialize this package */
X
Xint
XWave_Init(Tcl_Interp *interp)
X{
X    Tcl_CreateCommand(interp, "wave_cl", (Tcl_CmdProc *)Tcl_CallWave,
X (ClientData) WAVE_INTERACT, (Tcl_CmdDeleteProc *) NULL);
X
X    Tcl_CreateCommand(interp, "wave", (Tcl_CmdProc *)Tcl_CallWave,
X (ClientData) WAVE_CALL, (Tcl_CmdDeleteProc *) NULL);
X
X    Tcl_CreateCommand(interp, "wave_close", (Tcl_CmdProc *)Tcl_CallWave,
X (ClientData) WAVE_CLOSE, (Tcl_CmdDeleteProc *) NULL);
X
X    Tcl_CreateCommand(interp, "wave_float", (Tcl_CmdProc *)Tcl_WaveFloatVar,
X (ClientData) NULL, (Tcl_CmdDeleteProc *) NULL);
```

```
X
X    Tcl_CreateCommand(interp, "wave_long", (Tcl_CmdProc *)Tcl_WaveLongVar,
X (ClientData) NULL, (Tcl_CmdDeleteProc *) NULL);
X
X    Tcl_CreateCommand(interp, "wave_int", (Tcl_CmdProc *)Tcl_WaveShortVar,
X (ClientData) NULL, (Tcl_CmdDeleteProc *) NULL);
X
X    return TCL_OK;
X}
X * tkAppInit.c --
X *
X * Provides a default version of the Tcl_AppInit procedure for
X * use in wish and similar Tk-based applications.
X *
X * Copyright (c) 1993 The Regents of the University of California.
X * All rights reserved.
X *
X * Permission is hereby granted, without written agreement and without
X * license or royalty fees, to use, copy, modify, and distribute this
X * software and its documentation for any purpose, provided that the
X * above copyright notice and the following two paragraphs appear in
X * all copies of this software.
X *
X * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
X * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING
OUT
X * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE
UNIVERSITY OF
X * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
X *
X * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
X * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
X * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED
HEREUNDER IS
X * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
X * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
MODIFICATIONS.
X */
X
X#ifndef lint
Xstatic char rcsid[] = "$Header: /scr/rsf/rsg_cvs/rsg/radar/src/rds/diag/fft/tkAppInit.c,v 1.2
1993/12/29 22:54:14 vanandel Exp $ SPRITE (Berkeley)";
X#endif /* not lint */
X
X#include "tk.h"
X
X/*
X * The following variable is a special hack that allows applications
```

```
X * to be linked using the procedure "main" from the Tk library.  The
X * variable generates a reference to "main", which causes main to
X * be brought in from the library (and all of Tk and Tcl with it).
X */
X
Xextern int main();
Xint *tclDummyMainPtr = (int *) main;
X
X/*
X  *---------------------------------------------------------- -----------
X *
X * Tcl_AppInit --
X *
X * This procedure performs application-specific initialization.
X * Most applications, especially those that incorporate additional
X * packages, will have their own version of this procedure.
X *
X * Results:
X * Returns a standard Tcl completion code, and leaves an error
X * message in interp->result if an error occurs.
X *
X * Side effects:
X * Depends on the startup script.
X *
X  *---------------------------------------------------------- -----------
X */
X
Xint
XTcl_AppInit(interp)
X    Tcl_Interp *interp;  /* Interpreter for application. */
X{
X    Tk_Window main;
X
X    main = Tk_MainWindow(interp);
X
X    /*
X     * Call the init procedures for included packages.  Each call should
X     * look like this:
X     *
X     * if (Mod_Init(interp) == TCL_ERROR) {
X     *     return TCL_ERROR;
X     * }
X     *
X     * where "Mod" is the name of the module.
X     */
X    if (Wave_Init(interp) == TCL_ERROR) {
X        return TCL_ERROR;
X    }
```

```
X
X
X    if (Tcl_Init(interp) == TCL_ERROR) {
X return TCL_ERROR;
X    }
X    if (Tk_Init(interp) == TCL_ERROR) {
X return TCL_ERROR;
X    }
X
X    /*
X     * Call Tcl_CreateCommand for application-specific commands, if
X     * they weren't already created by the init procedures called above.
X     */
X
X    /*
X     * Specify a user-specific startup file to invoke if the application
X     * is run interactively.  Typically the startup file is "~/.apprc"
X     * where "app" is the name of the application.  If this line is deleted
X     * then no user-specific startup file will be run under any conditions.
X     */
X
X    tcl_RcFileName = "~/.wave_tkrc";
X    return TCL_OK;
X}
X;
X } else {
X     sprintf(buf, "%s (%s)", argv[2], argv[1]);
X }
X valuePtr = WaveGetFloatVar(buf);
X if (!valuePtr )
X {
X   wave_util.cX * via LINKNLOAD.  It modifies the values of the
X * parameters, passed to it, to show that a C function
X * can access and modify parameters passed to it by
X * PV-WAVE CL. It also calls the C function wavevars
X * which returns the names of, and pointers to, all
X * PV-WAVE CL variables. It then accesses the main
X * level PV-WAVE CL variable, wmainfltarr and modifies
X * it's value to show that a C program can access
X * PV-WAVE CL's variable data space directly.
X */
X
X/* Include the standard C i/o library */
X#include <stdio.h>
X
X/* Include wavevars structure definition and
X * PV-WAVE CL type definitions.
X */
```

```
X#include "wavevars.h"
X
X#include "wave.h"
X
X
X
X
X
X
Xchar *
XWaveName(char *common, int type, int num)
X{
X    static char buf[30];
X    if (type == VAR_I) {
X sprintf(buf, "I%d (%s)", num, common);
X    } else {
X sprintf(buf, "Q%d (%s)", num, common);
X    }
X    return buf;
X}
X
Xstatic WaveVariable *Vars = NULL;
Xstatic int Nvars;
Xstatic char         **WaveNames = NULL;
X/* #define DEBUG1 */
X
X
X
Xfloat* WaveGetFloatArray(char *name, int minSize)
X{
X
X    int i;
X    int found = 0;
X    float *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
```

```
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
X#endif
X     if (v->type ==  (TYP_FLOAT | TYP_ARRAY)) {
X  dataptr = ((float *) v->data);
X  if (v->numelems < minSize)
X  {
X    fprintf (stderr, " %s has size %d, too small! \n",
X   name, v->numelems);
X    return NULL;
X  }
X  found = 1;
X     }
X     break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
X
Xfloat* WaveGetFloatVar(char *name)
X{
X
X    int i;
X    int found = 0;
X    float *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
```

```
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
X
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
X#endif
X     if (v->type ==  (TYP_FLOAT )) {
X  dataptr = ((float *) v->data);
X  found = 1;
X     }
X     break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
X
Xlong* WaveGetLongVar(char *name)
X{
X
X    int i;
X    int found = 0;
X    long *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
```

```
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
X
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
X#endif
X     if (v->type ==  (TYP_LONG )) {
X  dataptr = ((long *) v->data);
X  found = 1;
X     }
X     break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
X
Xshort* WaveGetShortVar(char *name)
X{
X
X    int i;
X    int found = 0;
X    short *dataptr = NULL;
```

```
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
X
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
X#endif
X     if (v->type ==  (TYP_INT )) {
X  dataptr = ((short *) v->data);
X  found = 1;
X     }
X     break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
XE CL's variable data space direcwave.hX
Xextern float * WaveGetFloatVar(char *name);
Xextern long * WaveGetLongVar(char *name);
Xextern short * WaveGetShortVar(char *name);
```

```
X
Xenum VAR_TYPE {VAR_I, VAR_Q};
Xextern char * WaveName(char *common, int type, int num);
X
Xextern float* WaveGetFloatArray(char *name, int minSize);
X
X
X break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
XE CL's variable data space direcX * via LINKNLOAD.  It modifies the values of the
X * parameters, passed to it, to show that a C function
X * can access and modify parameters passed to it by
X * PV-WAVE CL. It also calls the C function wavevars
X * which returns the names of, and pointers to, all
X * PV-WAVE CL variables. It then accesses the main
X * level PV-WAVE CL variable, wmainfltarr and modifies
X * it's value to show that a C program can access
X * PV-WAVE CL's variable data space directly.
X */
X
X/* Include the standard C i/o library */
X#include <stdio.h>
X
X/* Include wavevars structure definition and
X * PV-WAVE CL type definitions.
X */
X#include "wavevars.h"
X
X#include "wave.h"
X
X
X
X
X
X
Xchar *
XWaveName(char *common, int type, int num)
X{
X    static char buf[30];
X    if (type == VAR_I) {
X sprintf(buf, "I%d (%s)", num, common);
```

```
X    } else {
X sprintf(buf, "Q%d (%s)", num, common);
X    }
X    return buf;
X}
X
Xstatic WaveVariable *Vars = NULL;
Xstatic int Nvars;
Xstatic char         **WaveNames = NULL;
X/* #define DEBUG1 */
X
X
X
Xfloat* WaveGetFloatArray(char *name, int minSize)
X{
X
X    int i;
X    int found = 0;
X    float *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
```

```
X#endif
X     if (v->type ==  (TYP_FLOAT | TYP_ARRAY)) {
X  dataptr = ((float *) v->data);
X  if (v->numelems < minSize)
X  {
X    fprintf (stderr, " %s has size %d, too small! \n",
X   name, v->numelems);
X    return NULL;
X  }
X  found = 1;
X     }
X     break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
X
Xfloat* WaveGetFloatVar(char *name)
X{
X
X    int i;
X    int found = 0;
X    float *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
X
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
```

```
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
X#endif
X     if (v->type ==  (TYP_FLOAT )) {
X  dataptr = ((float *) v->data);
X  found = 1;
X     }
X     break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
X
Xlong* WaveGetLongVar(char *name)
X{
X
X    int i;
X    int found = 0;
X    long *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
```

```
X
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
X#endif
X     if (v->type ==  (TYP_LONG )) {
X  dataptr = ((long *) v->data);
X  found = 1;
X     }
X     break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
X
Xshort* WaveGetShortVar(char *name)
X{
X
X    int i;
X    int found = 0;
X    short *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
```

```
X       return (NULL);
X }
X    }
X
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++)END_OF_FILE
if test 24576 -ne `wc -c <'wave_tk.tar'`; then
    echo shar: \"'wave_tk.tar'\" unpacked with wrong size!
fi
# end of 'wave_tk.tar'
fi
if test -f 'tcl_wave.c' -a "${1}" != "-c" ; then
  echo shar: Will not clobber existing file \"'tcl_wave.c'\"
else
echo shar: Extracting \"'tcl_wave.c'\" \(5996 characters\)
sed "s/^X//" >'tcl_wave.c' <<'END_OF_FILE'
X#include <stdio.h>
X
X#include <tcl.h>
X
X#include "wave.h"
X
X
 X/****************************************************** ********************
X*                                                      *
X* Copyright (c) 1993,             *
X*  National Center for Atmospheric Research                        *
X*                                                      *
 X****************************************************** ********************/
X
X#if !defined(lint) && !defined(VX)
Xstatic char rcsid[]=
X"$Header: /scr/rsf/rsg_cvs/rsg/radar/src/rds/diag/fft/tcl_wave.c,v 1.3 1993/12/29 22:54:13
vanandel Exp $";
X#endif
X
 X/****************************************************** ********************
X* Revision history:                                             *
X* $Log: tcl_wave.c,v $
X * Revision 1.3  1993/12/29  22:54:13  vanandel
X * :  Makefile tcl_wave.c tkAppInit.c ts.pro wave.h wave_util.c
X * : Removed Files:
X * :  main.c
X * :  ---------------------------------------------------------- ----------
X * Cleanup to use Tk's main().  Added support for accessing WAVE's short
X * variables.
```

```
X *
X * Revision 1.2  1993/10/20  15:08:14  vanandel
X * *** empty log message ***
X *
X * Revision 1.1  1993/09/21  18:05:33  vanandel
X * Initial revision
X *
X*                                                              *
X*********************************END OF RCS INFO*****************************/
X
X/*
X * utility routine to allow controlling VISUAL NUMERICS WAVE CL via TCL
X *
X */
X
Xextern int cwavec(int action, int argc, char *argv[]);
Xenum WAVE_CALL {WAVE_INTERACT=1,WAVE_CALL=2, WAVE_CLOSE=3};
X
Xint
XTcl_CallWave(ClientData *clientData, Tcl_Interp *interp, int argc,
X char *argv [ ])
X{
X    int action = (int) clientData;
X    char *cmd = Tcl_Concat(argc-1, argv+1);
X    int stat;
X
X    if ((action < 1) || (action > 3))
X    {
X Tcl_AppendResult (interp, argv[0], "ERROR: action must be 1, 2, or 3",
X  (char *) NULL);
X return TCL_ERROR;
X    }
X    /* call PVWAVE to invoke the command */
X    stat = cwavec(action, 1, &cmd);
X
X    ckfree(cmd);
X    return TCL_OK;
X}
X#define MAX_NAME 80
X
Xint
XTcl_WaveFloatVar(ClientData *clientData, Tcl_Interp *interp, int argc,
X char *argv [ ])
X{
X    char *common;
X    char *varName;
X    char buf[MAX_NAME];
X    float *valuePtr;
```

```
X    double newVal;
X
X    if (argc == 3 || argc == 4) {
X if (argv[1][0] == '\0') {
X     strcpy(buf, argv[2]);
X } else {
X     sprintf(buf, "%s (%s)", argv[2], argv[1]);
X }
X valuePtr = WaveGetFloatVar(buf);
X if (!valuePtr )
X {
X     Tcl_AppendResult (interp, argv[0],
X  "ERROR: can't find  floating point value with name ",
X  buf, (char *) NULL);
X     return TCL_ERROR;
X }
X if (argc == 3)
X {
X     /* retrieving */
X     sprintf(buf, "%f", *valuePtr);
X     Tcl_SetResult(interp, buf, TCL_VOLATILE);
X     return TCL_OK;
X } else {
X     if (Tcl_GetDouble(interp, argv[3], &newVal) != TCL_OK)
X     {
X  return TCL_ERROR;
X     }
X     /* set the WAVE variable */
X     *valuePtr = (float) newVal;
X }
X
X    } else  {
X Tcl_AppendResult (interp, "usage: ", argv[0],
X  "common_tag var_name ?value?" ,  (char *) NULL);
X return TCL_ERROR;
X    }
X}
X
Xint
XTcl_WaveLongVar(ClientData *clientData, Tcl_Interp *interp, int argc,
X char *argv [ ])
X{
X    char *common;
X    char *varName;
X    char buf[MAX_NAME];
X    long *valuePtr;
X    long newVal;
X
```

```
X    if (argc == 3 || argc == 4) {
X if (argv[1][0] == '\0') {
X     strcpy(buf, argv[2]);
X } else {
X     sprintf(buf, "%s (%s)", argv[2], argv[1]);
X }
X valuePtr = WaveGetLongVar(buf);
X if (!valuePtr )
X {
X     Tcl_AppendResult (interp, argv[0],
X  "ERROR: can't find  long value with name ",
X  buf, (char *) NULL);
X     return TCL_ERROR;
X }
X if (argc == 3)
X {
X     /* retrieving */
X     sprintf(buf, "%d", *valuePtr);
X     Tcl_SetResult(interp, buf, TCL_VOLATILE);
X     return TCL_OK;
X } else {
X     if (Tcl_GetInt(interp, argv[3], (int *) &newVal) != TCL_OK)
X     {
X  return TCL_ERROR;
X     }
X     /* set the WAVE variable */
X     *valuePtr =  newVal;
X }
X
X    } else  {
X Tcl_AppendResult (interp, "usage: ", argv[0],
X  "common_tag var_name ?value?" ,  (char *) NULL);
X return TCL_ERROR;
X    }
X}
Xint
XTcl_WaveShortVar(ClientData *clientData, Tcl_Interp *interp, int argc,
X char *argv [ ])
X{
X    char *common;
X    char *varName;
X    char buf[MAX_NAME];
X    short *valuePtr;
X    long newVal;
X
X    if (argc == 3 || argc == 4) {
X if (argv[1][0] == '\0') {
X     strcpy(buf, argv[2]);
```

```
X } else {
X     sprintf(buf, "%s (%s)", argv[2], argv[1]);
X }
X valuePtr = WaveGetShortVar(buf);
X if (!valuePtr )
X {
X     Tcl_AppendResult (interp, argv[0],
X  "ERROR: can't find  long value with name ",
X  buf, (char *) NULL);
X     return TCL_ERROR;
X }
X if (argc == 3)
X {
X     /* retrieving */
X     sprintf(buf, "%d", *valuePtr);
X     Tcl_SetResult(interp, buf, TCL_VOLATILE);
X     return TCL_OK;
X } else {
X     if (Tcl_GetInt(interp, argv[3], (int *) &newVal) != TCL_OK)
X     {
X  return TCL_ERROR;
X     }
X     /* set the WAVE variable */
X     *valuePtr =  newVal;
X }
X
X    } else  {
X Tcl_AppendResult (interp, "usage: ", argv[0],
X  "common_tag var_name ?value?" ,  (char *) NULL);
X return TCL_ERROR;
X    }
X}
X/* initialize this package */
X
Xint
XWave_Init(Tcl_Interp *interp)
X{
X    Tcl_CreateCommand(interp, "wave_cl", (Tcl_CmdProc *)Tcl_CallWave,
X (ClientData) WAVE_INTERACT, (Tcl_CmdDeleteProc *) NULL);
X
X    Tcl_CreateCommand(interp, "wave", (Tcl_CmdProc *)Tcl_CallWave,
X (ClientData) WAVE_CALL, (Tcl_CmdDeleteProc *) NULL);
X
X    Tcl_CreateCommand(interp, "wave_close", (Tcl_CmdProc *)Tcl_CallWave,
X (ClientData) WAVE_CLOSE, (Tcl_CmdDeleteProc *) NULL);
X
X    Tcl_CreateCommand(interp, "wave_float", (Tcl_CmdProc *)Tcl_WaveFloatVar,
X (ClientData) NULL, (Tcl_CmdDeleteProc *) NULL);
```

```
X
X    Tcl_CreateCommand(interp, "wave_long", (Tcl_CmdProc *)Tcl_WaveLongVar,
X (ClientData) NULL, (Tcl_CmdDeleteProc *) NULL);
X
X    Tcl_CreateCommand(interp, "wave_int", (Tcl_CmdProc *)Tcl_WaveShortVar,
X (ClientData) NULL, (Tcl_CmdDeleteProc *) NULL);
X
X    return TCL_OK;
X}
END_OF_FILE
if test 5996 -ne `wc -c <'tcl_wave.c'`; then
    echo shar: \"'tcl_wave.c'\" unpacked with wrong size!
fi
# end of 'tcl_wave.c'
fi
if test -f 'tkAppInit.c' -a "${1}" != "-c" ; then
  echo shar: Will not clobber existing file \"'tkAppInit.c'\"
else
echo shar: Extracting \"'tkAppInit.c'\" \(3464 characters\)
sed "s/^X//" >'tkAppInit.c' <<'END_OF_FILE'
X/*
X * tkAppInit.c --
X *
X * Provides a default version of the Tcl_AppInit procedure for
X * use in wish and similar Tk-based applications.
X *
X * Copyright (c) 1993 The Regents of the University of California.
X * All rights reserved.
X *
X * Permission is hereby granted, without written agreement and without
X * license or royalty fees, to use, copy, modify, and distribute this
X * software and its documentation for any purpose, provided that the
X * above copyright notice and the following two paragraphs appear in
X * all copies of this software.
X *
X * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
X * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING
OUT
X * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE
UNIVERSITY OF
X * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
X *
X * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
X * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
X * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED
HEREUNDER IS
X * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
X * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
```

```
MODIFICATIONS.
X */
X
X#ifndef lint
Xstatic char rcsid[] = "$Header: /scr/rsf/rsg_cvs/rsg/radar/src/rds/diag/fft/tkAppInit.c,v 1.2
1993/12/29 22:54:14 vanandel Exp $ SPRITE (Berkeley)";
X#endif /* not lint */
X
X#include "tk.h"
X
X/*
X * The following variable is a special hack that allows applications
X * to be linked using the procedure "main" from the Tk library.  The
X * variable generates a reference to "main", which causes main to
X * be brought in from the library (and all of Tk and Tcl with it).
X */
X
Xextern int main();
Xint *tclDummyMainPtr = (int *) main;
X
X/*
X  *---------------------------------------------------------- -----------
X *
X * Tcl_AppInit --
X *
X * This procedure performs application-specific initialization.
X * Most applications, especially those that incorporate additional
X * packages, will have their own version of this procedure.
X *
X * Results:
X * Returns a standard Tcl completion code, and leaves an error
X * message in interp->result if an error occurs.
X *
X * Side effects:
X * Depends on the startup script.
X *
X  *---------------------------------------------------------- -----------
X */
X
Xint
XTcl_AppInit(interp)
X    Tcl_Interp *interp;  /* Interpreter for application. */
X{
X    Tk_Window main;
X
X    main = Tk_MainWindow(interp);
X
X    /*
```

```
X     * Call the init procedures for included packages.  Each call should
X     * look like this:
X     *
X     * if (Mod_Init(interp) == TCL_ERROR) {
X     *     return TCL_ERROR;
X     * }
X     *
X     * where "Mod" is the name of the module.
X     */
X    if (Wave_Init(interp) == TCL_ERROR) {
X        return TCL_ERROR;
X    }
X
X
X    if (Tcl_Init(interp) == TCL_ERROR) {
X return TCL_ERROR;
X    }
X    if (Tk_Init(interp) == TCL_ERROR) {
X return TCL_ERROR;
X    }
X
X    /*
X     * Call Tcl_CreateCommand for application-specific commands, if
X     * they weren't already created by the init procedures called above.
X     */
X
X    /*
X     * Specify a user-specific startup file to invoke if the application
X     * is run interactively.  Typically the startup file is "~/.apprc"
X     * where "app" is the name of the application.  If this line is deleted
X     * then no user-specific startup file will be run under any conditions.
X     */
X
X    tcl_RcFileName = "~/.wave_tkrc";
X    return TCL_OK;
X}
END_OF_FILE
if test 3464 -ne `wc -c <'tkAppInit.c'`; then
    echo shar: \"'tkAppInit.c'\" unpacked with wrong size!
fi
# end of 'tkAppInit.c'
fi
if test -f 'wave_util.c' -a "${1}" != "-c" ; then
  echo shar: Will not clobber existing file \"'wave_util.c'\"
else
echo shar: Extracting \"'wave_util.c'\" \(5600 characters\)
sed "s/^X//" >'wave_util.c' <<'END_OF_FILE'
X/* This is a 'C' function to be called by PV-WAVE CL,
```

```
X * via LINKNLOAD.  It modifies the values of the
X * parameters, passed to it, to show that a C function
X * can access and modify parameters passed to it by
X * PV-WAVE CL. It also calls the C function wavevars
X * which returns the names of, and pointers to, all
X * PV-WAVE CL variables. It then accesses the main
X * level PV-WAVE CL variable, wmainfltarr and modifies
X * it's value to show that a C program can access
X * PV-WAVE CL's variable data space directly.
X */
X
X/* Include the standard C i/o library */
X#include <stdio.h>
X
X/* Include wavevars structure definition and
X * PV-WAVE CL type definitions.
X */
X#include "wavevars.h"
X
X#include "wave.h"
X
X
X
X
X
X
Xchar *
XWaveName(char *common, int type, int num)
X{
X    static char buf[30];
X    if (type == VAR_I) {
X sprintf(buf, "I%d (%s)", num, common);
X    } else {
X sprintf(buf, "Q%d (%s)", num, common);
X    }
X    return buf;
X}
X
Xstatic WaveVariable *Vars = NULL;
Xstatic int Nvars;
Xstatic char         **WaveNames = NULL;
X/* #define DEBUG1 */
X
X
X
Xfloat* WaveGetFloatArray(char *name, int minSize)
X{
X
```

```
X    int i;
X    int found = 0;
X    float *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
X#endif
X     if (v->type ==  (TYP_FLOAT | TYP_ARRAY)) {
X  dataptr = ((float *) v->data);
X  if (v->numelems < minSize)
X  {
X    fprintf (stderr, " %s has size %d, too small! \n",
X   name, v->numelems);
X    return NULL;
X  }
X  found = 1;
X     }
X     break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
```

```
X       dataptr);
X#endif
X    return dataptr;
X}
X
Xfloat* WaveGetFloatVar(char *name)
X{
X
X    int i;
X    int found = 0;
X    float *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
X
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
X#endif
X     if (v->type ==  (TYP_FLOAT )) {
X  dataptr = ((float *) v->data);
X  found = 1;
X     }
X     break;
X }
```

```
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
X
Xlong* WaveGetLongVar(char *name)
X{
X
X    int i;
X    int found = 0;
X    long *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
X
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
X#endif
X     if (v->type ==  (TYP_LONG )) {
X  dataptr = ((long *) v->data);
X  found = 1;
```

```
X     }
X     break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
X
Xshort* WaveGetShortVar(char *name)
X{
X
X    int i;
X    int found = 0;
X    short *dataptr = NULL;
X    int result;
X#ifdef NOTDEF
X    fprintf(stderr, "WaveGetFloatArray called for %s, calling wavevars\n", name);
X#endif
X
X    if (!Vars)
X    {
X result = wavevars (&Nvars, &WaveNames, &Vars);
X
X if (result == 0) {
X
X       /* Bad return value from wavevars. Print error message
X        * and return control to PV-WAVE CL.
X        */
X       printf ("Bad return value from wavevars! \n");
X       return (NULL);
X }
X    }
X
X#ifdef DEBUG1
X    fprintf(stderr, "searching %d variables \n", Nvars);
X#endif
X    for (i = 0; i < Nvars; i++) {
X#ifdef DEBUG1
X fprintf(stderr, "checking %s against %s\n", name, WaveNames[i]);
X#endif
X if (strcmp(name, WaveNames[i]) == 0) {
X     WaveVariable *v = &Vars[i];
X#ifdef DEBUG1
X    fprintf (stderr, "found variable for  %s ! \n", name);
X#endif
```

```
X     if (v->type ==  (TYP_INT )) {
X  dataptr = ((short *) v->data);
X  found = 1;
X     }
X     break;
X }
X    }
X#ifdef DEBUG1
X   fprintf (stderr, "returning pointer for  %s of 0x%x ! \n", name,
X       dataptr);
X#endif
X    return dataptr;
X}
END_OF_FILE
if test 5600 -ne `wc -c <'wave_util.c'`; then
    echo shar: \"'wave_util.c'\" unpacked with wrong size!
fi
# end of 'wave_util.c'
fi
if test -f 'wave.h' -a "${1}" != "-c" ; then
  echo shar: Will not clobber existing file \"'wave.h'\"
else
echo shar: Extracting \"'wave.h'\" \(335 characters\)
sed "s/^X//" >'wave.h' <<'END_OF_FILE'
Xextern int cwavec(int action, int argc, char *argv[]);
X
Xextern float * WaveGetFloatVar(char *name);
Xextern long * WaveGetLongVar(char *name);
Xextern short * WaveGetShortVar(char *name);
X
Xenum VAR_TYPE {VAR_I, VAR_Q};
Xextern char * WaveName(char *common, int type, int num);
X
Xextern float* WaveGetFloatArray(char *name, int minSize);
X
X
END_OF_FILE
if test 335 -ne `wc -c <'wave.h'`; then
    echo shar: \"'wave.h'\" unpacked with wrong size!
fi
# end of 'wave.h'
fi
echo shar: End of shell archive.
exit 0
```

Joe VanAndel Internet:vanandel@ncar.ucar.edu
National Center for Atmospheric Research