KAPL-P-000067
(K-98035)

REAL TIME PROGRAMMING ENVIRONMENT FOR WINDOWS, APPENDIX A

CONF-980448--

April 5, 1998

NOTICE

KAPL ATOMIC POWER LABORATORY SCHENECTADY, NEW YORK 12301

Operated for the U. S. Department of Energy
by KAPL, Inc. a Lockheed Martin company

DISCLAIMER

```
; IN NO EVENT SHALL THE AUTHOR BE LIABLE TO ANY PARTY FOR DIRECT,
; INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF
; THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHOR HAS
; BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
;
; THE AUTHOR SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT
; LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS"
; BASIS, AND THE AUTHOR HAS NO OBLIGATION TO PROVIDE MAINTENANCE,
; SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
; ------------------------------------------------------------

NAME          rtsched
DESCRIPTION 'Real Time Scheduler for Windows'
CODE          FIXED PRELOAD
DATA          FIXED MULTIPLE PRELOAD
HEAPSIZE    1024
STACKSIZE   8192

SECTIONS
              SHAREDAT1 READ WRITE SHARED
```

This appendix contains all source code for the RTProE system. The following file contents are included.

pdb.h
hgen.l
hgen.y
igen.l
igen.y
pdm.l
pdm.y
rtdata.l
rtdata.y
framegen.c
librt.c
librt.h
rtsched.c
build.tsh
sde.tcl
rtsched.def

```
/*
  pdb.h - This file contains the structures for the
                        Programmer's Data Base (PDB).

                        The PDB is manipulated using the Programmer's
                        Data Manager (PDM) of RTProE.
# ----------------------------------------------------------------------
# Copyright (C) 1996 Dennis R. LaBelle (drlabell@albany.net) All Rights Reserved.
#
# PERMISSION IS GRANTED TO DISTRIBUTE THIS SOFTWARE FREELY, WITH THE
# EXCEPTION THAT ONE MAY NOT CHARGE FOR IT OR INCLUDE IT WITH SOFTWARE
# WHICH IS SOLD. Permission to use, copy, modify, and distribute this software and
# its documentation for educational, research, internal corporate and non-profit
# purposes, without fee, and without a written agreement is hereby granted for all
# cases that do not conflict with the restriction in the first sentence of this
# paragraph, provided that the above copyright notice, this paragraph, and the
# following three paragraphs appear in all copies.
#
# Permission to incorporate this software into commercial products may be obtained
# from the author (e-mail drlabell@albany.net).
#
# IN NO EVENT SHALL THE AUTHOR BE LIABLE TO ANY PARTY FOR DIRECT,
# INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF
# THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHOR HAS
# BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# THE AUTHOR SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS"
# BASIS, AND THE AUTHOR HAS NO OBLIGATION TO PROVIDE MAINTENANCE,
# SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
# ----------------------------------------------------------------------

*/

#ifndef PDBHEADER

#define PDBHEADER

#define PART_NONE      0
#define TYPE_NONE      0
#define TYPE_LOGICAL   1
#define TYPE_UCHAR     2
#define TYPE_SHORT     3
#define TYPE_LONG      4
#define TYPE_FLOAT     5
#define TYPE_DOUBLE    6
#define TYPE_GLOB_CON  7
#define TYPE_GLOB_IC   8
#define TYPE_GLOB_REG  9
#define TYPE_STRUCT    10

#define SIZE_NONE      0
#define SIZE_LOGICAL   1
#define SIZE_UCHAR     1
#define SIZE_SHORT     2
#define SIZE_LONG      4
#define SIZE_FLOAT     4
#define SIZE_DOUBLE    8
#define SIZE_GLOBAL    32

#define PTIDSIZE       16
#define BRIEFSIZE      32
#define FULLSIZE       360
#define SYSTEMSIZE     2
#define FORMATSIZE     5
#define UNITSIZE       6

#define MAXGLOBS       100

/* Run status values */
#define FREEZE                  0
#define RUN                             1
#define RESET                   3
#define TERM                    4
```

```
#define SYNC                    5

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <sys/types.h>
#include <malloc.h>
#include <search.h>
#include <sys/stat.h>


typedef struct          {
                                long addr;
                                long prevaddr;
                                long dim[7];            /* dimension info is stored in C order */
                                char ndim;
                                char ptid[PTIDSIZE + 1];
                                char partition;
                                char format[FORMATSIZE + 1];
                                char vartype;
                               } PDB;

typedef struct          {
                                char ptid[PTIDSIZE + 1];
                                char brief[BRIEFSIZE + 1];
                                char full[FULLSIZE + 1];
                                char sys[SYSTEMSIZE + 1];
                                char units[UNITSIZE + 1];
                               } PDBEXT;

typedef struct          {
                                long addr;
                                long size;
                                char ptid[PTIDSIZE + 1];
                                char partition;
                                char vartype;
                                char flag;
                               } ICHEADER;

typedef struct {
                                int         totglobs;
                                char signstr[16];
                        } SIGNHDR;

#ifdef MAINDEF
struct      {
                    long        size;
                    char *typestr;
                   } pdbtype[12]        = {
                                                0, "none",
                                                1, "char   logical",
                                                1, "uchar  byte",
                                                2, "short  integer*2",
                                                4, "long   integer*4",
                                                4, "float  real*4",
                                                8, "double real*8",
                                               32, "global_constant character*32",
                                               32, "global_IC character*32",
                                               32, "global    character*32",
                                               32, "global_IO character*32",
                                                8, "struct character*8"
                                           };

struct      {
                    long        size;
                    char *typestr;
                   } pdbtype_C[12]      = {
                                                0, "none",
                                                1, "char",
                                                1, "uchar",
                                                2, "short",
                                                4, "long",
                                                4, "float",
                                                8, "double",
```

```
                                                    32, "global_constant",
                                                    32, "global_IC",
                                                    32, "global",
                                                    32, "global_IO",
                                                     8, "struct"
                                                };

ICHEADER globinfo[MAXGLOBS];

#else
extern struct           {
                         int        size;
                         char *typestr;
                        } pdbtype[12];

extern struct           {
                         long       size;
                         char *typestr;
                        } pdbtype_C[12];

extern ICHEADER globinfo[MAXGLOBS];

#endif

#endif%{
/*
-------------------------------------------------------------------
```



```
-------------------------------------------------------------------

 hgen.l     : Lexical definitions for C header generation program of
                                    RTProE.

*/

int             totlines;
char toksval[256];

/*****************************************************************************/
%}
A               [Aa]
B               [Bb]
C               [Cc]
D               [Dd]
E               [Ee]
F               [Ff]
G               [Gg]
H               [Hh]
I               [Ii]
J               [Jj]
K               [Kk]
L               [Ll]
```

```
M       [Mm]
N       [Nn]
O       [Oo]
P       [Pp]
Q       [Qq]
R       [Rr]
S       [Ss]
T       [Tt]
U       [Uu]
V       [Vv]
W       [Ww]
X       [Xx]
Y       [Yy]
Z       [Zz]

cstart  "/*"
cend    "*/"
quote   "\""
leftp   "("
%s CODE COMMENT INSTRING PREPROC
%%
<CODE,PREPROC>{cstart}                  {
                                          BEGIN COMMENT;
                                        }

<CODE>{quote}                   {
                                          BEGIN INSTRING;
                                        }

<INSTRING>{quote}       {
                                          BEGIN CODE;
                                        }

<INSTRING>[^"]                  {
                                          ;
                                        }

<COMMENT>{cend}                 {
                                          BEGIN CODE;
                                        }

<CODE>[a-zA-Z][a-zA-Z0-9_]*     {
                                          strcpy(toksval, yytext);
                                          return( PTID );
                                        }

<CODE>[a-zA-Z][a-zA-Z0-9_]*[ ]*{leftp}          {
                                          strcpy(toksval, yytext);
                                          return( FUNCTION );
                                        }

<CODE>^#include                  {
                                          BEGIN PREPROC;
                                        }

<CODE>^#if                              {
                                          BEGIN PREPROC;
                                        }

<CODE>^#end                             {
                                          BEGIN PREPROC;
                                        }

<CODE>^#pragma                  {
                                          BEGIN PREPROC;
                                        }

<CODE>^#define                  {
                                          BEGIN PREPROC;
                                        }

<PREPROC>[\n]                   {
                                          BEGIN CODE;
                                          return EOL;
                                        }
```

```
<COMMENT,PREPROC>[^\n]                  {
                                                ;
                                               }

<COMMENT>[\n]                   {
                                                 return EOL;
                                               }

<CODE>[^ \t\n]                  {
                                                ;
                                               }

<CODE>[ \t]                                ;
<CODE>\n                        return EOL;

%%
/*****************************************************************************/
%{
/*
```

Copyright (C) 1996 Dennis R. LaBelle (drlabell@albany.net) All Rights Reserved.

PERMISSION IS GRANTED TO DISTRIBUTE THIS SOFTWARE FREELY, WITH THE EXCEPTION THAT ONE MAY NOT CHARGE FOR IT OR INCLUDE IT WITH SOFTWARE WHICH IS SOLD. Permission to use, copy, modify, and distribute this software and its documentation for educational, research, internal corporate and non-profit purposes, without fee, and without a written agreement is hereby granted for all cases that do not conflict with the restriction in the first sentence of this paragraph, provided that the above copyright notice, this paragraph, and the following three paragraphs appear in all copies.

Permission to incorporate this software into commercial products may be obtained from the author (e-mail drlabell@albany.net).

IN NO EVENT SHALL THE AUTHOR BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHOR SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE AUTHOR HAS NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

```
 hgen.y      : Grammar specification for C header generation program of
                                        RTProE.
*/

#include <stdio.h>
#include <stdlib.h>
#include <search.h>
#include <string.h>

#include "librt.h"
#include "hgen_l.c"


extern char toksval[];

extern int totlines;

#define RESERVED 32
char *resvd_words[RESERVED] = {
                                                        "auto",
                                                        "break",
                                                        "case",
                                                        "char",
                                                        "const",
                                                        "continue",
                                                        "default",
                                                        "do",
                                                        "double",
```

```
                                                "else",
                                                "enum",
                                                "extern",
                                                "float",
                                                "for",
                                                "goto",
                                                "if",
                                                "int",
                                                "long",
                                                "register",
                                                "return",
                                                "short",
                                                "signed",
                                                "sizeof",
                                                "static",
                                                "struct",
                                                "switch",
                                                "typedef",
                                                "union",
                                                "unsigned",
                                                "void",
                                                "volatile",
                                                "while"
                                        };

#define PTIDSIZE   16
#define MAXPTID 50000

PDB varinfo[MAXPTID + 1];

long int nextptid;
int                     allglobs;               /* set to non-zero to generate full set of global definitions */

%}
%token EOL
%token PTID
%token STARTCMT
%token ENDCMT
%token DONOTHING
%token FUNCTION
%%
file    : /* empty */
        | file plist EOL
          {
            ++totlines;
          }
        | file EOL
          {
            ++totlines;
          }
        ;

plist   : plist pitem
                | pitem
        ;

pitem   : ptid
                | FUNCTION
                | DONOTHING
        ;

ptid    : PTID
          {
            int i;

                /* check whether item is a reserved word */
                for (i = 0; i < RESERVED; ++i)
                   if (!stricmp(resvd_words[i],toksval))
                                break;
                if (i >= RESERVED)
                  { /* not a reserved word, so check if already in list */
                        strncpy(varinfo[nextptid].ptid, toksval, PTIDSIZE);
                        varinfo[nextptid].ptid[PTIDSIZE] = 0;
                        for (i = 0; i < nextptid ; ++i)
                        if (!stricmp(varinfo[i].ptid, varinfo[nextptid].ptid))
```

```
                                        break;
                                if (i >= nextptid)
                                        { /* item not in list yet, so add it */
                                          ++nextptid;
                                        }
                        }
                }
                ;
%%
void *getglobaddr(char vname)
{
  return NULL;
}

int make_getglobaddr()
{ /* This function generates source code to retrieve global partition addresses */
  int  i;
  char globname[PTIDSIZE + 1];

  for (i = 0; i < maxrec; ++i)
                if (            (pdbrec[i].vartype == TYPE_GLOB_CON)
                                ||      (pdbrec[i].vartype == TYPE_GLOB_IC)
                                ||      (pdbrec[i].vartype == TYPE_GLOB_REG)
                        )
                        {
                          strcpy(globname, pdbrec[i].ptid);
                          rt_strupr(globname);
                          printf("_declspec(dllimport) char %s[%d];\n",
                                        globname, rt_gettotsize(&pdbrec[i]));
                        }
  printf("\n");
  printf("\nvoid *getglobaddr(char partition)\n");
  printf("{\n");
  for (i = 0; i < maxrec; ++i)
                if (            (pdbrec[i].vartype == TYPE_GLOB_CON)
                                ||      (pdbrec[i].vartype == TYPE_GLOB_IC)
                                ||      (pdbrec[i].vartype == TYPE_GLOB_REG)
                        )
                        {
                          strcpy(globname, pdbrec[i].ptid);
                          rt_strupr(globname);
                          printf("  if (partition == %d) return %s;\n",
                                        pdbrec[i].partition, globname);
                        }
  printf("  return NULL;\n");
  printf("}\n");
  printf("\n");
}


int compare_addr(const void *arg1, const void *arg2)
{
  PDB *rec1;
  PDB *rec2;

  rec1 = arg1;
  rec2 = arg2;
  if (rec1->partition < rec2->partition)
                return -1;
  if (rec1->partition > rec2->partition)
                return 1;
  if (rec1->addr < rec2->addr)
                return -1;
  if (rec1->addr > rec2->addr)
                return 1;
  return 0;
}


void create_header()
{
  PDB   *record;
  int           i;
  int           dummynum;
  int           dummysize;
```

```
int         nextaddr;
int         pos;
char        lastpart;
char        globname[PTIDSIZE + 1];

/* retrieve variable info from PDB */
for (i = 0; i < nextptid; ++i)
        {
    record = rt_getrecord(varinfo[i].ptid);
                if (record)
                        memcpy(&varinfo[i], record, sizeof(PDB));
        }

/* sort variable info by partition and address */
qsort((const void*)varinfo, nextptid, sizeof(PDB), compare_addr);

/* print out operating system specific declarations */
printf("#pragma data_seg(\"SHAREDAT1\")\n");
printf("\n");

/* print out structure definitions */
lastpart = 0;
for (pos = 0; pos < nextptid; ++pos)
        {
          if (varinfo[pos].partition != 0)
                  {
                        if (varinfo[pos].partition != lastpart)
                                {
                                  if (lastpart != 0)
                                          { /* finish off structure definition for last partition */

                                            /* pad structure to full size of partition */
                                            dummysize = rt_getglobsize(lastpart) - nextaddr;
                                            if (dummysize > 0)
                                                    printf("                        char        dummy%d[%d];\n", dummynum, dummysize);

                                            strcpy(globname, rt_getglobname(lastpart));
                                            rt_strupr(globname);
                                            printf("                        } %s", globname);
                                            if (allglobs)
                                                    printf(" = {0}");
                                            printf(";\n");
                                            printf("\n");
                                            rt_setglobflag(lastpart);
                                          }
                                  /* start structure definition for new partition */
                                  printf("__declspec (dllexport) struct {\n");          /* OS specific */
                                  lastpart = varinfo[pos].partition;
                                  nextaddr = 0;
                                  dummynum = 0;
                                }
                        if (varinfo[pos].addr > nextaddr)
                                {
                                  dummysize = varinfo[pos].addr - nextaddr;
                                  printf("                        char        dummy%d[%d];\n", dummynum, dummysize);
                                  ++dummynum;
                                  nextaddr += dummysize;
                                }
                        printf("                        ");
                        switch (varinfo[pos].vartype)
                                {
                                        case TYPE_NONE:
                                                printf("void        ");
                                                break;
                                        case TYPE_LOGICAL:
                                                printf("char        ");
                                                break;
                                        case TYPE_UCHAR:
                                                printf("unsigned char ");
                                                break;
                                        case TYPE_SHORT:
                                                printf("short int    ");
                                                break;
                                        case TYPE_LONG:
                                                printf("long  int    ");
```

```
                                                        break;
                                                case TYPE_FLOAT:
                                                        printf("float       ");
                                                        break;
                                                case TYPE_DOUBLE:
                                                        printf("double      ");
                                                        break;
                                                case TYPE_GLOB_CON:
                                                        printf("void        ");
                                                        break;
                                                case TYPE_GLOB_IC:
                                                        printf("void        ");
                                                        break;
                                                case TYPE_GLOB_REG:
                                                        printf("void        ");
                                                        break;
                                                case TYPE_STRUCT:
                                                        printf("void        ");
                                                        break;
                                        }
                                printf("%s", varinfo[pos].ptid);
                                if (varinfo[pos].ndim)
                                        {
                                         for (i = 0; i < varinfo[pos].ndim; ++i)
                                                {
                                                        printf("[%d]", varinfo[pos].dim[i]);
                                                }
                                        }
                                printf(";\n");
                                nextaddr += rt_gettotsize(&varinfo[pos]);
                        }
                }

if (lastpart != 0)
        { /* finish off structure definition for last partition */

         /* pad structure to full size of partition */
         dummysize = rt_getglobsize(lastpart) - nextaddr;
         if (dummysize > 0)
                printf("                        char        dummy%d[%d];\n", dummynum, dummysize);

         strcpy(globname, rt_getglobname(lastpart));
         rt_strupr(globname);
         printf("                } %s", globname);
         if (allglobs)
                printf(" = {0}");
         printf(";\n");
         printf("\n");
         rt_setglobflag(lastpart);
        }

/* print out definitions for non-used globals, if requested */
if (allglobs)
        {
                for (i = 0; i < totpart; ++i)
                        if (!globinfo[i].flag)
                          {
                                strcpy(globname, globinfo[i].ptid);
                                rt_strupr(globname);
                                printf("__declspec (dllexport) char %s[%d] = {0};\n",
                                                globname, globinfo[i].size);
                          }
                printf("\n");
        }

/* print out operating system specific declarations */
printf("#pragma data_seg()\n");

/* print out global information structure, if requested */
if (allglobs)
        {
                printf("\n");
                printf("#define TOTPART             %d\n", totpart);
                printf("\n");
                printf("ICHEADER rt_globinfo[TOTPART] = {\n");
```

```
                lastpart = totpart - 1;
                for (i = 0; i < totpart; ++i)
                        {
                         strcpy(globname, globinfo[i].ptid);
                         rt_strupr(globname);
                         printf("                        {(long)");
                         if (globinfo[i].flag)
                           printf("&");
                         else
                           printf(" ");
                         printf("%s, %ld, \"%s\", %d, %d, %d}",
                                                        globname,
                                                        globinfo[i].size,
                                                        globname,
                                                        globinfo[i].partition,
                                                        globinfo[i].vartype,
                                                        0);
                         if (i != lastpart)
                           printf(",");
                         printf("\n");
                        }
                printf("                        };\n");
        }

/* print out #defines for individual variables */
printf("\n");
for (pos = 0; pos < nextptid; ++pos)
        {
         if (varinfo[pos].partition != 0)
                {
                        strcpy(globname, rt_getglobname(varinfo[pos].partition));
                        rt_strupr(globname);
                  printf("#define %-*s  %s.%s\n",
                                                PTIDSIZE,
                                                varinfo[pos].ptid,
                                                globname,
                                                varinfo[pos].ptid);
                }
        }

 return;
}

void init()
{
 char *cptr;

 totlines = 0;
 nextptid = 0;

 cptr = getenv("ODSPATH");
 if (cptr == NULL)
   strcpy(odspath, ".");
 else
   strcpy(odspath, cptr);
 pdbrec = rt_makelocalpdb();
 if (pdbrec == NULL)
   {
                printf("Unable to read %s/pdb.dat file into memory.", odspath);
                exit(1);
        }
}


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

 init();

 /* process command line arguments */
 if (argc > 1)
   {
        for (i = 1; i < argc; ++i)
```

```
                {
                        if (!strcmp(argv[i],"-g"))
                                { /* generate getglobaddr function source code
                                        for data server
                                  */
                                  make_getglobaddr();
                                    return 0;
                                  }
                        if (!strcmp(argv[i],"-f"))
                                { /* 1) generate full global listing with structure
                                                initialization
                                        2) generate initial condition load and save code
                                        3) generate code for loading constant data
                                  */
                                  allglobs = 1;
                                  }
                }
        }

 BEGIN CODE;
 result = yyparse();
 create_header();
 return result;
}
%{
/*
```

```
 igen.l       : Lexical definitions for FORTRAN include file generation
                                program of RTProE.
*/

int             totlines;
int  lastype;
int             paren;

/*****************************************************************************/
%}
A               [Aa]
B               [Bb]
C               [Cc]
D               [Dd]
E               [Ee]
F               [Ff]
G               [Gg]
H               [Hh]
I               [Ii]
J               [Jj]
K               [Kk]
L               [Ll]
```

```
M       [Mm]
N       [Nn]
O       [Oo]
P       [Pp]
Q       [Qq]
R       [Rr]
S       [Ss]
T       [Tt]
U       [Uu]
V       [Vv]
W       [Ww]
X       [Xx]
Y       [Yy]
Z       [Zz]

cstart  "!"
cend    "\n"
quote   "\""
leftp   "("
rightp  ")"
sp      " "
star    "*"
%s CODE COMMENT GETARG INFDEF LABEL INSTRING
%%
<CODE>{quote}                                   {
                                                  BEGIN INSTRING;
                                                }

<INSTRING>{quote}                               {
                                                  BEGIN CODE;
                                                }

<INSTRING>[^\"]                                 {
                                                  ;
                                                }

<CODE>{cstart}                                  {
                                                  BEGIN COMMENT;
                                                }

<CODE>^{sp}{sp}{sp}{sp}{sp}[^ \t]   {
                                                  if (lastype != 0)
                                                          curtype = lastype;
                                                }

<CODE>^[0-9]+                                   {
                                                  return GOTOLABEL;
                                                }

<CODE>^[^0-9 \t\n][^\n]*                        {
                                                  BEGIN COMMENT;
                                                }

<CODE>{F}{U}{N}{C}{T}{I}{O}{N}          {
                                                  BEGIN INFDEF;
                                                  return FUNCTION;
                                                }

<CODE>{S}{U}{B}{R}{O}{U}{T}{I}{N}{E}    {
                                                  BEGIN INFDEF;
                                                  return SUBROUTINE;
                                                }

<CODE>{C}{H}{A}{R}{A}{C}{T}{E}{R}{      
                                                  return T_LOGICAL;
                                                }

<CODE>{C}{O}{M}{P}{L}{E}{X}             {
                                                  return T_FLOAT;
                                                }

<CODE>{L}{O}{G}{I}{C}{A}{L}             {
                                                  return T_LOGICAL;
                                                }
```

```
<CODE>{I}{N}{T}{E}{G}{E}{R}{star}1 {
                                                    return T_UCHAR;
                                                  }

<CODE>{I}{N}{T}{E}{G}{E}{R}{star}2 {
                                                    return T_SHORT;
                                                  }

<CODE>{I}{N}{T}{E}{G}{E}{R}{star}4 {
                                                    return T_LONG;
                                                  }

<CODE>{I}{N}{T}{E}{G}{E}{R}{star}  {
                                                    return T_LONG;
                                                  }

<CODE>{R}{E}{A}{L}{star}4          {
                                                    return T_FLOAT;
                                                  }

<CODE>{R}{E}{A}{L}{star}8          {
                                                    return T_DOUBLE;
                                                  }

<CODE>{R}{E}{A}{L}{star}           {
                                                    return T_FLOAT;
                                                  }

<CODE>[^ \t\n]                                {
                                                    ;
                                                  }

<CODE>[ \t]                                        {
                                                    ;
                                                  }

<CODE>\n                                      {
                                                    lastype = curtype;
                                                    curtype = 0;
                                                    return EOL;
                                                  }

<COMMENT>{cend}                               {
                                                    BEGIN CODE;
                                                    ++totlines;
                                                  }

<CODE,GETARG,INFDEF>[a-zA-Z][a-zA-Z0-9_]*  {
                                                    strcpy(yylval.string, yytext);
                                                    return( PTID );
                                                  }

<INFDEF>{leftp}                               {
                                                    BEGIN GETARG;
                                                    return PAREN_LEFT;
                                                  }

<GETARG>{rightp}                          {
                                                    ++nextfarg;
                                                    BEGIN CODE;
                                                    return PAREN_RIGHT;
                                                  }

<GETARG>,                                         {
                                                    ++nextfarg;
                                                  }

<GETARG>[ \t]*\n                          {
                                                    ++totlines;
                                                  }

<INFDEF>[ \t]*\n                          {
                                                    BEGIN CODE;
                                                    return EOL;
                                                  }
```

```
<COMMENT,GETARG,INFDEF>[^\n]            {
                                                                    ;
                                                                    }

%%
/*****************************************************************/
%{
/*
```



```
 igen.y      : Grammar specification for FORTRAN include file generation
                                    program of RTProE.

*/


#include "librt.h"

#define strncasecmp strnicmp
#define strcasecmp stricmp

extern int paren;
extern int totlines;

#define  PTIDSIZE       16
#define  MAXARGS        50
#define  MAXPTID        50000
#define  MAXLOCAL       1000
#define  MAXFUNC        1000
#define  LINE_LEN       4096

struct {
                char ptid[PTIDSIZE + 1 + 20];
        } src_name[MAXPTID + 1];

struct {
                int  vartype;
                char defined;
                char ptid[PTIDSIZE + 1];
        } func_args[MAXFUNC + 1] = {0};

struct {
                int  vartype;
                char ptid[PTIDSIZE + 1];
        } local_args[MAXLOCAL + 1];


typedef struct {
                                char  name[PTIDSIZE + 1];
```

```
                                        char  vartype[MAXARGS];
                                        char  totargs;
                                } arg_rec;

arg_rec curargs = {0};   /* actual arguments for function or subroutine currently under examination */
arg_rec curfunc = {0};   /* required arguments for function or subroutine */
int                      argsfound;             /* number of arguments for func/sub under examination */
char  *progname;                 /* program name */
long        nextptid  = 0;       /* total number of variables found */
long        nextfarg  = 0;       /* total number of local function arguments found */
long        nextlocal = 0;       /* total number of local arguments found */
int                      allglobs;              /* set to non-zero to generate full set of global definitions */
int                      curtype;               /* las seen variable definition type */
char        lcl_partition;/* set to non-zero to consider non-saved globals as local */
char        subname[100];        /* name of current subroutine or function */
char        line[LINE_LEN]; /* input line buffer */
char        wordbuf[LINE_LEN];   /* word buffer */
PDB varinfo[MAXPTID + 1]; /* information structure for variables found */

%}
%union      {
                                        int             ival;
                                        float fval;
                                        char  string[512];
                        }

%token CHKARGS
%token COMMA
%token ENDCMT
%token EOL
%token FUNCNAME
%token FUNCTION
%token GOTOLABEL
%token PAREN_LEFT
%token PAREN_RIGHT
%token <string> PTID
%token STARTCMT
%token <string> SUBARG
%token SUBROUTINE
%token T_LOGICAL
%token T_UCHAR
%token T_SHORT
%token T_LONG
%token T_FLOAT
%token T_DOUBLE

%type  <string> varname

%%
file        : /* empty */
            | file plist EOL
              {
                ++totlines;
              }
            | file EOL
              {
                ++totlines;
              }
            ;

plist       : plist pitem
                        | pitem
            ;

pitem       : chkargs
                        | func
                        | ptid
                        | sub
                        | vtype
                        | GOTOLABEL
            ;

arglist     : arglist PTID
                          {
                            strcpy(func_args[nextfarg].ptid, $2);
```

```
                        func_args[nextfarg].defined = 0;
                }
                | PTID
                {
                  strcpy(func_args[nextfarg].ptid, $1);
                        func_args[nextfarg].defined = 0;
                }
        ;

chkargs : CHKARGS
                {
                  if (stricmp(curfunc.name, subname))
                        {
                                ++argsfound;
                        }
                }
        ;

func    : FUNCTION PTID PAREN_LEFT arglist PAREN_RIGHT
                {
                        strcpy(subname, $2);
                }
                | FUNCTION PTID
                {
                        strcpy(subname, $2);
                }
        ;

sub             : SUBROUTINE PTID PAREN_LEFT arglist PAREN_RIGHT
                {
                        strcpy(subname, $2);
                }
                | SUBROUTINE PTID
                {
                        strcpy(subname, $2);
                }
        ;

vtype   : T_LOGICAL
                {
                        curtype = TYPE_LOGICAL;
                }
                | T_UCHAR
                {
                        curtype = TYPE_UCHAR;
                }
                | T_SHORT
                {
                        curtype = TYPE_SHORT;
                }
                | T_LONG
                {
                        curtype = TYPE_LONG;
                }
                | T_FLOAT
                {
                        curtype = TYPE_FLOAT;
                }
                | T_DOUBLE
                {
                        curtype = TYPE_DOUBLE;
                }
        ;

ptid    : varname
                {
                        int i;

                        for (i = 0; i < nextfarg; ++i)
                                if (!stricmp(func_args[i].ptid, $1))
                                        {
                                          if (!func_args[i].defined)
                                                {
                                                        func_args[i].vartype = curtype;
                                                        func_args[i].defined = 1;
```

```
                                        }
                                        break;
                                    }
                        if (i >= nextfarg)
                                { /* probably a variable, save it */
                                  strncpy(src_name[nextptid].ptid, $1, sizeof(src_name[0].ptid));
                                  src_name[nextptid].ptid[sizeof(src_name[0].ptid)] = 0;
                                  strncpy(varinfo[nextptid].ptid, src_name[nextptid].ptid, PTIDSIZE);
                                  varinfo[nextptid].ptid[PTIDSIZE] = 0;
                                  rt_strlwr(varinfo[nextptid].ptid);
                                  varinfo[nextptid].prevaddr = nextptid;
                                  for (i = 0; i < nextptid; ++i)
                                        if (!stricmp(varinfo[i].ptid, varinfo[nextptid].ptid))
                                            break;
                                  if (i >= nextptid)
                                        { /* item not yet in list, keep checking */
                                          if (curtype != 0)
                                                {
                                                    strcpy(local_args[nextlocal].ptid, varinfo[nextptid].ptid);
                                                    local_args[nextlocal].vartype = curtype;
                                                    ++nextlocal;
                                                }
                                          else
                                                ++nextptid;
                                        }
                                }
                        }
                ;

varname     : PTID
                        {
                                strcpy($$, $1);
                        }
                ;
%%
#include <string.h>
#include "igen_l.c"

void *getglobaddr(char vname)
{
  return NULL;
}

int make_getglobaddr()
{ /* This function generates source code to retrieve global partition addresses */
  int i;

  for (i = 0; i < maxrec; ++i)
                if (            (pdbrec[i].vartype == TYPE_GLOB_CON)
                                ||      (pdbrec[i].vartype == TYPE_GLOB_IC)
                                ||      (pdbrec[i].vartype == TYPE_GLOB_REG)
                        )
                        {
                          printf("_declspec(dllimport) char %s[%d];\n",
                                        pdbrec[i].ptid, rt_getlotsize(&pdbrec[i]));
                        }
  printf("\n");
  printf("\nvoid *getglobaddr(char partition)\n");
  printf("{\n");
  for (i = 0; i < maxrec; ++i)
                if (            (pdbrec[i].vartype == TYPE_GLOB_CON)
                                ||      (pdbrec[i].vartype == TYPE_GLOB_IC)
                                ||      (pdbrec[i].vartype == TYPE_GLOB_REG)
                        )
                        {
                          printf("  if (partition == %d) return %s;\n",
                                        pdbrec[i].partition, pdbrec[i].ptid);
                        }
  printf("  return NULL;\n");
  printf("}\n");
  printf("\n");
}


int compare_addr(const void *arg1, const void *arg2)
```

```
{
 PDB *rec1;
 PDB *rec2;

 rec1 = arg1;
 rec2 = arg2;
 if (rec1->partition < rec2->partition)
        return -1;
 if (rec1->partition > rec2->partition)
        return 1;
 if (rec1->addr < rec2->addr)
        return -1;
 if (rec1->addr > rec2->addr)
        return 1;
 return 0;
}


void create_header()
{
 PDB   *record;
 int        i;
 int        dummynum;
 int        dummysize;
 int        nextaddr;
 int        pos;
 char       lastpart;
 char       globname[PTIDSIZE + 1];
 char       eqvstr[100];
 int        needeqv;

 /* retrieve variable info from PDB */
 for (i = 0; i < nextptid; ++i)
        {
    record = rt_getrecord(varinfo[i].ptid);
                if (record)
                  if (record->vartype < TYPE_GLOB_CON)
                        {
                                pos = varinfo[i].prevaddr;
                                memcpy(&varinfo[i], record, sizeof(PDB));
                                varinfo[i].prevaddr = pos;
                        }
        }

 /* sort variable info by partition and address */
 qsort((const void*)varinfo, nextptid, sizeof(PDB), compare_addr);

 /* print out COMMON and EQUIVALENCE statements */
 printf("\n");
 lastpart = 0;
 for (pos = 0; pos < nextptid; ++pos)
        {
          if (varinfo[pos].partition != 0)
                {
                        needeqv = 1;
                        if (lcl_partition && (rt_getpartype(varinfo[pos].partition) == TYPE_GLOB_REG))
                                needeqv = 0;
                        if (varinfo[pos].partition != lastpart)
                                {
                                  if (needeqv)
                                        { /* print out COMMON block definition */
                                          strcpy(globname, rt_getglobname(varinfo[pos].partition));
                                          rt_strupr(globname);
                                          printf("      character*1 %sX(%d)\n", globname,
                                                                        rt_getglobsize(varinfo[pos].partition));
                                          printf("      COMMON /%s/%sX\n", globname, globname);
                                          printf("\n");
                                          lastpart = varinfo[pos].partition;
                                        }
                                }

                        sprintf(eqvstr, "      EQUIVALENCE(%s", src_name[varinfo[pos].prevaddr].ptid);
                        printf("      ");           /* 6 spaces of initial indent for FORTRAN */
                        switch (varinfo[pos].vartype)
                                {
```

```
                                        case TYPE_LOGICAL:
                                                printf("logical*1   ");
                                                break;
                                        case TYPE_UCHAR:
                                                printf("integer*1   ");
                                                break;
                                        case TYPE_SHORT:
                                                printf("integer*2   ");
                                                break;
                                        case TYPE_LONG:
                                                printf("integer*4   ");
                                                break;
                                        case TYPE_FLOAT:
                                                printf("real*4      ");
                                                break;
                                        case TYPE_DOUBLE:
                                                printf("real*8      ");
                                                break;
                                        default:
                                                printf("UNKNOWN     ");
                                                break;
                                }
                        printf("%s", src_name[varinfo[pos].prevaddr].ptid);
                        if (varinfo[pos].ndim)
                                {
                                 for (i = 0; i < varinfo[pos].ndim; ++i)
                                        { /* print dimensions */
                                                if (i == 0)
                                                 {
                                                        printf("(%d", varinfo[pos].dim[i]);
                                                        strcat(eqvstr, "(1");
                                                 }
                                                else
                                                 {
                                                        printf(",%d", varinfo[pos].dim[i]);
                                                        strcat(eqvstr, ",1");
                                                 }
                                        }
                                 printf(")");  /* Close parentheses */
                                 strcat(eqvstr, ")");
                                }
                        printf("\n");
                        if (needeqv)
                                printf("%s, %sX(%d))\n",
                                                eqvstr,
                                                rt_getglobname(varinfo[pos].partition),
                                                varinfo[pos].addr + 1);
                        printf("\n");
                        }
                }
 return;
}

void init()
{
 char *cptr;

 totlines = 0;
 nextptid = 0;

 cptr = getenv("ODSPATH");
 if (cptr == NULL)
   strcpy(odspath, ".");
 else
   strcpy(odspath, cptr);
 pdbrec = rt_makelocalpdb();
 if (pdbrec == NULL)
   {
                printf("Unable to read %s/pdb.dat file into memory.", odspath);
                exit(1);
        }
}


main(int argc, char *argv[])
```

```
{
 int result;
 int i;

 /* process command line arguments */
 if (argc > 1)
   {
            for (i = 1; i < argc; ++i)
                     {
                              if (!strcmp(argv[i],"-l"))
                                       { /* Consider non-saved globals as local only
                                         */
                                 lcl_partition = 1;
                                       }
                     }
           }

 init();
 BEGIN CODE;
 result = yyparse();
 create_header();
 return 0;
}
%{
/*
```

---

pdm.l - LEX input for for Programmer's Database Manager (PDM) of RTProE

```
*/

extern int yylval;
extern int showmsg;
extern int showeom;

/* function prototypes */
void rl_strlwr(char *string);

char tokcval[256];
char toksval[256];
char *lexptr;
long tokival;
long itemp;

void eosel ()
{
 if (showmsg && showeom)
   {
            printf("\n@#eosel#@\n");
            fflush(stdout);
```

```
    }
}

int newgetc(FILE *stream)
{
 int c;

 c = getc(stream);
 if (stream != stdin)
    {
            if (feof(stream))
              {
                    fclose(yyin);
                    yyin = stdin;
                    eosel();
                    c = getc(yyin);
              }
    }
 return c;
}

/* Replacement for LEX input function.
   This code allows temporary reading from a file while in interactive mode.
 */
#ifdef YY_INPUT
#undef YY_INPUT
#endif
#define YY_INPUT(buf,result,max_size) \
                    { \
                    int c = newgetc( yyin ); \
                    result = c == EOF ? 0 : 1; \
                    buf[0] = (char) c; \
                    }

/*************************************************************************/
%}
A           [Aa]
B           [Bb]
C           [Cc]
D           [Dd]
E           [Ee]
F           [Ff]
G           [Gg]
H           [Hh]
I           [Ii]
J           [Jj]
K           [Kk]
L           [Ll]
M           [Mm]
N           [Nn]
O           [Oo]
P           [Pp]
Q           [Qq]
R           [Rr]
S           [Ss]
T           [Tt]
U           [Uu]
V           [Vv]
W           [Ww]
X           [Xx]
Y           [Yy]
Z           [Zz]
star "*"

%s GETCMD GETPTID GETSTR GETBRIEF GETFULL GETDIM GETTYPE
%%
<GETCMD>[/]?{A}{D}{D}                   {
                                                BEGIN GETPTID;
                                                return (ADD);
                                        }

<GETCMD>[/]?{D}{E}{L}           {
                                                BEGIN GETPTID;
                                                return (DEL);
                                        }
```

```
<GETCMD>.{D}{E}{S}{C}                    {
                                             BEGIN GETBRIEF;
                                             return (DESC);
                                         }

<GETCMD>.{D}{I}{M}                       {
                                             BEGIN GETDIM;
                                             return (DIM);
                                         }

<GETCMD>.{D}{O}{N}{E}                    {
                                             return (DONE);
                                         }

<GETCMD>{D}{U}{M}{P}                     {
                                             BEGIN GETPTID;
                                             return (DUMP);
                                         }

<GETCMD>{E}{O}{M}{S}?{G}?                {
                                             BEGIN GETPTID;
                                             return (EOMSG);
                                         }

<GETCMD>[/]?{E}{X}{I}{T}                 {
                                             return (EXIT);
                                         }

<GETCMD>[/]?{E}{S}{C}{A}?{P}?{E}?        {
                                             return (ESCAPE);
                                         }

<GETCMD>{F}{I}{N}{D}{F}{I}{R}{S}{T}      {
                                             BEGIN GETSTR;
                                             return (FINDFIRST);
                                         }

<GETCMD>.{F}{O}{R}{M}                    {
                                             BEGIN GETSTR;
                                             return (FORM);
                                         }

<GETCMD>[/]?{L}{M}{D}{D}                 {
                                             BEGIN GETPTID;
                                             return (LOOK);
                                         }

<GETCMD>[/]?{L}{O}{O}{K}                 {
                                             BEGIN GETPTID;
                                             return (LOOK);
                                         }

<GETCMD>[/]?{M}{O}{D}                    {
                                             BEGIN GETPTID;
                                             return (MOD);
                                         }

<GETCMD>[/]?{N}{E}{W}                    {
                                             return (NEW);
                                         }

<GETCMD>[/]?{P}{A}{C}{K}                 {
                                             return (PACK);
                                         }

<GETCMD>.{P}{R}{E}{D}                    {
                                             BEGIN GETPTID;
                                             return (PRED);
                                         }

<GETCMD>[/]?{W}{R}?{I}?{T}?{E}?{M}{O}{D}{E} {
                                             return (WRITEMODE);
                                         }

<GETCMD>[/]?{R}{E}?{A}?{D}?{M}{O}{D}{E}  {
```

```
                                                    return (READMODE);
                                                }

<GETCMD>[/]?{S}{E}{L}           {
                                                    BEGIN GETSTR;
                                                    return (SEL);
                                                }

<GETCMD>.{S}{Y}{S}[ \t]*                    {
                                                    BEGIN GETSTR;
                                                    return (SYSTEM);
                                                }

<GETCMD>\{T}{Y}{P}{E}               {
                                                    BEGIN GETTYPE;
                                                    return (TYPE);
                                                }

<GETCMD>[/]?{U}{N}{D}{E}{L}             {
                                                    BEGIN GETPTID;
                                                    return (UNDEL);
                                                }

<GETCMD>\.{U}{N}{I}{T}              {
                                                    BEGIN GETSTR;
                                                    return (UNITS);
                                                }

<GETCMD>.{V}{A}{L}{U}               {
                                                    return (VALUE);
                                                }

<GETTYPE>{C}{H}{A}{R}               {
                                                    return (T_LOGICAL);
                                                }

<GETTYPE>{L}{star}1                 {
                                                    return (T_LOGICAL);
                                                }

<GETTYPE>{U}{C}{H}{A}{R}        {
                                                    return (T_UCHAR);
                                                }

<GETTYPE>{I}{star}1                 {
                                                    return (T_UCHAR);
                                                }

<GETTYPE>{S}{H}{O}{R}{T}        {
                                                    return (T_SHORT);
                                                }

<GETTYPE>{I}{star}2                 {
                                                    return (T_SHORT);
                                                }

<GETTYPE>{I}{N}{T}                  {
                                                    return (T_LONG);
                                                }

<GETTYPE>{L}{O}{N}{G}               {
                                                    return (T_LONG);
                                                }

<GETTYPE>{I}{star}4                 {
                                                    return (T_LONG);
                                                }

<GETTYPE>{F}{L}{O}{A}{T}        {
                                                    return (T_FLOAT);
                                                }

<GETTYPE>{R}{star}4                 {
                                                    return (T_FLOAT);
                                                }
```

```
<GETTYPE>{D}{O}{U}{B}{L}{E}     {
                                                        return (T_DOUBLE);
                                                }

<GETTYPE>{R}{star}8                     {
                                                        return (T_DOUBLE);
                                                }

<GETTYPE>{G}{L}{O}{B}{A}{L}     {
                                                        return (T_GLOB_REG);
                                                }

<GETTYPE>{G}{L}{O}{B}{C}{O}{N}  {
                                                        return (T_GLOB_CON);
                                                }

<GETTYPE>{G}{L}{O}{B}{I}{C}     {
                                                        return (T_GLOB_IC);
                                                }

<GETTYPE>{S}{T}{R}{U}{C}{T}     {
                                                        return (T_STRUCT);
                                                }

<GETDIM>[ \t]*\[[ \t]*[0-9]+[ \t]*\]    {
                                                        sscanf(yytext," [ %d ] ",&tokival);
                                                        return (CINDEX);
                                                }

<GETDIM>[(,][ \t]*[0-9]+{
                                                        yytext[0] = ' ';
                                                        sscanf(yytext," %d",&tokival);
                                                        return (FINDEX);
                                                }

<GETPTID>[a-zA-Z][a-zA-Z0-9_]*  {
                                                        strcpy(toksval, tokcval);
                                                        strcpy(tokcval, yytext);
                                                        rt_strlwr(tokcval);
                                                        return (PTID);
                                                }

<GETPTID>[ \t]*[0-9]+[ \t]*             {
                                                        yytext[0] = ' ';
                                                        sscanf(yytext," %d",&tokival);
                                                        return (INTEGER);
                                                        }

<GETPTID>,[ \t]*[0-9]+[ \t]*            {
                                                        yytext[0] = ' ';
                                                        sscanf(yytext," %d",&tokival);
                                                        return (B_OFFSET);
                                                        }

<GETPTID>,[ \t]*[0-9]+[ \t]*{B}         {
                                                        yytext[0] = ' ';
                                                        sscanf(yytext," %d",&tokival);
                                                        return (B_OFFSET);
                                                        }

<GETPTID>,[ \t]*[0-9]+[ \t]*{H}         {
                                                        yytext[0] = ' ';
                                                        sscanf(yytext," %d",&tokival);
                                                        return (H_OFFSET);
                                                        }

<GETPTID>,[ \t]*[0-9]+[ \t]*{W}         {
                                                        yytext[0] = ' ';
                                                        sscanf(yytext," %d",&tokival);
                                                        return (W_OFFSET);
                                                        }

<GETSTR>[^\n]*                                  {
                                                        for (lexptr = yytext; isspace(*lexptr); ++lexptr);
```

```
                                                    strcpy(tokcval, lexptr);
                                                    return (STRING);
                                                }

<GETBRIEF>[^\n]*                        {
                                                    BEGIN GETFULL;
                                                    if (*yytext == ' ')
                                                      strcpy(tokcval, yytext + 1);      /* skip first space */
                                                    else
                                                      strcpy(tokcval, yytext);
                                                    return (STRING);
                                                }

<GETFULL>^[ \t]*[./]                    {
                                                    BEGIN GETCMD;
                                                    itemp = strlen(yytext);
                                                    ungetc(yytext[itemp - 1], yyin);
                                                }

<GETFULL>^[ \t]*[^./\n][^\n]*           {
                                                    strcpy(tokcval, yytext);
                                                    return (LONGDESC);
                                                }

<GETFULL>[\n]                           {
                                                    return EOL;
                                                }

<GETCMD,GETSTR,GETDIM,GETTYPE>\n        {
                                                    BEGIN GETCMD;
                                                    return EOL;
                                                }

<GETPTID>\n             {
                                                    BEGIN GETCMD;
                                                    return EOL;
                                                }

<GETCMD,GETPTID,GETTYPE,GETDIM>[^\n]    {
                                                    ;
                                                }

%%
/*****************************************************************************/
%{
/*
```

```
*/
#include "librt.h"
#include "pdm_l.c"

#define GLOBLIST "ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxyz0123456789"

extern char   odspath[];          /* path to Official Development System */
extern long   maxrec;                     /* the number of records in PDB */
extern PDB    *pdbrec;                    /* pointer to PDB array */
extern PDBEXT *pdbext;                            /* pointer to PDB extension array */

int           cancel = 0;                 /* this variable is set to non-zero to cancel all changes */
int           packit = 0;                 /* this variable is set to non-zero to remove deleted records */
int           showmsg = 1;                /* flag for controlling message display
                                                    1 = display messages
                                                    0 = silent mode, no messages
                                                */
int           showeom = 0;                /* flag for controlling end-of-message prompt
                                                    1 = display end-of-message prompt
                                                    0 = silent mode, no end-of-messages
                                                */
int           recsadd;                    /* number of new records added */
int           recsdel;                    /* number of records deleted  */
int           recsmod;                    /* number of records modified  */
int           recsundel;                  /* number of records undeleted */
int           totmods;                    /* number of data base modifications made */
PDB           newrec;                           /* record buffer for new record information */
PDBEXT  newext;                           /* record buffer for new extended information */
char    curptid[PTIDSIZE + 1];  /* current working variable name */
char    orgptid[PTIDSIZE + 1];  /* original name of renamed variable */
char    tempstr[256];                     /* temporary working string */
long    currec;                           /* index for currently modified record
                                                                        set to -1 when adding
                                                                        set to record number when modifying
                                                */
long    absaddr;                          /* absolute address
                                                                        set to -1 with auto address selection (normal)

                                                                        set to desired absolute address when forcing
                                                                        a variable to a specific location
                                                */
char    dbmode;                           /* access mode of data base
                                                                        R = read only mode
                                                                        W = write/read mode
                                                                        F = failure mode
                                                */

/* prototypes */
void  finish();
void  message(char *);
void  eomsg();
int   display(char *);
void *getrecord(char *ptid);
void *getrecext(char *ptid);
int          close_record(char *ptid);
void  showrec(PDB *record, PDBEXT *recext);
char assign_partition(char *globname);
void make_onemap(PDB *record,int i);
extern void eosel();

%}
%token ADD
%token B_OFFSET
%token CINDEX
%token DEL
%token DESC
%token DIM
%token DONE
%token DUMP
%token EOL
%token EOMSG
%token ESCAPE
%token EXIT
%token FINDEX
%token FINDFIRST
```

```
%token FORM
%token GLOBNAME
%token H_OFFSET
%token INTEGER
%token LOOK
%token LONGDESC
%token MOD
%token NEW
%token PACK
%token PRED
%token PTID
%token READMODE
%token REAL
%token SEL
%token STRING
%token SYSTEM
%token T_LOGICAL
%token T_UCHAR
%token T_SHORT
%token T_LONG
%token T_FLOAT
%token T_DOUBLE
%token T_GLOB_CON
%token T_GLOB_IC
%token T_GLOB_REG
%token T_STRUCT
%token TYPE
%token UNDEL
%token UNITS
%token VALUE
%token W_OFFSET
%token WRITEMODE
%%
file            : /* empty */
                        | file cmdlist EOL
                          {
                            ;
                          }
                        | file error EOL
                          {
                            ;
                          }
                        | file EOL
                          {
                            ;
                          }
                ;

cmdlist         : cmdlist cmditem
                        | cmditem
                ;

cmditem         : add
                        | del
                        | desc
                        | dim
                        | done
                        | dump
                        | eomsg
                        | escape
                        | exit
                        | find
                        | format
                        | ldesc
                        | look
                        | mod
                        | new
                        | pack
                        | pred
                        | rmode
                        | sel
                        | system
                        | type
                        | undel
                        | units
```

```
                | value
                | wmode
        ;

add             : ADD PTID
                 { /* Add a data base record
                   */
                   PDB *record;

                        if (curptid[0])
                          {
                            sprintf(tempstr,"Cannot ADD %s while working on %s", tokcval, curptid);
                                message(tempstr);
                          }
                        else
                          {
                            record = getrecord(tokcval);
                                if (record)
                                        {
                                         sprintf(tempstr,"%s already found in data base.", tokcval);
                                         message(tempstr);
                                        }
                                else
                                  {
                                  currec  = -1;
                                        absaddr = -1;

                                        strncpy(curptid, tokcval, PTIDSIZE);
                                        curptid[PTIDSIZE] = 0;
                                        sprintf(tempstr, "ADDing %s\n", curptid);
                                        message(tempstr);
                                        strcpy(newrec.ptid, curptid);
                                        strcpy(newext.ptid, curptid);

                                        /* fill in default values for new item */
                                        newrec.vartype   = TYPE_FLOAT;
                                        newrec.partition = rt_getparco("global01");
                                        newrec.format[0] = 0;
                                        newrec.addr      = 0;
                                        newrec.prevaddr  = 0;
                                        newrec.ndim      = 0;
                                        newrec.dim[0]    = 0;
                                        newrec.dim[1]    = 0;
                                        newrec.dim[2]    = 0;
                                        newrec.dim[3]    = 0;
                                        newrec.dim[4]    = 0;
                                        newrec.dim[5]    = 0;
                                        newrec.dim[6]    = 0;
                                        newext.brief[0]  = 0;
                                        newext.full[0]   = 0;
                                        newext.sys[0]    = 0;
                                        newext.units[0]  = 0;
                                       }
                          }
                  eomsg();
                 }
        ;

del             : DEL PTID
                 { /* Mark a data base record for deletion.  The prevaddr field is set to a
                              negative value to indicate the record has been marked for deletion.
                   */
                   PDB *record;

                        if (curptid[0])
                          {
                            sprintf(tempstr,"Cannot DEL %s while working on %s", tokcval, curptid);
                                message(tempstr);
                          }
                        else
                          {
                                record = getrecord(tokcval);
                                if (record)
                                        {
                                 record->prevaddr = -1; /* set prevaddr to -1 for deleted records */
```

```
                                        ++recsdel;
                                    }
                                else
                                    {
                                    sprintf(tempstr, "%s not found in data base.", tokcval);
                                        message(tempstr);
                                    }
                            }
                        eomsg();
                        }
                ;

desc    : DESC STRING
                { /* Specify brief description of data base item
                  */
                  if (curptid[0])
                    {
                      strncpy(newext.brief, tokcval, BRIEFSIZE);
                                newext.brief[BRIEFSIZE] = 0;
                                newext.full[0] = 0;       /* blank out long desc */
                            }
                }
                ;

dim             : DIM
                  { /* Specify dimensioning information for variable.
                            Dimensioning information is stored in the data base
                            record assuming a FORTRAN style ordering.
                    */
                    if (curptid[0])
                      newrec.ndim = 0;
                  }
                | DIM findex
                  {
                    if (curptid[0])
                            {
                                newrec.ndim = 1;
                                newrec.dim[0] = $2;
                            }
                  }
                | DIM findex findex
                  {
                    if (curptid[0])
                            {
                                newrec.ndim = 2;
                                newrec.dim[0] = $2;
                                newrec.dim[1] = $3;
                            }
                  }
                | DIM findex findex findex
                  {
                    if (curptid[0])
                            {
                                newrec.ndim = 3;
                                newrec.dim[0] = $2;
                                newrec.dim[1] = $3;
                                newrec.dim[2] = $4;
                            }
                  }
                | DIM findex findex findex findex
                  {
                    if (curptid[0])
                            {
                                newrec.ndim = 4;
                                newrec.dim[0] = $2;
                                newrec.dim[1] = $3;
                                newrec.dim[2] = $4;
                                newrec.dim[3] = $5;
                            }
                  }
                | DIM findex findex findex findex findex
                  {
                    if (curptid[0])
                            {
                                newrec.ndim = 5;
```

```
                newrec.dim[0] = $2;
                newrec.dim[1] = $3;
                newrec.dim[2] = $4;
                newrec.dim[3] = $5;
                newrec.dim[4] = $6;
        }
 }
| DIM findex findex findex findex findex findex
 {
  if (curptid[0])
        {
                newrec.ndim = 6;
                newrec.dim[0] = $2;
                newrec.dim[1] = $3;
                newrec.dim[2] = $4;
                newrec.dim[3] = $5;
                newrec.dim[4] = $6;
                newrec.dim[5] = $7;
        }
 }
| DIM findex findex findex findex findex findex findex
 {
  if (curptid[0])
        {
                newrec.ndim = 7;
                newrec.dim[0] = $2;
                newrec.dim[1] = $3;
                newrec.dim[2] = $4;
                newrec.dim[3] = $5;
                newrec.dim[4] = $6;
                newrec.dim[5] = $7;
                newrec.dim[6] = $8;
        }
 }
| DIM cindex
 {
  if (curptid[0])
        {
        newrec.ndim = 1;
                newrec.dim[0] = $2;
        }
 }
| DIM cindex cindex
 {
  if (curptid[0])
        {
        newrec.ndim = 2;
                newrec.dim[0] = $3;
                newrec.dim[1] = $2;
        }
 }
| DIM cindex cindex cindex
 {
  if (curptid[0])
        {
                newrec.ndim = 3;
                newrec.dim[0] = $4;
                newrec.dim[1] = $3;
                newrec.dim[2] = $2;
        }
 }
| DIM cindex cindex cindex cindex
 {
  if (curptid[0])
        {
                newrec.ndim = 4;
                newrec.dim[0] = $5;
                newrec.dim[1] = $4;
                newrec.dim[2] = $3;
                newrec.dim[3] = $2;
        }
 }
| DIM cindex cindex cindex cindex cindex
 {
  if (curptid[0])
```

```
                        {
                                newrec.ndim = 5;
                                newrec.dim[0] = $6;
                                newrec.dim[1] = $5;
                                newrec.dim[2] = $4;
                                newrec.dim[3] = $3;
                                newrec.dim[4] = $2;
                        }
                }
                | DIM cindex cindex cindex cindex cindex cindex
                {
                  if (curptid[0])
                        {
                                newrec.ndim = 6;
                                newrec.dim[0] = $7;
                                newrec.dim[1] = $6;
                                newrec.dim[2] = $5;
                                newrec.dim[3] = $4;
                                newrec.dim[4] = $3;
                                newrec.dim[5] = $2;
                        }
                }
                | DIM cindex cindex cindex cindex cindex cindex cindex
                {
                  if (curptid[0])
                        {
                                newrec.ndim = 7;
                                newrec.dim[0] = $8;
                                newrec.dim[1] = $7;
                                newrec.dim[2] = $6;
                                newrec.dim[3] = $5;
                                newrec.dim[4] = $4;
                                newrec.dim[5] = $3;
                                newrec.dim[6] = $2;
                        }
                }
        ;

done    : DONE
                { /* Indicate end of record entry
                  */
                  close_record(curptid);
                  eomsg();
                }
        ;

dump    : DUMP PTID
                {
                  dump(tokcval);
                  eomsg();
                }
                | DUMP
                {
                  dump("");
                  eomsg();
                }
        ;

cindex : CINDEX
                {
                  $$ = tokival;
                }
        ;

findex : FINDEX
                {
                  $$ = tokival;
                }
        ;

escape  : ESCAPE
                {
                  cancel = 1;
                        finish();
                }
```

```
        ;

eomsg           : EOMSG integer
                {
                  showeom = $2;
                }
        ;

exit    : EXIT
                {
                  finish();
                }
        ;

find    : FINDFIRST STRING
                {
                        int   i;
                        int   slen;

        slen = strlen(tokcval);
        rt_strlwr(tokcval);
                        for (i = 0; i < maxrec; ++i)
                          {
           if (!strncmp(pdbrec[i].ptid, tokcval, slen))
                                  {
                                          sprintf(tempstr, "%s", pdbrec[i].ptid);
                                          message(tempstr);
                                          break;
                                  }
                          }
                        eomsg();
                }
                | FINDFIRST
                {
                        sprintf(tempstr, "%s", pdbrec[0].ptid);
                        message(tempstr);
                        eomsg();
                }
        ;

format  : FORM STRING
                { /* Specify C-style format string for printing value of variable
                  */
                  if (curptid[0])
                          {
                                  strncpy(newrec.format, tokcval, FORMATSIZE);
                                  newrec.format[FORMATSIZE] = 0;
                          }
                }
        ;

ldesc   : LONGDESC
                { /* Add a line of text to the record's full description
                  */
                  int slen1;
                        int slen2;
                        int addlen;

                  if (curptid[0])
                          {
                                  slen1 = strlen(newext.full);
                                  slen2 = strlen(tokcval);
                                  addlen = FULLSIZE - slen1 - 1;
                                  if (addlen < slen2)
                                          tokcval[addlen] = 0;
                                  if (newext.full[0])
                                    sprintf(newext.full, "%s\n%s", newext.full, tokcval);
                                  else
                                    strcpy(newext.full, tokcval);
                          }
                }
        ;

look    : LOOK PTID
                { /* List data base information for the specified variable, PTID
```

```
                */
                        if (curptid[0])
                          {
                            sprintf(tempstr,"Cannot LOOK %s while working on %s", tokcval, curptid);
                                message(tempstr);
                          }
                        else
                          {
                                display(tokcval);
                          }
                        eomsg();
                }
        | LOOK integer
                {
                        display_recno($2);
                        eomsg();
                }
        ;

mod             : MOD PTID
                {       /* Modify the specified variable's record
                 */
                  PDB           *record;
                  PDBEXT *recext;

                        if (curptid[0])
                          {
                            sprintf(tempstr,"Cannot MOD %s while working on %s", tokcval, curptid);
                                message(tempstr);
                          }
                        else
                          {
                            record = getrecord(tokcval);
                                if (record)
                                        {
                                 currec  = record - pdbrec;
                                          absaddr = -1;
                                          memcpy(&newrec, record, sizeof(PDB));
                                          strcpy(curptid, newrec.ptid);
                                          recext = getrecext(tokcval);
                                          if (recext)
                                         {
                                                  memcpy(&newext, recext, sizeof(PDBEXT));
                                                  strcpy(curptid, newrec.ptid);
                                           sprintf(tempstr, "MODing %s\n", curptid);
                                                  message(tempstr);
                                                }
                                          else
                                         {
                                                  sprintf(tempstr, "%s not found in data base extension.", tokcval);
                                                }
                                        }
                                else
                                        {
                                          sprintf(tempstr, "%s not found in data base.", tokcval);
                                          message(tempstr);
                                        }
                                }
                        eomsg();
                }
                | MOD PTID PTID
                 {
                   PDB          *record;
                   PDBEXT *recext;

                        if (curptid[0])
                          {
                            sprintf(tempstr,"Cannot rename %s while working on %s", toksval, curptid);
                                message(tempstr);
                          }
                        else
                          {
                                record = getrecord(toksval);
                                if (record)
                                        {
```

```
                                        currec = record - pdbrec;
                                        memcpy(&newrec, record, sizeof(PDB));
                                        strcpy(curptid, newrec.ptid);
                                        recext = getrecext(toksval);
                                        if (recext)
                                          {
                                                memcpy(&newext, recext, sizeof(PDBEXT));
                                                strcpy(orgptid, newrec.ptid);
                                                strncpy(curptid, tokcval, PTIDSIZE);
                                                curptid[PTIDSIZE] = 0;
                                                strcpy(newrec.ptid, curptid);
                                                strcpy(newext.ptid, curptid);
                                                sprintf(tempstr, "Renaming %s to %s", orgptid, curptid);
                                                message(tempstr);
                                          }
                                        else
                                                {
                                                        sprintf(tempstr, "%s not found in data base extension.", tokcval);
                                                        message(tempstr);
                                                }
                                        }
                                else
                                        {
                                         sprintf(tempstr, "%s not found in data base.", tokcval);
                                         message(tempstr);
                                        }
                          }
                        eomsg();
                }
        ;

new             : NEW
                { /* Create a new, minimal set of data base files named
                        pdbnew.dat and pdbexnew.dat
                   */
                  int result;

                        if (curptid[0])
                          {
                            sprintf(tempstr,"Cannot create new data bases while working on %s", curptid);
                                message(tempstr);
                          }
                        else
                          {
                         result = rt_new("pdbnew.dat", "pdbexnew.dat");
                                switch (result)
                                {
                                        case 0: message("New data bases saved to files pdbnew.dat and pdbexnew.dat");
                                                                        break;
                                        case 1: message("Unable to create file pdbnew.dat");
                                                                        break;
                                        case 2: message("Unable to create file pdbexnew.dat");
                                                                        break;
                                }
                          }
                        eomsg();
                }
        ;

offset  : B_OFFSET
                { /* Pass 1-byte offset value back up the tree
                   */
                        $$ = tokival;
                }
                | H_OFFSET
                {       /* Convert half-word (2-byte) offset value to a 1-byte offset value
                   */
                        $$ = tokival * 2;
                }
                | W_OFFSET
                {       /* Convert word (4-byte) offset value to a 1-byte offset value
                   */
                        $$ = tokival * 4;
                }
        ;
```

```
pack        : PACK
                    { /* Remove records marked for deletion from the data base files
                     */
                            if (curptid[0])
                              {
                                sprintf(tempstr,"Cannot PACK data bases while working on %s", curptid);
                                        message(tempstr);
                              }
                            else
                              {
                             packit = 1;
                              }
                            eomsg();
                    }
            ;

pred        : PRED
                    {
                    }
                   | PRED PTID
                    { /* Specify the base location for the variable
                     */
                      char newpartition;

                      if (curptid[0])
                                {
                                newpartition = rt_getpartition(tokcval);
                                        if (!newpartition)
                                          if (       (newrec.vartype == TYPE_GLOB_CON)
                                                        || (newrec.vartype == TYPE_GLOB_IC)
                                                        || (newrec.vartype == TYPE_GLOB_REG)
                                                       )
                                                  {
                                                        newpartition = assign_partition(tokcval);
                                                        if (newpartition)
                                                                {
                                                                  globinfo[totpart].size = rt_gettotsize(&newrec);
                                                                  globinfo[totpart].partition = newpartition;
                                                                  strcpy(globinfo[totpart].ptid, tokcval);
                                                                  globinfo[totpart].vartype = newrec.vartype;
                                                                  globinfo[totpart].addr = 0;
                                                                  make_onemap(&newrec, totpart);
                                                                  ++totpart;
                                                                }
                                                        else
                                                                {
                                                                  sprintf(tempstr,"Unable to create global %s", tokcval);
                                                                  message(tempstr);
                                                                  curptid[0] = 0;
                                                                }
                                                  }
                                        if (newpartition)
                                                {
                                                  newrec.partition = newpartition;
                                                  close_record(curptid);
                                                }
                                        else
                                                {
                                                  sprintf(tempstr,"Invalid predecessor -> %s\n", tokcval);
                                                  message(tempstr);
                                                  curptid[0] = 0;
                                                }
                                }
                            eomsg();
                    }
                   | PRED PTID offset
                    {           /* Specify the base location and offset for the variable
                     */
                      unsigned char newpartition;

                      if (curptid[0])
                                {
                                        newpartition = rt_getpartition(tokcval);
                                        if (newpartition == PART_NONE)
```

```
                                        {
                                          sprintf(tempstr,"Invalid predecessor -> %s", tokcval);
                                          message(tempstr);
                                          curptid[0] = 0;
                                        }
                                else
                                        {
                                          absaddr = $3;
                                          newrec.partition = newpartition;
                                          close_record(curptid);
                                        }
                        }
                  eomsg();
                }
        ;

rmode   : READMODE
                {
                        dbmode = 'R';
                }

sel             : SEL
                 {
                 }
                | SEL STRING
                 { /* Read a file and execute the PDM commands that are contained in it.
                   */
                   FILE *fin;

                   fin = fopen(tokcval, "rt");
                   if (fin == NULL)
                        {
                          sprintf(tempstr,"Unable to open SELect file %s", tokcval);
                          message(tempstr);
                        }
                   else
                        {
                         sprintf(tempstr,"Processing SELect file %s", tokcval);
                         message(tempstr);
                         yyin = fin;
                        }
                   eomsg();
                   if (fin == NULL)
                        eosel();
                 }
        ;

system  : SYSTEM
                {
                }
                | SYSTEM STRING
                 { /* Specify the "system" the variable is associated with.
                   */
                   if (curptid[0])
                        {
                                stmcpy(newext.sys, tokcval, SYSTEMSIZE);
                                newext.sys[SYSTEMSIZE] = 0;
                        }
                 }

type    : typecmd
                 { /* The specify the variable's data type
                   */
                   if (curptid[0])
                        if (!newrec.format[0])
                          switch (newrec.vartype)
                                {
                                        case TYPE_LOGICAL:
                                                strcpy(newrec.format, "%d");
                                                break;
                                        case TYPE_UCHAR:
                                                strcpy(newrec.format, "%u");
                                                break;
                                        case TYPE_SHORT:
                                                strcpy(newrec.format, "%d");
```

```
                                                        break;
                                                case TYPE_LONG:
                                                        strcpy(newrec.format, "%d");
                                                        break;
                                                case TYPE_FLOAT:
                                                        strcpy(newrec.format, "%.5g");
                                                        break;
                                                case TYPE_DOUBLE:
                                                        strcpy(newrec.format, "%.5g");
                                                        break;
                                        }
                        }
        ;

typecmd : TYPE
                        {
                         ;
                        }
                        | TYPE T_LOGICAL
                        { /* 1 byte, integer */
                                newrec.vartype = TYPE_LOGICAL;
                        }
                        | TYPE T_UCHAR
                        {       /* 1 byte, unsigned integer */
                                newrec.vartype = TYPE_UCHAR;
                        }
                        | TYPE T_SHORT
                        { /* 2 byte, signed integer */
                                newrec.vartype = TYPE_SHORT;
                        }
                        | TYPE T_LONG
                        { /* 4 byte, signed integer */
                                newrec.vartype = TYPE_LONG;
                        }
                        | TYPE T_FLOAT
                        { /* 4 byte, floating point real */
                                newrec.vartype = TYPE_FLOAT;
                        }
                        | TYPE T_DOUBLE
                        { /* 8 byte, floating point real */
                                newrec.vartype = TYPE_DOUBLE;
                        }
                        | TYPE T_GLOB_CON
                        { /* 32 byte global partition */
                                newrec.vartype = TYPE_GLOB_CON;
                        }
                        | TYPE T_GLOB_IC
                        { /* 32 byte global partition */
                                newrec.vartype = TYPE_GLOB_IC;
                        }
                        | TYPE T_GLOB_REG
                        { /* 32 byte global partition */
                                newrec.vartype = TYPE_GLOB_REG;
                        }
                        | TYPE T_STRUCT
                        { /* 8 byte structure variable */
                                newrec.vartype = TYPE_STRUCT;
                        }
        ;

undel   : UNDEL PTID
                        { /* Mark a record for deletion.  The record is not removed until the PACK
                                command is issued.  The prevaddr field must be non-negative for a record
                                to be active.
                                */
                          PDB *record;

                                if (curptid[0])
                                  {
                                    sprintf(tempstr,"Cannot UNDEL %s while working on %s", tokcval, curptid);
                                    message(tempstr);
                                  }
                                else
                                  {
                                        record = getrecord(tokcval);
```

```
                                        if (record)
                                          {
                                                record->prevaddr = 0;  /* set prevaddr to 0 to undelete records */
                                                ++recsundel;
                                                sprintf(tempstr, "%s undeleted.", tokcval);
                                                message(tempstr);
                                               }
                                        else
                                                {
                                                 sprintf(tempstr, "%s not found in data base.", tokcval);
                                                 message(tempstr);
                                                }
                                }
                        eomsg();
                       }
            ;

units       : UNITS STRING
                       {
                        if (curptid[0])
                          {
                           strncpy(newext.units, tokcval, UNITSIZE);
                                        newext.units[UNITSIZE] = 0;
                           }
                       }
            ;

value       : VALUE real
                       {
                       }
                      | VALUE integer
                       {
                       }
            ;

wmode       : WRITEMODE
                       {
                                dbmode = 'W';
                       }

real                   : REAL
                       {
                       }
            ;

integer     : INTEGER
                       {
                                $$ = tokival;
                       }
            ;

%%
char *globmap[200];                     /* global partition usage maps */
long  nextrec;                                  /* index of next available PDB record */
long  lastrec;                                  /* index of last available PDB record */

void message(char *msg)
{
  if (showmsg)
    {
            printf("%s\n", msg);
            fflush(stdout);
    }
}

void eomsg ()
{
  if (showmsg && showeom)
    {
            printf("@#eomsg#@\n");
            fflush(stdout);
    }
}
```

```
void *getrecord(char *ptid)
{ /* This function returns a pointer to the PDB record having a PTID of ptid.

            Returns pointer value, on success.
            Returns NULL pointer , on failure
 */

 void *recptr;
 long  i;

 /* check original PDB */
 recptr = rt_getrecord(ptid);
 if (recptr == NULL)                    /* check newly added items */
   {
     for (i = maxrec; i < nextrec; ++i)
                        {
                if (!strcmp(ptid, pdbrec[i].ptid))
                                        return (void *)(pdbrec + i);
                        }
              return NULL;
            }
 else
    return recptr;
}


void *getrecext(char *ptid)
{ /* This function returns a pointer to the PDBEXT record having a PTID of ptid.

            Returns pointer value, on success.
            Returns NULL pointer , on failure
 */

 void *recptr;
 long  i;

 /* check original PDB */
 recptr = rt_getrecext(ptid);
 if (recptr == NULL)                    /* check newly added items */
   {
     for (i = maxrec; i < nextrec; ++i)
                  {
                if (!strcmp(ptid, pdbext[i].ptid))
                                                return (void *)(pdbext + i);
                        }
              return NULL;
            }
 else
    return recptr;
}


void *getglobaddr(char partition)
{
  return NULL;
}


char assign_partition(char *globname)
{  /* This function assigns a partition code for a new global

            Returns    On success, the newly assigned partition code
                                                On failure, 0
  */
  char glist[100];
  int i;
  int slen;
  int pos;

  glist[0] = 0;
  for (i = 0; i < totpart; ++i)
     {
                glist[i] = globinfo[i].partition;
              }
  glist[i] = 0;
```

```
  slen = strlen(GLOBLIST);
  for (i = 0; i < slen; ++i)
    if (!strchr(glist, GLOBLIST[i]))
                        {
                return GLOBLIST[i];
                        }
  return 0;
}


void make_onemap(PDB *record, int i)
{
  int  index;
  long  recno;
  long  size;
  long  itemsize;
  char *cptr;

        size = rt_gettotsize(record);
        globinfo[i].size = size;
        globmap[i] = malloc(size + 1);
        if (globmap[i] == NULL)
          {
                printf("Unable to allocate memory for map of %s.\n", record->ptid);
                eomsg();
                exit(2);
          }
        strcpy(globinfo[i].ptid, record->ptid);
        memset(globmap[i], 'a', size);          /* mark all items available */
        globmap[i][size] = 0;                           /* terminate buffer with zero byte */
        for (recno = 0; recno < maxrec; ++recno)
                if (   (pdbrec[recno].partition == globinfo[i].partition)
                        && (pdbrec[recno].vartype != TYPE_GLOB_CON)
                        && (pdbrec[recno].vartype != TYPE_GLOB_IC)
                        && (pdbrec[recno].vartype != TYPE_GLOB_REG)
                 )
            {
                        cptr = globmap[i] + pdbrec[recno].addr;
                        itemsize = pdbtype[pdbrec[recno].vartype].size;
                        for (index = 0; index < pdbrec[recno].ndim; ++index)
                                {
                                        itemsize *= pdbrec[recno].dim[index];
                          }
                        memset(cptr, ' ', itemsize); /* mark memory space as used */
                }
}


void makemaps()
{ /* This function constructs maps of the global memory usage.

        The allocation of memory in the global partitions is mapped with
        dynamically allocated memory of the same size as the partitions.

        Available memory locations are marked with the character 'a'.

        Memory locations which are already in use are marked with a space
        character.
  */
  int  i;
  int        globno;
  PDB  *record;

  globno = 0;
  for (i =0; i < maxrec; ++i)                   /* find each global */
        {
          if (      (pdbrec[i].vartype == TYPE_GLOB_CON)
                        || (pdbrec[i].vartype == TYPE_GLOB_IC)
                        || (pdbrec[i].vartype == TYPE_GLOB_REG)
                 )
            {
                        make_onemap(pdbrec + i, globno);
                        ++globno;
                }
        }
```

```
}


int free_varspace(char partition, int bytes2clear, long addr)
{ /* This function frees space in a global partition.

            partition   = global partition
            bytes2clear = total number of bytes to free
            addr        = offset into partition for space to free

            Returns  0, on success.
            Returns -1, on failure.
 */

 int  globindex;
 char *cptr;

 for (globindex = 0; globindex < totpart; ++globindex)
    if (partition == globinfo[globindex].partition)
            break;
 if (globindex >= totpart)
   return -1;
 if ((addr + bytes2clear) > globinfo[globindex].size)
   bytes2clear = globinfo[globindex].size - addr;
 cptr = globmap[globindex] + addr;
 memset(cptr, 'a', bytes2clear);       /* make space available in map */
}

long reserve_varspace(char partition, char vartype, int totitems, long absaddr)
{ /* This function reserves space in a global partition.

            partition = global partition
            vartype   = variable type
            totitems  = total number of items, of type vartype, for which
                                                space will be reserved
            absaddr   = if not equal to -1, place item at this offset

            Returns the offset into the partition, on success.
            Returns -1, on failure.
 */

 long  offset;
 long  addbytes;
 long  bytesfound;
 long  bytesneeded;
 int   globindex;
 char *cptr;
 char *maxptr;

 /* global partitions are a special case */
 if (           (vartype == TYPE_GLOB_CON)
             || (vartype == TYPE_GLOB_IC)
             || (vartype == TYPE_GLOB_REG)
            )
            return 0;    /* set global partition start addresses to 0 */

 for (globindex = 0; globindex < totpart; ++globindex)
    if (partition == globinfo[globindex].partition)
             break;
 if (globindex >= totpart)
   return -1;
 cptr = globmap[globindex];
 maxptr = cptr + globinfo[globindex].size;
 bytesneeded = pdbtype[vartype].size * totitems;
 if (absaddr != -1)
   { /* absolute placement requested */
            cptr += absaddr;

            /* check for proper byte alignment */
            addbytes = absaddr % pdbtype[vartype].size;
            if (addbytes)
              return -1;            /* item not properly aligned */

            /* check for sufficient space to end of partition */
            for ( bytesfound = 0;
```

```
                    (bytesfound < bytesneeded) && cptr[bytesfound];
                        ++bytesfound
                    );

            if (bytesfound == bytesneeded)
                    { /* sufficient space found */
                        memset(cptr, ' ', bytesneeded);      /* reserve space in map */
                        return absaddr;
                    }
            else
              return -1;
          }
 while (cptr < maxptr)
                    {
                     cptr = strchr(cptr, 'a');
                     if (cptr == NULL)
                       cptr = maxptr;
                     else
                       {
                            /* adjust pointer to proper boundary depending on variable type */
                            offset = cptr - globmap[globindex];
                            addbytes = offset % pdbtype[vartype].size;
                            if (addbytes)
                              {
                                addbytes = pdbtype[vartype].size - addbytes;
                                        cptr += addbytes;
                                }

                            if (cptr < maxptr)
                              {
                                        /* check for sufficient available space */
                                        for ( bytesfound = 0;
                                            (bytesfound < bytesneeded) && (cptr[bytesfound] == 'a');
                                                ++bytesfound
                                            );
                                        if (bytesfound == bytesneeded)
                                          { /* sufficient space found */
                                                memset(cptr, ' ', bytesneeded);      /* reserve space in map */
                                                return (cptr - globmap[globindex]);
                                          }
                                        cptr += bytesfound;
                                }
                        }
                    }
 return -1;
}


int addrec(PDB *record, PDBEXT *recext, long absaddr)
{ /* This function adds the contents of record to the data bases.

   Returns  0, on success.
            Returns -1, on failure.
 */
 int   totitems;
 int           index;
 long  offset;
 void *vptr;

 totitems = 1;
 for (index = 0; index < record->ndim; ++index)
             totitems *= record->dim[index];
 offset = reserve_varspace(record->partition, record->vartype, totitems, absaddr);
 if (offset < 0)
            return -1;
 record->addr = offset;

 if (nextrec >= lastrec)
   { /* need to expand PDB in memory */
             vptr = realloc((void *)pdbrec, (nextrec + 100) * sizeof(PDB));
             if (vptr == NULL)
               return -1;
             pdbrec = vptr;
             vptr = realloc((void *)pdbext, (nextrec + 100) * sizeof(PDBEXT));
             if (vptr == NULL)
```

```
                return -1;
              pdbext = vptr;
              lastrec = nextrec + 100;
            }
  memcpy(pdbrec + nextrec, record, sizeof(PDB));
  memcpy(pdbext + nextrec, recext, sizeof(PDBEXT));
  ++nextrec;
  return 0;
}


int modrec(PDB *record, PDBEXT *recext, long recno, long absaddr)
{ /* This function modifies the contents of a record in the data bases.

    Returns  0, on success.
             Returns -1, on failure.
  */
  int   totitems;
  int           index;
  long  offset;

  if (  (record->partition != pdbrec[recno].partition)
              || (record->ndim     != pdbrec[recno].ndim)
              || (record->dim[0]   != pdbrec[recno].dim[0])
              || (record->dim[1]   != pdbrec[recno].dim[1])
              || (record->dim[2]   != pdbrec[recno].dim[2])
              || (record->dim[3]   != pdbrec[recno].dim[3])
              || (record->dim[4]   != pdbrec[recno].dim[4])
              || (record->dim[5]   != pdbrec[recno].dim[5])
              || (record->dim[6]   != pdbrec[recno].dim[6])
              || (record->vartype   != pdbrec[recno].vartype)
             )
             {          /* need to free then reallocate space in map */
                        free_varspace(pdbrec[recno].partition,
                                                        rt_gettotsize(pdbrec + recno),
                                                        pdbrec[recno].addr);
                        totitems = 1;
                        for (index = 0; index < record->ndim; ++index)
                                totitems *= record->dim[index];
                        offset = reserve_varspace(record->partition, record->vartype, totitems, absaddr);
                        if (offset < 0)
                                return -1;
                        record->addr = offset;
             }
  memcpy(pdbrec + recno, record, sizeof(PDB));
  memcpy(pdbext + recno, recext, sizeof(PDBEXT));
  return 0;
}


void showrec(PDB *record, PDBEXT *recext)
{ /* This function displays the information in the record structures
    given as arguments.
  */
  char tstr[512];
  char delstr[6];
  int  i;

  sprintf(tstr, "Record No.   = %d", record - &pdbrec[0]);
  message(tstr);
  if (record->prevaddr == -1)
    strcpy(delstr, "*del*");
  else
    delstr[0] = 0;
  sprintf(tstr, "PTID        = %-*s   %s", PTIDSIZE, record->ptid, delstr);
  message(tstr);
  sprintf(tstr, "Brief desc   = %s", recext->brief);
  message(tstr);
  sprintf(tstr, "Units       = %s", recext->units);
  message(tstr);
  sprintf(tstr, "System       = %s", recext->sys);
  message(tstr);
  sprintf(tstr, "Type        = %s", pdbtype[record->vartype].typestr);
  message(tstr);
  strcpy(tstr,  "Dimensions   = ");
```

```
 if (record->ndim)
   {
            for (i = 0; i < record->ndim; ++i)
                    {
                            if (!i)
                                    strcat(tstr,"(");
                            else
                                    strcat(tstr,", ");
                            sprintf(tstr, "%s%d", tstr, record->dim[i]);
                    }
            strcat(tstr,")");
           }
 else
           strcat(tstr,"None");
 message(tstr);
 sprintf(tstr, "Total bytes  = %d", rt_gettotsize(record));
 message(tstr);
 sprintf(tstr, "Partition    = %s", rt_getglobname(record->partition));
 message(tstr);
 sprintf(tstr, "Offset       = 0x%X", record->addr);
 message(tstr);
 sprintf(tstr, "Print format = %s", record->format);
 message(tstr);
 sprintf(tstr, "Full desc    = %s", recext->full);
 message(tstr);
 message("");
}


int close_record(char *ptid)
{
 if (*ptid)
   {
            if (currec < 0)
                    { /* adding new record */
                     if (!addrec(&newrec, &newext, absaddr))
                            {
                             ++recsadd;
                             display_recno(nextrec - 1);
                            }
                     else
                            {
                             sprintf(tempstr, "Error adding record for %s.", curptid);
                             message(tempstr);
                            }
                    }
            else
                    { /* modifying an existing record */
                     if (!modrec(&newrec, &newext, currec, absaddr))
                            {
                             ++recsmod;
                             display_recno(currec);
                            }
                     else
                            {
                             sprintf(tempstr, "Error modifying record for %s.", curptid);
                             message(tempstr);
                            }
                    }
             curptid[0] = 0;
           }
 return 0;
}


int display(char *ptid)
{ /* This function prints to stdout the contents of the record associated
            with ptid

            Returns  0, on success.
            Returns -1, on error;
 */

 PDB     *record;
 PDBEXT *recext;
```

```
 record = getrecord(ptid);
 if (record)
   {
                recext = getrecext(ptid);
                if (recext)
                  showrec(record, recext);
                else
                  return -1;
               }
 else
   return -1;
 return 0;
}


int display_recno(int index)
{ /* This function prints to stdout the information for a specific record number

               Returns  0, on success.
               Returns -1, on error;
 */

 PDB       *record;
 PDBEXT *recext;

 if ((index >= 0) && (index < nextrec))
   {
                record = &pdbrec[index];
                recext = getrecext(record->ptid);
                if (recext)
                  showrec(record, recext);
                else
                  return -1;
               }
 else
   return -1;
 return 0;
}


void dumprec(PDB *record, PDBEXT *recext)
{ /* This function displays the information in the record structures
    given as arguments.
 */
 char tstr[512];
 char delstr[6];
 int  i;

 sprintf(tstr, "/cmd  %s", record->ptid);
 message(tstr);
 sprintf(tstr, ".desc %s", recext->brief);
 message(tstr);
 sprintf(tstr, "%s", recext->full);
 message(tstr);
 sprintf(tstr, ".unit %s", recext->units);
 message(tstr);
 sprintf(tstr, ".sys  %s", recext->sys);
 message(tstr);
 sprintf(tstr, ".type %s", pdbtype_C[record->vartype].typestr);
 message(tstr);
 strcpy(tstr,  ".dim  ");
 if (record->ndim)
   {
                for (i = 0; i < record->ndim; ++i)
                                {
                                                if (!i)
                                                                strcat(tstr,"(");
                                                else
                                                                strcat(tstr,", ");
                                                sprintf(tstr, "%s%d", tstr, record->dim[i]);
                                }
                strcat(tstr,")");
               }
 message(tstr);
```

```
  sprintf(tstr, ".form %s", record->format);
  message(tstr);
  sprintf(tstr, ".pred %s, %d", rt_getglobname(record->partition), record->addr);
  message(tstr);
  message("");
}


int dump(char *ptid)
{ /* This function prints to stdout the record definition associated
             with ptid. If ptid is an empty string, all records are printed.

             Returns  0, on success.
             Returns -1, on error;
  */

  PDB       *record;
  PDBEXT *recext;
  int        startpos;
  int        stop_pos;
  int        i;

  if (ptid[0])
    {
              record = rt_getrecord(ptid);
              if (record)
                {
                  startpos = record - &pdbrec[0];
                                    stop_pos = startpos;
                }
              else
                return -1;
            }
  else
    {
              startpos = 0;
              stop_pos = maxrec;
              record = &pdbrec[0];
            }
  for (i = startpos; i <= stop_pos; ++i)
    {
              recext = getrecext(pdbrec[i].ptid);
              if (recext)
                dumprec(&pdbrec[i], recext);
              else
                return -1;
            }
  return 0;
}


void init()
{
  char *cptr;

  cptr = getenv("ODSPATH");
  if (cptr == NULL)
    strcpy(odspath, ".");
  else
    strcpy(odspath, cptr);
  dbmode = 'F';
  pdbrec = rt_makelocalpdb();
  if (pdbrec == NULL)
    {
                    sprintf(tempstr,"Unable to read %s/pdb.dat file into memory.", odspath);
                    message(tempstr);
                    exit(1);
            }
  nextrec = maxrec;
  lastrec = maxrec;
  pdbext = rt_makelocalpdbext();
  if (pdbrec == NULL)
    {
                    sprintf(tempstr,"Unable to read %s/pdbext.dat file into memory.", odspath);
                    message(tempstr);
```

```
                    exit(2);
            }
 dbmode = 'R';
 makemaps();
 recsadd  = 0;
 recsdel  = 0;
 recsmod  = 0;
 recsundel = 0;
}


int write_database()
{ /* This function writes the modified PDB to disk.

   Returns number of records written, on success.
           Returns 0, on failure.
 */

 FILE  *fout1;
 FILE  *fout2;
 char   fullpath1[256];
 char   fullpath2[256];
 long        recs_out;
 long        recno;
 long        delrecs;
 void  *vptr;

 /* check for read or write mode */
 if (dbmode != 'W')
   {
            sprintf(tempstr,"No records written. Program not in write mode.\n");
            message(tempstr);
            return 0;
           }

 /* sort data items alphabetically */
 vptr = rt_compare_ptid;
 qsort((void *)pdbrec,
                    (size_t)nextrec,
                    sizeof(PDB),
                    vptr);
 vptr = rt_compare_ptid_ext;
 qsort((void *)pdbext,
                    (size_t)nextrec,
                    sizeof(PDBEXT),
                    vptr);

 sprintf(fullpath1, "%s/pdb.dat", odspath);
 fout1 = fopen(fullpath1, "wb");
 if (fout1 == NULL)
   return 0;
 sprintf(fullpath2, "%s/pdbext.dat", odspath);
 fout2 = fopen(fullpath2, "wb");
 if (fout2 == NULL)
   return 0;

 for ( recno = 0, delrecs = 0, recs_out = 0;
          recno < nextrec;
          ++recno
   )
           if (pdbrec[recno].prevaddr < 0)
                    {
                     ++delrecs;
                     if (!packit)
                       {
                        fwrite(pdbrec + recno, sizeof(PDB), 1, fout1);
                        fwrite(pdbext + recno, sizeof(PDBEXT), 1, fout2);
                                        ++recs_out;
                                }
                    }
            else
                    {
                fwrite(pdbrec + recno, sizeof(PDB), 1, fout1);
                fwrite(pdbext + recno, sizeof(PDBEXT), 1, fout2);
                    ++recs_out;
```

```
                        }
 fclose(fout1);
 fclose(fout2);
 if (packit)
   {
    sprintf(tempstr,"%ld deleted records removed.\n", delrecs);
              message(tempstr);
            }
 else
   {
    sprintf(tempstr,"%ld records currently marked for deletion.\n", delrecs);
              message(tempstr);
            }
 sprintf(tempstr,"%ld records written to files %s and %s\n", recs_out, fullpath1, fullpath2);
 message(tempstr);
 return recs_out;
}


void finish()
{
 if (!cancel)
   {
              totmods = recsadd + recsdel + recsmod + recsundel;
              if (totmods || packit)
      write_database();
            }
 exit (0);
}


main()
{
          int result;

   init();
   BEGIN GETCMD;
   result = yyparse();
          return result;
}
%{
/*
```

```
 rtdata.l - LEX input for Real Time Data Server of RTProE

*/

/* function prototypes */
void  rt_strlwr(char *string);
```

```
char tokcval[256];
char toksval[256];

#define YY_INPUT(buf,result,max_size) \
                        { \
                        int c = getc( yyin ); \
                        result = c == EOF ? 0 : 1; \
                        buf[0] = (char) c; \
                        }
/*****************************************************************************/
%}
A       [Aa]
B       [Bb]
C       [Cc]
D       [Dd]
E       [Ee]
F       [Ff]
G       [Gg]
H       [Hh]
I       [Ii]
J       [Jj]
K       [Kk]
L       [Ll]
M       [Mm]
N       [Nn]
O       [Oo]
P       [Pp]
Q       [Qq]
R       [Rr]
S       [Ss]
T       [Tt]
U       [Uu]
V       [Vv]
W       [Ww]
X       [Xx]
Y       [Yy]
Z       [Zz]
star "*"

%s GETCMD GETPTID GETSTR GETBRIEF GETFULL GETDIM GETTYPE GETINT GETREAL FILENAME
%%
<GETCMD>{A}{D}{D}{T}{O}{L}{I}{S}{T}                     {
                                                        BEGIN GETPTID;
                                                        return (ADDTOLIST);
                                                }

<GETCMD>{C}{O}{N}{L}{O}{A}{D}                           {
                                                        BEGIN GETSTR;
                                                        return (CONST_LOAD);
                                                }

<GETCMD>{C}{O}{N}{S}{A}{V}{E}                           {
                                                        BEGIN GETSTR;
                                                        return (CONST_SAVE);
                                                }

<GETCMD>{D}{U}{M}{P}                                    {
                                                        BEGIN GETPTID;
                                                        return (DUMP);
                                                }

<GETCMD>{E}{C}{H}{O}                    {
                                                        BEGIN GETSTR;
                                                        return (ECHO);
                                                }

<GETCMD>{E}{X}{I}{T}            {
                                                        return (EXIT);
                                                }

<GETCMD>{F}{A}{S}{T}            {
                                                        BEGIN GETINT;
                                                        return (FAST);
                                                }
```

```
<GETCMD>{F}{I}{N}{D}{I}{N}        {
                                        BEGIN GETSTR;
                                        return (FINDIN);
                                }

<GETCMD>{F}{I}{N}{D}{S}{T}{A}{R}{T}        {
                                        BEGIN GETSTR;
                                        return (FINDSTART);
                                }

<GETCMD>{F}{R}{E}?{E}?{Z}{E}?    {
                                        return (CMD_FREEZE);
                                }

<GETCMD>{G}{E}{T}{I}{N}{F}{O}    {
                                        BEGIN GETPTID;
                                        return (GETINFO);
                                }

<GETCMD>{G}{E}{T}{L}{I}{S}{T}    {
                                        BEGIN GETINT;
                                        return (GETLIST);
                                }

<GETCMD>{G}{E}{T}{P}{O}{S}       {
                                        BEGIN GETPTID;
                                        return (GETPOS);
                                }

<GETCMD>{G}{E}{T}{R}{A}{N}{G}{E}        {
                                        BEGIN GETPTID;
                                        return (GETRANGE);
                                }

<GETCMD>{G}{E}{T}{S}{T}{A}{T}    {
                                        return (GETSTAT);
                                }

<GETCMD>{G}{E}{T}{V}{A}?{L}?     {
                                        BEGIN GETPTID;
                                        return (GET);
                                }

<GETCMD>{M}{F}{I}{N}{F}{O}               {
                                        BEGIN GETPTID;
                                        return (MFINFO);
                                }

<GETCMD>{N}{E}{W}{L}{I}{S}{T}    {
                                        BEGIN GETINT;
                                        return (NEWLIST);
                                }

<GETCMD>{R}{E}?{S}{E}?{T}        {
                                        BEGIN FILENAME;
                                        return (CMD_RESET);
                                }

<GETCMD>{R}{F}{I}{N}{F}{O}               {
                                        BEGIN GETPTID;
                                        return (RFINFO);
                                }

<GETCMD>{R}{U}{N}                {
                                        return (CMD_RUN);
                                }

<GETCMD>{S}{E}{T}                {
                                        BEGIN GETPTID;
                                        return (SET);
                                }

<GETCMD>{S}{E}{T}{P}{O}{S}               {
                                        BEGIN GETPTID;
```

```
                                                    return (SETPOS);
                                                }

<GETCMD>{S}{L}{O}{W}                    {
                                                    BEGIN GETINT;
                                                    return (SLOW);
                                                }

<GETCMD>{S}{N}{A}{P}                    {
                                                    BEGIN FILENAME;
                                                    return (SNAP);
                                                }

<GETCMD>{T}{E}{R}{M}                    {
                                                    return (CMD_TERM);
                                                }

<GETPTID,GETDIM>[ \t]*\[[ \t]*[0-9]+[ \t]*\]    {
                                                    sscanf(yytext," [ %d ] ",&yylval.ival);
                                                    return (CINDEX);
                                                }

<GETPTID,GETDIM>[(,][ \t]*[0-9]+    {
                                                    yytext[0] = ' ';
                                                    sscanf(yytext," %d",&yylval.ival);
                                                    return (FINDEX);
                                                }

<GETPTID,GETINT>[ \t]*\.{F}\.       {
                                                    yylval.ival = 0;
                                                    return (INTEGER);
                                                }

<GETPTID,GETINT>[ \t]*\.{T}\.       {
                                                    yylval.ival = 1;
                                                    return (INTEGER);
                                                }

<GETPTID,GETINT>[ \t]*-?\+?[0-9]+{
                                                    yytext[0] = ' ';
                                                    sscanf(yytext," %d",&yylval.ival);
                                                    return (INTEGER);
                                                }

<GETPTID,GETREAL>[ \t]*-?\+?[0-9]*\.[0-9]*      {
                                                    yytext[0] = ' ';
                                                    sscanf(yytext," %f",&yylval.fval);
                                                    return (REAL);
                                                }

<GETPTID,GETREAL>[ \t]*-?\+?[0-9]*[.]?[0-9]*{E}[+-]?[0-9][0-9]?[0-9]?    {
                                                    yytext[0] = ' ';
                                                    sscanf(yytext," %f",&yylval.fval);
                                                    return (REAL);
                                                }

<GETPTID>[a-zA-Z][a-zA-Z0-9_]*      {
                                                    strcpy(yylval.string, yytext);
                                                    rt_strlwr(yylval.string);
                                                    return (PTID);
                                                }

<FILENAME>[bBit][cC][0-9][0-9]      {
                                                    strcpy(yylval.string, yytext);
                                                    return (ICNAME);
                                                }

<GETSTR>[^\n]*                                  {
                                                    if (*yytext == ' ')
                                                      strcpy(yylval.string, yytext + 1);     /* skip first space */
                                                    else
                                                      strcpy(yylval.string, yytext);
                                                    return (STRING);
                                                }
```

```
<FILENAME,GETINT,GETCMD,GETPTID,GETSTR,GETDIM,GETTYPE>\n        {
                                                                    BEGIN GETCMD;
                                                                    return EOL;
                                                                }

<FILENAME,GETINT,GETCMD,GETPTID,GETTYPE,GETDIM>[^\n]    {
                                                                    ;
                                                                }

%%
/*****************************************************************/
%{
/*
-------------------------------------------------------------------
```



```
-------------------------------------------------------------------

 rtdata.y - YACC Grammar specification for Real Time Data Server of RTProE

*/

#include <Windows.h>
#include "librt.h"

#define MONITOR_NONE        0
#define MONITOR_EVENT       1
#define MONITOR_OOS         2
#define MONITOR_IO  3

/* prototypes */
void finish();
long newlist(int listnum);
char *getlist(int listnum, char *rtnbuf);
int addtolist(int listnum, char *vname, int index);
void showquery(int i);

/* pointers to some simulation load variables */
char        *status;
char        *icname;
char        *icrw;
char        *loadname;
char        *slofac;
long int *seconds;
long int *curframe;
float       *avg_cpu_usage;

PDB                 record;                 /* current working PDB record */
int                 showmsg;                /* flag for message display */
char                tempstr[512];           /* temporary work string */
long                nextrec;
long                lastrec;
int                 lindex;
```

```
float                   fl;
double      dbtemp;
long                    listid;
char                    workstr[5000];
char                    sharedfile[256];    /* shared memory EXE or DLL file to attach to */
HANDLE      FRAMEDONE;                      /* Event to signal that frame work completed */
HANDLE      rtdata_thread;
DWORD                   rtdata_thread_id;
int                     monrec;

/* structure definitions */
typedef struct monitor   {
                                            void  *addr;
                                            int           index;
                                            double  limitlo;
                                            double  limithi;
                                            void  *mon_addr;
                                            char   mon_type;
                                            char   vartype;
                                            char   scantype;
                                            char   delta[PTIDSIZE + 1];
                                            char   ptid[PTIDSIZE + 1];
                                            char   format[FORMATSIZE + 1];
                                           } monitor;

typedef struct dispinfo {
                                            void                    *varaddr;
                                            int                             index;
                                            char                    ptid[PTIDSIZE + 1];
                                            char                    format[FORMATSIZE + 1];
                                            char                    vartype;
                                            struct dispinfo *nextitem;
                                           } dispinfo;

typedef struct listname   {
                                            long                    listid;
                                            dispinfo                *nextitem;
                                            struct listname *nextlist;
                                           } listname;
monitor   scanlist[1000];
listname  displist;
listname *first_disp_list;

%}
%union      {
                        int             ival;
                        float   fval;
                        char            string[500];
                       }


%token ADDTOLIST
%token <ival> CINDEX
%token CMD_FREEZE
%token CMD_RESET
%token CMD_RUN
%token CMD_TERM
%token CONST_LOAD
%token CONST_SAVE
%token DUMP
%token ECHO
%token EOL
%token FAST
%token FINDIN
%token FINDSTART
%token EXIT
%token <ival> FINDEX
%token GET
%token GETEVENTS
%token GETINFO
%token GETLIST
%token GETOOS
%token GETPOS
%token GETRANGE
%token GETSTAT
```

```
%token <string> ICNAME
%token <ival> INTEGER
%token MFINFO
%token NEWLIST
%token <string> PTID
%token <fval> REAL
%token RFINFO
%token SET
%token SETPOS
%token SLOW
%token SNAP
%token <string> STRING
%token T_LOGICAL
%token T_UCHAR
%token T_SHORT
%token T_LONG
%token T_FLOAT
%token T_DOUBLE
%token T_GLOBAL

%type  <ival> cindex
%type  <ival> findex
%type  <string> ptid

%%
file        : /* empty */
                    | file cmdlist EOL
                     {
                       ;
                     }
                    | file error EOL
                     {
                       ;
                     }
                    | file EOL
                     {
                       ;
                     }
            ;

cmdlist     : cmdlist cmditem
                    | cmditem
            ;

cmditem     : add2list
                    | const_ld
                    | const_sv
                    | dump
                    | echo
                    | exit
                    | fast
                    | find
                    | freeze
                    | get
                    | getinfo
                    | getlist
                    | getpos
                    | getrange
                    | getstat
                    | mfinfo
                    | newlist
                    | reset
                    | rfinfo
                    | run
                    | set
                    | setpos
                    | slow
                    | snap
                    | term
            ;

add2list: ADDTOLIST INTEGER ptid
                     {
                              lindex = rt_Clinearaddress($3, record.ndim, record.dim);
                              addtolist($2, $3, lindex);
```

```
                }
                | ADDTOLIST INTEGER
                | ADDTOLIST
                {
                        ;
                }
        ;

const_ld    : CONST_LOAD STRING
                {
                        rt_load_constants($2);
                }
                | CONST_LOAD
                {
                        rt_load_constants("globcon.dat");
                }
        ;

const_sv    : CONST_SAVE STRING
                {
                        rt_save_constants($2);
                }
                | CONST_SAVE
                {
                        rt_save_constants("globcon.dat");
                }
        ;

dump        : DUMP PTID
                {
                        int i, k;
                        unsigned char parco;
                        unsigned char partype;

                        if (!strcmpi($2, "ic"))
                          partype = TYPE_GLOB_IC;
                        else
                          if (!strcmpi($2, "constants"))
                                partype = TYPE_GLOB_CON;
                          else
                                partype = 0;
                        if (partype)
                          for (k = 0; k < totpart; ++k)
                                if (globinfo[k].vartype == partype)
                                  {
                                        parco = globinfo[k].partition;
                                        for (i = 0; i < maxrec; ++i)
                                          {
                                                if (    (pdbrec[i].partition == parco)
                                                        && (pdbrec[i].vartype != partype)
                                                   )
                                                  {
                                                        dumpvar(i);
                                                  }
                                          }
                                  }
                }
                | DUMP
                {
                }
        ;

echo            : ECHO STRING
                {
                        printf("%s\n", $2);
                        fflush(stdout);
                }
                | ECHO
                {
                        printf("\n");
                        fflush(stdout);
                }
        ;

fast        : FAST INTEGER
```

```
                {
                        rt_set_value("slofac", 0, -$2);
                }
                | FAST
                {
                }
        ;

find    : FINDIN STRING
                {
                        int i;
                        int slen;

                        rt_strlwr($2);
                        for (i = 0; i < maxrec; ++i)
                                {
                                 if (strstr(pdbrec[i].ptid, $2))
                                        showquery(i);
                                }
                        printf("#@eofind@#\n");
                        fflush(stdout);
                }
                | FINDSTART STRING
                {
                        int   i;
                        int   slen;

                        slen = strlen($2);
                        rt_strlwr($2);
                        for (i = 0; i < maxrec; ++i)
                                {
                                 if (!strncmp(pdbrec[i].ptid, $2, slen))
                                        showquery(i);
                                }
                        printf("#@eofind@#\n");
                        fflush(stdout);
                }
                | FINDIN
                {
                        printf("#@eofind@#\n");
                        fflush(stdout);
                }
                | FINDSTART
                {
                        printf("#@eofind@#\n");
                        fflush(stdout);
                }
        ;

freeze  : CMD_FREEZE
                {
                        rt_set_value("status", 0, FREEZE);
                }
        ;

get     : GET ptid
                {
                        lindex = rt_Clinearaddress($2, record.ndim, record.dim);
                        if (lindex < 0)
                          { /* only print out INVALID string */
                                printf("%s\n", INVALID);
                          }
                        else
                          {
                                printf("%s\n", rt_get_avalue($2, lindex));
                          }
                        fflush(stdout);
                }
                | GET
                 { /* only print out expected end-of-line */
                        printf("\n");
                        fflush(stdout);
                 }
        ;
```

```
getinfo     : GETINFO PTID
                    {
                            if (display($2))
                                    { /* had error, so print out blank lines */
                                      printf("\n\n\n\n\n\n\n\n\n\n");
                                      fflush(stdout);
                                    }
                    }
                   | GETINFO
                    { /* print out the expected end-of-lines */
                      printf("\n\n\n\n\n\n\n\n\n\n");
                            fflush(stdout);
                    }
            ;

getlist     : GETLIST INTEGER
                    {
                      static char rtnval[20000];

                            printf("%s\n", getlist($2, rtnval));
                            fflush(stdout);
                    }
                   | GETLIST
                    {
                            printf("\n");
                            fflush(stdout);
                    }
            ;

getpos      : GETPOS ptid
                    { /* This command retrieves the current enumerated position of
                              the specified variable
                      */
                      char poslabel[256];

                      lindex = rt_Clinearaddress($2, record.ndim, record.dim);
                      if (lindex < 0)
                              { /* only print out INVALID string */
                                printf("%s\n", INVALID);
                              }
                      else
                              {
                                printf("%s\n", rt_getposition($2, lindex));
                              }
                      fflush(stdout);
                    }
                   | GETPOS
                    {
                      printf("\n");
                      fflush(stdout);
                    }
            ;

getrange    : GETRANGE PTID
                    { /* This command retrieves the current enumerated positions
                              or low-high limits of the specified variable
                      */
                      char range[256];
                      int  rangetype;

                      rangetype = rt_getrange($2, range);
                      printf("%d %s\n", rangetype, range);
                      fflush(stdout);
                    }
                   | GETRANGE
                    {
                      printf("\n");
                      fflush(stdout);
                    }
            ;

getstat     : GETSTAT
                    { /* get general status information */
                      int           hours, minutes, secs;
                            float factor;
```

```
                if (status)
                        {
                         switch (*status)
                                        {
                                                case FREEZE:
                                                        printf("Freeze\n");
                                                        break;
                                                case RUN:
                                                        printf("Run\n");
                                                        break;
                                                case RESET:
                                                        printf("Reset\n");
                                                        break;
                                                case TERM:
                                                        printf("Term\n");
                                                        break;
                                                case SYNC:
                                                        printf("Sync\n");
                                                        break;
                                                default:
                                                        printf("Unknown\n");
                                                        break;
                                        }
                         }
                        else
                          printf("Unknown\n");

                        if (*slofac < 1)
                                factor = (float)abs(*slofac);
                        else
                           factor = 1 / (float)(*slofac);
                        printf("%4.2f\n", factor);
                 if (seconds)
                          {
                           hours   = *seconds / 3600;
                                   minutes = (*seconds % 3600) / 60;
                                   secs    = *seconds % 60;
                                   printf("%02d:%02d:%02d\n", hours, minutes, secs);
                                   printf("%d\n", *seconds);
                          }
                        else
                          printf("\n\n");
                        if (avg_cpu_usage)
                          {
                           printf("%f\n", *avg_cpu_usage);
                          }
                        else
                          printf("\n");
                        if (loadname)
                                printf("%s\n", loadname);
                        else
                                printf("\n");
                        if (icname)
                                printf("%s\n", icname);
                        else
                                printf("\n");
                         fflush(stdout);
                }
        ;

mfinfo      : MFINFO plid
                { /* This command retrieves information about a remote function. */
                  char info[256];

                  lindex = rt_Clinearaddress($2, record.ndim, record.dim);
                  if (lindex < 0)
                          { /* only print out blank line */
                            printf("\n");
                          }
                  else
                          {
                            rt_func_info(4, 4, $2, lindex, info);
                            printf("%s\n", info);
                          }
```

```
                    fflush(stdout);
                  }
                | MFINFO
                  {
                    int i;
                    char info[256];

                    for (i = 0; i < maxrec; ++i)
                            if (rt_func_info(4, 4, pdbrec[i].ptid, 0, info))
                                    printf("%s\n", info);
                    fflush(stdout);
                  }
        ;

newlist     : NEWLIST INTEGER
                  {
                            printf("%d\n", newlist($2));
                            fflush(stdout);
                  }
                | NEWLIST
                  {
                            printf("%d\n", newlist(0));
                            fflush(stdout);
                  }
        ;

reset               : CMD_RESET ICNAME
                  {
                            strncpy(icname, $2, 4);
                            icname[4] = 0;
                            *icrw = 'R';
                  }
                | CMD_RESET ICNAME ICNAME
                  {         /* Reset to first IC but call it by the second name */
                            int ctr;

                            strncpy(icname, $2, 4);
                            icname[4] = 0;
                            *icrw = 'R';

                            /* wait for Reset to complete */
                            for (ctr = 0; (*status != SYNC) && (ctr < 20); ctr++)
                               Sleep(100);

                            /* Rename current IC to second name */
                            strncpy(icname, $3, 4);
                            icname[4] = 0;
                  }

                | CMD_RESET
        ;

rfinfo      : RFINFO ptid
                  { /* This command retrieves information about a remote function. */
                    char info[256];

                    lindex = rt_Clinearaddress($2, record.ndim, record.dim);
                    if (lindex < 0)
                            { /* only print out blank line */
                              printf("\n");
                            }
                    else
                            {
                              rt_func_info(1, 2, $2, lindex, info);
                              printf("%s\n", info);
                            }
                    fflush(stdout);
                  }
                | RFINFO
                  {
                    int i;
                    char info[256];

                    for (i = 0; i < maxrec; ++i)
                            if (rt_func_info(1, 2, pdbrec[i].ptid, 0, info))
```

```
                                        printf("%s\n", info);
                        fflush(stdout);
                      }
        ;

run                   : CMD_RUN
                      {
                              rt_set_value("status", 0, RUN);
                      }
        ;

set             : SET ptid INTEGER
                      {
                              lindex = rt_Clinearaddress($2, record.ndim, record.dim);
                              rt_set_value($2, lindex, (double)$3);
                      }
                      | SET ptid REAL
                      {
                              lindex = rt_Clinearaddress($2, record.ndim, record.dim);
                              rt_set_value($2, lindex, (double)$3);
                      }
        ;

setpos          : SETPOS ptid PTID
                      { /* This command sets the enumerated position of
                              the specified variable
                       */
                       char poslabel[256];

                       lindex = rt_Clinearaddress($2, record.ndim, record.dim);
                       if (lindex >= 0)
                              rt_setposition($2, $3, lindex);
                      }
        ;

slow            : SLOW INTEGER
                      {
                              rt_set_value("slofac", 0, $2);
                      }
                      | SLOW
                      {
                      }
        ;

snap            : SNAP
                      {
                       *icrw = 'W';
                      }
                      | SNAP ICNAME
                      {
                       strncpy(icname, $2, 4);
                              icname[4] = 0;
                       *icrw = 'W';
                      }
        ;

term            : CMD_TERM
                      {
                              rt_set_value("status", 0, TERM);
                      }
        ;


ptid            : PTID
                      { /* Specify Dimensioning information for variable
                       */
                       strcpy($$,$1);
                      record.ndim             = 0;
                              record.dim[0]           = 0;
                              record.dim[1]           = 0;
                              record.dim[2]           = 0;
                              record.dim[3]           = 0;
                              record.dim[4]           = 0;
                              record.dim[5]           = 0;
                              record.dim[6]           = 0;
```

```
                }
              | PTID offset
                {
                  strcpy($$,$1);
                }
        ;

offset      : cindex
                {
              record.ndim             = 1;
                        record.dim[0]         = $1;
                        record.dim[1]         = 0;
                        record.dim[2]         = 0;
                        record.dim[3]         = 0;
                        record.dim[4]         = 0;
                        record.dim[5]         = 0;
                        record.dim[6]         = 0;
                }
              | findex
                {
              record.ndim             = 1;
                        record.dim[0]         = $1;
                        record.dim[1]         = 0;
                        record.dim[2]         = 0;
                        record.dim[3]         = 0;
                        record.dim[4]         = 0;
                        record.dim[5]         = 0;
                        record.dim[6]         = 0;
                }
              | cindex cindex
                {
              record.ndim             = 2;
                        record.dim[0]  = $1;
                        record.dim[1]  = $2;
                        record.dim[2]  = 0;
                        record.dim[3]  = 0;
                        record.dim[4]  = 0;
                        record.dim[5]  = 0;
                        record.dim[6]         = 0;
                }
              | cindex cindex cindex
                {
                        record.ndim           = 3;
                        record.dim[0]         = $1;
                        record.dim[1]         = $2;
                        record.dim[2]         = $3;
                        record.dim[3]         = 0;
                        record.dim[4]         = 0;
                        record.dim[5]         = 0;
                        record.dim[6]         = 0;
                }
              | cindex cindex cindex cindex
                {
                        record.ndim           = 4;
                        record.dim[0]         = $1;
                        record.dim[1]         = $2;
                        record.dim[2]         = $3;
                        record.dim[3]         = $4;
                        record.dim[4]         = 0;
                        record.dim[5]         = 0;
                        record.dim[6]         = 0;
                }
              | cindex cindex cindex cindex cindex
                {
                        record.ndim           = 5;
                        record.dim[0]         = $1;
                        record.dim[1]         = $2;
                        record.dim[2]         = $3;
                        record.dim[3]         = $4;
                        record.dim[4]         = $5;
                        record.dim[5]         = 0;
                        record.dim[6]         = 0;
                }
              | cindex cindex cindex cindex cindex cindex
                {
```

```
            record.ndim             = 6;
            record.dim[0]           = $1;
            record.dim[1]           = $2;
            record.dim[2]           = $3;
            record.dim[3]           = $4;
            record.dim[4]           = $5;
            record.dim[5]           = $6;
            record.dim[6]           = 0;
 }
| cindex cindex cindex cindex cindex cindex cindex
 {
            record.ndim             = 7;
            record.dim[0]           = $1;
            record.dim[1]           = $2;
            record.dim[2]           = $3;
            record.dim[3]           = $4;
            record.dim[4]           = $5;
            record.dim[5]           = $6;
            record.dim[6]           = $7;
 }
| findex findex
 {
record.ndim             = 2;
            record.dim[0]           = $2;
            record.dim[1]           = $1;
            record.dim[2]           = 0;
            record.dim[3]           = 0;
            record.dim[4]           = 0;
            record.dim[5]           = 0;
            record.dim[6]           = 0;
 }
| findex findex findex
 {
            record.ndim             = 3;
            record.dim[0]           = $3;
            record.dim[1]           = $2;
            record.dim[2]           = $1;
            record.dim[3]           = 0;
            record.dim[4]           = 0;
            record.dim[5]           = 0;
            record.dim[6]           = 0;
 }
| findex findex findex findex
 {
            record.ndim             = 4;
            record.dim[0]           = $4;
            record.dim[1]           = $3;
            record.dim[2]           = $2;
            record.dim[3]           = $1;
            record.dim[4]           = 0;
            record.dim[5]           = 0;
            record.dim[6]           = 0;
 }
| findex findex findex findex findex
 {
            record.ndim             = 5;
            record.dim[0]           = $5;
            record.dim[1]           = $4;
            record.dim[2]           = $3;
            record.dim[3]           = $2;
            record.dim[4]           = $1;
            record.dim[5]           = 0;
            record.dim[6]           = 0;
 }
| findex findex findex findex findex findex
 {
            record.ndim             = 6;
            record.dim[0]           = $6;
            record.dim[1]           = $5;
            record.dim[2]           = $4;
            record.dim[3]           = $3;
            record.dim[4]           = $2;
            record.dim[5]           = $1;
            record.dim[6]           = 0;
 }
```

```
                | findex findex findex findex findex findex findex
                {
                        record.ndim             = 7;
                        record.dim[0]           = $7;
                        record.dim[1]           = $6;
                        record.dim[2]           = $5;
                        record.dim[3]           = $4;
                        record.dim[4]           = $3;
                        record.dim[5]           = $2;
                        record.dim[6]           = $1;
                }
        ;

cindex  : CINDEX
                {
                  $$ = $1;
                }

findex  : FINDEX
                {
                  $$ = $1 - 1;
                }

exit    : EXIT
                {
                  finish();
                }
        ;

%%
#include "rtdata_l.c"

void message(char *msg)
{
  if (showmsg)
    {
      printf("%s\n", msg);
                fflush(stdout);
              }
}


void *getglobaddr(char partition)
{
  int i;

  for (i = 0; i < totpart; ++i)
            {
                if (globinfo[i].partition == partition)
                        return (void *)globinfo[i].addr;
            }
  return NULL;
}


void showquery(int i)
{
 int k;
 int firstime;
 PDBEXT recext;

 tempstr[0] = 0;
 if (pdbrec[i].ndim > 0) {
            firstime = 1;
            for (k = 0; k < pdbrec[i].ndim; ++k)
              {
                if (firstime)
                  {
                    strcat(tempstr, "(");
                    firstime = 0;
                  }
                else
                  strcat(tempstr, ",");
                sprintf(tempstr, "%s%d", tempstr, pdbrec[i].dim[k]);
              }
```

```
            strcat(tempstr, ")");
    }
  if (!rt_getextrec_fromfile(pdbrec[i].ptid, &recext))
    printf("%-*s  %-10s  %s\n", PTIDSIZE, pdbrec[i].ptid, tempstr, recext.brief);
}



void showrec(PDB *record, PDBEXT *recext)
{ /* This function displays the information in the record structures
     given as arguments.
  */
  char tstr[256];
  char delstr[6];
  int  i;

  if (record->prevaddr == -1)
    strcpy(delstr, "*del*");
  else
    delstr[0] = 0;
  sprintf(tstr, "Name        = %-16s   %s", record->ptid, delstr);
  message(tstr);
  sprintf(tstr, "Brief desc   = %s", recext->brief);
  message(tstr);
  sprintf(tstr, "Units       = %s", recext->units);
  message(tstr);
  sprintf(tstr, "System      = %s", recext->sys);
  message(tstr);
  sprintf(tstr, "Type        = %s", pdbtype[record->vartype].typestr);
  message(tstr);
  strcpy(tstr,  "Dimensions  =");
  for (i = 0; i < record->ndim; ++i)
            sprintf(tstr, "%s [%d]", tstr, record->dim[i]);
  message(tstr);
  sprintf(tstr, "Total bytes  = %d", rt_gettotsize(record));
  message(tstr);
  sprintf(tstr, "Partition    = %s", rt_getglobname(record->partition));
  message(tstr);
  sprintf(tstr, "Offset      = 0x%X", record->addr);
  message(tstr);
  sprintf(tstr, "Print format = %s", record->format);
  message(tstr);
  sprintf(tstr, "Full desc    = %s", recext->full);
  message(tstr);
  message("");
}


int display(char *ptid)
{ /* This function prints to stdout the contents of the record associated
            with ptid

            Returns  0, on success.
            Returns -1, on error;
  */

  PDB       *record;
  PDBEXT  recext;

  record = rt_getrecord(ptid);
  if (record)
    {
             if (rt_getextrec_fromfile(ptid, &recext))
               return -1;
             else
               showrec(record, &recext);
            }
  else
    return -1;
  return 0;
}


int dumpvar(int index)
{
```

```
 int i;
 int ndim;
 int curdim;
 int item;
 int totitems;

 ndim = pdbrec[index].ndim;
 totitems = 1;
 for (i = 0; i < ndim; ++i)
            totitems *= pdbrec[index].dim[i];
 if  (ndim)
            {
              for (i = 0; i < totitems; ++i)
                        {
                 printf("set %s[%d] %s\n",
                                    pdbrec[index].ptid,
                                    i,
                                    rt_get_avalue(pdbrec[index].ptid, i)
                                   );
                        }
            }
 else
   printf("set %s %s\n", pdbrec[index].ptid, rt_get_avalue(pdbrec[index].ptid, 0));
}


long newlist(int listnum)
{
  int                    curlist;
  listname   *listptr;
  dispinfo    *itemptr;
  dispinfo  *lastptr;

  if (listnum == 0)
    {
                        listptr = malloc(sizeof(listname));
                        listptr->nextlist = first_disp_list;
                        first_disp_list = listptr;
                        first_disp_list->listid   = listid;
                        first_disp_list->nextitem = NULL;
                        curlist = listid;
                        ++listid;
            }
  else
    {
                        curlist = listnum;
                        for (listptr = first_disp_list;
                                    listptr != NULL;
                                    listptr = listptr->nextlist
                                   )
                                   if (listptr->listid == curlist)
                                     { /* truncate the display list */
                                       itemptr = listptr->nextitem;
                                                  while (itemptr != NULL)
                                                             { /* free up memory for display list records */
                                                               lastptr = itemptr;
                                                                          itemptr = itemptr->nextitem;
                                                                          free(lastptr);
                                                             }
                                                  listptr->nextitem = NULL;
                                                  break;
                                     }
            }
  return curlist;
}

int addtolist(int listnum, char *vname, int index)
{
  listname   *listptr;
  dispinfo    *itemptr;

  for (       listptr = first_disp_list;
                        listptr != NULL;
                        listptr = listptr->nextlist
             )
```

```
                        if (listptr->listid == listnum)
                          { /* add new item to end of list */
                                if (listptr->nextitem == NULL)
                                        {
                                         listptr->nextitem = malloc(sizeof(dispinfo));
                                         itemptr = listptr->nextitem;
                                        }
                                else
                                        {
                                         /* find end of linked list */
                                         itemptr = listptr->nextitem;
                                         while (itemptr->nextitem != NULL)
                                                        itemptr = itemptr->nextitem;
                                         itemptr->nextitem = malloc(sizeof(dispinfo));
                                         itemptr = itemptr->nextitem;
                                        }

                                strcpy(itemptr->ptid, vname);
                                itemptr->index          = index;
                                itemptr->vartype  = rt_get_vartype(vname);
                                rt_get_format(vname, itemptr->format);
                                itemptr->varaddr  = rt_getaddress(vname);
                                itemptr->nextitem = NULL;
                                break;
                          }
 return 0;
}


char *getlist(int listnum, char *rtnbuf)
{
 listname   *listptr;
 dispinfo   *itemptr;
 char        aval[32];
 int                pos;
 char               *cptr;

 rtnbuf[0] = 0;
 cptr = rtnbuf;
 for (      listptr = first_disp_list;
                        listptr != NULL;
                        listptr = listptr->nextlist
             )
                        if (listptr->listid == listnum)
                          { /* build return value list */
                                itemptr = listptr->nextitem;
                                while (itemptr != NULL)
                                        {       /* add new item to return list */
                                                *cptr = ' ';
                                                ++cptr;
                                                rt_get_aval2(itemptr->varaddr,
                                                                        itemptr->index,
                                                                        itemptr->vartype,
                                                                        itemptr->format,
                                                                        cptr);
                                                cptr += strlen(cptr);
                                                itemptr  = itemptr->nextitem;
                                        }
                                break;
                          }
 return rtnbuf;
}


int rtdata_cyclic()
{
 int        i;
 char laststatus;

 while (1)
   {
            if (WaitForSingleObject(FRAMEDONE, INFINITE) != WAIT_OBJECT_0)
              {
                        printf("Wait for FRAMEDONE failed");
                        fflush(stdout);
```

```
                return -1;
        }
    if (*curframe == 1)
        {
            ;
        }
        switch (*status)
            {
              case RUN:
                    break;
              case RESET: /* perform reset functions */
                    break;
              case SYNC:
                    break;
              case FREEZE:
                    break;
              case TERM:
                    break;
            }
        laststatus = *status;
    }
}


int makelist(char *buffer, char *ltype, char *list)
{ /*  Search buffer for the ltype delimited list

        Returns:    0, if the ltype label is not found
                    1, if the ltype label is found

        Side effect: Fills the buffer pointed to by list with the
                     ASCII list.
 */
 char   searchbuf[1024];
 char  *cptr;
 char  *cptr2;
 int        ctr;
 int    rtnval;

 list[0] = 0;
 rt_strlwr(buffer);
 /* find enumerated positions list */
 strncpy(searchbuf, buffer, sizeof(searchbuf));
 searchbuf[sizeof(searchbuf) - 1] = 0;
 rtnval = 0;
 if (cptr = strstr(searchbuf, ltype) )
   {
        rtnval = 1;
        /* find opening parentheses */
        for (cptr += strlen(ltype); isspace(*cptr); ++cptr);
        if (*cptr == '(')
          {
            ++cptr;
            /* find closing parentheses */
            cptr2 = strchr(cptr, ')');
            if (cptr2 != NULL)
              {
                *cptr2 = 0; /* cptr now points to a list of positions */

                /* copy list and remove any separating commas */
                for (ctr = 0;  cptr[ctr]; ++ctr)
                    if (cptr[ctr] == ',')
                        list[ctr] = ' ';
                    else
                        list[ctr] = cptr[ctr];
                list[ctr] = 0;
              }
        }
   }
 return rtnval;
}


void build_monitor_tables()
{ /*  This subroutine scans the data base to determine which parameters must be
```

```
            monitored for:

                        1) Out-of-Specification conditions
                        2) Event tracking
                        3) Input/Output activity (necessary for Replay feature)
                        4) switch check
*/
char  list[256];
int   recno;
int   items;
int   currec;
PDBEXT recext;

recno  = 0;
items  = 0;
monrec = 0;
for (currec = 0; currec < maxrec; ++currec)
        {
         fseek(pdbextfile, currec * sizeof(PDBEXT), SEEK_SET);
         fread(&recext, sizeof(PDBEXT), 1, pdbextfile);
         if (makelist(recext.full, "@event", list))
           {
                        strcpy(scanlist[monrec].ptid, recext.ptid);
                        scanlist[monrec].mon_type = MONITOR_EVENT;
                        ++monrec;
           }
         if (makelist(recext.full, "@oos", list))
           {
                        ++monrec;
           }
         if (makelist(recext.full, "@io", list))
           {
                        ++monrec;
           }
   }

}


void finish()
{
 exit(0);
}


void init()
{
 char *cptr;
 char  fullpath[256];
 char  framend[65];
 char  fname[_MAX_FNAME];

 cptr = getenv("ODSPATH");
 if (cptr == NULL)
   strcpy(odspath, ".");
 else
   strcpy(odspath, cptr);
 pdbrec = rt_makelocalpdb();
 if (pdbrec == NULL)
   {
                        sprintf(tempstr,"Unable to read %s/pdb.dat file into memory.", odspath);
                        message(tempstr);
                        exit(2);
        }
 sprintf(fullpath, "%s/pdbext.dat", odspath);
 pdbextfile = fopen(fullpath, "rb");
 if (pdbextfile == NULL)
   {
                        sprintf(tempstr,"Unable to open %s/pdbext.dat file.", odspath);
                        message(tempstr);
                        exit(3);
        }
 nextrec = maxrec;
 lastrec = maxrec;
 showmsg = 1;
```

```
/* connect program to shared memory of real time program  */
rt_connect_to_shmem(sharedfile);

/* Get handle to real time application FRAMEDONE event */
_splitpath(sharedfile, NULL, NULL, fname, NULL);
rt_strupr(fname);
sprintf(framend, "framedone_%s", fname);
if ((FRAMEDONE = CreateEvent(NULL, TRUE, FALSE, framend)) == NULL)
  {
        printf("Unable to attach to FRAMEDONE event\n");
        fflush(stdout);
  }

/* spawn a thread to respond to real time application FRAMEDONE event */
if ((rtdata_thread = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)rtdata_cyclic,
                                                        NULL, 0, &rtdata_thread_id)) == NULL)
        {
          printf("Unable to create rtdata_cyclic thread\n");
          fflush(stdout);
        }

/* initialize display lists */
listid = 1;
first_disp_list = NULL;

/* access some simulation load variables */
status          = rt_getaddress("status");
seconds         = rt_getaddress("seconds");
avg_cpu_usage   = rt_getaddress("avg_cpu_usage");
icname          = rt_getaddress("icname");
loadname        = rt_getaddress("loadname");
icrw            = rt_getaddress("icrw");
slofac          = rt_getaddress("slofac");
curframe        = rt_getaddress("curframe");

/* build_monitor_tables(); */
}


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

  if (argc > 1)
          { /* get name of real time program to connect to */
            strcpy(sharedfile, argv[1]);
          }
        else
          { /* use default name for real time program */
            strcpy(sharedfile, "rtsched.exe");
          }
  init();
  BEGIN GETCMD;
  result = yyparse();
  return result;
}
/* framegen.c -        This program generates the C source code for the
                                the frame calling sequence.

                                The calling sequence is taken from information
                                in the "frameseq.in" file.

                                The results are stored in the file "frameseq.c".

# ------------------------------------------------------------
# Copyright (C) 1996 Dennis R. LaBelle (drlabell@albany.net) All Rights Reserved.
#
# PERMISSION IS GRANTED TO DISTRIBUTE THIS SOFTWARE FREELY, WITH THE
# EXCEPTION THAT ONE MAY NOT CHARGE FOR IT OR INCLUDE IT WITH SOFTWARE
# WHICH IS SOLD. Permission to use, copy, modify, and distribute this software and
# its documentation for educational, research, internal corporate and non-profit
# purposes, without fee, and without a written agreement is hereby granted for all
# cases that do not conflict with the restriction in the first sentence of this
# paragraph, provided that the above copyright notice, this paragraph, and the
```

```
# following three paragraphs appear in all copies.
#
# Permission to incorporate this software into commercial products may be obtained
# from the author (e-mail drlabell@albany.net).
#
# IN NO EVENT SHALL THE AUTHOR BE LIABLE TO ANY PARTY FOR DIRECT,
# INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF
# THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHOR HAS
# BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# THE AUTHOR SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS"
# BASIS, AND THE AUTHOR HAS NO OBLIGATION TO PROVIDE MAINTENANCE,
# SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
# ----------------------------------------------------------------------

*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define strcasecmp(a, b)  _strcmpi(a, b)

#define TOTCALLS      1000

struct      {
                        int         frameno;
                        char        modname[65];
                      } framelist[TOTCALLS];

main(int argc, char *argv[])
{
 FILE *fin;
 FILE *fout;
 int  freq;
 char       line[256];
 char       cmd[256];
 char  modname[256];
 char  blank[256];
 char       protolist[16386];
 int  modfreq;
 int        modgroup;
 int        result;
 int        groups;
 int        totcalls;
 int        ctr1;
 int        frameno;
 struct tm *newtime;
 time_t aclock;


 freq = 12;
 totcalls = 0;
 if (argc > 1)
        {
                freq = atoi(argv[1]);
                if (freq <= 0)
                  {
                        printf("Invalid Maximum calling frequency = %d\n", freq);
                        exit (1);
                  }
        }
 fin = fopen("frameseq.in", "r");
 if (fin == NULL)
        {
                printf("Unable to open file = frameseq.in\n");
                exit (2);
        }
 fout = fopen("frameseq.c", "w");
 if (fout == NULL)
        {
                printf("Unable to open file = frameseq.c\n");
                exit (3);
```

```
        }
while (!feof(fin))
                {
                 line[0] = 0;
                 fgets(line, sizeof(line), fin);
                 cmd[0]     = 0;
                 modname[0] = 0;
                 modfreq    = 0;
                 modgroup   = 0;
                 result = sscanf(line, "load %s %d %d", modname, &modfreq, &modgroup);
                 if (result >= 3)
                   {
                                if (freq % modfreq)
                                  {
                                        printf("Invalid frequency = %d for module %s\n", modfreq, modname);
                                  }
                                else
                                  {
                                        groups = freq / modfreq;
                                        if ((modgroup < 1) || (modgroup > groups))
                                                {
                                                  printf("Invalid group = %d for module %s\n", modgroup, modname);
                                                }
                                        else
                                          {
                                                  /* Add a record for each frame in which the module will be called */
                                                  for (ctr1 = 0; ctr1 < modfreq; ++ctr1)
                                                          {
                                                                  framelist[totcalls].frameno = (ctr1 * groups) + modgroup;
                                                                  strcpy(framelist[totcalls].modname, modname);
                                                                  ++totcalls;
                                                          }
                                                }
                                  }
                        }
                }
fclose(fin);

/* build output file */
/*                                      */

/* print informational header */
fprintf(fout, "/*\n");
fprintf(fout, "  This file generated by %s program\n", argv[0]);
time( &aclock );
newtime = localtime( &aclock );
fprintf(fout, "  on %s\n", asctime(newtime));
fprintf(fout, "*/\n");
fprintf(fout, "\n");

/* print function prototypes */
strcpy(protolist, ",");
for (ctr1 = 0; ctr1 < totcalls; ++ctr1)
        {
          if (strstr(protolist, framelist[ctr1].modname) == NULL)
                  { /* print out each prototype only once */
                          fprintf(fout, "void %s();\n", framelist[ctr1].modname);
                          sprintf(protolist, "%s,%s", protolist, framelist[ctr1].modname);
                  }
        }
fprintf(fout, "\n");

/* generate function that retrieves the number of frames per second */
fprintf(fout, "int rt_getfreq()\n");
fprintf(fout, "{\n");
fprintf(fout, "  return %d;\n", freq);
fprintf(fout, "}\n");
fprintf(fout, "\n");

/* generate individual frame calls */
for (frameno = 1; frameno <= freq; ++frameno)
        {
          fprintf(fout, "int doframe%d()\n", frameno);
          fprintf(fout, "{\n", frameno);
                  for (ctr1 = 0; ctr1 < totcalls; ++ctr1)
```

```
                                        if (framelist[ctr1].frameno == frameno)
                                          {
                                                fprintf(fout, "  %s();\n", framelist[ctr1].modname);
                                          }
                        fprintf(fout, "  return 0;\n");
                  fprintf(fout, "}\n");
                  fprintf(fout, "\n");
                }

  /* generate master frame calling subroutine */
  fprintf(fout, "int callframe(int frameno)\n");
  fprintf(fout, "{\n");
  fprintf(fout, "  switch (frameno)\n");
  fprintf(fout, "      {\n");
  for (frameno = 1; frameno <= freq; ++frameno)
                {
                        fprintf(fout, "      case %3d: doframe%d();\n", frameno, frameno);
                        fprintf(fout, "              break;\n");
                }
  fprintf(fout, "      }\n");
  fprintf(fout, "  return 0;\n");
  fprintf(fout, "}\n");
  fprintf(fout, "\n");

  fclose(fout);
}/*
   librt.c - Library of functions to support real time modeling with RTProE

# ----------------------------------------------------------------------
# Copyright (C) 1996 Dennis R. LaBelle (drlabell@albany.net) All Rights Reserved.
#
# PERMISSION IS GRANTED TO DISTRIBUTE THIS SOFTWARE FREELY, WITH THE
# EXCEPTION THAT ONE MAY NOT CHARGE FOR IT OR INCLUDE IT WITH SOFTWARE
# WHICH IS SOLD. Permission to use, copy, modify, and distribute this software and
# its documentation for educational, research, internal corporate and non-profit
# purposes, without fee, and without a written agreement is hereby granted for all
# cases that do not conflict with the restriction in the first sentence of this
# paragraph, provided that the above copyright notice, this paragraph, and the
# following three paragraphs appear in all copies.
#
# Permission to incorporate this software into commercial products may be obtained
# from the author (e-mail drlabell@albany.net).
#
# IN NO EVENT SHALL THE AUTHOR BE LIABLE TO ANY PARTY FOR DIRECT,
# INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF
# THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHOR HAS
# BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# THE AUTHOR SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS"
# BASIS, AND THE AUTHOR HAS NO OBLIGATION TO PROVIDE MAINTENANCE,
# SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
# ----------------------------------------------------------------------

*/
#define LIBRTSRC

#define  MAINDEF

#include <Windows.h>
#include <ctype.h>
#include <search.h>
#include <string.h>
#include "pdb.h"

#include "librt.h"

void rt_strlwr(char *string)
{ /* This function converts the specified string to lower case.
  */
  char *cptr;

  for (cptr = string; *cptr; ++cptr)
     *cptr = tolower(*cptr);
```

```
}


void rt_strupr(char *string)
{ /* This function converts the specified string to upper case.
  */
  char *cptr;

  for (cptr = string; *cptr; ++cptr)
     *cptr = toupper(*cptr);
}


void rt_rtrim(char *string)
{ /* This function trims white space from the end of a string */
  int   slen;

  for (slen = strlen(string) - 1; (slen >= 0) && isspace(string[slen]); --slen)
                {
                  string[slen] = 0;
                }
}


void rt_lindex(char *poslist, int index, char *label)
{ /* Copies the indexed item from the list into the label */
  char *cptr;
  char worklist[256];
  int   pos;

  strcpy(worklist, poslist);
  label[0] = 0;
  cptr = strtok(worklist, ", ");
  for (pos = 0; (cptr != NULL) && (pos < index); ++pos)
                cptr = strtok(NULL, ", ");
  if  (cptr != NULL)
                strcpy(label, cptr);
}


int rt_lsearch(char *poslist, char *item)
{ /* Returns the index position of an item in a list.
     Returns -1 if item does not appear in the list.
  */
  char *cptr;
  char worklist[256];
  int   pos;

  strcpy(worklist, poslist);
  cptr = strtok(worklist, ", ");
  for (pos = 0; (cptr != NULL); ++pos)
                {
                  if (!strcmpi(cptr, item))
                     return pos;
                  cptr = strtok(NULL, ", ");
                }
  return -1;
}


int rt_getlist(char *ltype, char *ptid, char *list)
{
  PDBEXT  recext;
  char   *cptr;
  char   *cptr2;
  int             ctr;

    list[0] = 0;
    if (!rt_getextrec_fromfile(ptid, &recext))
                {
                  rt_striwr(recext.full);
                  /* find enumerated positions list */
                  if (cptr = strstr(recext.full, ltype) )
                                {
                                  /* find opening parentheses */
```

```
                        for (cptr += strlen(ltype); isspace(*cptr); ++cptr);
                        if (*cptr == '(')
                          {
                                ++cptr;
                                /* find closing parentheses */
                                cptr2 = strchr(cptr, ')');
                                if (cptr2 != NULL)
                                  {
                                        *cptr2 = 0; /* cptr now points to a list of positions */

                                        /* copy list and remove any separating commas */
                                        for (ctr = 0;  cptr[ctr]; ++ctr)
                                           if (cptr[ctr] == ',')
                                                list[ctr] = ' ';
                                           else
                                                list[ctr] = cptr[ctr];
                                        list[ctr] = 0;
                                  }
                          }
                        }
                }
   return 0;
}


int rt_connect_to_shmem(char *filename)
{
  int                   i;
  char                  globname[PTIDSIZE + 1];
  HINSTANCE hInst;

  hInst = LoadLibrary(filename);
  if (hInst == 0)
    {
                        printf("Unable to find and attach to %s file.", filename);
                        exit (1);
                }
  for (i = 0; i < totpart; ++i)
                {
                  strcpy(globname, globinfo[i].ptid);
                        rt_strupr(globname);
                  globinfo[i].addr = (long)GetProcAddress(hInst, globname);
                }

}


int rt_new(char *fname1, char *fname2)
{ /* This function creates a new, empty PDM data file set.

                Returns 0, on success.
                Returns 1, if unable to open fname1 file
                Returns 2, if unable to open fname2 file
  */

  PDB    record;
  PDBEXT  recext;
  FILE   *fout1;
  FILE   *fout2;
  int            i;

  if (fname1 == NULL)
    return 1;
  if (!fname1[0])
    return 1;
  fout1 = fopen(fname1, "w");
  if (fout1 == NULL)
    {
                  return 1;
                }
  if (fname2 == NULL)
    return 2;
  if (!fname2[0])
    return 2;
  fout2 = fopen(fname2, "w");
```

```
  if (fout2 == NULL)
    {
                return 2;
              }
  fclose(fout1);
  fclose(fout2);
  return 0;
}

int rt_Clinearaddress(char *ptid, int ndim, long *dim)
{ /* This function converts a set of array dimensions into an equivalent
              linear, single dimension offset for a given named variable.

                        INPUT:    ptid =      variable name
                                  ndim =      number of array dimensions specified
                                  dim  =      an array of dimension indexes. These dimensions
                                              should be in C specification order.

                        Returns: On success, the equivalent single dimension array index.
                                              On failure, <0
  */
  PDB   *record;
  int           i;
  int           curidx;
  int           product;
  int           num;

  if (record = rt_getrecord(ptid))
              {
               if (record->ndim != ndim)
                 {
                        if ((ndim == 0) && (record->ndim == 1))
                                return 0;
                              else
                          return -1;
                       }
               num = 0;
               for (curidx = 0; curidx < ndim; ++curidx)
                        {
                         product = dim[curidx];
                                for (i = curidx + 1; i < ndim; ++i)
                                  {
                                          product *= record->dim[i];
                                        }
                                num += product;
                       }
               return num;
             }
  else
              return -1;
}


int rt_readpdb(PDB *buffer, long totrecs)
{ /* This function reads the number of specified records from the PDB file
              into memory pointed to by buffer */

  FILE  *fin;
  char  fullpath[256];
  long          recs_read;
  long          i;

  sprintf(fullpath, "%s/pdb.dat", odspath);
  fin = fopen(fullpath, "r");
  if (fin == NULL)
    return 0;
  recs_read = fread(buffer, sizeof(PDB), totrecs, fin);
  fclose(fin);

  /* update shared memory global information */
  totpart = 0;
  for (i = 0; i < totrecs; ++i)
              {
               if (      (buffer[i].vartype == TYPE_GLOB_CON)
                                ||          (buffer[i].vartype == TYPE_GLOB_IC)
```

```
                        ||          (buffer[i].vartype == TYPE_GLOB_REG)
                    )
                    {
                        globinfo[totpart].size = rt_gettotsize(buffer + i);
                        globinfo[totpart].partition = buffer[i].partition;
                        strcpy(globinfo[totpart].ptid, buffer[i].ptid);
                        globinfo[totpart].vartype = buffer[i].vartype;

                        /* get global address */
                        /* Source code for the getglobaddr function is generated */
                        /* using the hgen program. */
                        globinfo[totpart].addr = (long)getglobaddr(buffer[i].partition);
                        ++totpart;
                    }
            }
 return recs_read;
}


int rt_readpdbext(void *buffer, long totrecs)
{/* This function reads the number of specified records from the PDB extension
            file into memory pointed to by buffer */

 FILE  *fin;
 char   fullpath[256];
 long         recs_read;

 sprintf(fullpath, "%s/pdbext.dat", odspath);
 fin = fopen(fullpath, "r");
 if (fin == NULL)
   return 0;
 recs_read = fread(buffer, sizeof(PDBEXT), totrecs, fin);
 fclose(fin);
 return recs_read;
}


void *rt_makelocalpdb()
{/* This function dynamically allocates memory and fills it with the PDB.

            On success, returns a pointer to the filled buffer.
            On error,   returns NULL
 */

 char   fullpath[256];
 long         totrecs;
 long         recsread;
 void  *bufptr;
 struct _stat statbuf;

 sprintf(fullpath, "%s/pdb.dat", odspath);
 if (_stat(fullpath, &statbuf))
            return NULL;
 totrecs = statbuf.st_size / sizeof(PDB);
 bufptr = malloc(statbuf.st_size);
 recsread = rt_readpdb(bufptr, totrecs);
 if (recsread < totrecs)
   {
             free(bufptr);
             return NULL;
            }
 maxrec = recsread;
 return bufptr;
}


void *rt_makelocalpdbext()
{/* This function dynamically allocates memory and fills it with the PDB extension.

            On success, returns a pointer to the filled buffer.
            On error,   returns NULL
 */

 char   fullpath[256];
 long         totrecs;
```

```
 long          recsread;
 void  *bufptr;
 struct _stat statbuf;

 sprintf(fullpath, "%s/pdbext.dat", odspath);
 if (_stat(fullpath, &statbuf))
          return NULL;
 totrecs = statbuf.st_size / sizeof(PDBEXT);
 bufptr = malloc(statbuf.st_size);
 recsread = rt_readpdbext(bufptr, totrecs);
 if (recsread < totrecs)
   {
            free(bufptr);
            return NULL;
          }
 return bufptr;
}


int rt_compare_ptid(const void *arg1, const void *arg2)
{
 PDB *rec1;
 PDB *rec2;

 rec1 = (PDB *)arg1;
 rec2 = (PDB *)arg2;
 return (strcmp(rec1->ptid, rec2->ptid));
}


int rt_compare_ptid_ext(const void *arg1, const void *arg2)
{
 PDBEXT *rec1;
 PDBEXT *rec2;

 rec1 = (PDBEXT *)arg1;
 rec2 = (PDBEXT *)arg2;
 return (strcmp(rec1->ptid, rec2->ptid));
}


int rt_gettotsize(PDB *record)
{ /*  This function returns the number of bytes used by a variable.
  */
 int itemsize;
 int index;

 itemsize = pdbtype[record->vartype].size;
 for (index = 0; index < record->ndim; ++index)
          {
                   itemsize *= record->dim[index];
          }
 return itemsize;
}


void *rt_getrecord(char *ptid)
{ /* This function returns a pointer to the PDB record having a PTID of ptid.

          Returns pointer value, on success.
          Returns NULL pointer , on failure
  */

 PDB   searchrec;
 void *recptr;

 /* check original PDB */
 strcpy(searchrec.ptid, ptid);
 recptr = bsearch((void *)&searchrec, (void *)pdbrec, (size_t)maxrec, (size_t)sizeof(PDB), rt_compare_ptid);
 if (recptr != NULL)
   return recptr;

 return NULL;
}
```

```
PDBEXT *rt_getrecext(char *ptid)
{ /* This function returns a pointer to the PDB extension record having a PTID of ptid.

        Returns pointer value, on success.
        Returns NULL pointer , on failure
 */

 PDBEXT  searchrec;
 PDBEXT  *recptr;

 /* check original PDB extension */
 strcpy(searchrec.ptid, ptid);
 recptr = bsearch((void *)&searchrec, (void *)pdbext, (size_t)maxrec,
                                                    (size_t)sizeof(PDBEXT), rt_compare_ptid_ext);
 if (recptr != NULL)
   return recptr;

 return NULL;
}


int rt_getextrec_fromfile(char *ptid, PDBEXT *recext)
{ /* This function fills a PDB extension record for a PTID of ptid.

        Returns    On success,  0
        Returns    On failure, -1
 */

 PDB  searchrec;
 PDB  *recptr;
 long       pos;

 /* check original PDB in memory to find position in file */
 strcpy(searchrec.ptid, ptid);
 recptr = bsearch((void *)&searchrec, (void *)pdbrec, (size_t)maxrec,
                                                    (size_t)sizeof(PDB), rt_compare_ptid);
 if (recptr != NULL)
   {
        pos = recptr - pdbrec;
        fseek(pdbextfile, pos * sizeof(PDBEXT), SEEK_SET);
        fread(recext, sizeof(PDBEXT), 1, pdbextfile);
    return 0;
   }

 return -1;
}


int rt_setglobflag(char parco)
{ /* This function sets the generic flag for a
        partition given its partition code.

        Returns    On success,  0
        Returns    On failure, -1
 */

 char i;

 for (i = 0; i < totpart; ++i)
   {
        if (globinfo[i].partition == parco)
                {
                        globinfo[i].flag = 1;
                        return 0;
                }
        }
 return -1;
}


char *rt_getglobname(char parco)
{ /* This function returns the name of the partition given the partition code.

        Returns On success, pointer to global name
```

```
                Returns On failure, pointer to null string
 */

 char i;

 for (i = 0; i < totpart; ++i)
    {
                if (globinfo[i].partition == parco)
                        return globinfo[i].ptid;
                }
 return "";
}


long rt_getglobsize(char parco)
{ /* This function returns the size of the partition given the partition code.

                Returns On success, global partition size
                Returns On failure, 0
 */

 char i;

 for (i = 0; i < totpart; ++i)
    {
                if (globinfo[i].partition == parco)
                        return globinfo[i].size;
                }
 return 0;
}


unsigned char rt_getparco(char *vname)
{ /* This function returns the partition number of a given variable.

                Returns global partition, on success.
                Returns 0, on failure.
 */
 PDB *record;

 record = rt_getrecord(vname);
 if (record)
    return record->partition;
 else
    return 0;
}


long rt_getoffset(char *vname)
{ /* This function returns the offset of a given variable.

                Returns global offset, on success.
                Returns -1, on failure.
 */
 PDB *record;

 record = rt_getrecord(vname);
 if (record)
    return record->addr;
 else
    return -1;
}


unsigned char rt_getpartition(char *vname)
{ /* This function returns the partition number given the name of the partition.

                Returns global partition, on success.
                Returns 0, on failure.
 */

 char i;

 for (i = 0; i < totpart; ++i)
    {
```

```
            if (!stricmp(globinfo[i].ptid,vname))
                    return globinfo[i].partition;
        }
 return 0;
}


unsigned char rt_getpartype(char parco)
{ /* This function returns the variable type of the given partition code

        Returns global partition, on success.
        Returns 0, on failure.
 */

 char i;

 for (i = 0; i < totpart; ++i)
   {
        if (globinfo[i].partition == parco)
                return globinfo[i].vartype;
        }
 return 0;
}


char *rt_get_globaddr(char parco)
{ /* This function returns the address of the GLOBAL given the partition.

        Returns global partition address, on success.
        Returns NULL, on failure..
 */

 char i;
 char tstr[12];

 for (i = 0; i < totpart; ++i)
   {
        if (globinfo[i].partition == parco)
                {
                        return (char *)globinfo[i].addr;
                }
        }
 return NULL;
}


void *rt_getaddress(char *vname)
{ /* This function returns a pointer to the specified variable

        Returns    On success, the address of the variable.
                                        On failure, NULL.
 */

 PDB   *record;
 char  *vptr;

 if (record = rt_getrecord(vname))
        {
         vptr = rt_get_globaddr(record->partition);
         if (vptr)
                return (vptr + record->addr);
        }
 return NULL;
}


char rt_get_vartype(char *vname)
{ /* This function retrieves the type of the specified variable.

        Returns    On success, the variable type.
                                        On failure, TYPE_NONE.
 */
 PDB   *record;

 if (record = rt_getrecord(vname))
```

```
        return record->vartype;
  else
        return TYPE_NONE;
}


void rt_get_format(char *vname, char *formstr)
{ /* This function retrieves the type of the specified variable.

        Returns     On success, copy format string into formstr
                                        On failure, blank out formstr.
  */
  PDB   *record;

  formstr[0] = 0;
  if (record = rt_getrecord(vname))
    strcpy(formstr, record->format);
  return;
}


char *rt_get_aval2(void *vptr, int index, char vartype, char *format, char *rtnval)
{
  char              *chptr;
  short             *shptr;
  long              *inptr;
  float             *fiptr;
  double      *dbptr;

  if (vptr == NULL)
    strcpy(rtnval, INVALID);
  else
    {
        rtnval[0] = 0;
        switch (vartype)
                                {
                                  case TYPE_NONE:
                                        chptr = vptr;
                                        if (format[0] != '%')
                                          strcpy(format, "%d");
                                        sprintf(rtnval, format, chptr[index]);
                                        break;
                                  case TYPE_LOGICAL:
                                        chptr =vptr;
                                        sprintf(rtnval, "%d", chptr[index]);
                                        break;
                                  case TYPE_UCHAR:
                                        chptr = vptr;
                                        if (format[0] != '%')
                                          strcpy(format, "%u");
                                        if (strcmp(format, "%s"))
                                                sprintf(rtnval, format, chptr[index]);
                                        else
                                          sprintf(rtnval, format, &chptr[index]);
                                        break;
                                  case TYPE_SHORT:
                                        shptr = vptr;
                                        if (format[0] != '%')
                                          strcpy(format, "%d");
                                        sprintf(rtnval, format, shptr[index]);
                                        break;
                                  case TYPE_LONG:
                                        inptr = vptr;
                                        if (format[0] != '%')
                                          strcpy(format, "%ld");
                                        sprintf(rtnval, format, inptr[index]);
                                        break;
                                  case TYPE_FLOAT:
                                        fiptr = vptr;
                                        if (format[0] != '%')
                                          strcpy(format, "%.5G");
                                        sprintf(rtnval, format, fiptr[index]);
                                        break;
                                  case TYPE_DOUBLE:
                                        dbptr = vptr;
```

```
                                        if (format[0] != '%')
                                          strcpy(format, "%.5G");
                                        sprintf(rtnval, format, dbptr[index]);
                                        break;
                                case TYPE_GLOB_IC:
                                case TYPE_GLOB_CON:
                                case TYPE_GLOB_REG:
                                        break;
                                }
        }
 return rtnval;
}


char *rt_get_avalue(char *vname, int index)
{ /* This function returns a pointer to a string which contains the
     ASCII formatted value of the specified variable.

             Returns    On success, the value string.
                                        On failure, the INVALID string.
 */
 char                           formstr[FORMATSIZE + 1];
 static char rtnval[32];

 void                   *vptr;
 char                   vartype;

 vptr = rt_getaddress(vname);
 strcpy(rtnval, INVALID);
 if (vptr)
        {
         vartype = rt_get_vartype(vname);
         rt_get_format(vname, formstr);
         rt_get_aval2(vptr, index, vartype, formstr, rtnval);
        }
 return rtnval;
}


int rt_set_value(char *vname, int index, double value)
{ /* This function sets the named variable to a specified value.

             Returns    On success, 0
                                        On failure, -1
 */
 int                            rtnval;
 void                   *vptr;
 char                   vartype;
 char                           format[10];
 char                   *chptr;
 short                  *shptr;
 long                   *inptr;
 float                  *fiptr;
 double         *dbptr;

 rtnval = -1;
 vptr = rt_getaddress(vname);
 if (vptr)
        {
         vartype = rt_get_vartype(vname);
         switch (vartype)
                                {
                                 case TYPE_NONE:
                                        break;
                                 case TYPE_LOGICAL:
                                        chptr =vptr;
                                        chptr[index] = (char)value;
                                        rtnval = 0;
                                        break;
                                 case TYPE_UCHAR:
                                        chptr = vptr;
                                        chptr[index] = (unsigned char)value;
                                        rtnval = 0;
                                        break;
                                 case TYPE_SHORT:
```

```
                                        shptr = vptr;
                                        shptr[index] = (short)value;
                                        rtnval = 0;
                                        break;
                                case TYPE_LONG:
                                        inptr = vptr;
                                        inptr[index] = (long)value;
                                        rtnval = 0;
                                        break;
                                case TYPE_FLOAT:
                                        flptr = vptr;
                                        flptr[index] = (float)value;
                                        rtnval = 0;
                                        break;
                                case TYPE_DOUBLE:
                                        dbptr = vptr;
                                        dbptr[index] = value;
                                        rtnval = 0;
                                        break;
                                case TYPE_GLOB_IC:
                                case TYPE_GLOB_CON:
                                case TYPE_GLOB_REG:
                                        break;
                                }
                }
 return rtnval;
}


int rt_load_constants(char *filename)
{
 /* Load CONSTANT values from a file.

                Return:     On success, 0
                                                On failure, -1
 */
 int                    i;
 int                    items2read;
 int                    totread;
 int                    pos;
 FILE           *fin;
 int                    rtnval;
 ICHEADER               nextglob;
 long                   size;
 SIGNHDR  signhdr;

 rtnval = -1;
 if (fin = fopen(filename, "rb"))
   {
                        /* read in signature header */
                        pos = sizeof(signhdr);
                        fseek(fin, -pos, SEEK_END);
                        items2read = fread(&signhdr, sizeof(signhdr), 1, fin);
                        if (!items2read || strcmp(signhdr.signstr, "CONSTANTS"))
                          {
                                return -1;
                          }

                        printf("Loading %s\n", filename);
                        items2read = signhdr.totglobs;
                        totread = 0;

                        /* read in global data based on header information */
                        while (totread < items2read)
                            {
                                        ++totread;

                                        /* read in the next global partition header */
                                        pos += sizeof(ICHEADER);
                                        fseek(fin, -pos, SEEK_END);
                                        fread(&nextglob, sizeof(ICHEADER), 1, fin);

                                        /* find address of matching global */
                                        pos += nextglob.size;
                                        for (i = 0; i < totpart; ++i)
```

```
                                        if (    (globinfo[i].partition == nextglob.partition)
                                                && (globinfo[i].vartype == TYPE_GLOB_CON)
                                           )
                                                { /* check partition sizes */
                                                 if (globinfo[i].size == nextglob.size)
                                                        {
                                                                size = globinfo[i].size;
                                                                }
                                                 else
                                                        if (globinfo[i].size < nextglob.size)
                                                          { /* read in smaller size */
                                                                size = globinfo[i].size;
                                                                printf("Program size for %s is less than IC file.\n", globinfo[i].ptid);
                                                               }
                                                        else
                                                                { /* read in smaller size */
                                                                 size = nextglob.size;
                                                                 printf("Program size for %s is greater than IC file.\n", globinfo[i].ptid);
                                                                }
                                                 fseek(fin, -pos, SEEK_END);
                                                 fread((void *)globinfo[i].addr, size, 1, fin);
                                                 break;
                                                }
                                        if (i >= totpart)
                                          { /* Global partition no longer used. Skip over it. */
                                                printf("%s no long used. Skipping...\n", nextglob.ptid);
                                          }
                                }
                        rtnval = 0;
                }
        else
          {
                        printf("Unable to load %s\n", filename);
                        fflush(stdout);
                }
        return rtnval;
}


int rt_save_constants(char *filename)
{ /* This function saves all CONSTANTS partitions to a file.

                Return:     On success, 0
                                        On failure, -1
        */
        int             i;
        FILE            *fout;
        int             rtnval;
        SIGNHDR  signhdr;

        if (fout = fopen(filename, "wb"))
          {
                        signhdr.totglobs = 0;
                        for (i = 0; i < totpart; ++i)
                                if (globinfo[i].vartype == TYPE_GLOB_CON)
                                  {
                                        fwrite((void *)globinfo[i].addr, globinfo[i].size, 1, fout);
                                        fwrite(&globinfo[i], sizeof(ICHEADER), 1, fout);
                                        ++signhdr.totglobs;
                                  }

                        /* write out signature header */
                        strcpy(signhdr.signstr, "CONSTANTS");
                        fwrite(&signhdr, sizeof(signhdr), 1, fout);

                        fclose(fout);
                        rtnval = 0;
                }
        else
          {
                        printf("Unable to save %s\n", filename);
                        fflush(stdout);
                        rtnval = -1;
                }
        return rtnval;
```

```
}


int rt_setswitchpos(char *ptid, int pos)
{ /*        This function sets the position of the switch identified by ptid.

            See the notes associated with the rt_getswitchpos()function.
 */
 char  *addr;
 PDB   *record;
 int   totpos;
 int   i;
 int   index;

 index = pos - 1;
 if (addr = rt_getaddress(ptid))
   {
    record = rt_getrecord(ptid);
    if ((record->ndim == 1) && (record->vartype == TYPE_LOGICAL) )
            { /* Must be a single dimension array of type char */
                    totpos = record->dim[0];

                    /* set value of all items in switch array */
                    for (i = 0; i < totpos; ++i)
                       if (i == index)
                                addr[i] = 1;
                       else
                                addr[i] = 0;
            }
   }
 return 0;
}


int rt_getswitchpos(char *poslist, char *ptid, char *posname)
{ /*        This function retrieves the current position for the switch
            identified by ptid.

            Input/Output (I/O) hardware switch variables are single dimension
            arrays with each item in the array representing a switch position.
            I/O hardware should be wired such that only one array index is non-zero
            when a hardware switch is in any given position. When a switch position
            is selected, the corresponding array item value is non-zero. When a
            switch position is not selected, its array item value is zero.

            Each switch variable should have a position list assigned to it.
            This list is defined using the @swpos() declarator in the full description
            field of the data base. Position labels are associated with array indexes
            based on their order in the list. The first item in the list represents
            the position label for the completely non-wired position of the switch.
            The remaining items in the list are assigned to the items of the switch
            array in order of their appearance.

            Example 1: @swpos("FAILED" OFF SLOW FAST)

                                Represents a three position hardware switch whose first
                                position is OFF, second position is SLOW and third position
                                is FAST. All positions of the hardware switch are wired to
                                the I/O system. If the switch is not in one of these
                                positions, the associated wiring or I/O card has failed.

            Example 2: @swpos(OFF SLOW FAST)

                                Represents a three position hardware switch whose first
                                position is OFF, second position is SLOW and third position
                                is FAST. The first position of the switch is not wired. The
                                I/O hardware is only attached to the second and third
                                positions of the switch. When the application source code
                                does not detect non-zero values on the second and third
                                positions, it assumes that the first position of the switch
                                is selected,

   Returns: 0, if switch ptid found
                                -1, if switch ptid not found
 */
```

```
  char *addr;
  char *cptr;
  int   rtnval;
  PDB  *record;
  int   totpos;
  int   i;

  rtnval -1;
  posname[0] = 0;
  if (addr = rt_getaddress(ptid))
    {
      record = rt_getrecord(ptid);
      if ((record->ndim == 1) && (record->vartype == TYPE_LOGICAL) )
              { /* Must be a single dimension array of type char */
                      cptr = posname;
                      totpos = record->dim[0];
                      for (i = 0; i < totpos; ++i)
                        { /* check all switch positions */
                                if (addr[i])
                                  {
                                          rt_lindex(poslist, i + 1, cptr);
                                          cptr += strlen(cptr);
                                          *cptr = ' ';
                                          ++cptr;
                                          *cptr = 0;
                                  }
                        }
                      if (!posname[0])          /* Must be in non-wired position */
                        rt_lindex(poslist, 0, posname);
              }
    }
  return rtnval;
}


char *rt_getposition(char *ptid, int array_index)
{ /*      This function retrieves the enumerated value of a variable.
          The positions are enumerated in the data base by placing a string of
          the following format in the full description field.

                  @pos(label0, label1, label2, ...)

          When the variable has a value of 0 string label0 will be returned.
          When the variable has a value of 1 string label1 will be returned.
          When the variable has a value of 2 string label1 will be returned.
          etc...
  */
  static  char posname[256];
  char   *addr;
  char    value;
  char    vartype;
  char    poslist[256];

  posname[0] = 0;
  vartype = rt_get_vartype(ptid);
  if (vartype == TYPE_NONE)
          return INVALID;
  else
    if (vartype == TYPE_UCHAR)
            {
              addr = rt_getaddress(ptid);
              if (addr == NULL)
                      return INVALID;
              rt_getlist("@rfpos", ptid, poslist);
              if (poslist[0])
                      { /* retrieve the correct item from the list */
                        rt_lindex(poslist, addr[array_index], posname);
                      }
              else
                      {
                        rt_getlist("@mfpos", ptid, poslist);
                        if (poslist[0])
                                { /* retrievé the correct item from the list */
                                  rt_lindex(poslist, addr[array_index], posname);
                                }
```

```
                        else
                                {
                                 rt_getlist("@swpos", ptid, poslist);
                                 if (poslist[0])
                                        rt_getswitchpos(poslist, ptid, posname);
                                }
                        }
                }
    else
                if (vartype == TYPE_LOGICAL)
                  {
                        rt_getlist("@swpos", ptid, poslist);
                        if (poslist[0])
                          rt_getswitchpos(poslist, ptid, posname);
                  }
  if (!posname[0])
    strcpy(posname, rt_get_avalue(ptid, array_index));
  return posname;
}


int rt_setposition(char *ptid, char *posname, int array_index)
{ /* This function sets the value of an enumerated variable to the value corresponding
     to posname.

     Returns:  0 on success,
                        -1 on failure
  */
  char   *addr;
  char   value;
  char   vartype;
  char   poslist[256];

  value = -1;
  vartype = rt_get_vartype(ptid);
  if (vartype == TYPE_UCHAR)
    {
                addr = rt_getaddress(ptid);
                if (addr != NULL)
                  {
                   rt_getlist("@rfpos", ptid, poslist);
                   if (!poslist[0])
                        {
                          rt_getlist("@mfpos", ptid, poslist);
                          if (!poslist[0])
                                rt_getlist("@swpos", ptid, poslist);
                        }
                   if (poslist[0])          /* retrieve the position of the item in the list */
                        value = rt_lsearch(poslist, posname);
                  }
    }
  else
    if (vartype == TYPE_LOGICAL)
                {
                  rt_getlist("@swpos", ptid, poslist);
                  if (poslist[0])
                        value = rt_lsearch(poslist, posname);
                  if (value != -1)
                        {
                          rt_setswitchpos(ptid, value);
                          return 0;
                        }
                }
  if (value == -1)
    return -1;
  addr[array_index] = value;
  return 0;
}


int rt_getrange(char *ptid, char *poslist)
{ /*            This function retrieves an ASCII formatted list of permitted values
                for a variable.

                For enumerated variables (i.e. those with @rfpos strings in the full
```

```
        description), the discrete position list will be returned.

        For floating point variables this will be a range specified by two
        floating point numbers. The first number being the low limit and the
        second consisting of the high limit. The low and high limits are
        defined in the data base by placing a string of the following format
        in the full description field.

                @limit(low_limit, high_limit)

        Logical variables will return a list consisting of "false true".

        For malfunction variables (i.e. those with @mfpos strings in the full
        description), the list of discrete malfunction states will be returned.

        For I/O switch variables (i.e. those with @swpos strings in the full
        description), the list of posible switch positions will be returned.

        Specifying any other type of variable will return an empty list.

        Returns:    0 for no specific range
                        1 for discrete, enumerated positional ranges
                        2 for low-high, variable ranges
                        3 for booleans, "false true" ranges
                        4 for discrete, malfunction states
                        5 for I/O switch positions
*/
static char posname[256];
char  *addr;
char  value;
char  vartype;

rt_getlist("@rfpos", ptid, poslist);
if (poslist[0])
        return 1;
rt_getlist("@limit", ptid, poslist);
if (poslist[0])
        return 2;
rt_getlist("@mfpos", ptid, poslist);
if (poslist[0])
        return 4;
rt_getlist("@swpos", ptid, poslist);
if (poslist[0])
        return 5;
if (rt_get_vartype(ptid) == TYPE_LOGICAL)
  {
        strcpy(poslist, "false true");
        return 3;
  }
return 0;
}


int rt_func_info(int startype, int endtype, char *ptid, int lindex, char *info)
{ /*        This subroutine retrieves information for:
      1) remote functions
      2) malfunctions
      3) I/O switch positions

        The "info" character buffer is filled with a string of the following format:

   ptid  feedbk_var  curval  rangetype  rangeval1  rangeval2...rangevalN  desc
   |  |       |      |       |          |                                 |
   var.  name of     current  range type  range values                    brief
   name  feedback    value    1=discrete  total number  depends on range type  description
     variable    of ptid  2=low-high
                          3=booleans
                          4=malf states
                          5=switch pos.

        Returns:    0, if no remote information found
                        1, if information found
*/
char  feedbk_var[PTIDSIZE + 20];
char  poslist[256];
```

```
PDB   record;
PDBEXT recext;
int   rangetype;
int           rtnval;

info[0] = 0;
rtnval  = 0;
if (!rt_getextrec_fromfile(ptid, &recext))
  {
          rangetype = rt_getrange(ptid, poslist);
          if ( (rangetype >= startype) && (rangetype <= endtype) )
            {
                  sprintf(feedbk_var, "%s_fb", ptid);
                  if (rt_get_vartype(feedbk_var) == TYPE_NONE)
                    strcpy(feedbk_var, ptid);
                  sprintf(info, "%s %s %s %d %s {%s}",
                                        ptid,
                                        feedbk_var,
                                        rt_get_avalue(ptid, lindex),
                                        rangetype,
                                        poslist,
                                        recext.brief
                                );
                  rtnval = 1;
            }
  }
 return rtnval;
}


/* librt.h
# ----------------------------------------------------------------------
# Copyright (C) 1996 Dennis R. LaBelle (drlabell@albany.net) All Rights Reserved.
#
# PERMISSION IS GRANTED TO DISTRIBUTE THIS SOFTWARE FREELY, WITH THE
# EXCEPTION THAT ONE MAY NOT CHARGE FOR IT OR INCLUDE IT WITH SOFTWARE
# WHICH IS SOLD. Permission to use, copy, modify, and distribute this software and
# its documentation for educational, research, internal corporate and non-profit
# purposes, without fee, and without a written agreement is hereby granted for all
# cases that do not conflict with the restriction in the first sentence of this
# paragraph, provided that the above copyright notice, this paragraph, and the
# following three paragraphs appear in all copies.
#
# Permission to incorporate this software into commercial products may be obtained
# from the author (e-mail drlabell@albany.net).
#
# IN NO EVENT SHALL THE AUTHOR BE LIABLE TO ANY PARTY FOR DIRECT,
# INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF
# THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHOR HAS
# BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# THE AUTHOR SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS"
# BASIS, AND THE AUTHOR HAS NO OBLIGATION TO PROVIDE MAINTENANCE,
# SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
# ----------------------------------------------------------------------
*/
#include "pdb.h"

#define INVALID "------"

/* prototype definitions for librt.c */
char *rt_getglobname(char parco);
void rt_strlwr(char *string);
void rt_rtrim(char *string);
int rt_new(char *fname1, char *fname2);
int rt_readpdb(void *buffer, long totrecs);
int rt_readpdbext(void *buffer, long totrecs);
void *rt_makelocalpdb();
void *rt_makelocalpdbext();
int rt_compare_ptid(const void *arg1, const void *arg2);
int rt_compare_ptid_ext(const void *arg1, const void *arg2);
int rt_gettotsize(PDB *record);
void *rt_getrecord(char *ptid);
```

```
PDBEXT *rt_getrecext(char *ptid);
unsigned char rt_getpartition(char *vname);
int rt_Clinearaddress(char *ptid, int ndim, long *dim);
void *getglobaddr(char);
int rt_set_value(char *vname, int index, double value);
unsigned char rt_getparco(char *vname);
void *rt_getaddress(char *vname);
char *rt_get_aval2(void *vptr, int index, char vartype, char *format, char *rtnval);
char *rt_getposition(char *ptid, int array_index);

/* variable definitions */
#ifdef LIBRTSRC

char            odspath[256];
char            libstr1[256];
long            maxrec;
PDB                     *pdbrec;
PDBEXT          *pdbext;
long            totpart;
FILE            *pdbextfile;

#else

extern char   odspath[256];     /* path to Official Development System */
extern char   libstr1[256];     /* temporary string */
extern long   maxrec;                   /* the number of records in PDB */
extern PDB    *pdbrec;                  /* pointer to PDB array */
extern PDBEXT *pdbext;                          /* pointer to PDB extension array */
extern long   totpart;                  /* total global partitions found in PDB */
extern FILE   *pdbextfile;              /* PDB extension file stream */

#endif

/*
        rtsched.c - Real Time Scheduler for RTProE

   This file contains a real time scheduler for Windows.
   The main() function definition is contained within this file.

# ----------------------------------------------------------------
# Copyright (C) 1996 Dennis R. LaBelle (drlabell@albany.net) All Rights Reserved.
#
# PERMISSION IS GRANTED TO DISTRIBUTE THIS SOFTWARE FREELY, WITH THE
# EXCEPTION THAT ONE MAY NOT CHARGE FOR IT OR INCLUDE IT WITH SOFTWARE
# WHICH IS SOLD. Permission to use, copy, modify, and distribute this software and
# its documentation for educational, research, internal corporate and non-profit
# purposes, without fee, and without a written agreement is hereby granted for all
# cases that do not conflict with the restriction in the first sentence of this
# paragraph, provided that the above copyright notice, this paragraph, and the
# following three paragraphs appear in all copies.
#
# Permission to incorporate this software into commercial products may be obtained
# from the author (e-mail drlabell@albany.net).
#
# IN NO EVENT SHALL THE AUTHOR BE LIABLE TO ANY PARTY FOR DIRECT,
# INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF
# THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHOR HAS
# BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# THE AUTHOR SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS"
# BASIS, AND THE AUTHOR HAS NO OBLIGATION TO PROVIDE MAINTENANCE,
# SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
# ----------------------------------------------------------------

*/

#include <Windows.H>
#include <StdIO.H>
#include <StdLib.H>
#include <ConIO.H>
#include <String.H>
#include <signal.h>
#include <limits.h>
```

```
#include "pdb.h"
#include "rtsched.h"

/* function prototypes */
int rt_getfreq();
int callframe(int frameno);

COORD XYpos;
char  line[256];

/* Run status values */
#define FRZ                             0
#define RUN                             1
#define RST                             3
#define TERM            4
#define SYNC            5


#define WERR(who,where)   {sprintf(Buf,"ERROR: %s returned %u, line: %u", who, GetLastError(), __LINE__);\
                sprintf(Buf2,"From within %s", where);\
                MessageBox(hwndMain, Buf, Buf2, MB_OK);}

void printstr();


HWND    hwndMain;               // Main hwnd.  Needed in callback
DWORD   framesize; // size, in milliseconds, of one frame

int                     laststatus;
int                     totoverruns;
int                     newover;
int             overtime;
DWORD   curtime;
DWORD   starttime;
DWORD   stoptime = 0;
char    tempstr[256];                           /* Temp storage to build strings */
float   *cpu_avg;                               /* average CPU usage array */
char    rt_curIC[5];                            /* name of current Initial Condition */
UINT    wTimerID;
CHAR    Buf[80];
CHAR    Buf2[80];
HANDLE  FRAMESTART;                             /* Event to signal start of a frame */
HANDLE  FRAMEDONE;                              /* Event to signal that frame work completed */
HANDLE  RTthread;
DWORD   RTloopID;
char    lastspeed;                      /* last simulation speed requested */
char    loaddrive[_MAX_DRIVE];          /* drive location for current load */
char    loaddir[_MAX_DIR];              /* directory location for current load */
char    loadfname[_MAX_FNAME];          /* load file name (without extension) */
char    odspath[256];                           /* Official Development System path */
int                     maxover;                                /* maximum allowed overruns before halting program */
int                     freq;                                   /* number of time slices (frames) per second*/

//---------------------------------------------------------------
// Function prototypes
//---------------------------------------------------------------

void CALLBACK FrameMsg(UINT, UINT, DWORD, DWORD, DWORD);
void rtctrl();

void rt_strupr(char *string)
{ /* This function converts the specified string to upper case.
  */
  char *cptr;

  for (cptr = string; *cptr; ++cptr)
     *cptr = toupper(*cptr);
}


int load_constants(char *filename)
{
  /* Load CONSTANT values from a file.
```

```
        Return:   On success, 0
                                        On failure, -1
*/
int             i;
int             items2read;
int             totread;
int             pos;
FILE    *fin;
int             rtnval;
ICHEADER        nextglob;
long            size;
SIGNHDR  signhdr;

rtnval = -1;
if (fin = fopen(filename, "rb"))
  {
                /* read in signature header */
                pos = sizeof(signhdr);
                fseek(fin, -pos, SEEK_END);
                items2read = fread(&signhdr, sizeof(signhdr), 1, fin);
                if (!items2read || strcmp(signhdr.signstr, "CONSTANTS"))
                  {
                        return -1;
                  }

                printf("Loading constants from %s\n", filename);
                items2read = signhdr.totglobs;
                totread = 0;

                /* read in global data based on header information */
                while (totread < items2read)
                   {
                                ++totread;

                                /* read in the next global partition header */
                                pos += sizeof(ICHEADER);
                                fseek(fin, -pos, SEEK_END);
                                fread(&nextglob, sizeof(ICHEADER), 1, fin);

                                /* find address of matching global */
                                pos += nextglob.size;
                                for (i = 0; i < TOTPART; ++i)
                                  if (      (rt_globinfo[i].partition == nextglob.partition)
                                                && (rt_globinfo[i].vartype == TYPE_GLOB_CON)
                                    )
                                        { /* check partition sizes */
                                          if (rt_globinfo[i].size == nextglob.size)
                                                {
                                                        size = rt_globinfo[i].size;
                                                        }
                                          else
                                                if (rt_globinfo[i].size < nextglob.size)
                                                  { /* read in smaller size */
                                                        size = rt_globinfo[i].size;
                                                        printf("Program size for %s is less than IC file.\n", rt_globinfo[i].ptid);
                                                      }
                                                else
                                                        { /* read in smaller size */
                                                          size = nextglob.size;
                                                          printf("Program size for %s is greater than IC file.\n", rt_globinfo[i].ptid);
                                                        }
                                          fseek(fin, -pos, SEEK_END);
                                          fread((void *)rt_globinfo[i].addr, size, 1, fin);
                                          break;
                                        }
                                if (i >= TOTPART)
                                  { /* Global partition no longer used. Skip over it. */
                                        printf("%s no long used. Skipping...\n", nextglob.ptid);
                                  }
                   }
          rtnval = 0;
        }
else
  {
                printf("Unable to load %s\n", filename);
```

```
                fflush(stdout);
        }
 return rtnval;
}


int ICsave(char *filename)
{ /* Save initial condition variables to a file.

        Return:  On success, 0
                                        On failure, -1
 */
 int  i;
 FILE *fout;
 int        rtnval;

 if (fout = fopen(filename, "wb"))
   {
     printf("Saving %s\n", filename);
     for (i = 0; i < TOTPART; ++i)
       if (rt_globinfo[i].vartype == TYPE_GLOB_IC)
                {
                        fwrite(&rt_globinfo[i], sizeof(ICHEADER), 1, fout);
                        fwrite((void *)rt_globinfo[i].addr, rt_globinfo[i].size, 1, fout);
                }
     fclose(fout);
     rtnval = 0;
   }
 else
   {
        printf("Unable to save %s\n", filename);
        fflush(stdout);
        rtnval = -1;
   }
 return rtnval;
}


int ICload(char *filename)
{
 /* Load initial condition variables from a file.

        Return:  On success, 0
                                        On failure, -1
 */
 int            i;
 int            itemsread;
 FILE       *fin;
 int            rtnval;
 ICHEADER       nextglob;
 long           size;
 long           pos;

 if (fin = fopen(filename, "rb"))
   {
     printf("Loading %s\n", filename);

        /* read header for first global */
        itemsread = fread(&nextglob, sizeof(ICHEADER), 1, fin);

        /* read in global data based on header information */
        pos = 0;
        while (itemsread)
          {
                pos += sizeof(ICHEADER);

                /* find address of matching global */
                for (i = 0; i < TOTPART; ++i)
                  if (      (rt_globinfo[i].partition == nextglob.partition)
                                && (rt_globinfo[i].vartype == TYPE_GLOB_IC)
                                )
                                { /* check partition sizes */
                                  if (rt_globinfo[i].size == nextglob.size)
                                        {
                                                size = rt_globinfo[i].size;
```

```
                                                }
                                        else
                                                if (rt_globinfo[i].size < nextglob.size)
                                                  { /* read in smaller size */
                                                        size = rt_globinfo[i].size;
                                                        printf("Program size for %s is less than IC file.\n", rt_globinfo[i].ptid);
                                                      }
                                                else
                                                        { /* read in smaller size */
                                                          size = nextglob.size;
                                                          printf("Program size for %s is greater than IC file.\n", rt_globinfo[i].ptid);
                                                        }
                                        fseek(fin, pos, SEEK_SET);
                                        fread((void *)rt_globinfo[i].addr, size, 1, fin);
                                        break;
                                      }
                                if (i >= TOTPART)
                                  { /* Global partition no longer used. Skip over it. */
                                        printf("%s no long used. Skipping...\n", nextglob.ptid);
                                      }

                                /* get next global partition header */
                                pos += nextglob.size;
                                fseek(fin, pos, SEEK_SET);
                                itemsread = fread((void *)&nextglob, sizeof(ICHEADER), 1, fin);
                        }
                fclose(fin);
                rtnval = 0;
              }
 else
   {
                printf("Unable to load %s\n", filename);
                fflush(stdout);
                rtnval = -1;
        }
 return rtnval;
}


void CALLBACK FrameMsg(UINT wTimerID, UINT wMsg, DWORD dwUser, DWORD dw1, DWORD dw2)
{
 /* activate the Real Time Thread */
 ++overruns;
 PulseEvent(FRAMESTART);

 /* check whether we have a run-away process */
 if ((overruns > maxover) && (status == RUN))
        {
                printf("%d overruns\n", overruns);
    printf("Going to FREEZE\n");
                fflush(stdout);
                status = FRZ;

                /* terminate the original, misbehaving thread */
                TerminateThread(RTthread, 1);

                /* start a new real-time loop thread */
            if ((RTthread = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)rtctrl,
                                NULL, 0, &RTloopID)) == NULL)
                  {
                                sprintf(tempstr,"Unable to create real-time loop thread");
                                MessageBox(hwndMain,(LPCSTR)tempstr,
                                                "rtsched Error!",MB_APPLMODAL | MB_ICONSTOP);
                  }

                overruns = 0;
        }
}


void cleanup()
{
 /* Even though Windows destroys timers created by this app, destroying
        the things the application creates is good practice. */
 if (wTimerID)
```

```
    {
        timeKillEvent(wTimerID);
        timeEndPeriod(1);
      }
  exit(0);
}

void rtctrl()
{
  int       i;
  float cpu_usage;
  UINT      newsize;
  int       result;
  int       waitframes;

  while (1)
    {
                if (WaitForSingleObject(FRAMESTART, INFINITE) != WAIT_OBJECT_0)
                  {
                        sprintf(tempstr,"Wait for FRAMESTART failed");
                        MessageBox(hwndMain,(LPCSTR)tempstr,
                                                "rtsched Error!",MB_APPLMODAL | MB_ICONSTOP);
                        cleanup();
                  }

                /* Check for IC read/write requests */
                if (icrw == 'W')
                  {
                        sprintf(tempstr, "%s%s%s", loaddrive, loaddir, icname);
                        ICsave(tempstr);
                        strcpy(icname, rt_curIC);
                        icrw = 0;
                      }
                else
                  if (icrw == 'R')
                        {
                                status = RST;
                                icrw = 0;
                        }

                /* Check for speed change requests */
                if (slofac == -1 || slofac == 0)
                        slofac = 1;
                if (slofac < -100)
                  slofac = -100;
                else
                  if (slofac > 100)
                                slofac = 100;
                if (slofac != lastspeed)
                        {
                          lastspeed = slofac;
                          if (slofac < 0)
                                newsize = framesize / abs(slofac);
                          else
                                newsize = framesize * slofac;
                          timeKillEvent(wTimerID);          /* remove current timer event */
                          wTimerID = timeSetEvent((UINT)newsize, 1, FrameMsg, (DWORD)hwndMain, TIME_PERIODIC);
                          if (!wTimerID)
                                MessageBox(hwndMain,"Unable to set up timer.",
                                        "rtsched Error!",MB_APPLMODAL | MB_ICONSTOP);
                        }

                curtime  = timeGetTime();
                stoptime = curtime + framesize;

                do
                        {
                          switch (status)
                                        {
                                                case RUN:
                                                        if (status != laststatus)
                                                          {
                                                            printf("Going to RUN\n");
```

```
                        fflush(stdout);
                    }
                ++totframes;
                curframe = (totframes % freq) + 1;
                starttime = timeGetTime();
                callframe(curframe);
                curtime = timeGetTime();
                cpu_avg[curframe - 1] = (float)(curtime - starttime);
                if (curframe == 1)
                    {
                     ++seconds;
                     for (i = 0, cpu_usage = (float)0.0; i < freq; ++i)
                            cpu_usage += cpu_avg[i];
                     avg_cpu_usage = cpu_usage / (float)10.0;
                    }
                /*
                if (curtime > stoptime)
                    {
                            overtime  = curtime - starttime;
                            newover   = overtime / framesize;
                            overruns += newover;
                            printf("overruns = %d\n", overruns);
                            stoptime += (newover * framesize);
                    }
                */
                if (overruns > 0)
                    {
                            --overruns;
                            if (overruns)
                                    ++catchups;
                    }
                break;
        case RST:
printf("RESETing\n");
                fflush(stdout);
                laststatus = status;
                totframes = 0;
                overruns  = 0;
                catchups  = 0;
                seconds   = 0;
                for (i = 0; i < freq; ++i)
                    cpu_avg[i] = (float)0.0;

                /* perform reset functions */
                sprintf(tempstr, "%s%s%s", loaddrive, loaddir, icname);
                if (result = ICload(tempstr))
                    {       /* Try loading from ODSPATH if not found in load dir. */
                            sprintf(tempstr, "%s%s", odspath, icname);
                            result = ICload(tempstr);
                    }

                /* if unable to load IC, reset to original IC name */
                if (result)
                    strcpy(icname, rt_curIC);
                else
                    strcpy(rt_curIC, icname);

                /* set up to perform synchronization pause for 2 seconds */
                waitframes = freq * 2;
                status = SYNC;
                overruns = 0;

                break;
        case SYNC:
                --waitframes;
                if (waitframes <= 0)
                  {
                    waitframes = 0;
                    status = FRZ;
                  printf("Going to FREEZE\n");
                    fflush(stdout);
                  }
                if (overruns > 0)
                    --overruns;
                break;
```

```
                                                    case FRZ:
                                                            if (status != laststatus)
                                                              {
                                                                printf("Going to FREEZE\n");
                                                                        fflush(stdout);
                                                                }
                                                            if (overruns > 0)
                                                                {
                                                                  overruns = 0;
                                                                }
                                                            break;
                                                    case TERM:
                                                            printf("Exiting program.\n");
                                                            fflush(stdout);
                                                            cleanup();
                                                            break;
                                                }
                                        PulseEvent(FRAMEDONE);
                                      }
                                while ((overruns > 0) && (overruns <= maxover));
                                laststatus = status;
    }
}

/*************************************************************************\
*
* FUNCTION    : main
*
* INPUTS      : __argc - argument count
*               __argv - array of argument strings
*
* RETURNS     : 0 on success or -1 on failure.
*
* COMMENTS    :
*
\*************************************************************************/
int main( int __argc, char** __argv )
{
  char *cptr;
  int             endpos;
  char  framebegin[65];
  char  framend[65];

  GetModuleFileName(NULL, loadname, sizeof(loadname) );
  _splitpath(loadname, loaddrive, loaddir, loadfname, NULL );
  cptr = getenv("ODSPATH");
  if (cptr == NULL || !cptr[0])
    strcpy(odspath, ".");
  else
    strcpy(odspath, cptr);
  endpos = strlen(odspath) - 1;
  if (odspath[endpos] != '\\')
    strcat(odspath, "\\");

  /* check if Win32s, if so, display notice and terminate */
  if( GetVersion() & 0x80000000 && (GetVersion() & 0xFF) ==3)
  {
    MessageBoxA( NULL,
      "This application cannot run on Windows 3.1.\n"
      "This application will now terminate.",
      loadname,
      MB_OK | MB_ICONSTOP | MB_SETFOREGROUND );
      return( 1 );
  }

  status       = FRZ;
  printf("Going to FREEZE\n");
  fflush(stdout);
  totframes = 0;
  overruns  = 0;
  catchups   = 0;
  seconds    = 0;
  lastspeed = 1;
  slofac       = 1;
  strcpy(rt_curIC, "IC50");
```

```
  strcpy(icname, rt_curIC);
  freq = rt_getfreq();
  maxover = freq * 2;

  cpu_avg = malloc(freq * sizeof(float));
  if (load_constants(loadname))
                        {
                          printf("Constants data not found at end of %s\n", loadname);
                          cleanup();
                        }


  /* change process to REALTIME_PRIORITY_CLASS */
  if (SetPriorityClass(GetCurrentProcess(), REALTIME_PRIORITY_CLASS) == FALSE)
                        {
                          printf("Unable to change priority class\n");
                        }
  SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_TIME_CRITICAL);


  /* Set up periodic frame timer */
  framesize = 1000 / freq;
  timeBeginPeriod(1);
  wTimerID = timeSetEvent((UINT)framesize, 1, FrameMsg, (DWORD)hwndMain, TIME_PERIODIC);
  if (!wTimerID)
            printf("Unable to set up timer.\n");

  /* create Events for controling real-time loop */
  sprintf(framebegin, "framestart_%s", loadfname);
  rt_strupr(loadfname);
  sprintf(framend, "framedone_%s", loadfname);
  if ((FRAMESTART = CreateEvent(NULL, TRUE, FALSE, framebegin)) == NULL)
            {
              printf("Unable to create FRAMESTART\n");
            }
  else
    if ((FRAMEDONE = CreateEvent(NULL, TRUE, FALSE, framend)) == NULL)
            {
              printf(tempstr,"Unable to create FRAMEDONE\n");
            }

  /* create real-time loop thread */
  if ((RTthread = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)rtctrl,
                                                NULL, 0, &RTloopID)) == NULL)
            {
              printf("Unable to create real-time loop thread\n");
            }

  Sleep(INFINITE);

  return( 0 );
}


#
# build.tsh - This TCL script compiles and links an RTProE real-time application
#                       from object files in the ODSPATH and the current directory.
#
# ----------------------------------------------------------------
# Copyright (C) 1996 Dennis R. LaBelle (drlabell@albany.net) All Rights Reserved.
#
# PERMISSION IS GRANTED TO DISTRIBUTE THIS SOFTWARE FREELY, WITH THE
# EXCEPTION THAT ONE MAY NOT CHARGE FOR IT OR INCLUDE IT WITH SOFTWARE
# WHICH IS SOLD. Permission to use, copy, modify, and distribute this software and
# its documentation for educational, research, internal corporate and non-profit
# purposes, without fee, and without a written agreement is hereby granted for all
# cases that do not conflict with the restriction in the first sentence of this
# paragraph, provided that the above copyright notice, this paragraph, and the
# following three paragraphs appear in all copies.
#
# Permission to incorporate this software into commercial products may be obtained
# from the author (e-mail drlabell@albany.net).
#
# IN NO EVENT SHALL THE AUTHOR BE LIABLE TO ANY PARTY FOR DIRECT,
# INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF
```

```
# THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHOR HAS
# BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# THE AUTHOR SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS"
# BASIS, AND THE AUTHOR HAS NO OBLIGATION TO PROVIDE MAINTENANCE,
# SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
# ----------------------------------------------------------------


proc build__init {} {
#       ____________________________________________
#            |          ATTENTION:          |
#       |                                    |
#       | This procedure contains Operating System or  |
#       | Compiler specific code.            |
#       |____________________________________________|
#
# This procedure performs necessary initilizations of variables
#
# This procedure returns the name of the application which will
# be built.
#
global argv
global env
global tcl_platform
global ODSPATH
global UDSPATH
global EXEOUT
global OBJOUT
global EXTEXE
global EXTLIB
global EXTOBJ
global LIBOPT
global CFLAGS
global FFLAGS
global BFLAGS
global CSCANNER
global FSCANNER
global APPEND
global LIBRARIES
global CC
global F77
global BUILD
global LIBCON
global LIBSEG
global LIBSUB
global ULIBCON
global ULIBSEG
global ULIBSUB
global SEP
global LINKNAME
global filesbuilt
global haderror
global verbose

# read ODSPATH environment variable
if {[info exists env(ODSPATH)]} {
            set ODSPATH $env(ODSPATH)
            regsub -all {\\} $ODSPATH / ODSPATH
  } {set ODSPATH [pwd] }
set ODSPATH [build__get_realpath $ODSPATH]

# set user level path to current directory
set UDSPATH [build__get_realpath [pwd]]

# set up operating system/compiler specific variables
switch $tcl_platform(platform) {
            windows   {
                              set EXEOUT                "/Fe"
                              set OBJOUT                "/Fo"
                              set EXTEXE          ".exe"
                              set EXTLIB          ".lib"
                              set EXTOBJ                ".obj"
```

```
                                set LIBOPT                      "/link"
                                set CFLAGS                      "/c /nologo"
                                set FFLAGS                      "/c /4Yd /WX /nologo"
                                set BFLAGS                      "/nologo "
                                set CSCANNER        "hgen"
                                set FSCANNER        "igen"
                                set APPEND                      "$env(COMSPEC) /c type"
                                regsub -all {\\} $APPEND / APPEND
                                set LIBRARIES       "winmm.lib kernel32.lib user32.lib gdi32.lib \
winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib \
 odbc32.lib odbccp32.lib /subsystem:console /incremental:no /def:$ODSPATH/rtsched.def"
                                set CC              "cl"
                                set F77             "fl32"
                                set BUILD           "fl32"
                                set SEP             "\\"
                                set LINKNAME        "rtsched.exe"
                               }
        unix                   {
                                set EXEOUT                      "-o "
                                set OBJOUT                      "-o "
                                set EXTEXE                      ""
                                set EXTLIB          ".a"
                                set EXTOBJ                      ".o"
                                set LIBOPT                      ""
                                set CFLAGS                      "-c -O2"
                                set FFLAGS                      "-c -O2"
                                set BFLAGS                      ""
                                set APPEND                      "cat"
                                set LIBRARIES       ""
                                set CC              "cc"
                                set F77             "f77"
                                set BUILD           "cc"
                                set SEP             "/"
                                set LINKNAME        "rtsched"
                               }
        }

# set up common variables
set LIBCON[file join $ODSPATH control]
set LIBSEG [file join $ODSPATH segment]
set LIBSUB [file join $ODSPATH subs]
set ULIBCON             [file join $UDSPATH control]
set ULIBSEG             [file join $UDSPATH segment]
set ULIBSUB             [file join $UDSPATH subs]
set CSCANNER            "hgen"
set FSCANNER            "igen"

# determine name for new program and check for verbose status
set buildname ""
set verbose 0
foreach arg $argv {
        switch -- $arg {
        "-v"            {
                                        set verbose 1
                                       }
          default       {
                                        set buildname $arg
                                       }
                        }
        }
if {$buildname == ""} {set buildname $LINKNAME}
set ext [file extension $buildname]
if {$ext != $EXTEXE} {set buildname "$buildname$EXTEXE"}

# initialize build counter and error flag
set filesbuilt 0
set haderror 0

return $buildname
}


proc build_lib {dir libname} {
#       ---------------------------------------
#       |             ATTENTION:              |
```

```
#       |                                       |
#       | This procedure contains Operating System or   |
#       | Compiler specific code.                       |
#       |_______________________________________________|
#
# This procedure is used to maintain object files in a library
#
# dir    = directory containing the library
# libname = file name for library
#
global tcl_platform
global EXTOBJ
global env
global verbose

        switch $tcl_platform(platform) {
                windows    {
                                set cmd0 "    lib /nologo $dir/*$EXTOBJ /OUT:$dir/$libname"
                                if {$verbose} {puts $cmd0}
                                set cmd "exec $cmd0"
                                if {[catch {eval $cmd} result]} {puts $result}
                           }
                unix       {
                                set cmd0 "ar -r $dir/*$EXTOBJ $dir/$libname"
                                if {$verbose} {puts $cmd0}
                                set cmd "exec $cmd0"
                                if {[catch {eval $cmd} result]} {puts $result}
                           }
        }
}


proc build__get_realpath {orgpath} {
#       ___________________________________________
#           |           ATTENTION:          |
#       |                                       |
#       | This procedure contains Operating System or   |
#       | Compiler specific code.                       |
#       |_______________________________________________|
#
global tcl_platform

        set rtnval $orgpath
        switch $tcl_platform(platform) {
                windows    {
                                # check for virtual drives created by SUBST command
                                set orgpath [string tolower $orgpath]
                                set subst_string [exec subst]
                                regsub -all {\\} $subst_string / rstring
                                set lstring [string tolower $rstring]
                                set substlist [split $lstring "\n"]
                                foreach subst $substlist {
                                        set drive [lindex $subst 0]
                                        if {[string first $drive $orgpath] == 0} {
                                                # replace virtual drive with actual path
                                                set substpath [lindex $subst 2]
                                                set rtnval [file join $substpath [string range $orgpath 3 end]]
                                        }
                                }
                           }
                unix       {}
           }
        return $rtnval
}


proc build__need_to_update {target args} {
        if {![file exists $target]} {return 1}
        set srclist $args
        foreach srcfile $srclist {
                if      {[file mtime $srcfile] > [file mtime $target]} {return 1}
                }
        return 0
}
```

```
proc build__compile_C {target srcfile} {
# This procedure compiles C files
#
# Returns:  0 on success
#                       -1 on error
#
global CC
global CFLAGS
global OBJOUT
global CSCANNER
global verbose

        set hdrfile [file root $srcfile].h
        set cmd0 "    $CSCANNER <$srcfile >$hdrfile"
        if {$verbose} {puts $cmd0}
        set cmd "exec $cmd0"
        set rtnval 0
        if {[catch {eval $cmd} result]} {
                puts $result
                set rtnval -1
          } {
                set cmd0 "    $CC $CFLAGS $srcfile ${OBJOUT}$target"
                puts $cmd0
                set cmd "exec $cmd0"
                if {[catch {eval $cmd} result]} {
                        puts $result
                        set rtnval -1
                  }
          }
        return $rtnval
}


proc build__compile_FORTRAN {target srcfile} {
# This procedure compiles FORTRAN files
#
# Returns:  0 on success
#                       -1 on error
#
global F77
global FFLAGS
global OBJOUT
global FSCANNER
global verbose

        set hdrfile [file root $srcfile].inc
        set cmd0 "    $FSCANNER <$srcfile >$hdrfile"
        if {$verbose} {puts $cmd0}
        set cmd "exec $cmd0"
        set rtnval 0
        if {[catch {eval $cmd} result]} {
                puts $result
                set rtnval -1
          } {
                set cmd0 "    $F77 $FFLAGS $srcfile ${OBJOUT}$target"
                puts $cmd0
                set cmd "exec $cmd0"
                if {[catch {eval $cmd} result]} {
                        puts $result
                        set rtnval -1
                  }
          }
        return $rtnval
}


proc build__check_needed_dirs {} {
global ULIBCON
global ULIBSEG
global ULIBSUB

        set dirlist "$ULIBCON $ULIBSEG $ULIBSUB"
        foreach dir $dirlist {
                if {![file exists $dir]} {
```

```
                    file mkdir $dir
                    if {![file isdir $dir]} {
                        puts "ABORT!  Unable to create directory named $dir."
                        exit 1
                    }
                } elseif {![file isdir $dir]} {
                        puts "ABORT!  Can not create directory named $dir."
                        puts "      A file by that name already exists"
                        exit 2
                }
            }
}


proc build__check_needed_files {} {
#  This procedure ensures the existance of necessary files in the
#  current directory. Missing files are copied from the ODSPATH.
#
global ODSPATH
global EXTOBJ

        set filelist "${ODSPATH}/frameseq.in"
        foreach fname $filelist {
            set shortname [file tail $fname]
            if {![file exists $shortname]} {
                if {![file exists $fname]} {
                        puts "ABORT!  Cannot find $fname"
                        exit 3
                    }
                file copy $fname $shortname
              } elseif {[file isdir $shortname]} {
                        puts "ABORT!  Cannot create file named $shortname."
                        puts "      A directory by that name already exists"
                        exit 4
                }
            }
}


proc build__make_modules {modtype} {
global ULIBCON
global ULIBSEG
global ULIBSUB
global EXTOBJ
global EXTLIB
global ARCHIVER
global filesbuilt
global haderror

        switch $modtype {
            control     {set typechars c; set LIB $ULIBCON}
            segment     {set typechars s; set LIB $ULIBSEG}
            subroutine  {set typechars "x y"; set LIB $ULIBSUB}
            default     {return}
          }

        puts "Compiling $modtype modules."
        set buildctr 0

        # compile C source files in current directory, as necessary
        foreach tchar $typechars {
            if {[catch {glob ??$tchar*.c} clist]} {set clist ""}
            foreach srcfile $clist {
                    set target $LIB/[file root $srcfile]$EXTOBJ
                    if {[build__need_to_update $target $srcfile]} {
                        if {[build__compile_C $target $srcfile]} {
                            set haderror 1
                        } {incr buildctr}
                    }
                }
            }

        # compile FORTRAN source files in current directory, as necessary
        foreach tchar $typechars {
            if {[catch {glob ??$tchar*.f} flist]} {set flist ""}
```

```
                foreach srcfile $flist {
                        set target $LIB/[file root $srcfile]$EXTOBJ
                        if {[build__need_to_update $target $srcfile]} {
                                if {[build__compile_FORTRAN $target $srcfile]} {
                                        set hadderror 1
                                } {incr buildctr}
                        }
                }
        }

    if {$haderror} {
        puts "ABORT!  Terminal error encountered. Cannot continue."
        exit 5
      } {
        if {$buildctr > 0} {
            # place new object files into library
            build__lib $LIB lib[string range $modtype 0 2]$EXTLIB
            incr filesbuilt $buildctr
        }
      }
}


proc build__make_frame_calls {} {
global EXTOBJ
global CC
global CFLAGS
global OBJOUT
global verbose

    set srcfile frameseq.in
    set target frameseq$EXTOBJ
    if {[build__need_to_update $target $srcfile]} {
        if {$verbose} {puts framegen}
        if {[catch {exec framegen} result]} {
            puts $result
          } {
            set srcfile frameseq.c
            set target  frameseq$EXTOBJ
            set cmd0 "$CC $CFLAGS $srcfile ${OBJOUT}$target"
            if {$verbose} {puts $cmd0}
            set cmd "exec $cmd0"
            if {[catch {eval $cmd} result]} {puts $result}
          }
      }
    puts "Frame sequencing file is up to date"
}


proc build__make_constants {} {
#  This procedure creates a constant coefficients file for attaching to
#  the application.
#
#  Returns the name of the file created.
#
global ODSPATH
global verbose

    # make temporary input file for rtdata program
    set fname build[pid]
    if {[catch {set fout [open $fname w]} result]} {
        puts "ABORT!  Unable to open temporary file $fname"
        exit 6
      } {
        puts $fout "conload $ODSPATH/globcon.dat"
        if {![catch {set fin [open c.constants r]} result]} {
            # copy c.constants file if found
            while {![eof $fin]} {
                gets $fin line
                puts $fout $line
                }
            close $fin
          }
        set fname2 attach[pid]
        puts $fout "consave $fname2"
```

```
                puts $fout "exit"
                close $fout
                set cmd0 "rtdata <$fname"
#       if {$verbose} {puts $cmd0}
                set cmd "exec $cmd0"
                if {[catch {eval $cmd} results]} {puts $result}
                file delete $fname
                return $fname2
        }
}

proc build__link_application {xname} {
global EXTEXE
global EXTOBJ
global LIBCON
global LIBSEG
global LIBSUB
global ULIBCON
global ULIBSEG
global ULIBSUB
global EXTLIB
global BUILD
global BFLAGS
global EXEOUT
global LIBOPT
global LIBRARIES
global APPEND
global ODSPATH
global UDSPATH
global verbose

        set target $xname
        if {$UDSPATH != $ODSPATH} {
                set isUDS 1
          } {set isUDS 0}
        set    srclist $ODSPATH/rtsched$EXTOBJ
        lappend srclist frameseq$EXTOBJ
        set udslib $ULIBCON/libcon$EXTLIB
        if {$isUDS && [file exists $udslib]} {lappend srclist $udslib}
        lappend srclist $LIBCON/libcon$EXTLIB
        set udslib $ULIBSEG/libseg$EXTLIB
        if {$isUDS && [file exists $udslib]} {lappend srclist $udslib}
        lappend srclist $LIBSEG/libseg$EXTLIB
        set udslib $ULIBSUB/libsub$EXTLIB
        if {$isUDS && [file exists $udslib]} {lappend srclist $udslib}
        lappend srclist $LIBSUB/libsub$EXTLIB
        set newconstants 0
        if {[file exists c.constants]} {
           if {[build__need_to_update $xname c.constants]} {set newconstants 1}
          }
        if {[eval "build__need_to_update $target $srclist"] || $newconstants} {
                puts "Linking program"
                if {$verbose} {puts ""}
                set cmd0 "$BUILD $BFLAGS $srclist $EXEOUT$target $LIBOPT $LIBRARIES"
                if {$verbose} {puts $cmd0}
                set cmd "exec $cmd0"
                if {[catch {eval $cmd} result]} {
                        puts $result
                        puts "\nFailed build of $xname"
                  } {
                        # attach constant coefficients data to end of executable file
                        if {$verbose} {puts ""}
                        puts "Attaching constant coefficients to application file."
                        set fname [build__make_constants]
                        set cmd0 "    [lindex $APPEND end] $fname >>$target"
                        if {$verbose} {puts $cmd0}
                        set cmd "exec $APPEND $fname >>$target"
                        if {[catch {eval $cmd} result]} {
                                puts $result
                                file delete $xname
                                puts "Failed build of $xname"
                          } {puts "Successful build of $xname"}
                        file delete $fname
                  }
```

```
            } {puts "$xname already up to date."}
}


proc build__main {} {
            set buildname [build__init]
            build__check_needed_dirs
            build__check_needed_files
            build__make_frame_calls
            build__make_modules control
            build__make_modules segment
            build__make_modules subroutine
            build__link_application $buildname
}

build__main
#!wish -f
# Program: sde
# Tcl version: 7.6 (Tcl/Tk/XF)
# Tk version: 4.2
# XF version: 4.0
#
# ----------------------------------------------------------------------
# Copyright (C) 1996 Dennis R. LaBelle (drlabell@albany.net) All Rights Reserved.
#
# PERMISSION IS GRANTED TO DISTRIBUTE THIS SOFTWARE FREELY, WITH THE
# EXCEPTION THAT ONE MAY NOT CHARGE FOR IT OR INCLUDE IT WITH SOFTWARE
# WHICH IS SOLD. Permission to use, copy, modify, and distribute this software and
# its documentation for educational, research, internal corporate and non-profit
# purposes, without fee, and without a written agreement is hereby granted for all
# cases that do not conflict with the restriction in the first sentence of this
# paragraph, provided that the above copyright notice, this paragraph, and the
# following three paragraphs appear in all copies.
#
# Permission to incorporate this software into commercial products may be obtained
# from the author (e-mail drlabell@albany.net).
#
# IN NO EVENT SHALL THE AUTHOR BE LIABLE TO ANY PARTY FOR DIRECT,
# INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF
# THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHOR HAS
# BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# THE AUTHOR SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS"
# BASIS, AND THE AUTHOR HAS NO OBLIGATION TO PROVIDE MAINTENANCE,
# SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
# ----------------------------------------------------------------------

# module inclusion
global env
global xfLoadPath
global xfLoadInfo
set xfMisc(separator) @
set xfLoadInfo 0
if {[info exists env(XF_LOAD_PATH)]} {
  if {[string first $env(XF_LOAD_PATH) c:/sim/tkprogs/xf] == -1} {
    set xfLoadPath $env(XF_LOAD_PATH):c:/sim/tkprogs/xf
  } {
    set xfLoadPath c:/sim/tkprogs/xf
  }
} {
  set xfLoadPath c:/sim/tkprogs/xf
}

global argc
global argv
set tmpArgv ""
for {set counter 0} {$counter < $argc} {incr counter 1} {
  case [string tolower [lindex $argv $counter]] in {
    {-xfloadpath} {
      incr counter 1
      set xfLoadPath "[lindex $argv $counter]:$xfLoadPath"
    }
    {-xfstartup} {
```

```
      incr counter 1
      source [lindex $argv $counter]
    }
    {-xfbindfile} {
      incr counter 1
      set env(XF_BIND_FILE) "[lindex $argv $counter]"
    }
    {-xfcolorfile} {
      incr counter 1
      set env(XF_COLOR_FILE) "[lindex $argv $counter]"
    }
    {-xfcursorfile} {
      incr counter 1
      set env(XF_CURSOR_FILE) "[lindex $argv $counter]"
    }
    {-xffontfile} {
      incr counter 1
      set env(XF_FONT_FILE) "[lindex $argv $counter]"
    }
    {-xfmodelmono} {
      tk colormodel . monochrome
    }
    {-xfmodelcolor} {
      tk colormodel . color
    }
    {-xfloading} {
      set xfLoadInfo 1
    }
    {-xfnoloading} {
      set xfLoadInfo 0
    }
    {default} {
      lappend tmpArgv [lindex $argv $counter]
    }
  }
}
set argv $tmpArgv
set argc [llength $tmpArgv]
unset counter
unset tmpArgv


# procedure to show window ShowWindow.about
proc ShowWindow.about { args} {
# xf ignore me 7

  # build widget .about
  if {"[info procs XFEdit]" != ""} {
    catch "XFDestroy .about"
  } {
    catch "destroy .about"
  }
  toplevel .about  -relief {raised}

  # Window manager configurations
  wm positionfrom .about program
  wm sizefrom .about program
  wm geometry .about +26+52
  wm maxsize .about 415 133
  wm minsize .about 415 133
  wm protocol .about WM_DELETE_WINDOW {XFProcError {Application windows can not be destroyed.
Please use the "Current widget path:" to show/hide windows.}}
  wm title .about {About RTProE Workbench}


  # build widget .about.frame1
  frame .about.frame1  -borderwidth {2}  -height {30}  -relief {ridge}  -width {30}

  # build widget .about.frame1.label2
  label .about.frame1.label2  -borderwidth {0}  -image {tclpowered}  -text {label2}

  # build widget .about.frame1.frame5
  frame .about.frame1.frame5  -height {30}  -width {30}

  # build widget .about.frame1.frame5.label3
```

```
  label .about.frame1.frame5.label3  -borderwidth {0}  -text {Created by Dennis R. LaBelle}

  # build widget .about.frame1.frame5.label7
  label .about.frame1.frame5.label7  -borderwidth {0}

  # build widget .about.frame1.frame5.label8
  label .about.frame1.frame5.label8  -borderwidth {0}  -font {-adobe-times-bold-i-normal--14-140-75-75-p-77-iso8859-1}  -foreground {red}  -text {bringing Real Time
programming to real people}

  # build widget .about.frame1.frame5.label9
  label .about.frame1.frame5.label9  -borderwidth {0}

  # build widget .about.frame1.frame5.label10
  label .about.frame1.frame5.label10  -borderwidth {0}  -text {with the power of TCL/TK}

  # build widget .about.frame1.frame5.frame11
  frame .about.frame1.frame5.frame11  -height {7}  -width {30}

  # build widget .about.button1
  button .about.button1  -borderwidth {1}  -command {DestroyWindow.about}  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1}  -highlightthickness {0}
-padx {11}  -pady {4}  -text {OK}

  # pack master .about.frame1
  pack configure .about.frame1.label2  -side left
  pack configure .about.frame1.frame5  -expand 1  -fill both  -side left

  # pack master .about.frame1.frame5
  pack configure .about.frame1.frame5.frame11  -fill x
  pack configure .about.frame1.frame5.label3  -fill x
  pack configure .about.frame1.frame5.label7  -fill x
  pack configure .about.frame1.frame5.label8  -fill x
  pack configure .about.frame1.frame5.label9  -fill x
  pack configure .about.frame1.frame5.label10  -fill x

  # pack master .about
  pack configure .about.frame1
  pack configure .about.button1  -pady 2

  if {"[info procs XFEdit]" != ""} {
    catch "XFMiscBindWidgetTree .about"
    after 2 "catch {XFEditSetShowWindows}"
  }
}

proc DestroyWindow.about {} {# xf ignore me 7
  if {"[info procs XFEdit]" != ""} {
    if {"[info commands .about]" != ""} {
      global xfShowWindow.about
      set xfShowWindow.about 0
      XFEditSetPath .
      after 2 "XFSaveAsProc .about; XFEditSetShowWindows"
    }
  } {
    catch "destroy .about"
    update
  }
}


# procedure to show window ShowWindow.options
proc ShowWindow.options { args} {
# xf ignore me 7

  # build widget .options
  if {"[info procs XFEdit]" != ""} {
    catch "XFDestroy .options"
  } {
    catch "destroy .options"
  }
  toplevel .options  -relief {raised}

  # Window manager configurations
  wm positionfrom .options program
  wm sizefrom .options program
  wm geometry .options +43+393
```

```
 wm maxsize .options 553 123
 wm minsize .options 553 123
 wm protocol .options WM_DELETE_WINDOW {XFProcError {Application windows can not be destroyed.
Please use the "Current widget path:" to show/hide windows.}}
 wm title .options {Options}

 # bindings
 bind .options <Reparent> {set loadname_req $loadname
set ODSPATH_req $ODSPATH
set UDSPATH_req $UDSPATH
set browse_cmd_req $browse_cmd
set edit_cmd_req $edit_cmd
set dbmode_req $dbmode
set packmode_req $packmode}

 # build widget .options.frame7
 frame .options.frame7 -height {30} -width {30}

 # build widget .options.frame7.button8
 button .options.frame7.button8 -borderwidth {1} -command {set loadname $loadname_req
set ODSPATH $ODSPATH_req
set UDSPATH $UDSPATH_req
set browse_cmd $browse_cmd_req
set edit_cmd $edit_cmd_req
set dbmode $dbmode_req
set packmode $packmode_req
DestroyWindow.options} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -padx {11} -pady {2} -state {active} -text {OK} -width {6}

 # build widget .options.frame7.button9
 button .options.frame7.button9 -borderwidth {1} -command {DestroyWindow.options} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -padx {11}
-pady {2} -text {Cancel}

 # build widget .options.frame7.button18
 button .options.frame7.button18 -borderwidth {1} -command {help_show rtpro_pdm.htm} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -padx
{11} -pady {2} -text {Help}

 # build widget .options.frame8
 frame .options.frame8 -height {30} -width {30}

 # build widget .options.frame8.canvas10
 canvas .options.frame8.canvas10 -height {16} -highlightthickness {0} -width {544}

 # build widget .options.frame8.frame11
 frame .options.frame8.frame11 -borderwidth {3} -height {30} -relief {ridge} -width {30}

 # build widget .options.frame8.frame11.frame10
 frame .options.frame8.frame11.frame10 -height {30} -width {30}

 # build widget .options.frame8.frame11.frame10.label2
 label .options.frame8.frame11.frame10.label2 -borderwidth {0} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -text {Application name: } -width
{16}

 # build widget .options.frame8.frame11.frame10.entry3
 entry .options.frame8.frame11.frame10.entry3 -background {azure} -textvariable {loadname_req}

 # build widget .options.frame8.frame11.frame1
 frame .options.frame8.frame11.frame1 -height {30} -width {30}

 # build widget .options.frame8.frame11.frame1.label2
 label .options.frame8.frame11.frame1.label2 -borderwidth {0} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -text {ODS path:          } -width
{16}

 # build widget .options.frame8.frame11.frame1.entry3
 entry .options.frame8.frame11.frame1.entry3 -background {azure} -textvariable {ODSPATH_req} -width {60}

 # build widget .options.frame8.frame11.frame4
 frame .options.frame8.frame11.frame4 -height {30} -width {30}

 # build widget .options.frame8.frame11.frame4.label2
 label .options.frame8.frame11.frame4.label2 -borderwidth {0} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -text {UDS path:          } -width
{16}

 # build widget .options.frame8.frame11.frame4.entry3
 entry .options.frame8.frame11.frame4.entry3 -background {azure} -textvariable {UDSPATH_req}
```

```
# build widget .options.frame8.frame12
frame .options.frame8.frame12 -borderwidth {3} -height {30} -relief {ridge} -width {30}

# build widget .options.frame8.frame12.frame5
frame .options.frame8.frame12.frame5 -height {30} -width {30}

# build widget .options.frame8.frame12.frame5.label2
label .options.frame8.frame12.frame5.label2 -borderwidth {0} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -text { HTML browser:    } -width
{16}

# build widget .options.frame8.frame12.frame5.entry3
entry .options.frame8.frame12.frame5.entry3 -background {azure} -textvariable {browse_cmd_req} -width {60}

# build widget .options.frame8.frame12.frame6
frame .options.frame8.frame12.frame6 -height {30} -width {30}

# build widget .options.frame8.frame12.frame6.label2
label .options.frame8.frame12.frame6.label2 -borderwidth {0} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -text { Editor:                } -width
{16}

# build widget .options.frame8.frame12.frame6.entry3
entry .options.frame8.frame12.frame6.entry3 -background {azure} -textvariable {edit_cmd_req}

# build widget .options.frame8.frame12.frame16
frame .options.frame8.frame12.frame16 -height {25} -width {30}

# build widget .options.frame8.frame13
frame .options.frame8.frame13 -borderwidth {3} -height {30} -relief {ridge} -width {30}

# build widget .options.frame8.frame13.frame14
frame .options.frame8.frame13.frame14 -height {30} -width {30}

# build widget .options.frame8.frame13.frame14.radiobutton2
radiobutton .options.frame8.frame13.frame14.radiobutton2 -borderwidth {1} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -pady {2} -selectcolor
{black} -text {Read Only  } -value {read} -variable {dbmode_req} -width {15}

# build widget .options.frame8.frame13.frame14.radiobutton3
radiobutton .options.frame8.frame13.frame14.radiobutton3 -borderwidth {1} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -pady {2} -selectcolor
{black} -text {Write on exit} -value {write} -variable {dbmode_req} -width {15}

# build widget .options.frame8.frame13.frame15
frame .options.frame8.frame13.frame15 -height {30} -width {30}

# build widget .options.frame8.frame13.frame15.radiobutton4
radiobutton .options.frame8.frame13.frame15.radiobutton4 -borderwidth {1} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -pady {2} -selectcolor
{black} -text {No pack     } -value {nopack} -variable {packmode_req} -width {15}

# build widget .options.frame8.frame13.frame15.radiobutton5
radiobutton .options.frame8.frame13.frame15.radiobutton5 -borderwidth {1} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -pady {2} -selectcolor
{black} -text {Pack on exit} -value {pack} -variable {packmode_req} -width {15}

# build widget .options.frame8.frame13.frame17
frame .options.frame8.frame13.frame17 -height {27} -width {547}

# pack master .options.frame7
pack configure .options.frame7.button18 -side right
pack configure .options.frame7.button9 -side right
pack configure .options.frame7.button8 -side right

# pack master .options.frame8
pack configure .options.frame8.canvas10
pack configure .options.frame8.frame11

# pack master .options.frame8.frame11
pack configure .options.frame8.frame11.frame10 -fill x
pack configure .options.frame8.frame11.frame1 -fill x
pack configure .options.frame8.frame11.frame4 -fill x

# pack master .options.frame8.frame11.frame10
pack configure .options.frame8.frame11.frame10.label2 -side left
pack configure .options.frame8.frame11.frame10.entry3 -expand 1 -fill x -side left

# pack master .options.frame8.frame11.frame1
pack configure .options.frame8.frame11.frame1.label2 -side left
pack configure .options.frame8.frame11.frame1.entry3 -expand 1 -fill x -side left
```

```
# pack master .options.frame8.frame11.frame4
pack configure .options.frame8.frame11.frame4.label2  -side left
pack configure .options.frame8.frame11.frame4.entry3  -expand 1  -fill x  -side left

# pack master .options.frame8.frame12
pack configure .options.frame8.frame12.frame5  -fill x
pack configure .options.frame8.frame12.frame6  -fill x
pack configure .options.frame8.frame12.frame16

# pack master .options.frame8.frame12.frame5
pack configure .options.frame8.frame12.frame5.label2  -side left
pack configure .options.frame8.frame12.frame5.entry3  -expand 1  -fill x  -side left

# pack master .options.frame8.frame12.frame6
pack configure .options.frame8.frame12.frame6.label2  -side left
pack configure .options.frame8.frame12.frame6.entry3  -expand 1  -fill x  -side left

# pack master .options.frame8.frame13
pack configure .options.frame8.frame13.frame14
pack configure .options.frame8.frame13.frame15
pack configure .options.frame8.frame13.frame17

# pack master .options.frame8.frame13.frame14
pack configure .options.frame8.frame13.frame14.radiobutton2  -side left
pack configure .options.frame8.frame13.frame14.radiobutton3  -side left

# pack master .options.frame8.frame13.frame15
pack configure .options.frame8.frame13.frame15.radiobutton4  -side left
pack configure .options.frame8.frame13.frame15.radiobutton5  -side left

# pack master .options
pack configure .options.frame8
pack configure .options.frame7  -fill x

# build canvas items .options.frame8.canvas10
set xfTmpTag [.options.frame8.canvas10 create polygon 55.0 14.0 59.0 0.0 139.0 0.0 143.0 14.0 55.0 14.0]
.options.frame8.canvas10 itemconfigure $xfTmpTag  -fill {grey68}  -outline {black}  -tags {tab4.1}
.options.frame8.canvas10 bind $xfTmpTag <Button-1> {tabselect 4 tab4.1}
set xfTmpTag [.options.frame8.canvas10 create line 55.0 14.0 59.0 0.0 139.0 0.0 143.0 14.0]
.options.frame8.canvas10 itemconfigure $xfTmpTag  -tags {tab4.1}
set xfTmpTag [.options.frame8.canvas10 create text 139.0 0.0]
.options.frame8.canvas10 itemconfigure $xfTmpTag  -anchor {ne}  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1}  -tags {tab4.1}  -text
{Commands }
.options.frame8.canvas10 bind $xfTmpTag <Button-1> {tabselect 4 tab4.1}
set xfTmpTag [.options.frame8.canvas10 create polygon 140.0 14.0 144.0 0.0 224.0 0.0 228.0 14.0 140.0 14.0]
.options.frame8.canvas10 itemconfigure $xfTmpTag  -fill {grey68}  -outline {black}  -tags {tab4.2}
.options.frame8.canvas10 bind $xfTmpTag <Button-1> {tabselect 4 tab4.2}
set xfTmpTag [.options.frame8.canvas10 create line 140.0 14.0 144.0 0.0 224.0 0.0 228.0 14.0]
.options.frame8.canvas10 itemconfigure $xfTmpTag  -tags {tab4.2}
set xfTmpTag [.options.frame8.canvas10 create text 224.0 0.0]
.options.frame8.canvas10 itemconfigure $xfTmpTag  -anchor {ne}  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1}  -tags {tab4.2}  -text {DataBase
}
.options.frame8.canvas10 bind $xfTmpTag <Button-1> {tabselect 4 tab4.2}
set xfTmpTag [.options.frame8.canvas10 create polygon 0.0 14.0 4.0 0.0 54.0 0.0 58.0 14.0 0.0 14.0]
.options.frame8.canvas10 itemconfigure $xfTmpTag  -fill {azure}  -outline {azure}  -tags {tab4.0}
.options.frame8.canvas10 bind $xfTmpTag <Button-1> {tabselect 4 tab4.0}
set xfTmpTag [.options.frame8.canvas10 create text 54.0 0.0]
.options.frame8.canvas10 itemconfigure $xfTmpTag  -anchor {ne}  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1}  -tags {tab4.0}  -text {Build }
.options.frame8.canvas10 bind $xfTmpTag <Button-1> {tabselect 4 tab4.0}
set xfTmpTag [.options.frame8.canvas10 create line 0.0 14.0 4.0 0.0 54.0 0.0 58.0 14.0]
.options.frame8.canvas10 itemconfigure $xfTmpTag  -tags {tab4.0}


 if {"[info procs XFEdit]" != ""} {
   catch "XFMiscBindWidgetTree .options"
   after 2 "catch {XFEditSetShowWindows}"
 }
}

proc DestroyWindow.options {} {# xf ignore me 7
 if {"[info procs XFEdit]" != ""} {
   if {"[info commands .options]" != ""} {
     global xfShowWindow.options
```

```
    set xfShowWindow.options 0
    XFEditSetPath .
    after 2 "XFSaveAsProc .options; XFEditSetShowWindows"
   }
  } {
   catch "destroy .options"
   update
  }
}


# procedure to show window ShowWindow.selbuf
proc ShowWindow.selbuf { args} {
# xf ignore me 7

 # build widget .selbuf
 if {"[info procs XFEdit]" != ""} {
  catch "XFDestroy .selbuf"
 } {
  catch "destroy .selbuf"
 }
 toplevel .selbuf

 # Window manager configurations
 wm positionfrom .selbuf program
 wm sizefrom .selbuf program
 wm geometry .selbuf +287+29
 wm maxsize .selbuf 1024 768
 wm minsize .selbuf 315 246
 wm protocol .selbuf WM_DELETE_WINDOW {XFProcError {Application windows can not be destroyed.
Please use the "Current widget path:" to show/hide windows.}}
 wm title .selbuf {Select file buffer}

 # bindings
 bind .selbuf <Destroy> {set selwin hide}
 bind .selbuf <Reparent> {.selbuf.frame.text2 delete 1.0 end
.selbuf.frame.text2 insert end $select_buffer
set selbuflast $select_buffer
set selwin show}

 # build widget .selbuf.frame
 frame .selbuf.frame -relief {raised}

 # build widget .selbuf.frame.scrollbar1
 scrollbar .selbuf.frame.scrollbar1 -activebackground {grey85} -borderwidth {1} -command {.selbuf.frame.text2 yview} -relief {raised} -takefocus {grey87} -troughcolor
{grey87}

 # build widget .selbuf.frame.text2
 text .selbuf.frame.text2 -background {azure} -height {16} -highlightthickness {0} -width {40} -wrap {none} -yscrollcommand {.selbuf.frame.scrollbar1 set}

 # build widget .selbuf.frame1
 frame .selbuf.frame1 -height {30} -width {30}

 # build widget .selbuf.frame1.button1
 button .selbuf.frame1.button1 -borderwidth {1} -command {set select_buffer ""
.selbuf.frame.text2 delete 1.0 end} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -padx {5} -pady {1} -text {Clear}

 # build widget .selbuf.frame1.button2
 button .selbuf.frame1.button2 -borderwidth {1} -command {selbuf_save} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -padx {5} -pady {1}
-text {Save}

 # build widget .selbuf.frame1.button3
 button .selbuf.frame1.button3 -borderwidth {1} -command {set select_buffer [.selbuf.frame.text2 get 1.0 end]
DestroyWindow.selbuf} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -padx {5} -pady {1} -text {OK}

 # build widget .selbuf.frame1.button4
 button .selbuf.frame1.button4 -borderwidth {1} -command {DestroyWindow.selbuf
set select_buffer $selbuflast} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -padx {5} -pady {1} -text {Cancel}

 # build widget .selbuf.frame1.button5
 button .selbuf.frame1.button5 -borderwidth {1} -command {help_show rtpro_pdm.htm} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -padx {5}
-pady {1} -text {Help}

 # build widget .selbuf.frame1.button0
```

```
 button .selbuf.frame1.button0 -borderwidth {1} -command {selbuf_load} -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} -padx {5} -pady {1} -text
{Load}

 # pack master .selbuf.frame
 pack configure .selbuf.frame.scrollbar1 -fill y -side right
 pack configure .selbuf.frame.text2 -expand 1 -fill both

 # pack master .selbuf.frame1
 pack configure .selbuf.frame1.button1 -side left
 pack configure .selbuf.frame1.button2 -side left
 pack configure .selbuf.frame1.button5 -side right
 pack configure .selbuf.frame1.button4 -side right
 pack configure .selbuf.frame1.button3 -side right
 pack configure .selbuf.frame1.button0 -side left

 # pack master .selbuf
 pack configure .selbuf.frame -expand 1 -fill both -pady 2
 pack configure .selbuf.frame1 -fill x

 .selbuf.frame.text2 insert end {}



 if {"[info procs XFEdit]" != ""} {
   catch "XFMiscBindWidgetTree .selbuf"
   after 2 "catch {XFEditSetShowWindows}"
 }
}

proc DestroyWindow.selbuf {} {# xf ignore me 7
 if {"[info procs XFEdit]" != ""} {
   if {"[info commands .selbuf]" != ""} {
     global xfShowWindow.selbuf
     set xfShowWindow.selbuf 0
     XFEditSetPath .
     after 2 "XFSaveAsProc .selbuf; XFEditSetShowWindows"
   }
 } {
   catch "destroy .selbuf"
   update
 }
}


# procedure to show window .
proc ShowWindow. {args} {# xf ignore me 7

 # Window manager configurations
 wm positionfrom . user
 wm sizefrom . ""
 wm maxsize . 1024 753
 wm minsize . 104 1
 wm protocol . WM_DELETE_WINDOW {XFProcError {Application windows can not be destroyed.
Please use the "Current widget path:" to show/hide windows.}}
 wm title . {RTProE Workbench}


 # build widget .frame0
 frame .frame0

 # build widget .frame0.menubutton1
 menubutton .frame0.menubutton1 \
   -activebackground {slateblue4} \
   -activeforeground {white} \
   -borderwidth {0} \
   -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
   -menu {.frame0.menubutton1.m} \
   -padx {5} \
   -pady {4} \
   -text {File} \
   -underline {0}

 # build widget .frame0.menubutton1.m
 menu .frame0.menubutton1.m \
   -activebackground {slateblue4} \
```

```
    -activeborderwidth {0} \
    -activeforeground {white} \
    -borderwidth {0} \
    -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
    -relief {flat} \
    -tearoff {0}
  .frame0.menubutton1.m add command \
    -command {file_new} \
    -label {New} \
    -underline {0}
  .frame0.menubutton1.m add command \
    -command {set index [.frame1.frame0.frame.listbox1 curselection]
if {$index != ""} {
            open_file [.frame1.frame0.frame.listbox1 get $index]
  }} \
    -label {Open} \
    -underline {0}
  .frame0.menubutton1.m add command \
    -command {file_copy} \
    -label {Copy from ODS} \
    -underline {0}
  .frame0.menubutton1.m add separator
  .frame0.menubutton1.m add command \
    -command {file_import} \
    -label {Import records} \
    -underline {0}
  .frame0.menubutton1.m add separator
  .frame0.menubutton1.m add command \
    -command {ShowWindow.options} \
    -label {Options} \
    -underline {1}
  .frame0.menubutton1.m add command \
    -command {options_save} \
    -label {Save Options} \
    -underline {0}
  .frame0.menubutton1.m add separator
  .frame0.menubutton1.m add command \
    -command {finish} \
    -label {Exit} \
    -underline {1}

  # build widget .frame0.menubutton6
  menubutton .frame0.menubutton6 \
    -activebackground {slateblue4} \
    -activeforeground {white} \
    -borderwidth {0} \
    -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
    -menu {.frame0.menubutton6.m} \
    -padx {5} \
    -pady {4} \
    -text {Build} \
    -underline {0}

  # build widget .frame0.menubutton6.m
  menu .frame0.menubutton6.m \
    -activebackground {slateblue4} \
    -activeforeground {white} \
    -borderwidth {0} \
    -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
    -relief {flat} \
    -tearoff {0}
  .frame0.menubutton6.m add command \
    -command {ShowWindow.options
tabselect 4 tab4.0} \
    -label {Options} \
    -underline {0}
  .frame0.menubutton6.m add separator
  .frame0.menubutton6.m add command \
    -command {after 1000 {buildload}} \
    -label {Build application} \
    -underline {0}

  # build widget .frame0.menubutton8
  menubutton .frame0.menubutton8 \
    -activebackground {slateblue4} \
```

```
  -activeforeground {white} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -menu {.frame0.menubutton8.m} \
  -padx {5} \
  -pady {4} \
  -text {Window} \
  -underline {0}

# build widget .frame0.menubutton8.m
menu .frame0.menubutton8.m \
  -activebackground {slateblue4} \
  -activeforeground {white} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {flat} \
  -tearoff {0}
.frame0.menubutton8.m add checkbutton \
  -command {if {$outwin == "hide"} {
            pack forget .frame2
 } {
            pack .frame2 -side bottom -expand 1 -fill both
  }} \
  -label {Output window} \
  -offvalue {hide} \
  -onvalue {show} \
  -variable {outwin} \
  -underline {1}
.frame0.menubutton8.m add checkbutton \
  -command {if {$selwin == "hide"} {
  if {[winfo exists .selbuf]} {
            set select_buffer [.selbuf.frame.text2 get 1.0 end]
            DestroyWindow.selbuf
    }
 } {        ShowWindow.selbuf }} \
  -label {Select buffer} \
  -offvalue {hide} \
  -onvalue {show} \
  -variable {selwin} \
  -underline {0}

# build widget .frame0.menubutton2
menubutton .frame0.menubutton2 \
  -activebackground {slateblue4} \
  -activeforeground {white} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightbackground {grey85} \
  -menu {.frame0.menubutton2.m} \
  -padx {5} \
  -pady {4} \
  -text {Help} \
  -underline {0}

# build widget .frame0.menubutton2.m
menu .frame0.menubutton2.m \
  -activebackground {slateblue4} \
  -activeborderwidth {0} \
  -activeforeground {white} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {flat} \
  -tearoff {0}
.frame0.menubutton2.m add command \
  -command {ShowWindow.about} \
  -label {About} \
  -underline {0}

# build widget .frame1
frame .frame1 \
  -height {30} \
  -width {30}

# build widget .frame1.frame3
frame .frame1.frame3 \
```

```
  -background {grey50} \
  -borderwidth {2} \
  -height {30} \
  -relief {ridge} \
  -width {30}

 # build widget .frame1.frame3.frame1
 frame .frame1.frame3.frame1 \
  -height {30} \
  -width {30}

 # build widget .frame1.frame3.frame1.frame7
 frame .frame1.frame3.frame1.frame7 \
  -borderwidth {2} \
  -height {30} \
  -relief {ridge} \
  -width {30}

 # build widget .frame1.frame3.frame1.frame7.label8
 label .frame1.frame3.frame1.frame7.label8 \
  -background {lightblue} \
  -borderwidth {1} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {raised} \
  -text {Find}

 # build widget .frame1.frame3.frame1.frame7.entry6
 entry .frame1.frame3.frame1.frame7.entry6 \
  -background {azure} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -textvariable {find_req}
 # bindings
 bind .frame1.frame3.frame1.frame7.entry6 <KeyRelease> {fillrec [findfirst]}

 # build widget .frame1.frame3.frame1.frame9
 frame .frame1.frame3.frame1.frame9 \
  -borderwidth {2} \
  -height {30} \
  -relief {ridge} \
  -width {30}

 # build widget .frame1.frame3.frame1.frame9.label8
 label .frame1.frame3.frame1.frame9.label8 \
  -background {lightblue} \
  -borderwidth {1} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {raised} \
  -text {Record No.}

 # build widget .frame1.frame3.frame1.frame9.canvas2
 canvas .frame1.frame3.frame1.frame9.canvas2 \
  -borderwidth {2} \
  -height {15} \
  -highlightthickness {0} \
  -relief {raised} \
  -width {18}
 # bindings
 bind .frame1.frame3.frame1.frame9.canvas2 <Button-1> {%W configure -relief sunken}
 bind .frame1.frame3.frame1.frame9.canvas2 <ButtonRelease-1> {%W configure -relief raised
if {[winfo containing %X %Y] == "%W"} {
            set recno_requested [chgnum $recno_requested down]
  }}
 bind .frame1.frame3.frame1.frame9.canvas2 <Enter> {%W configure -bg systembuttonhighlight}
 bind .frame1.frame3.frame1.frame9.canvas2 <Leave> {%W configure -bg systembuttonface}

 # build widget .frame1.frame3.frame1.frame9.entry3
 entry .frame1.frame3.frame1.frame9.entry3 \
  -background {azure} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -textvariable {recno_requested} \
  -width {6}
 # bindings
 bind .frame1.frame3.frame1.frame9.entry3 <Key-Return> {set recno_requested [chgnum $recno_requested set]}
```

```
# build widget .frame1.frame3.frame1.frame9.canvas5
canvas .frame1.frame3.frame1.frame9.canvas5 \
  -borderwidth {2} \
  -height {15} \
  -highlightthickness {0} \
  -relief {raised} \
  -width {18}
# bindings
bind .frame1.frame3.frame1.frame9.canvas5 <Button-1> {%W configure -relief sunken}
bind .frame1.frame3.frame1.frame9.canvas5 <ButtonRelease-1> {%W configure -relief raised
if {[winfo containing %X %Y] == "%W"} {
            set recno_requested [chgnum $recno_requested up]
 }}
bind .frame1.frame3.frame1.frame9.canvas5 <Enter> {%W configure -bg systembuttonhighlight}
bind .frame1.frame3.frame1.frame9.canvas5 <Leave> {%W configure -bg systembuttonface}

# build widget .frame1.frame3.frame1.label13
label .frame1.frame3.frame1.label13 \
  -borderwidth {3} \
  -font {-adobe-helvetica-bold-r-normal--14-140-75-75-p-82-iso8859-1} \
  -foreground {red} \
  -textvariable {del_req} \
  -width {6}

# build widget .frame1.frame3.frame10
frame .frame1.frame3.frame10 \
  -height {30} \
  -width {30}

# build widget .frame1.frame3.frame10.label11
label .frame1.frame3.frame10.label11 \
  -borderwidth {1} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {raised} \
  -text {Name} \
  -width {10}

# build widget .frame1.frame3.frame10.entry12
entry .frame1.frame3.frame10.entry12 \
  -background {azure} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -textvariable {ptid_req}

# build widget .frame1.frame3.frame15
frame .frame1.frame3.frame15 \
  -height {30} \
  -width {30}

# build widget .frame1.frame3.frame15.label11
label .frame1.frame3.frame15.label11 \
  -borderwidth {1} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {raised} \
  -text {Description} \
  -width {10}

# build widget .frame1.frame3.frame15.entry12
entry .frame1.frame3.frame15.entry12 \
  -background {azure} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -textvariable {desc_req}

# build widget .frame1.frame3.frame2
frame .frame1.frame3.frame2 \
  -height {30} \
  -width {30}

# build widget .frame1.frame3.frame2.label11
label .frame1.frame3.frame2.label11 \
  -borderwidth {1} \
```

```
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {raised} \
  -text {System} \
  -width {10}

# build widget .frame1.frame3.frame2.entry12
entry .frame1.frame3.frame2.entry12 \
  -background {azure} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -textvariable {sys_req}

# build widget .frame1.frame3.frame16
frame .frame1.frame3.frame16 \
  -height {30} \
  -width {30}

# build widget .frame1.frame3.frame16.label11
label .frame1.frame3.frame16.label11 \
  -borderwidth {1} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {raised} \
  -text {Type} \
  -width {10}

# build widget .frame1.frame3.frame16.entry12
entry .frame1.frame3.frame16.entry12 \
  -background {azure} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -takefocus {1} \
  -textvariable {type_req}

# build widget .frame1.frame3.frame16.canvas4
canvas .frame1.frame3.frame16.canvas4 \
  -borderwidth {1} \
  -height {15} \
  -highlightthickness {0} \
  -relief {raised} \
  -width {16}
# bindings
bind .frame1.frame3.frame16.canvas4 <Button-1> {tk_popup .frame1.frame3.frame16.canvas4.m [expr %X - %x + 1] [expr %Y - %y + 2 + [%W cget -height]]}
bind .frame1.frame3.frame16.canvas4 <Enter> {%W configure -bg systembuttonhighlight}
bind .frame1.frame3.frame16.canvas4 <Leave> {%W configure -bg systembuttonface}

# build widget .frame1.frame3.frame16.canvas4.m
menu .frame1.frame3.frame16.canvas4.m \
  -activeborderwidth {0} \
  -borderwidth {0} \
  -tearoff {0}
.frame1.frame3.frame16.canvas4.m add command \
  -command {set type_req uchar} \
  -label {unsigned char}
.frame1.frame3.frame16.canvas4.m add command \
  -command {set type_req char} \
  -label {char}
.frame1.frame3.frame16.canvas4.m add command \
  -command {set type_req short} \
  -label {short integer}
.frame1.frame3.frame16.canvas4.m add command \
  -command {set type_req long} \
  -label {long integer}
.frame1.frame3.frame16.canvas4.m add command \
  -command {set type_req float} \
  -label {float}
.frame1.frame3.frame16.canvas4.m add command \
  -command {set type_req double} \
  -label {double}
.frame1.frame3.frame16.canvas4.m add cascade \
  -label {global partitions} \
  -menu {.frame1.frame3.frame16.canvas4.m.m}

# build widget .frame1.frame3.frame16.canvas4.m.m
```

```
menu .frame1.frame3.frame16.canvas4.m.m \
  -borderwidth {0} \
  -tearoff {0}
.frame1.frame3.frame16.canvas4.m.m add command \
  -command {set type_req globic} \
  -label {Initial Condition variables}
.frame1.frame3.frame16.canvas4.m.m add command \
  -command {set type_req globcon} \
  -label {constant coefficients}
.frame1.frame3.frame16.canvas4.m.m add command \
  -command {set type_req global} \
  -label {non-saved variables}

# build widget .frame1.frame3.frame17
frame .frame1.frame3.frame17 \
  -height {30} \
  -width {30}

# build widget .frame1.frame3.frame17.label11
label .frame1.frame3.frame17.label11 \
  -borderwidth {1} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {raised} \
  -text {Dimensions} \
  -width {10}

# build widget .frame1.frame3.frame17.entry12
entry .frame1.frame3.frame17.entry12 \
  -background {azure} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -textvariable {dim_req}

# build widget .frame1.frame3.frame18
frame .frame1.frame3.frame18 \
  -height {30} \
  -width {30}

# build widget .frame1.frame3.frame18.label11
label .frame1.frame3.frame18.label11 \
  -borderwidth {1} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {raised} \
  -text {Formating} \
  -width {10}

# build widget .frame1.frame3.frame18.entry12
entry .frame1.frame3.frame18.entry12 \
  -background {azure} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -textvariable {form_req}

# build widget .frame1.frame3.frame18.canvas4
canvas .frame1.frame3.frame18.canvas4 \
  -borderwidth {1} \
  -height {15} \
  -highlightthickness {0} \
  -relief {raised} \
  -width {16}
# bindings
bind .frame1.frame3.frame18.canvas4 <Button-1> {tk_popup .frame1.frame3.frame18.canvas4.m [expr %X - %x + 1] [expr %Y - %y + 2 + [%W cget -height]]}
bind .frame1.frame3.frame18.canvas4 <Enter> {%W configure -bg systembuttonhighlight}
bind .frame1.frame3.frame18.canvas4 <Leave> {%W configure -bg systembuttonface}

# build widget .frame1.frame3.frame18.canvas4.m
menu .frame1.frame3.frame18.canvas4.m \
  -activeborderwidth {0} \
  -borderwidth {0} \
  -tearoff {0}
.frame1.frame3.frame18.canvas4.m add command \
  -command {set form_req %u} \
  -label {%u}
```

```
.frame1.frame3.frame18.canvas4.m add command \
 -command {set form_req %d} \
 -label {%d}
.frame1.frame3.frame18.canvas4.m add command \
 -command {set form_req %f} \
 -label {%f}
.frame1.frame3.frame18.canvas4.m add command \
 -command {set form_req %.2f} \
 -label {%.2f}
.frame1.frame3.frame18.canvas4.m add command \
 -command {set form_req %.5G} \
 -label {%.5G}
.frame1.frame3.frame18.canvas4.m add command \
 -command {set form_req %.5E} \
 -label {%.5E}

# build widget .frame1.frame3.frame19
frame .frame1.frame3.frame19 \
 -height {30} \
 -width {30}

# build widget .frame1.frame3.frame19.label11
label .frame1.frame3.frame19.label11 \
 -borderwidth {1} \
 -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
 -relief {raised} \
 -text {Units} \
 -width {10}

# build widget .frame1.frame3.frame19.entry12
entry .frame1.frame3.frame19.entry12 \
 -background {azure} \
 -borderwidth {0} \
 -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
 -highlightthickness {0} \
 -textvariable {unit_req}

# build widget .frame1.frame3.frame20
frame .frame1.frame3.frame20 \
 -height {30} \
 -width {30}

# build widget .frame1.frame3.frame20.label11
label .frame1.frame3.frame20.label11 \
 -borderwidth {1} \
 -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
 -relief {raised} \
 -text {Partition} \
 -width {10}

# build widget .frame1.frame3.frame20.entry12
entry .frame1.frame3.frame20.entry12 \
 -background {azure} \
 -borderwidth {0} \
 -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
 -highlightthickness {0} \
 -relief {flat} \
 -textvariable {pred_req}

# build widget .frame1.frame3.frame22
frame .frame1.frame3.frame22 \
 -height {30} \
 -width {30}

# build widget .frame1.frame3.frame22.label11
label .frame1.frame3.frame22.label11 \
 -borderwidth {1} \
 -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
 -relief {raised} \
 -text {Notes} \
 -width {10}

# build widget .frame1.frame3.frame22.text24
text .frame1.frame3.frame22.text24 \
 -background {azure} \
```

```
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -height {5} \
  -highlightthickness {1} \
  -pady {0} \
  -relief {flat} \
  -width {40}

# build widget .frame1.frame3.frame0
frame .frame1.frame3.frame0 \
  -height {30} \
  -width {30}

# build widget .frame1.frame3.frame0.button1
button .frame1.frame3.frame0.button1 \
  -borderwidth {1} \
  -command {DBase_Out_add [modrec add]} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -padx {5} \
  -pady {1} \
  -text {Add}

# build widget .frame1.frame3.frame0.button4
button .frame1.frame3.frame0.button4 \
  -borderwidth {1} \
  -command {DBase_Out_add [modrec mod]} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -padx {5} \
  -pady {1} \
  -text {Modify}

# build widget .frame1.frame3.frame0.button5
button .frame1.frame3.frame0.button5 \
  -borderwidth {1} \
  -command {delrec} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -padx {5} \
  -pady {1} \
  -text {Delete}

# build widget .frame1.frame3.frame0.button6
button .frame1.frame3.frame0.button6 \
  -borderwidth {1} \
  -command {undelrec} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -padx {5} \
  -pady {1} \
  -text {Undelete}

# build widget .frame1.frame3.frame0.button19
button .frame1.frame3.frame0.button19 \
  -borderwidth {1} \
  -command {ShowWindow.options
tabselect 4 tab4.2} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -padx {5} \
  -pady {1} \
  -text {Options}

# build widget .frame1.frame3.frame0.button7
button .frame1.frame3.frame0.button7 \
  -borderwidth {1} \
  -command {selbuf_add [cmdrec]} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -padx {5} \
  -pady {1} \
  -text {Copy}

# build widget .frame1.frame3.frame0.button0
```

```
button .frame1.frame3.frame0.button0 \
  -borderwidth {1} \
  -command {file_import} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -padx {5} \
  -pady {1} \
  -text {Import}

# build widget .frame1.frame3.frame0.button2
button .frame1.frame3.frame0.button2 \
  -borderwidth {1} \
  -command {set answer [tk_messageBox -message "Continuing this operation will copy all data base records to the select buffer!" -type okcancel -icon warning]
if {$answer == "ok"} {pdm_dump}} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -padx {5} \
  -pady {1} \
  -text {Dump}

# build widget .frame1.frame3.frame3
frame .frame1.frame3.frame3 \
  -height {30} \
  -width {30}

# build widget .frame1.frame3.frame3.label11
label .frame1.frame3.frame3.label11 \
  -borderwidth {1} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -relief {raised} \
  -text {Offset} \
  -width {10}

# build widget .frame1.frame3.frame3.entry12
entry .frame1.frame3.frame3.entry12 \
  -background {azure} \
  -borderwidth {0} \
  -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
  -highlightthickness {0} \
  -state {disabled} \
  -textvariable {addr_req}

# build widget .frame1.frame0
frame .frame1.frame0 \
  -borderwidth {2} \
  -height {30} \
  -relief {ridge} \
  -width {30}

# build widget .frame1.frame0.frame
frame .frame1.frame0.frame \
  -background {azure} \
  -borderwidth {4} \
  -relief {ridge}

# build widget .frame1.frame0.frame.scrollbar3
scrollbar .frame1.frame0.frame.scrollbar3 \
  -activebackground {grey85} \
  -borderwidth {0} \
  -command {.frame1.frame0.frame.listbox1 xview} \
  -elementborderwidth {2} \
  -highlightthickness {0} \
  -orient {horizontal} \
  -relief {raised} \
  -troughcolor {grey87}

# build widget .frame1.frame0.frame.scrollbar2
scrollbar .frame1.frame0.frame.scrollbar2 \
  -activebackground {grey85} \
  -borderwidth {0} \
  -command {.frame1.frame0.frame.listbox1 yview} \
  -elementborderwidth {2} \
  -highlightthickness {0} \
  -relief {raised} \
  -troughcolor {grey87}
```

```
# build widget .frame1.frame0.frame.listbox1
listbox .frame1.frame0.frame.listbox1 \
  -background {azure} \
  -exportselection {0} \
  -height {6} \
  -highlightthickness {0} \
  -width {23} \
  -xscrollcommand {.frame1.frame0.frame.scrollbar3 set} \
  -yscrollcommand {.frame1.frame0.frame.scrollbar2 set}
# bindings
bind .frame1.frame0.frame.listbox1 <B1-Motion> {%W select anchor [%W nearest %y]}
bind .frame1.frame0.frame.listbox1 <Double-Button-1> {open_file [%W get [%W nearest %y]]}
bind .frame1.frame0.frame.listbox1 <Shift-B1-Motion> {%W select anchor [%W nearest %y]}
bind .frame1.frame0.frame.listbox1 <Shift-Button-1> {%W select anchor [%W nearest %y]}

# build widget .frame1.frame0.canvas0
canvas .frame1.frame0.canvas0 \
  -height {16} \
  -highlightthickness {0} \
  -width {182}

# build widget .frame2
frame .frame2 \
  -height {30} \
  -width {30}

# build widget .frame2.frame
frame .frame2.frame \
  -borderwidth {2} \
  -relief {ridge}

# build widget .frame2.frame.scrollbar1
scrollbar .frame2.frame.scrollbar1 \
  -activebackground {grey85} \
  -borderwidth {1} \
  -command {.frame2.frame.text2 yview} \
  -relief {raised} \
  -troughcolor {grey87}

# build widget .frame2.frame.text2
text .frame2.frame.text2 \
  -background {azure} \
  -borderwidth {0} \
  -height {6} \
  -highlightthickness {0} \
  -padx {0} \
  -pady {0} \
  -relief {flat} \
  -wrap {none} \
  -xscrollcommand {.frame2.frame.scrollbar3 set} \
  -yscrollcommand {.frame2.frame.scrollbar1 set}

# build widget .frame2.frame.scrollbar3
scrollbar .frame2.frame.scrollbar3 \
  -activebackground {grey85} \
  -borderwidth {0} \
  -command {.frame2.frame.text2 xview} \
  -elementborderwidth {2} \
  -highlightthickness {0} \
  -orient {horizontal} \
  -relief {raised} \
  -troughcolor {grey87}

# build widget .frame2.frame.canvas0
canvas .frame2.frame.canvas0 \
  -height {16} \
  -highlightthickness {0} \
  -width {385}

# pack master .frame0
pack configure .frame0.menubutton1 \
  -side left
pack configure .frame0.menubutton6 \
  -side left
```

```
pack configure .frame0.menubutton8 \
  -side left
pack configure .frame0.menubutton2 \
  -side left

# pack master .frame1
pack configure .frame1.frame0 \
  -fill y \
  -side left
pack configure .frame1.frame3 \
  -expand 1 \
  -fill both \
  -side left

# pack master .frame1.frame3
pack configure .frame1.frame3.frame0 \
  -fill x
pack configure .frame1.frame3.frame1 \
  -fill x
pack configure .frame1.frame3.frame10 \
  -fill x
pack configure .frame1.frame3.frame15 \
  -fill x
pack configure .frame1.frame3.frame2 \
  -fill x
pack configure .frame1.frame3.frame16 \
  -fill x
pack configure .frame1.frame3.frame17 \
  -fill x
pack configure .frame1.frame3.frame18 \
  -fill x
pack configure .frame1.frame3.frame19 \
  -fill x
pack configure .frame1.frame3.frame3 \
  -fill x
pack configure .frame1.frame3.frame20 \
  -fill x
pack configure .frame1.frame3.frame22 \
  -expand 1 \
  -fill both

# pack master .frame1.frame3.frame1
pack configure .frame1.frame3.frame1.frame9 \
  -side left
pack configure .frame1.frame3.frame1.frame7 \
  -side left
pack configure .frame1.frame3.frame1.label13 \
  -side right

# pack master .frame1.frame3.frame1.frame7
pack configure .frame1.frame3.frame1.frame7.label8 \
  -ipadx 4 \
  -side left
pack configure .frame1.frame3.frame1.frame7.entry6 \
  -side left

# pack master .frame1.frame3.frame1.frame9
pack configure .frame1.frame3.frame1.frame9.label8 \
  -ipadx 4 \
  -side left
pack configure .frame1.frame3.frame1.frame9.canvas2 \
  -side left
pack configure .frame1.frame3.frame1.frame9.entry3 \
  -side left
pack configure .frame1.frame3.frame1.frame9.canvas5 \
  -side left

# pack master .frame1.frame3.frame10
pack configure .frame1.frame3.frame10.label11 \
  -side left
pack configure .frame1.frame3.frame10.entry12 \
  -side left

# pack master .frame1.frame3.frame15
pack configure .frame1.frame3.frame15.label11 \
```

```
  -side left
pack configure .frame1.frame3.frame15.entry12 \
  -side left

# pack master .frame1.frame3.frame2
pack configure .frame1.frame3.frame2.label11 \
  -side left
pack configure .frame1.frame3.frame2.entry12 \
  -side left

# pack master .frame1.frame3.frame16
pack configure .frame1.frame3.frame16.label11 \
  -side left
pack configure .frame1.frame3.frame16.entry12 \
  -side left
pack configure .frame1.frame3.frame16.canvas4 \
  -side left

# pack master .frame1.frame3.frame17
pack configure .frame1.frame3.frame17.label11 \
  -side left
pack configure .frame1.frame3.frame17.entry12 \
  -side left

# pack master .frame1.frame3.frame18
pack configure .frame1.frame3.frame18.label11 \
  -side left
pack configure .frame1.frame3.frame18.entry12 \
  -side left
pack configure .frame1.frame3.frame18.canvas4 \
  -side left

# pack master .frame1.frame3.frame19
pack configure .frame1.frame3.frame19.label11 \
  -side left
pack configure .frame1.frame3.frame19.entry12 \
  -side left

# pack master .frame1.frame3.frame20
pack configure .frame1.frame3.frame20.label11 \
  -side left
pack configure .frame1.frame3.frame20.entry12 \
  -side left

# pack master .frame1.frame3.frame22
pack configure .frame1.frame3.frame22.label11 \
  -anchor nw \
  -side left
pack configure .frame1.frame3.frame22.text24 \
  -expand 1 \
  -fill both \
  -side left

# pack master .frame1.frame3.frame0
pack configure .frame1.frame3.frame0.button1 \
  -side left
pack configure .frame1.frame3.frame0.button4 \
  -side left
pack configure .frame1.frame3.frame0.button5 \
  -side left
pack configure .frame1.frame3.frame0.button6 \
  -side left
pack configure .frame1.frame3.frame0.button2 \
  -side right
pack configure .frame1.frame3.frame0.button0 \
  -side right
pack configure .frame1.frame3.frame0.button7 \
  -side right
pack configure .frame1.frame3.frame0.button19 \
  -side right

# pack master .frame1.frame3.frame3
pack configure .frame1.frame3.frame3.label11 \
  -side left
pack configure .frame1.frame3.frame3.entry12 \
```

```
  -side left

 # pack master .frame1.frame0
 pack configure .frame1.frame0.frame \
  -expand 1 \
  -fill y
 pack configure .frame1.frame0.canvas0

 # pack master .frame1.frame0.frame
 pack configure .frame1.frame0.frame.scrollbar2 \
  -fill y \
  -side right
 pack configure .frame1.frame0.frame.listbox1 \
  -expand 1 \
  -fill both
 pack configure .frame1.frame0.frame.scrollbar3 \
  -fill x \
  -side bottom

 # pack master .frame2
 pack configure .frame2.frame \
  -expand 1 \
  -fill both \
  -side bottom

 # pack master .frame2.frame
 pack configure .frame2.frame.scrollbar1 \
  -fill y \
  -side right
 pack configure .frame2.frame.text2 \
  -expand 1 \
  -fill both
 pack configure .frame2.frame.canvas0 \
  -side left
 pack configure .frame2.frame.scrollbar3 \
  -expand 1 \
  -fill x \
  -side left

 # pack master .
 pack configure .frame0 \
  -fill x
 pack configure .frame1 \
  -expand 1 \
  -fill both
 pack configure .frame2 \
  -expand 1 \
  -fill both \
  -side bottom

 # build canvas items .frame1.frame3.frame1.frame9.canvas2
 set xfTmpTag [.frame1.frame3.frame1.frame9.canvas2 create polygon 15.0 4.0 5.0 10.0 15.0 15.0 15.0 4.0]
 .frame1.frame3.frame1.frame9.canvas2 itemconfigure $xfTmpTag \
  -outline {black}
 # build canvas items .frame1.frame3.frame1.frame9.canvas5
 set xfTmpTag [.frame1.frame3.frame1.frame9.canvas5 create polygon 6.0 4.0 16.0 10.0 6.0 15.0 6.0 4.0]
 .frame1.frame3.frame1.frame9.canvas5 itemconfigure $xfTmpTag \
  -outline {black}
 # build canvas items .frame1.frame3.frame16.canvas4
 set xfTmpTag [.frame1.frame3.frame16.canvas4 create polygon 5.0 4.0 12.0 4.0 9.0 12.0 5.0 4.0]
 .frame1.frame3.frame16.canvas4 itemconfigure $xfTmpTag \
  -outline {black} \
  -tags {current}
 # build canvas items .frame1.frame3.frame18.canvas4
 set xfTmpTag [.frame1.frame3.frame18.canvas4 create polygon 5.0 4.0 13.0 4.0 9.0 13.0 5.0 4.0]
 .frame1.frame3.frame22.text24 insert end {}
 # build canvas items .frame1.frame0.canvas0
 set xfTmpTag [.frame1.frame0.canvas0 create polygon 70.0 0.0 74.0 14.0 114.0 14.0 118.0 0.0 70.0 0.0]
 .frame1.frame0.canvas0 itemconfigure $xfTmpTag \
  -fill {grey68} \
  -outline {black} \
  -tags {tab2.2}
 .frame1.frame0.canvas0 bind $xfTmpTag <Button-1> {tabselect 2 tab2.2}
 set xfTmpTag [.frame1.frame0.canvas0 create text 114.0 14.0]
 .frame1.frame0.canvas0 itemconfigure $xfTmpTag \
```

```
 -anchor {se} \
 -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
 -tags {tab2.2} \
 -text {Help }
.frame1.frame0.canvas0 bind $xfTmpTag <Button-1> {tabselect 2 tab2.2}
set xfTmpTag [.frame1.frame0.canvas0 create line 70.0 0.0 74.0 14.0 114.0 14.0 118.0 0.0]
.frame1.frame0.canvas0 itemconfigure $xfTmpTag \
 -tags {tab2.2}
set xfTmpTag [.frame1.frame0.canvas0 create polygon 35.0 0.0 39.0 14.0 69.0 14.0 73.0 0.0 35.0 0.0]
.frame1.frame0.canvas0 itemconfigure $xfTmpTag \
 -fill {grey68} \
 -outline {black} \
 -tags {tab2.1}
.frame1.frame0.canvas0 bind $xfTmpTag <Button-1> {tabselect 2 tab2.1}
set xfTmpTag [.frame1.frame0.canvas0 create line 35.0 0.0 39.0 14.0 69.0 14.0 73.0 0.0]
.frame1.frame0.canvas0 itemconfigure $xfTmpTag \
 -tags {tab2.1}
set xfTmpTag [.frame1.frame0.canvas0 create text 69.0 14.0]
.frame1.frame0.canvas0 itemconfigure $xfTmpTag \
 -anchor {se} \
 -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
 -tags {tab2.1} \
 -text {ODS}
.frame1.frame0.canvas0 bind $xfTmpTag <Button-1> {tabselect 2 tab2.1}
set xfTmpTag [.frame1.frame0.canvas0 create polygon 0.0 0.0 4.0 14.0 34.0 14.0 38.0 0.0 0.0 0.0]
.frame1.frame0.canvas0 itemconfigure $xfTmpTag \
 -fill {azure} \
 -outline {azure} \
 -tags {tab2.0}
.frame1.frame0.canvas0 bind $xfTmpTag <Button-1> {tabselect 2 tab2.0}
set xfTmpTag [.frame1.frame0.canvas0 create line 0.0 0.0 4.0 14.0 34.0 14.0 38.0 0.0]
.frame1.frame0.canvas0 itemconfigure $xfTmpTag \
 -tags {tab2.0}
set xfTmpTag [.frame1.frame0.canvas0 create text 34.0 14.0]
.frame1.frame0.canvas0 itemconfigure $xfTmpTag \
 -anchor {se} \
 -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
 -tags {tab2.0} \
 -text {UDS}
.frame1.frame0.canvas0 bind $xfTmpTag <Button-1> {tabselect 2 tab2.0}
.frame2.frame.text2 insert end {}
# build canvas items .frame2.frame.canvas0
set xfTmpTag [.frame2.frame.canvas0 create polygon 0.0 0.0 4.0 14.0 44.0 14.0 48.0 0.0 0.0 0.0]
.frame2.frame.canvas0 itemconfigure $xfTmpTag \
 -fill {grey68} \
 -outline {black} \
 -tags {tab0.0}
.frame2.frame.canvas0 bind $xfTmpTag <Button-1> {tabselect 0 tab0.0}
set xfTmpTag [.frame2.frame.canvas0 create text 44.0 14.0]
.frame2.frame.canvas0 itemconfigure $xfTmpTag \
 -anchor {se} \
 -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
 -tags {tab0.0} \
 -text {Build }
.frame2.frame.canvas0 bind $xfTmpTag <Button-1> {tabselect 0 tab0.0}
set xfTmpTag [.frame2.frame.canvas0 create line 0.0 0.0 4.0 14.0 44.0 14.0 48.0 0.0]
.frame2.frame.canvas0 itemconfigure $xfTmpTag \
 -tags {tab0.0}
set xfTmpTag [.frame2.frame.canvas0 create polygon 45.0 0.0 49.0 14.0 113.0 14.0 117.0 0.0 45.0 0.0]
.frame2.frame.canvas0 itemconfigure $xfTmpTag \
 -fill {azure} \
 -outline {azure} \
 -tags {tab0.1}
.frame2.frame.canvas0 bind $xfTmpTag <Button-1> {tabselect 0 tab0.1}
set xfTmpTag [.frame2.frame.canvas0 create line 45.0 0.0 49.0 14.0 113.0 14.0 117.0 0.0]
.frame2.frame.canvas0 itemconfigure $xfTmpTag \
 -tags {tab0.1}
set xfTmpTag [.frame2.frame.canvas0 create text 113.0 14.0]
.frame2.frame.canvas0 itemconfigure $xfTmpTag \
 -anchor {se} \
 -font {-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1} \
 -tags {tab0.1} \
 -text {DataBase}
.frame2.frame.canvas0 bind $xfTmpTag <Button-1> {tabselect 0 tab0.1}
```

```
 if {"[info procs XFEdit]" != ""} {
   catch "XFMiscBindWidgetTree ."
   after 2 "catch {XFEditSetShowWindows}"
 }
}


# User defined procedures


# Procedure: Build_Out
proc Build_Out {} {
global build_msg
global output_type

.frame2.frame.text2 delete 1.0 end
.frame2.frame.text2 insert end $build_msg
set output_type build
}


# Procedure: Build_Out_add
proc Build_Out_add { msg} {
global build_msg
global output_type

set build_msg "$build_msg$msg\n"
if {$output_type == "build"} {
   .frame2.frame.text2 insert end "$msg\n"
  }
}


# Procedure: DBase_Out
proc DBase_Out {} {
global dbase_msg
global output_type

.frame2.frame.text2 delete 1.0 end
.frame2.frame.text2 insert end $dbase_msg
set output_type dbase
}


# Procedure: DBase_Out_add
proc DBase_Out_add { msg} {
global dbase_msg
global output_type

set dbase_msg "$dbase_msg$msg\n"
if {$output_type == "dbase"} {
   .frame2.frame.text2 insert end "$msg\n"
   .frame2.frame.text2 see end
  }
}


# Procedure: Help_files
proc Help_files {} {
global helplist
global filetype

set filetype help
.frame1.frame0.frame.listbox1 delete 0 end
eval ".frame1.frame0.frame.listbox1 insert end $helplist"
}


# Procedure: ODS_files
proc ODS_files {} {
global ODSPATH
global filetype
```

```
set filetype ODS
set extlist ".exe .com .dll .obj .lib .exp"
.frame1.frame0.frame.listbox1 delete 0 end
if {[catch {glob $ODSPATH/*} flist]} {set flist ""}
set flist [lsort $flist]
foreach fname $flist {
        set tail [file tail $fname]
        if {[file isfile $fname]} {
          set ext [file extension $fname]
          if    {[lsearch $extlist $ext] == -1} {
                .frame1.frame0.frame.listbox1 insert end $tail
                }
         }
    }
}


# Procedure: UDS_files
proc UDS_files {} {
global UDSPATH
global filetype

set filetype UDS
set extlist ".exe .com .dll .obj .lib .exp"
.frame1.frame0.frame.listbox1 delete 0 end
if {[catch {glob $UDSPATH/*} flist]} {set flist ""}
set flist [lsort $flist]
foreach fname $flist {
        set tail [file tail $fname]
        if {[file isfile $fname]} {
          set ext [file extension $fname]
          if    {[lsearch $extlist $ext] == -1} {
                .frame1.frame0.frame.listbox1 insert end $tail
                }
         }
    }
}


# Procedure: build__check_needed_dirs
proc build__check_needed_dirs {} {
global ULIBCON
global ULIBSEG
global ULIBSUB
global redir

        set dirlist "$ULIBCON $ULIBSEG $ULIBSUB"
        foreach dir $dirlist {
                if {![file exists $dir]} {
                        file mkdir $dir
                        if {![file isdir $dir]} {
                                sendout "ABORT!  Unable to create directory named $dir."
                                exit 1
                          }
                  } elseif {![file isdir $dir]} {
                                sendout "ABORT!  Can not create directory named $dir."
                                sendout "       A file by that name already exists"
                                exit 2
                  }
                }
}


# Procedure: build__check_needed_files
proc build__check_needed_files {} {
#  This procedure ensures the existance of necessary files in the
#  current directory. Missing files are copied from the ODSPATH.
#
global ODSPATH
global EXTOBJ
global redir

        set filelist "${ODSPATH}/frameseq.in"
        foreach fname $filelist {
                set shortname [file tail $fname]
```

```
                if {![file exists $shortname]} {
                        if {![file exists $fname]} {
                                sendout "ABORT!  Cannot find $fname"
                                exit 3
                        }
                        file copy $fname $shortname
                } elseif {[file isdir $shortname]} {
                                sendout "ABORT!  Cannot create file named $shortname."
                                sendout "       A directory by that name already exists"
                                exit 4
                }
        }
}


# Procedure: build__compile_C
proc build__compile_C { target srcfile} {
#  This procedure compiles C files
#
#  Returns:  0 on success
#                       -1 on error
#
global CC
global CFLAGS
global OBJOUT
global CSCANNER
global verbose
global redir
global SHELL

        set hdrfile [file root $srcfile].h
        set cmd0 "    $CSCANNER <$srcfile >$hdrfile"
        if {$verbose} {sendout $cmd0}
        set cmd "exec $SHELL $cmd0"
        set rtnval 0
        if {[catch {eval $cmd} result]} {
                sendout $result
                set rtnval -1
         }{
                set cmd0 "    $CC $CFLAGS $srcfile ${OBJOUT}$target"
                sendout $cmd0
                set cmd "exec $SHELL $cmd0"
                if {[catch {eval $cmd} result]} {
                        sendout $result
                        set rtnval -1
                 }
         }
        return $rtnval
}


# Procedure: build__compile_FORTRAN
proc build__compile_FORTRAN { target srcfile} {
#  This procedure compiles FORTRAN files
#
#  Returns:  0 on success
#                       -1 on error
#
global F77
global FFLAGS
global OBJOUT
global FSCANNER
global verbose
global redir
global SHELL

        set hdrfile [file root $srcfile].inc
        set cmd0 "    $FSCANNER <$srcfile >$hdrfile"
        if {$verbose} {sendout $cmd0}
        set cmd "exec $cmd0"
        set rtnval 0
        if {[catch {eval $cmd} result]} {
                sendout $result
                set rtnval -1
         }{
```

```
                set cmd0 "    $F77 $FFLAGS $srcfile ${OBJOUT}$target"
                sendout $cmd0
                set cmd "exec $cmd0"
                if {[catch {eval $cmd} result]} {
                        sendout $result
                        set rtnval -1
                }
        }
        return $rtnval
}

# Procedure: build__get_realpath
proc build__get_realpath { orgpath} {
#       ____________________________________________
#       |           ATTENTION:                     |
#       |                                          |
#       | This procedure contains Operating System or  |
#       | Compiler specific code.                  |
#       |__________________________________________|
#
global tcl_platform

        set rtnval $orgpath
        switch $tcl_platform(platform) {
                windows    {
                                # check for virtual drives created by SUBST command
                                set orgpath [string tolower $orgpath]
                                set subst_string [exec subst]
                                regsub -all {\\} $subst_string / rstring
                                set lstring [string tolower $rstring]
                                set substlist [split $lstring "\n"]
                                foreach subst $substlist {
                                        set drive [lindex $subst 0]
                                        if {[string first $drive $orgpath] == 0} {
                                                # replace virtual drive with actual path
                                                set substpath [lindex $subst 2]
                                                set rtnval [file join $substpath [string range $orgpath 3 end]]
                                        }
                                }
                        }
                unix                    {}
        }
        return $rtnval
}

# Procedure: build__init
proc build__init {} {
#       ____________________________________________
#       |           ATTENTION:                     |
#       |                                          |
#       | This procedure contains Operating System or  |
#       | Compiler specific code.                  |
#       |__________________________________________|
#
# This procedure performs necessary initilizations of variables
#
# This procedure returns the name of the application which will
# be built.
#
global argv
global env
global tcl_platform
global ODSPATH
global UDSPATH
global EXEOUT
global OBJOUT
global EXTEXE
global EXTLIB
global EXTOBJ
global LIBOPT
global CFLAGS
global FFLAGS
global BFLAGS
```

```
global CSCANNER
global FSCANNER
global SHELL
global APPEND
global LIBRARIES
global CC
global F77
global BUILD
global LIBCON
global LIBSEG
global LIBSUB
global ULIBCON
global ULIBSEG
global ULIBSUB
global LINKNAME
global filesbuilt
global haderror
global verbose

# read ODSPATH environment variable
if {[info exists env(ODSPATH)]} {
        set ODSPATH $env(ODSPATH)
        regsub -all {\\} $ODSPATH / ODSPATH
  } {set ODSPATH [pwd] }
set ODSPATH [build__get_realpath $ODSPATH]

# set user level path to current directory
set UDSPATH [build__get_realpath [pwd]]

# set up operating system/compiler specific variables
switch $tcl_platform(platform) {
        windows    {
                        set EXEOUT              "/Fe"
                        set OBJOUT              "/Fo"
                        set EXTEXE      ".exe"
                        set EXTLIB      ".lib"
                        set EXTOBJ              ".obj"
                        set LIBOPT              "/link"
                        set CFLAGS              "/c /nologo"
                        set FFLAGS              "/c /4Yd /WX /nologo"
                        set BFLAGS              "/nologo "
                        regsub -all {\\}        "$env(ComSpec) /c" / SHELL
                        set APPEND              "$env(ComSpec) /c type"
                        regsub -all {\\} $APPEND / APPEND
                        set LIBRARIES   "winmm.lib kernel32.lib user32.lib gdi32.lib  winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib
oleaut32.lib uuid.lib  odbc32.lib odbccp32.lib /subsystem:console /incremental:no /def:$ODSPATH/rtsched.def"
                        set CC          "cl"
                        set F77         "fl32"
                        set BUILD       "fl32"
                        set LINKNAME    "rtsched.exe"
                       }
        unix            {
                        set EXEOUT              "-o "
                        set OBJOUT              "-o "
                        set EXTEXE              ""
                        set EXTLIB      ".a"
                        set EXTOBJ              ".o"
                        set LIBOPT              ""
                        set CFLAGS              "-c -O2"
                        set FFLAGS              "-c -O2"
                        set BFLAGS              ""
                        set SHELL       "/bin/csh -c"
                        set APPEND              "cat"
                        set LIBRARIES   ""
                        set CC          "cc"
                        set F77         "f77"
                        set BUILD       "cc"
                        set LINKNAME    "rtsched"
                       }
        }

# set up common variables
set LIBCON[file join $ODSPATH control]
set LIBSEG [file join $ODSPATH segment]
set LIBSUB [file join $ODSPATH subs]
```

```
set ULIBCON             [file join $UDSPATH control]
set ULIBSEG             [file join $UDSPATH segment]
set ULIBSUB             [file join $UDSPATH subs]
set CSCANNER            "hgen"
set FSCANNER            "igen"

# determine name for new program and check for verbose status
set buildname ""
set verbose 0
foreach arg $argv {
        switch -- $arg {
    "-v"        {
                                        set verbose 1
                                      }
                default    {
                                        set buildname $arg
                                      }
                }
        }
if {$buildname == ""} {set buildname $LINKNAME}
set ext [file extension $buildname]
if {$ext != $EXTEXE} {set buildname "$buildname$EXTEXE"}

# initialize build counter and error flag
set filesbuilt 0
set haderror 0

return $buildname
}


# Procedure: build__lib
proc build__lib { dir libname} {
#       ____________________________________________
#       |           ATTENTION:                     |
#       |                                          |
#       | This procedure contains Operating System or  |
#       | Compiler specific code.                  |
#       |__________________________________________|
#
#  This procedure is used to maintain object files in a library
#
#  dir    = directory containing the library
#  libname = file name for library
#
global tcl_platform
global EXTOBJ
global env
global verbose
global redir
global SHELL

        switch $tcl_platform(platform) {
                windows    {
                                        set cmd0 "    lib /nologo $dir/*$EXTOBJ /OUT:$dir/$libname"
                                        if {$verbose} {sendout $cmd0}
                                        set cmd "exec $SHELL $cmd0"
                                        if {[catch {eval $cmd} result]} {sendout $result}
                                      }
                unix                  {
                                        set cmd0 "ar -r $dir/*$EXTOBJ $dir/$libname"
                                        if {$verbose} {sendout $cmd0}
                                        set cmd "exec $cmd0"
                                        if {[catch {eval $cmd} result]} {sendout $result}
                                      }
                }
}


# Procedure: build__link_application
proc build__link_application { xname} {
global EXTEXE
global EXTOBJ
global LIBCON
global LIBSEG
```

```
global LIBSUB
global ULIBCON
global ULIBSEG
global ULIBSUB
global EXTLIB
global BUILD
global BFLAGS
global EXEOUT
global LIBOPT
global LIBRARIES
global APPEND
global ODSPATH
global UDSPATH
global verbose
global redir
global SHELL

        set target $xname
        if {$UDSPATH != $ODSPATH} {
                set isUDS 1
          } {set isUDS 0}
        set    srclist $ODSPATH/rtsched$EXTOBJ
        lappend srclist frameseq$EXTOBJ
        set udslib $ULIBCON/libcon$EXTLIB
        if {$isUDS && [file exists $udslib]} {lappend srclist $udslib}
        lappend srclist $LIBCON/libcon$EXTLIB
        set udslib $ULIBSEG/libseg$EXTLIB
        if {$isUDS && [file exists $udslib]} {lappend srclist $udslib}
        lappend srclist $LIBSEG/libseg$EXTLIB
        set udslib $ULIBSUB/libsub$EXTLIB
        if {$isUDS && [file exists $udslib]} {lappend srclist $udslib}
        lappend srclist $LIBSUB/libsub$EXTLIB
        set newconstants 0
        if {[file exists c.constants]} {
           if {[build__need_to_update $xname c.constants]} {set newconstants 1}
          }
        if {[eval "build__need_to_update $target $srclist"] || $newconstants} {
                sendout "Linking program"
                if {$verbose} {sendout ""}
                set cmd0 "$BUILD $BFLAGS $srclist $EXEOUT$target $LIBOPT $LIBRARIES"
                if {$verbose} {sendout $cmd0}
                set cmd "exec $SHELL $cmd0"
                if {[catch {eval $cmd} result]} {
                        sendout $result
                        sendout "\nFailed build of $xname"
                  } {
                        # attach constant coefficients data to end of executable file
                        if {$verbose} {sendout ""}
                        sendout "Attaching constant coefficients to application file."
                        set fname [build__make_constants]
                        set cmd0 "    [lindex $APPEND end] $fname >>$target"
                        if {$verbose} {sendout $cmd0}
                        set cmd "exec $APPEND $fname >>$target"
                        if {[catch {eval $cmd} result]} {
                                sendout $result
                                file delete $xname
                                sendout "Failed build of $xname"
                          } {sendout "Successful build of $xname"}
                        file delete $fname
                  }
          } {sendout "$xname already up to date."}
}


# Procedure: build__main
proc build__main {} {
        set buildname [build__init]
        build__check_needed_dirs
        build__check_needed_files
        build__make_frame_calls
        build__make_modules control
        build__make_modules segment
        build__make_modules subroutine
        build__link_application $buildname
}
```

```
# Procedure: build__make_constants
proc build__make_constants {} {
#  This procedure creates a constant coefficients file for attaching to
#  the application.
#
#  Returns the name of the file created.
#
global ODSPATH
global verbose
global redir

        # make temporary input file for rtdata program
        set fname build[pid]
        if {[catch {set fout [open $fname w]} result]} {
                sendout "ABORT!  Unable to open temporary file $fname"
                exit 6
          } {
                puts $fout "conload $ODSPATH/globcon.dat"
                if {![catch {set fin [open c.constants r]} result]} {
                        # copy c.constants file if found
                        while {![eof $fin]} {
                                gets $fin line
                                puts $fout $line
                               }
                        close $fin
                  }
                set fname2 attach[pid]
                puts $fout "consave $fname2"
                puts $fout "exit"
                close $fout
                set cmd0 "rtdata <$fname"
#       if {$verbose} {sendout $cmd0}
                set cmd "exec $cmd0"
                if {[catch {eval $cmd} results]} {sendout $result}
                file delete $fname
                return $fname2
          }
}


# Procedure: build__make_frame_calls
proc build__make_frame_calls {} {
global EXTOBJ
global CC
global CFLAGS
global OBJOUT
global verbose
global redir
global SHELL

        set srcfile frameseq.in
        set target frameseq$EXTOBJ
        if {[build__need_to_update $target $srcfile]} {
                if {$verbose} {sendout framegen}
                if {[catch {exec framegen} result]} {
                        sendout $result
                  } {
                        set srcfile frameseq.c
                        set target  frameseq$EXTOBJ
                        set cmd0 "$CC $CFLAGS $srcfile ${OBJOUT}$target"
                        if {$verbose} {sendout $cmd0}
                        set cmd "exec $SHELL $cmd0"
                        if {[catch {eval $cmd} result]} {sendout $result}
                   }
          }
        sendout "Frame sequencing file is up to date"
}


# Procedure: build__make_modules
proc build__make_modules { modtype} {
global ULIBCON
global ULIBSEG
```

```
global ULIBSUB
global EXTOBJ
global EXTLIB
global ARCHIVER
global filesbuilt
global haderror
global redir

        switch $modtype {
                control     {set typechars c; set LIB $ULIBCON}
                segment     {set typechars s; set LIB $ULIBSEG}
                subroutine  {set typechars "x y"; set LIB $ULIBSUB}
                default     {return}
          }

        sendout "Compiling $modtype modules."
        set buildctr 0

        # compile C source files in current directory, as necessary
        foreach tchar $typechars {
                if {[catch {glob ??$tchar*.c} clist]} {set clist ""}
                foreach srcfile $clist {
                                set target $LIB/[file root $srcfile]$EXTOBJ
                                if {[build__need_to_update $target $srcfile]} {
                                        if {[build__compile_C $target $srcfile]} {
                                                set haderror 1
                                        } {incr buildctr}
                                }
                        }
                }

        # compile FORTRAN source files in current directory, as necessary
        foreach tchar $typechars {
                if {[catch {glob ??$tchar*.f} flist]} {set flist ""}
                foreach srcfile $flist {
                                set target $LIB/[file root $srcfile]$EXTOBJ
                                if {[build__need_to_update $target $srcfile]} {
                                        if {[build__compile_FORTRAN $target $srcfile]} {
                                                set hadderror 1
                                        } {incr buildctr}
                                }
                        }
                }

        if {$haderror} {
                sendout "ABORT!  Terminal error encountered. Cannot continue."
                exit 5
          } {
                if {$buildctr > 0} {
                        # place new object files into library
                        build__lib $LIB lib[string range $modtype 0 2]$EXTLIB
                        incr filesbuilt $buildctr
                 }
          }
}


# Procedure: build__need_to_update
proc build__need_to_update { target args} {
        if {![file exists $target]} {return 1}
        set srclist $args
        foreach srcfile $srclist {
                if      {[file mtime $srcfile] > [file mtime $target]} {return 1}
                }
        return 0
}


# Procedure: buildload
proc buildload {} {
global loadname
global UDSPATH
global build_msg
global buildsrc_loaded
```

```
if {$buildsrc_loaded} {
        tabselect 0 tab0.0
        .frame2.frame.text2 delete 1.0 end
        set build_msg ""
        set orgdir [pwd]
        cd $UDSPATH
        build__main
        cd $orgdir
        return
  }
set fpath [findinpath build.tsh]
if {$fpath != ""} {
        set orgdir [pwd]
        if {[catch {source $fpath} build_msg]} {
                tabselect 0 tab0.0
          } {
                tabselect 0 tab0.0
                .frame2.frame.text2 delete 1.0 end
                set build_msg ""
                cd $UDSPATH
                eval "proc sendout {arg} {\nBuild_Out_add \$arg\nupdate\n}"
                set buildsrc_loaded 1
                build__main
                cd $orgdir
          }
  } {
        tabselect 0 tab0.0
        .frame2.frame.text2 delete 1.0 end
        set build_msg ""
        set orgdir [pwd]
        cd $UDSPATH
        build__main
        cd $orgdir
    }
}


# Procedure: chgnum
proc chgnum { num action} {
# check for valid number
global lastgoodnum

set temp $num
if {[catch {incr temp} result]} {return $lastgoodnum}
if {$num < 1} {return 1}
switch $action {
        up      {incr num}
        down    {
                 incr num -1
                 if {$num < 1} {set num 1}
                }
        set     {}
    }
if {![fillrec [expr $num - 1]]} {set num $lastgoodnum}
set lastgoodnum $num
return $num
}


# Procedure: cmdrec
proc cmdrec {} {
# This procedure creates a command record for the select buffer
#
# Returns: a command string for the select buffer
#
global ptid_req
global ptid_org
global desc_req
global type_req
global dim_req
global form_req
global unit_req
global sys_req
global pred_req
global note_req
```

```
global del_req
global recno_requested
global pdmsrc

set ptid [string trim $ptid_req]
set line(1) "/cmd $ptid"
set line(2) ".desc $desc_req"
set line(3) [.frame1.frame3.frame22.text24 get 1.0 end]
set line(4) ".type [string trim $type_req]"
set dim [string tolower [string trim $dim_req]]
if {$dim == "none"} {set dim ""}
set line(5) ".dim  $dim"
set form [string trim $form_req]
set line(6) ".form $form"
set unit [string trim $unit_req]
set line(7) ".unit $unit"
set sys [string trim $sys_req]
set line(8) ".sys  $sys"
set pred [string trim $pred_req]
set line(9) ".pred $pred"
set resp "$line(1)\n$line(2)\n$line(3)\n$line(4)\n$line(5)\n$line(6)\n$line(7)\n$line(8)\n$line(9)\n"
return $resp
}


# Procedure: delrec
proc delrec {} {
global ptid_req
global pdmsrc

set ptid [string trim $ptid_req]
if {[catch {expr $ptid - 1} record]} {set record $ptid}
if {$ptid != ""} {
        puts $pdmsrc "del $record"
        flush $pdmsrc
        set response [pdm_getresponse]
        fillrec $record
  }
}


# Procedure: file_copy
proc file_copy {} {
global filetype
global ODSPATH
global UDSPATH
global edit_cmd

if {$filetype == "ODS"}   {
   set index [.frame1.frame0.frame.listbox1 curselection]
   if       {$index != ""} {
        set fname [.frame1.frame0.frame.listbox1 get $index]
        set oname [file join $ODSPATH $fname]
        set uname [file join $UDSPATH $fname]
        if {[file exists $oname]} {
                catch {file copy $oname $uname}
          }
        }
   }
}


# Procedure: file_import
proc file_import {} {
#  This procedure asks the PDM to process a SELect file
#
global pdmsrc
global UDSPATH

set types {
        {"Text files"           {.txt}      }
        {"All files"            "}
   }
regsub -all / $UDSPATH {\\} ipath
set result [tk_getOpenFile -filetypes $types -parent . -initialdir $ipath -title "Process data base commands from:" -defaultextension .txt]
```

```
if {$result != ""} {
        puts $pdmsrc "sel $result"
        flush $pdmsrc
        set line [lindex [pdm_getresponse] 0]
        while {$line != "@#eose#@"} {
                DBase_Out_add $line
                update
                set line [lindex [pdm_getresponse] 0]
            }
  }
}


# Procedure: file_new
proc file_new {} {
global filetype
global ODSPATH
global UDSPATH
global edit_cmd

set fname Untitled
switch $filetype {
        ODS     {
                  set fpath [file join $ODSPATH $fname]
                  if {![file exists $fpath]} {
                          set fout [open $fpath w]
                          close $fout
                    }
                  exec $edit_cmd $fpath &
              }
        UDS     {
                  set fpath [file join $UDSPATH $fname]
                 puts $fpath
                  if {![file exists $fpath]} {
                          set fout [open $fpath w]
                          close $fout
                    }
                  exec $edit_cmd $fpath &
                 }
   }
}


# Procedure: fillrec
proc fillrec { recno} {
# Fills the current data base record with the specified record number
#
# Returns:  1 on success
#                 0 on failure
#
global ptid_req
global ptid_org
global desc_req
global type_req
global dim_req
global form_req
global unit_req
global pred_req
global sys_req
global note_req
global del_req
global addr_req
global recno_requested

if {$recno == ""} {return 0}
set rlist [pdm_look $recno]
set numrec [lindex [lindex $rlist 0] 3]
set ptid [lindex [lindex $rlist 1] 2]
if {($numrec == $recno) || ($ptid == $recno)} {
        set recno_requested [expr $numrec + 1]
        set ptid_req [lindex [lindex $rlist 1] 2]
        set ptid_org $ptid_req
        set del_req  [lindex [lindex $rlist 1] 3]
        set desc_req [lrange [lindex $rlist 2] 3 end]
        set type_req [lindex [lindex $rlist 5] 2]
```

```
            set dim_req  [lrange [lindex $rlist 6] 2 end]
            set form_req [lindex [lindex $rlist 10] 3]
            set unit_req [lindex [lindex $rlist 3] 2]
            set pred_req [lindex [lindex $rlist 8] 2]
            set sys_req  [lindex [lindex $rlist 4] 2]
            set addr_req [lindex [lindex $rlist 9] 2]
            set pos [string first = [lindex $rlist 11]]
    incr pos 2
            set lastline [string range [lindex $rlist 11] $pos end]
            set note_req ""
    foreach line [lrange $rlist 12 end] {
                        set note_req "$note_req$lastline\n"
                        set lastline $line
                      }
            if {[string trim $lastline] != ""} {set note_req "$note_req$lastline\n"}
            .frame1.frame3.frame22.text24 delete 1.0 end
            .frame1.frame3.frame22.text24 insert end $note_req
            set rtnval 1
  } {       set rtnval 0 }
return $rtnval
}


# Procedure: findfirst
proc findfirst {} {
global find_req
global pdmsrc

set ptid [string trim $find_req]
puts $pdmsrc "findfirst $ptid"
flush $pdmsrc
set ptid [lindex [pdm_getresponse] 0 ]
return $ptid
}


# Procedure: findinpath
proc findinpath { fname} {
global env

if {[info exists env(PATH)]} {
            set pathval $env(PATH)
  }         {
             if {[info exists env(Path)]} {
                        set pathval $env(Path)
               } {set pathval [pwd]}
            }
set plist [split $pathval \;]
foreach path $plist {
            regsub -all {\\} $path / dir
            set fpath [file join $dir $fname]
            if {[file exists $fpath]} {
                        return $fpath
              }
    }
return ""
}


# Procedure: finish
proc finish {} {
global packmode
global dbmode
global pdmsrc

if {$packmode == "pack"} {
            puts $pdmsrc pack
            flush $pdmsrc
            pdm_getresponse
  }
if {$dbmode == "write"} {
            puts $pdmsrc writemode
            flush $pdmsrc
  }
pdm_quit
```

```
exit
}


# Procedure: fixed_tkMenuFind
proc fixed_tkMenuFind { w char} {
   global tkPriv
   set char [string tolower $char]

   foreach child [winfo child $w] {
            switch [winfo class $child] {
               Menubutton {
                        set char2 [string index [$child cget -text]  [$child cget -underline]]
                        if {([string compare $char [string tolower $char2]] == 0)
                                 || ($char == "")} {
                           if {[$child cget -state] != "disabled"} {
                                 return $child
                           }
                        }
               }

               default {
                        set match [tkMenuFind $child $char]
                        if {$match != ""} {
                           return $match
                        }
               }
            }
   }
   return {}
}


# Procedure: help_show
proc help_show { fname} {
global rtproe_root
global browse_cmd
global ODSPATH
global UDSPATH

set fpath [file join $rtproe_root rtproe doc $fname]
regsub -all / $browse_cmd {\\} cmd
exec $cmd $fpath &
}


# Procedure: modrec
proc modrec { action} {
# Modify the current record in the data base
# or add a new record.
#
# Returns:   1 on success
#                        0 on failure
#
global ptid_req
global ptid_org
global desc_req
global type_req
global dim_req
global form_req
global unit_req
global sys_req
global pred_req
global note_req
global del_req
global recno_requested
global pdmsrc

set ptid [string trim $ptid_req]
switch $action {
            mod  {
                        set line(1) "/mod $ptid_org"
                        if {$ptid != $ptid_org} {lappend line(1) $ptid}
                        puts $pdmsrc $line(1)
                        flush $pdmsrc
```

```
                    set resp [lindex [pdm_getresponse] 0]
                    if {[string first "not found" [lindex $resp 0]] != -1} {return $resp}
                }
            add  {
                    set line(1) "/add $ptid"
                    puts $pdmsrc $line(1)
                    flush $pdmsrc
                    set resp [lindex [pdm_getresponse] 0]
                    if {[string first "already found" $resp] != -1} {return $resp}
                }
    }
set line(2) ".desc $desc_req"
set line(3) [.frame1.frame3.frame22.text24 get 1.0 "end -1 chars"]
set line(4) ".type [string trim $type_req]"
set dim [string tolower [string trim $dim_req]]
if {$dim == "none"} {set dim ""}
set line(5) ".dim  $dim"
set form [string trim $form_req]
set line(6) ".form $form"
set unit [string trim $unit_req]
set line(7) ".unit $unit"
set sys [string trim $sys_req]
set line(8) ".sys  $sys"
set pred [string trim $pred_req]
set line(9) ".pred $pred"
for {set ctr 2} {$ctr < 10} {incr ctr} {
            puts $pdmsrc $line($ctr)
    }
flush $pdmsrc
set resp [pdm_getresponse]
fillrec $ptid
return $resp
}


# Procedure: open_file
proc open_file { fname} {
global rtproe_root
global browse_cmd
global edit_cmd
global filetype
global ODSPATH
global UDSPATH

switch $filetype {
            help        {
                        set fpath [file join $rtproe_root rtproe doc $fname]
                        regsub -all / $browse_cmd {\\} cmd
                        exec $cmd $fpath &
                        }
            ODS         {
                        set fpath [file join $ODSPATH $fname]
                        exec $edit_cmd $fpath &
                    }
            UDS         {
                        set fpath [file join $UDSPATH $fname]
                        exec $edit_cmd $fpath &
                        }
    }
}


# Procedure: options_read
proc options_read {} {
global loadname
global ODSPATH
global UDSPATH
global browse_cmd
global edit_cmd
global dbmode
global packmode

if {![catch {set fin [open rtproe.ini "r"]}] {
    while {![eof $fin]} {
            gets $fin line
```

```
            switch [lindex $line 0] {
                    loadname { set loadname   [lrange $line 1 end] }
                    ODSPATH{ set ODSPATH    [lrange $line 1 end] }
                    UDSPATH{ set UDSPATH    [lrange $line 1 end] }
                    browse_cmd              { set browse_cmd [lrange $line 1 end] }
                    edit_cmd  { set edit_cmd   [lrange $line 1 end] }
                    dbmode    { set dbmode     [lrange $line 1 end] }
                    packmode { set packmode   [lrange $line 1 end] }
                  }
            }
   close $fin
  }
}


# Procedure: options_save
proc options_save {} {
global loadname
global ODSPATH
global UDSPATH
global browse_cmd
global edit_cmd
global dbmode
global packmode

if {![catch {set fout [open rtproe.ini "w"]}] {
            puts $fout "loadname $loadname"
            puts $fout "ODSPATH $ODSPATH"
            puts $fout "UDSPATH $UDSPATH"
            puts $fout "browse_cmd $browse_cmd"
            puts $fout "edit_cmd $edit_cmd"
            puts $fout "dbmode $dbmode"
            puts $fout "packmode $packmode"
            close $fout
  }
}


# Procedure: options_tab_build
proc options_tab_build {} {
pack forget .options.frame8.frame12
pack forget .options.frame8.frame13
pack .options.frame8.frame11 -side top -after .options.frame8.canvas10
}


# Procedure: options_tab_cmds
proc options_tab_cmds {} {
pack forget .options.frame8.frame11
pack forget .options.frame8.frame13
pack .options.frame8.frame12 -side top -after .options.frame8.canvas10
}


# Procedure: options_tab_dbase
proc options_tab_dbase {} {
pack forget .options.frame8.frame11
pack forget .options.frame8.frame12
pack .options.frame8.frame13 -side top -after .options.frame8.canvas10
}


# Procedure: pdm_connect
proc pdm_connect {} {
global pdmsrc

if {$pdmsrc == ""} {
   if {[catch {open {|pdm} "r+"} pdmsrc]} {
            puts "CONNECT ERROR! $pdmsrc"
            set pdmsrc ""
    }{
                    # turn on end-of-message prompts from PDM
                    #
                    # This is VERY important and must be the first PDM command issued
                    # in order to have proper communication with PDM. The string
```

```
                    # @#eomsg#@ serves as an end of message marker from the PDM. It is
                    # used to prevent blocking on from PDM when PDM has no more information
                    # to send.
                    puts $pdmsrc "eomsg 1"
                    flush $pdmsrc
                }
  }
}


# Procedure: pdm_dump
proc pdm_dump {} {
global ptid_req
global pdmsrc

puts $pdmsrc dump
flush $pdmsrc
gets $pdmsrc line
while {$line != "@#eomsg#@"} {
        selbuf_add $line
        gets $pdmsrc line
    }
}


# Procedure: pdm_getresponse
proc pdm_getresponse {} {
global pdmsrc

set rtnval ""
if {$pdmsrc != ""} {
        gets $pdmsrc line
        while {$line != "@#eomsg#@"} {
                lappend rtnval $line
                gets $pdmsrc line
                if {$line == "@#eosel#@"} {
                        set rtnval $line
                        set line "@#eomsg#@"
                  }
            }
  }
return $rtnval
}


# Procedure: pdm_look
proc pdm_look { ptid} {
global pdmsrc

set rtnval ""
if {$pdmsrc != ""} {
        puts $pdmsrc "look $ptid"
        flush $pdmsrc
        set rtnval [pdm_getresponse]
  }
return $rtnval
}


# Procedure: pdm_quit
proc pdm_quit {} {
global pdmsrc

if {$pdmsrc != ""} {
        puts $pdmsrc exit
        close $pdmsrc
        set pdmsrc ""
  }
}


# Procedure: selbuf_add
proc selbuf_add { data} {
global select_buffer
```

```
set select_buffer "$select_buffer$data\n"
if {[winfo exists .selbuf.frame.text2]} {.selbuf.frame.text2 insert end "$data\n"}
}


# Procedure: selbuf_load
proc selbuf_load {} {
global UDSPATH

set types {
        {"Text files"       {.txt}      }
        {"All files"        *}
  }
regsub -all / $UDSPATH {\\} ipath
set result [tk_getOpenFile -filetypes $types -parent .selbuf -initialdir $ipath -title "Load select buffer from:" -defaultextension .txt]
if {$result != ""} {
        set fin [open $result r]
        while {![eof $fin]} {
                gets $fin line
                .selbuf.frame.text2 insert insert "$line\n"
            }
        close $fin
  }
}


# Procedure: selbuf_save
proc selbuf_save {} {
global UDSPATH

set types {
        {"Text files"       {.txt}      }
        {"All files"        *}
  }
regsub -all / $UDSPATH {\\} ipath
set result [tk_getSaveFile -filetypes $types -parent .selbuf -initialdir $ipath -title "Save select buffer to:"  -initialfile Untitled -defaultextension .txt]
if {$result != ""} {
        set fout [open $result w]
        set selbuf [.selbuf.frame.text2 get 1.0 end]
        puts $fout $selbuf
        close $fout
  }
}


# Procedure: sendout
proc sendout { arg} {
Build_Out_add $arg
update
}


# Procedure: tabcreate
proc tabcreate { cvname tablist cmdlist direction} {
global tab_list
global tab_canvas
global tab_cmd
global tab_num

if {[info exists tab_num]} {
        incr tab_num
  } {   set tab_num 0}

set xorigin 0
if {$direction == "down"} {
        set yorigin 0
        set yheight 14
        set anchorpt "se"
  } {
        set yorigin 14
        set yheight -14
        set anchorpt "ne"
   }

set xoffset $xorigin
```

```
set color [$cvname cget -bg]
set tab_list($tab_num) ""
set tab_canvas($tab_num) $cvname
set ctr 0
foreach tab $tablist {
        set tlen [string length $tab]
        set x1 $xoffset
        incr xoffset 4
        set y1 $yorigin
        set x2 $xoffset
        incr xoffset [expr 10 * $tlen]
        set y2 $yorigin
        incr y2 $yheight
        set x3 $xoffset
        incr xoffset 4
        set y3 $y2
        set x4 $xoffset
        incr xoffset -3
        set y4 $yorigin
        set tagno tab$tab_num.$ctr
        incr ctr
        set itemtag [$cvname create polygon $x1 $y1 $x2 $y2 $x3 $y3 $x4 $y4 -fill $color -outline $color -tag $tagno]
        lappend tab_list($tab_num) $tagno
        $cvname bind $itemtag <Button-1> "tabselect $tab_num $tagno"
        $cvname create line $x1 $y1 $x2 $y2 $x3 $y3 $x4 $y4 -fill black  -tag $tagno
        set itag [$cvname create text $x3 $y3 -text $tab -anchor $anchorpt -tag $tagno -font -adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1]
        $cvname bind $itag <Button-1> "tabselect $tab_num $tagno"
    }
set tab_cmd($tab_num) $cmdlist
return $tab_num
}


# Procedure: tabselect
proc tabselect { tabid itemtag} {
global tab_list
global tab_cmd
global tab_canvas

set color [$tab_canvas($tabid) cget -bg]
set color grey68
set ctr 0
foreach item $tab_list($tabid) {
        set tlist [$tab_canvas($tabid) find withtag $item]
        if {$item == $itemtag} {
           eval [lindex $tab_cmd($tabid) $ctr]
           set base ""
           set stack ""
           foreach inum $tlist {
                   set type [$tab_canvas($tabid) type $inum]
                   if      {$type == "polygon"} {
                           $tab_canvas($tabid) itemconfigure $inum -fill azure -outline azure
                           set base $inum
                           } {lappend stack $inum}
                  }
                $tab_canvas($tabid) raise $base
                foreach object $stack {$tab_canvas($tabid) raise $object $base}
          } {
                foreach inum $tlist {
                        set type [$tab_canvas($tabid) type $inum]
                        if {$type == "polygon"} {
                           $tab_canvas($tabid) itemconfigure $inum -fill $color -outline black
                           }
                  }
           }
        incr ctr
    }
}


# Procedure: undelrec
proc undelrec {} {
global ptid_req
global pdmsrc
```

```
set ptid [string trim $ptid_req]
if {[catch {expr $ptid - 1} record]} {set record $ptid}
if {$ptid != ""} {
                puts $pdmsrc "undel $record"
                flush $pdmsrc
                set response [pdm_getresponse]
                fillrec $record
  }
}


# User defined images

image create photo tclpowered \
 -file {pwrdlogo100.gif} \
 -gamma {1.0} \
 -height {0} \
 -width {0}


# Internal procedures

# application parsing procedure
proc XFLocalParseAppDefs {xfAppDefFile} {
 global xfAppDefaults tcl_platform

 # basically from: Michael Moore
 if {[file exists $xfAppDefFile] &&
    [file readable $xfAppDefFile] &&
    "[file type $xfAppDefFile]" == "link"} {
  catch "file type $xfAppDefFile" xfType
  while {"$xfType" == "link"} {
   if {[catch "file readlink $xfAppDefFile" xfAppDefFile]} {
    return
   }
   catch "file type $xfAppDefFile" xfType
  }
 }
 if {!("$xfAppDefFile" != "" &&
     [file exists $xfAppDefFile] &&
     [file readable $xfAppDefFile] &&
     "[file type $xfAppDefFile]" == "file")} {
  return
 }
 if {![catch "open $xfAppDefFile r" xfResult]} {
  set xfAppFileContents [read $xfResult]
  close $xfResult
  foreach line [split $xfAppFileContents "\n"] {
   # backup indicates how far to backup.  It applies to the
   # situation where a resource name ends in . and when it
   # ends in *.  In the second case you want to keep the *
   # in the widget name for pattern matching, but you want
   # to get rid of the . if it is the end of the name.
   set backup -2
   set line [string trim $line]
   if {[string index $line 0] == "#" || "$line" == ""} {
    # skip comments and empty lines
    continue
   }
   if {![string compare "windows" $tcl_platform(platform)]} {
    set list [split $line ";"]
   } {
    set list [split $line ":"]
   }
   set resource [string trim [lindex $list 0]]
   set i [string last "." $resource]
   set j [string last "*" $resource]
   if {$j > $i} {
    set i $j
    set backup -1
   }
   incr i
   set name [string range $resource $i end]
   incr i $backup
   set widname [string range $resource 0 $i]
```

```
    set value [string trim [lindex $list 1]]
    if {"$widname" != "" && "$widname" != "*"} {
      # insert the widget and resourcename to the application
      # defaults list.
      if {![info exists xfAppDefaults]} {
        set xfAppDefaults ""
      }
      lappend xfAppDefaults [list $widname [string tolower $name] $value]
    }
   }
  }
}

# application loading procedure
proc XFLocalLoadAppDefs {{xfClasses "*"} {xfPriority "startupFile"} {xfAppDefFile ""}} {
  global env tcl_platform

  if {![string compare "windows" $tcl_platform(platform)]} {
    set separator ";"
  } {
    set separator ":"
  }
  if {"$xfAppDefFile" == ""} {
    set xfFileList ""
    if {[info exists env(XUSERFILESEARCHPATH)]} {
      eval lappend xfFileList [split $env(XUSERFILESEARCHPATH) $separator]
    }
    if {[info exists env(XAPPLRESDIR)]} {
      eval lappend xfFileList [split $env(XAPPLRESDIR) $separator]
    }
    if {[info exists env(XFILESEARCHPATH)]} {
      eval lappend xfFileList [split $env(XFILESEARCHPATH) $separator]
    }
    append xfFileList " /usr/lib/X11/app-defaults"
    append xfFileList " /usr/X11/lib/X11/app-defaults"

    foreach xfCounter1 $xfClasses {
      foreach xfCounter2 $xfFileList {
        set xfPathName $xfCounter2
        if {[regsub -all "%N" "$xfPathName" "$xfCounter1" xfResult]} {
          set xfPathName $xfResult
        }
        if {[regsub -all "%T" "$xfPathName" "app-defaults" xfResult]} {
          set xfPathName $xfResult
        }
        if {[regsub -all "%S" "$xfPathName" "" xfResult]} {
          set xfPathName $xfResult
        }
        if {[regsub -all "%C" "$xfPathName" "" xfResult]} {
          set xfPathName $xfResult
        }
        if {[file exists $xfPathName] &&
            [file readable $xfPathName] &&
            ("[file type $xfPathName]" == "file" ||
             "[file type $xfPathName]" == "link")} {
          catch "option readfile $xfPathName $xfPriority"
          if {"[info commands XFParseAppDefs]" != ""} {
            XFParseAppDefs $xfPathName
          } {
            if {"[info commands XFLocalParseAppDefs]" != ""} {
              XFLocalParseAppDefs $xfPathName
            }
          }
        } {
          if {[file exists $xfCounter2/$xfCounter1] &&
              [file readable $xfCounter2/$xfCounter1] &&
              ("[file type $xfCounter2/$xfCounter1]" == "file" ||
               "[file type $xfCounter2/$xfCounter1]" == "link")} {
            catch "option readfile $xfCounter2/$xfCounter1 $xfPriority"
            if {"[info commands XFParseAppDefs]" != ""} {
              XFParseAppDefs $xfCounter2/$xfCounter1
            } {
              if {"[info commands XFLocalParseAppDefs]" != ""} {
                XFLocalParseAppDefs $xfCounter2/$xfCounter1
              }
```

```
        }
       }
      }
     }
    }
   }{
    # load a specific application defaults file
    if {[file exists $xfAppDefFile] &&
       [file readable $xfAppDefFile] &&
       ("[file type $xfAppDefFile]" == "file" ||
        "[file type $xfAppDefFile]" == "link")} {
      catch "option readfile $xfAppDefFile $xfPriority"
      if {"[info commands XFParseAppDefs]" != ""} {
       XFParseAppDefs $xfAppDefFile
      } {
       if {"[info commands XFLocalParseAppDefs]" != ""} {
        XFLocalParseAppDefs $xfAppDefFile
       }
      }
    }
   }
}

# application setting procedure
proc XFLocalSetAppDefs {{xfWidgetPath "."}} {
  global xfAppDefaults

  if {![info exists xfAppDefaults]} {
    return
  }
  foreach xfCounter $xfAppDefaults {
    if {"$xfCounter" == ""} {
      break
    }
    set widname [lindex $xfCounter 0]
    if {[string match $widname ${xfWidgetPath}] ||
       [string match "${xfWidgetPath}*" $widname]} {
      set name [string tolower [lindex $xfCounter 1]]
      set value [lindex $xfCounter 2]
      # Now lets see how many tcl commands match the name
      # pattern specified.
      set widlist [info command $widname]
      if {"$widlist" != ""} {
        foreach widget $widlist {
          # make sure this command is a widget.
          if {![catch "winfo id $widget"] &&
             [string match "${xfWidgetPath}*" $widget]} {
           catch "$widget configure -$name $value"
          }
        }
      }
    }
  }
}

# initialize bindings for all widgets
proc XFInitAllBindings {} {
  # bindings
  bind all <Alt-Key> {
    tkTraverseToMenu %W %A
}
  bind all <Key-F10> {
    tkFirstMenu %W
}
  bind all <Key-Tab> {focus [tk_focusNext %W]}
  bind all <Shift-Key-Tab> {focus [tk_focusPrev %W]}
}


# end source
proc EndSrc {} {
global pdmsrc
global lastgoodnum
global buildsrc_loaded
global build_msg
```

```
global dbase_msg
global QDSPATH
global UDSPATH
global env
global find_req
global tk_version

wm protocol . WM_DELETE_WINDOW {finish}

# set up default ODSPATH and UDSPATH
if {[info exists env(ODSPATH)]} {
        set ODSPATH $env(ODSPATH)
  } { set ODSPATH [pwd] }
set UDSPATH [pwd]

# look for initialization file
if {[file exists rtproe.ini]} {
        options_read
  }

# connect to data base manager
set lastgoodnum 1
set pdmsrc ""
pdm_connect
fillrec 0
set find_req ""

# set initial build flag
set buildsrc_loaded 0

# initialize output window
set build_msg ""
set dbase_msg ""
tabselect 0 tab0.1

# fix a bug in the TCL libraries
if {$tk_version <= "4.2"} {
        eval "proc tkMenuFind {w char} {\n[info body fixed_tkMenuFind]\n}"
  }
}

# prepare auto loading
global auto_path
global tk_library
global xfLoadPath
foreach xfElement [eval list [split $xfLoadPath @] $auto_path] {
  if {[file exists $xfElement/tclIndex]} {
    lappend auto_path $xfElement
  }
}

# initialize global variables
proc InitGlobals {} {
  global {APPEND}
  set {APPEND} {C:/WINNT/system32/cmd.exe /c type}
  global {BFLAGS}
  set {BFLAGS} {/nologo }
  global {BUILD}
  set {BUILD} {fi32}
  global {CC}
  set {CC} {cl}
  global {CFLAGS}
  set {CFLAGS} {/c /nologo}
  global {CSCANNER}
  set {CSCANNER} {hgen}
  global {EXEOUT}
  set {EXEOUT} {/Fe}
  global {EXTEXE}
  set {EXTEXE} {.exe}
  global {EXTLIB}
  set {EXTLIB} {.lib}
  global {EXTOBJ}
  set {EXTOBJ} {.obj}
  global {F77}
  set {F77} {fi32}
```

```
global {FFLAGS}
set {FFLAGS} {/c /4Yd /WX /nologo}
global {FSCANNER}
set {FSCANNER} {igen}
global {LIBCON}
set {LIBCON} {s:/odstest/control}
global {LIBOPT}
set {LIBOPT} {/link}
global {LIBRARIES}
set {LIBRARIES} {winmm.lib kernel32.lib user32.lib gdi32.lib  winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib  odbc32.lib odbccp32.lib
/subsystem:console /incremental:no /def:s:/odstest/rtsched.def}
global {LIBSEG}
set {LIBSEG} {s:/odstest/segment}
global {LIBSUB}
set {LIBSUB} {s:/odstest/subs}
global {LINKNAME}
set {LINKNAME} {rtsched.exe}
global {OBJOUT}
set {OBJOUT} {/Fo}
global {ODSPATH}
set {ODSPATH} {s:/odstest}
global {ODSPATH_req}
set {ODSPATH_req} {s:/odstest}
global {SHELL}
set {SHELL} {C:/WINNT/system32/cmd.exe /c}
global {UDSPATH}
set {UDSPATH} {s:/demo}
global {UDSPATH_req}
set {UDSPATH_req} {s:/demo}
global {ULIBCON}
set {ULIBCON} {s:/demo/control}
global {ULIBSEG}
set {ULIBSEG} {s:/demo/segment}
global {ULIBSUB}
set {ULIBSUB} {s:/demo/subs}
global {addr_req}
set {addr_req} {0x8}
global {answer}
set {answer} {cancel}
global {browse_cmd}
set {browse_cmd} {C:/Program Files/Netscape/Navigator/Program/NETSCAPE.EXE}
global {browse_cmd_req}
set {browse_cmd_req} {C:/Program Files/Netscape/Navigator/Program/NETSCAPE.EXE}
global {build_msg}
set {build_msg} {}
global {buildsrc_loaded}
set {buildsrc_loaded} {0}
global {dbase_msg}
set {dbase_msg} {}
global {dbmode}
set {dbmode} {read}
global {dbmode_req}
set {dbmode_req} {read}
global {del_req}
set {del_req} {}
global {desc_req}
set {desc_req} {CPU usage in percent}
global {dim_req}
set {dim_req} {None}
global {edit_cmd}
set {edit_cmd} {write.exe}
global {edit_cmd_req}
set {edit_cmd_req} {write.exe}
global {filesbuilt}
set {filesbuilt} {1}
global {filetype}
set {filetype} {UDS}
global {find_req}
set {find_req} {}
global {flist}
set {flist} {}
global {fname}
set {fname} {Untitled.txt}
global {form_req}
set {form_req} {}
```

```
 global {fout}
 set {fout} {file5}
 global {fpath}
 set {fpath} {c:/sim/demo/Untitled.txt}
 global {haderror}
 set {haderror} {0}
 global {helplist}
 set {helplist} {rtpro_copy.htm rtpro_faq.htm rtpro_files.htm rtpro_frames.htm rtpro_goals.htm rtpro_graph.htm rtpro_hgen.htm rtpro_home.htm rtpro_ics.htm rtpro_igen.htm
rtpro_make.htm rtpro_memm.htm rtpro_pdm.htm rtpro_rtdata.htm rtpro_rtsched.htm rtpro_sum.htm}
 global {index}
 set {index} {5}
 global {lastgoodnum}
 set {lastgoodnum} {1}
 global {loadname}
 set {loadname} {rtsched}
 global {loadname_req}
 set {loadname_req} {rtsched}
 global {note_req}
 set {note_req} {
}
 global {output_type}
 set {output_type} {dbase}
 global {outwin}
 set {outwin} {show}
 global {packmode}
 set {packmode} {nopack}
 global {packmode_req}
 set {packmode_req} {nopack}
 global {pdmsrc}
 set {pdmsrc} {file92}
 global {pred_req}
 set {pred_req} {global30}
 global {ptid_org}
 set {ptid_org} {avg_cpu_usage}
 global {ptid_req}
 set {ptid_req} {avg_cpu_usage}
 global {recno_requested}
 set {recno_requested} {1}
 global {redir}
 set {redir} {stdout}
 global {rtproe_root}
 set {rtproe_root} {s:/}
 global {selbuflast}
 set {selbuflast} {
}
 global {select_buffer}
 set {select_buffer} {
}
 global {selwin}
 set {selwin} {hide}
 global {sys_req}
 set {sys_req} {}
 global {tab_canvas}
 set {tab_canvas(0)} {.frame2.frame.canvas0}
 set {tab_canvas(1)} {.frame1.frame0.canvas0}
 set {tab_canvas(2)} {.frame1.frame0.canvas0}
 set {tab_canvas(4)} {.options.frame8.canvas10}
 global {tab_cmd}
 set {tab_cmd(0)} {Build_Out DBase_Out}
 set {tab_cmd(1)} {UDS_files ODS_files Help_files}
 set {tab_cmd(2)} {UDS_files ODS_files Help_files}
 set {tab_cmd(4)} {options_tab_build options_tab_cmds options_tab_dbase}
 global {tab_list}
 set {tab_list(0)} {tab0.0 tab0.1}
 set {tab_list(1)} {tab1.0 tab1.1 tab1.2}
 set {tab_list(2)} {tab2.0 tab2.1 tab2.2}
 set {tab_list(4)} {tab4.0 tab4.1 tab4.2}
 global {tab_num}
 set {tab_num} {4}
 global {type_req}
 set {type_req} {float}
 global {unit_req}
 set {unit_req} {}
 global {verbose}
 set {verbose} {0}
```

```
  # please don't modify the following
  # variables. They are needed by xf.
  global {autoLoadList}
  set {autoLoadList(main.tcl)} {0}
  set {autoLoadList(sde.tcl)} {0}
  global {internalAliasList}
  set {internalAliasList} {}
  global {moduleList}
  set {moduleList(sde.tcl)} {}
  global {preloadList}
  set {preloadList(xfInternal)} {}
  global {symbolicName}
  set {symbolicName(root)} {.}
  global {xfWmSetPosition}
  set {xfWmSetPosition} {.about .options .selbuf}
  global {xfWmSetSize}
  set {xfWmSetSize} {}
  global {xfAppDefToplevels}
  set {xfAppDefToplevels} {}
}

# initialize global variables
InitGlobals

# display/remove toplevel windows.
ShowWindow.

global xfShowWindow.about
set xfShowWindow.about 0

global xfShowWindow.options
set xfShowWindow.options 0

global xfShowWindow.selbuf
set xfShowWindow.selbuf 0

# load default bindings.
if {[info exists env(XF_BIND_FILE)] &&
    "[info procs XFShowHelp]" == ""} {
  source $env(XF_BIND_FILE)
}

# initialize bindings for all widgets.
XFInitAllBindings


# parse and apply application defaults.
XFLocalLoadAppDefs Sde
XFLocalSetAppDefs

# end source
EndSrc

# eof
#

;-------------------------------------------------------------
;  rtsched - Real Time Scheduler for Windows
;-------------------------------------------------------------
; ------------------------------------------------------------
; Copyright (C) 1996 Dennis R. LaBelle (drlabell@albany.net) All Rights Reserved.
;
; PERMISSION IS GRANTED TO DISTRIBUTE THIS SOFTWARE FREELY, WITH THE
; EXCEPTION THAT ONE MAY NOT CHARGE FOR IT OR INCLUDE IT WITH SOFTWARE
; WHICH IS SOLD. Permission to use, copy, modify, and distribute this software and
; its documentation for educational, research, internal corporate and non-profit
; purposes, without fee, and without a written agreement is hereby granted for all
; cases that do not conflict with the restriction in the first sentence of this
; paragraph, provided that the above copyright notice, this paragraph, and the
; following three paragraphs appear in all copies.
;
; Permission to incorporate this software into commercial products may be obtained
; from the author (e-mail drlabell@albany.net).
;
```