Analgesia Database:
Error message Library for PalmOS PDA

Version 0.90

J.M. van Schalkwyk

February 27, 2009

Contents

1 Error library introduction

This library shouldn't really exist. All it really used to do was provide error messages, which could as easily have been done by looking up records in a database. My initial justification for its existence was that I was experimenting with libraries, and thought that this trivial one would be an ideal starting point. In consequence, the whole library was excessively boring, simply providing an easy introduction to creating and using libraries!

In order to help us with debugging of that perennial C problem — memory leaks — we've modified things a little. We moved the ErrWrite routine out of the main C++ program into the Error library as ErrorWrite.

[What ErrorWrite now does is to write to a previously established pointer to the CYCERR PalmOs 'file', as it did in the main program previously]

1.1 Library codes

The following library codes are registered against my name with PalmOS:

- JovS : used for main program
- ErLi : used for this error library
- NuLi : used for numeric library
- CnLi : used for console library
- ScLi : used for scripting library
- 4sql : used for sql library

2 Error: the C program

Although it is possible to write error libraries in C++, we've chosen the simpler (and better documented) option of just using ordinary old C, with all of its limitations. As the GNU C Compiler supports use of the C++ style comment (two slashes on a line), we use it within our C code, as usage of /* comments */ is excessively tedious.

We include two pre-defined PalmOS headers (SystemMgr and LibTraps), necessary for creating our libraries, and then include our own important header file, discussed in a later section (5). We also need to reference the console header, for later writing of error messages to our console program.

```
#include <SystemMgr.h>
#include <PalmOS.h>
// PalmOS.h only required for StrLen!
#define ERRDEBUG_TRAP(trapno)
#include <LibTraps.h>
#include "ERRDEBUG.h"
#include "../console/CONSOLE.h"
#include "../palmsql3A.h"

#define ErrorBufferName         "CYCERROR"
#define CYCSIZE                 32752
```

CYCSIZE is a clumsy constant, as is ErrorBufferName [export me]! We include palmsql3A.h to tell us about constants such as DBCREATOR.

The following three functions are necessary for the normal operation of the library. Most important is the `start` function, called when the library is initialised. It's pretty standard from library to library (as we shall see on creating more complex libraries) so just accept the code for now.

```
Err start (UInt16 refnum, SysLibTblEntryPtr entryP) {
  extern void *jmptable ();
  entryP->dispatchTblP = (void *) jmptable;
  entryP->globalsP = NULL;
  return 0;
  }
```

The opening and closing functions were previously simple stubs, but after some initial experimentation with Sql3lib, we now set up a console handle so we can write error messages to our console program. Our fourth function gives us a 'nothing' function which we can use as a further stub! (See section 7). First, the type definition ...

```
typedef struct
 { UInt16    CONSOLE;
   UInt16    refcount;
   DmOpenRef ERRBUF;
   MemHandle ERRHANDLE;
   Char *    ERRPTR;
 } ErrLib_globals;
```

Next, some preliminary wrapped functions:

```
Int16 e_DmWrite (void *recordP, UInt32 offset, Char *srcP,
                      UInt32 bytes)
{ return (! DmWrite (recordP, offset, srcP, bytes));
}
```

```
LocalID e_DmFindDatabase (const Char *nameP)
{ return DmFindDatabase (0, nameP);
}

DmOpenRef e_DmOpenDatabase (LocalID dbID, UInt16 mode)
{
  return DmOpenDatabase (0, dbID, mode);
}

MemHandle e_DmGetRecord (DmOpenRef dbP, UInt16 index)
{
  return DmGetRecord (dbP, index);
}

Int16 e_DmCreateDatabase (const Char *nameP)
{ return (! DmCreateDatabase (0, nameP, DBCREATOR, DBTYPE, false));
}

MemHandle e_DmNewRecord (DmOpenRef dbP, UInt16 *atP, UInt32 size)
{
  return DmNewRecord (dbP, atP, size);
}
```

A subsidiary routine:

```
Int16 FindRecordZero (ErrLib_globals *gl, LocalID mydbid,
                      Int16 oldrec)
{ Int16 i;
  UInt16 idx=0;
  UInt16* pIdx; //point to index
  pIdx = &idx;

  if (! mydbid)
     { return -2;
     };
  // assumes gl is valid [?]
  gl->ERRBUF = e_DmOpenDatabase (mydbid, dmModeReadWrite);
  if (! gl->ERRBUF) { return -3; };

  if (oldrec) // unless force new record
     { gl->ERRHANDLE = e_DmGetRecord (gl->ERRBUF, 0);
       if (gl->ERRHANDLE)
          {
            gl->ERRPTR = (Char *) MemHandleLock(gl->ERRHANDLE);
            return 0; //ok!
     };   }; // else, make the record:

  gl->ERRHANDLE = e_DmNewRecord(gl->ERRBUF, pIdx, CYCSIZE);
```

```
  if (! gl->ERRHANDLE)
     { return -4; // failed to make rec 0
     };

  gl->ERRPTR = (Char *) MemHandleLock(gl->ERRHANDLE);
  i = CYCSIZE/8;
  while (i)
    { i --;
      e_DmWrite(gl->ERRPTR, i*8, "????????", 8);
    };
  idx = 2; // use for different purpose [ugh]
  if (! e_DmWrite(gl->ERRPTR, 2, (Char *) &idx, 2))
     { return -5; // write failed
     };
  idx = 0x8001;
  e_DmWrite(gl->ERRPTR, 0, (Char *) &idx, 2);
  return 0; // ok
}
```

Now the actual opening routine.

```
Err ERRDEBUGOpen (UInt16 refnum)
{ LocalID mydbid;
  Int16 oldrec = 1; // default: old rec 0 exists..

  SysLibTblEntryPtr entryP = SysLibTblEntry (refnum);
  ErrLib_globals *gl = entryP->globalsP;
  if (gl)
     { gl->refcount ++;
       return 0; // ok ??hmm [actually, should fail!]
     };

  gl = entryP->globalsP = MemPtrNew (sizeof (ErrLib_globals));
  MemPtrSetOwner (gl, 0); ///
  gl->refcount = 1;  // for this one!
  gl->CONSOLE = 0;
  gl->ERRBUF = 0;
  gl->ERRHANDLE = 0;

  mydbid = e_DmFindDatabase(ErrorBufferName); // "CYCERROR"
  if (! mydbid)
     { if (! e_DmCreateDatabase(ErrorBufferName))
          { return -1;
          };
       oldrec = 0;
       mydbid = e_DmFindDatabase(ErrorBufferName);
     };
```

```
  return (FindRecordZero(gl, mydbid, oldrec));
}
```

In the previous incarnation of the error buffer initialisation, we also invoked ErrWrite to write 0xfffE to the buffer, signalling the 'start point'. We do this no longer.[1]

Let's look at the corresponding closure routine, but first some subsidiary wraps:

```
Int16 e_MemHandleUnlock (MemHandle h)
{ return (! MemHandleUnlock (h) );
}

Int16 e_DmReleaseRecord (DmOpenRef dbP, UInt16 index,
                                Boolean dirty)
{ return (! DmReleaseRecord (dbP, index, dirty));
}

Int16 e_DmCloseDatabase (DmOpenRef dbP)
{ return (! DmCloseDatabase (dbP));
}
```

The actual closure routine:

```
Err ERRDEBUGClose (UInt16 refnum, UInt16 *numappsP)
{
  SysLibTblEntryPtr entryP = SysLibTblEntry (refnum);
  ErrLib_globals *gl = entryP->globalsP;
  Int16 err = 0;

  if (!gl)
     { return 8;  // bit 3 : (global not found)
     };

  *numappsP = --gl->refcount; // byref return count!
  if (*numappsP)  // if non-zero [bad, can be -ve]
     { return 16; // bit 4 : err lib left open ??
     };

  if (! e_MemHandleUnlock(gl->ERRHANDLE))
     { err |= 32;  // bit 5
     };
  if (! e_DmReleaseRecord(gl->ERRBUF, 0, true))
     { err |= 64;  // bit 6
     };
  if (! e_DmCloseDatabase(gl->ERRBUF))
```

[1] In fact, consider flushing everything back to the start on re-entering, and even write 0x3F3F to top on exit?

```
     { err |= 128; // bit 7
     };
  MemChunkFree (entryP->globalsP);
  entryP->globalsP = NULL;
  return err;
}
```

Okay, okay we should wrap MemChunkFree as well. [Fix me, and other similar?]

```
Err nothing (UInt16 refnum)
{
  return 0;
}
```

Next, ErrPassConsole accepts and stores a handle on our console library, for printing of error messages . . .

```
Int16 ErrPassConsole (UInt16 refnum, UInt16 cons)
{
  SysLibTblEntryPtr entryP = SysLibTblEntry (refnum);
  ErrLib_globals *gl = entryP->globalsP;
  if (!gl)
     {  return 1; // fail
     };
  if (gl->CONSOLE)
     { return 1; // fail if already set
     };
  gl->CONSOLE = cons; // set console
  return 0; // success
}
```

Here's how we write to the console (We do this whenever we encounter an error)!

```
void XWriteConsoleText(UInt16 refnum, Char * txt, Int16 txlen)
{
  UInt16 CONSOLE;
  SysLibTblEntryPtr entryP;
  ErrLib_globals *gl;

  entryP = SysLibTblEntry (refnum);
  gl = entryP->globalsP;
  if (! gl)
     { return;
     };
```

```
  CONSOLE = gl->CONSOLE;
  if (! CONSOLE)
     { return;  // fail
     };
   ConWrite(CONSOLE, txt, txlen);
}
```

3 Cyclical error write

The write routine. We here must acquire the stored ERRPTR value from the globals.

```
void ErrorWrite(UInt16 refnum, Int16 e)
{ Int16 howfar;
  SysLibTblEntryPtr entryP;
  ErrLib_globals *gl;
  Char * ERRPTR;

  entryP = SysLibTblEntry (refnum);
  gl = entryP->globalsP;
  if (! gl)
     { return; // fail
     };
  ERRPTR = gl->ERRPTR;
  if (! ERRPTR)
     { return; // fail
     };

  howfar = * ((Int16 *) (ERRPTR+2));
  if (howfar >= CYCSIZE)
     { howfar = 4;        // wrap
     };
  e_DmWrite(ERRPTR, howfar, (Char *) &e, 2);
  howfar += 2;
  e_DmWrite(ERRPTR, 2, (Char *) &howfar, 2); // new offset
}
```

3.0.1 A minor function

Our old friend, string length:

```
Int16 e_StrLen (Char * txt)
{
  if (! txt) { return 0; };
```

```
  return StrLen(txt);
};
```

4 ErrorString

Finally, we include our single, horrendous error function called ErrorString. This cumbersome function places a text string into the buffer provided, given a corresponding error code. And that's more or less it (apart from a write to CONSOLE, if present)! The error messages themselves are far from friendly, and would benefit from revision.

See how ErrorString contains an enormous list of constant strings (which might profitably be put almost anywhere else, for example, in a database). The only merit of such an approach is the single error library file. You will most certainly want to jump over this long list of boring strings to the next section.

It is permissible to supply a NULL value in dest in the following, provided the length lgth is also zero. Under these circumstances the error string should still be written to the console!

```
Int16 ErrorString (UInt16 refnum, Char * dest, Int16 lgth,
                   Int16 errnum)
{
  Char * sTestingError = "*Testing Error*";
  Char * sNullMemoryRequest = "No zero length block of memory";
  Char * sNoListMemory = "No memory available for Linked list node";
  Char * sFailInsertList = "List node insertion failed";
  Char * sListCreationFailed = "Failed to create list. Memory?";
  Char * sNoScratchBuffer = "No SCRATCH1 buffer found. Bad";
  Char * sPalmDbNotFound = "Failed to find Palm Database";
  Char * sPalmDbNotOpen = "Could not open palm DB";
  Char * sPalmLinkDbNotOpen = "Can't open DB at link creation";
  Char * sNotClosePalmDb = "Failed to close Palm database!";
  Char * sBadDbSizeCount = "Database Record count failed";
  Char * sNulInt16Write = "Tried to write Int16 to null ptr";
  Char * sNulInt32Write = "Tried to write Int32 to null ptr";
  Char * sScratchOverflow = "Overflow during write to scratch";
  Char * sScratchOverflow2 = "Can't write -ve length to scratch?";
  Char * sCannotCreateScratch= "Cannot make SCRATCH1 buffer";
  Char * sTextComparisonMode = "Bad text comparison mode";
  Char * sFloatComparison = "Should not compare floats as =";
  Char * sFloatComparison2 = "Bad float comparison mode";
  Char * sQueryNoIFdelimiter = "No 1F in i.f. query list!";
  Char * sQueryNodeNoMake = "Failed to make query node!";
  Char * sIntComparisonMode = "Bad integer comparison mode";
  Char * sSQLTranslation1 = "SQL txlation to i.f. failed";
```

```
Char * sSQLNoWhere = "SQL txlation error no WHERE clause found";
Char * sSQLBadWhere = "SQL Defective WHERE statement";
Char * sSQLselect = "SQL Bad SELECT component";
Char * sSQLleftSpaceCondition = "SQL No L space before condition";
Char * sSQLBadCondition = "SQL parse error bad condition";
Char * sUnknownSQLTest = "Unknown SQL condition";
Char * sSQLComparator = "Unknown SQL comparator";
Char * sSQLComparator2 = "Unknown SQL comparator(2)";
Char * sSQLComparator3 = "Unknown SQL comparator(3)";
Char * sSQLComparatorStop = "Premature end SQL comparison";
Char * sSQLNoCfSpace = "No space for SQL intermediate string";
Char * sKillLongUpQueryNode = "Up Node query branch on deletion";
  /* this may not always be an error */
Char * sDefectNextPtr = "Bad join (up) ptr. Worrying!";
Char * sQueryTooShort = "SQL Query phrase too short";
Char * sQueryNoSpace = "Space missing from SQL query";
Char * sSQLnoColumn ="SQL column not in query clause";
Char * sSQLBadDatum ="SQL datum bad format or value";
Char * sBadSQLtype = "SQL bad datum type?";
Char * sDatumXlateFailed = "SQL datum translation failed";
Char * sSQLNoQuote="SQL datum has no terminal quote";
Char * sSQLNoCol="SQL join column not found on right";
Char * sSQLlogic="Bad SQL logic";
Char * sNoMemory="Failed to allocate memory. Full?";
Char * sNoMemLock="Failed to lock memory!";
Char * sMemoryNoUnlock="Failed to unlock memory";
Char * sMemoryNoFree="Failed to free memory";
Char * sBadControlStyle="Unknown style: new control";
Char * sPalmDbCantClose="Failed to close Palm database!";
Char * sFailNewRecord="Failed to create new record";
Char * sFailWriteRecord="Failed to write record";
Char * sFailUnlockRecord="Failed to unlock record";
Char * sFailReleaseRecord="Failed to release record";
Char * sLockNullHandle="Cannot lock null handle";
Char * sUnlockNullHandle="Cannot unlock null handle";
Char * sFailUnlockNulHandle="Failed to unlock null handle";
Char * sNoFreeNulPtr="Cannot free a null ptr";
Char * sFailFreeMemPtr="Failed to free memory ptr";
Char * sCannotUnlockNulPtr="Cannot unlock null mem ptr";
Char * sFailUnlockMemPtr="Failed to unlock memory ptr";
Char * sCantReleaseNulRec="Cannot release null ptr to record";
Char * sFailReleaseRec="Failed to release record";
Char * sCloseNulDb="Cannot close null database reference";
Char * sFailCloseDb="Failed to close database";
Char * sNulDbName="Null database name seems silly";
Char * sFailMakeDb="Failed to create palm database";
Char * sDelNulDb="Can't delete: null database reference";
Char * sDelDbFail="Failed to delete database";
```

```
Char * sWriteNulRec="Cannot write to null record";
Char * sWriteNulSrc="Cannot write record from null source";
Char * sFailRecCheck="Check on record write failed";
Char * sFailRecWrite="Failed to write record";
Char * sSortNulDb="Attempt to sort on null database";
Char * sSortNoRec="No record provided for record sort";
Char * sSortNoFx="No comparator callback fx: record sort";
Char * sNotRemoveNulRec="Null ptr at record removal";
Char * sFailRecRemove="Failed to remove record";
Char * sNulCopyDestin="Cannot copy to null destination";
Char * sNulCopySrc="Cannot copy from null source";
Char * sNoScratch="No scratch buffer found";
Char * sFailScratchUnlock="Failed to unlock scratch buffer";
Char * sFailScratchFree="Failed to free scratch buffer";
Char * sFailScratchClose="Failed to close scratch buffer";
Char * sDelNulPtr="Tried to delete null ptr?";
Char * sBadAscIntLen="Bad length ASC string Int32 convert";
Char * sLongAscInt="ASC string too long: Int32 convert";
Char * sBadAscIntString="Bad character(s) in ASC Int32 string";
Char * sSizeInt32="We only allow Int32s 0--999999999";
Char * sFloatTooLong="Float too long for our buffer";
Char * sDateLen="Bad date length. Not 10. YYYY-MM-DD format";
Char * sDateSeparator="Date sep (-). First separator bad";
Char * sDateSeparator2="Date sep (-). Second separator bad";
Char * sImbalancedPars="Imbalanced ()s logical expression";
Char * sBadConditional="Bad conditional in SQL";
Char * sBadConditional2="Bad conditional in SQL (2)";
Char * sLogicOverflow="SQL logic too complex. Wow!";
Char * sWhileOverflow="Overflow in WHILE statement";
Char * sFewParentheses="SQL ((too few right parentheses";
Char * sQuoteImbalance="Imbalanced 'quotes";
Char * sBadLogic="Bad SQL logic";
Char * sStoreCondition="Storage of SQL condition failed";
Char * sSelCleanFail="Failed cleanup after SELECT statement";
Char * sDbCleanFail="Failed cleanup databases after SELECT";
Char * sNoCol="Result column not in SQL query. Hmm";
Char * sBadTable="No/bad ref to table. ? tablename.columname!";
Char * sNoDbTbl="Database table not found";
Char * sBadDbMatch="Bad database match";
Char * sBadColumnName="Bad col name offset in (?bad) PDB file";
Char * sColNotFound="Table col not found (generic,SQL)";
Char * sNoQueryOp="No query op: intermediate format string";
Char * sBadNumericType="Bad numeric type";
Char * sBadType="Bad datum type";
Char * sScratchWrite="Scratch write failed";
Char * sScratchTerminal="FFFF stopper failed: scratch buffer";
Char * sInScratch="Internal error in scratch buffer";
Char * sScratchDest="NO copy from scratch: destin is too small";
```

```
Char * sBadScratchDatum="Bad datum type: scratch buffer";
Char * sPullScratch="Internal err remove raw data: scratch";
Char * sScratchDest2="NO copy from scratch: destin too small";
Char * sSizeInt32A="We only allow Int32s= 0--999999999 [A]";
Char * sBadTblCREATE="Bad CREATE TABLE";
Char * sCreateTermRpar="CREATE TABLE: no end right parenthesis";
Char * sCREATENoMem="Memory runout in CREATE TABLE statement";
Char * sCREATEfailed="CREATE TABLE statement failed";
Char * sBadTblNameCreate="Bad table name: CREATE TABLE";
Char * sCreateTableExists="DB table already exists. Cannot CREATE";
Char * sBadColNameCreate="Bad col name in table creation";
Char * sCreateBadColType="Bad col type in CREATE TABLE";
Char * sCreateBadCol="Bad column spec in CREATE TABLE";
Char * sCreateNoComma="Absent comma in CREATE TABLE";
Char * sCreateNoPrecision="Precision no spec in eg NUMERIC type";
Char * sCreateTooManyCols="Too many cols in CREATE stmt. 63 max";
Char * sCreateFailed="CREATE TABLE statement failed";
Char * sCreateOpenFail="Not open CREATEd table!";
Char * sFailMakeCreateHdr="Not create header: created DB";
Char * sFailWriteCreateHdr="Not write header: CREATEd table";
Char * sCreateNotCloseHdr="Not close header: created DB";
Char * sInsertNoTblname="No table name: INSERT statement";
Char * sNoInsertTbl="Insertion table not found";
Char * sInsertNoHdrAccess="No hdr access: insertion table";
Char * sInsColsNoRpar="No right par: insertion columns";
Char * sInsNoVALUES="Not found ')VALUES('";
Char * sInsRemainingColData="Extra data in INSERT";
Char * sInsBadKey="Bad key in INSERT";
Char * sBadInsDatum="Bad datum in INSERT";
Char * sInsExtraCol="Extra column/name+space in INSERT";
Char * sRowInsFailed="Row insertion failed";
Char * sDuplicateKey="Duplicate key";
Char * sRowInsFailed2="Row insertion failed (2)";
Char * sBadStringQuotes = "Badly quoted string";
Char * sString1Quote="Single quote in string. Use ''";
Char * sStringTrunc="WARNING: String truncated to fit";
Char * sIntNoSpace="No space to store 4-byte integer";
Char * sSizeInt32B="We only allow Int32s: 0--999999999 [B]";
Char * sNumericNoSpace="No space for Numeric datum";
Char * sNumericTooLong="Numeric too long to fit in";
Char * sNumericTooLong2="Numeric too long to fit in [B]";
Char * sNoDateSpace="No space to insert 8-character date";
Char * sBadDateFormat="Bad date format not YYYY-MM-DD";
Char * sTimeNoSpace="No space: insert 12 character time";
Char * sTimeBadFormat="Bad TIME 'hh:mm:ss[.nnnnnn]'";
Char * sTimestampNoSpace="No space for 20 char timestamp";
Char * sTimestampFormat="TIMESTAMP yyyy-mm-dd hh:mm:ss[.nnnnnn]";
Char * sFloatNoSpace="No space to store 8byte float";
```

```
Char * sFloatFailedConversion="Failed to convert float";

Char * sAdjustFail1="Row update failed [A]";
Char * sAdjustFail2="Row update failed [B]";
Char * sAdjustFail3="Row update failed [C]";
Char * sFailNewRecord2="Failed to create empty record";
Char * sFailReleaseRecord2="Failed to release empty record";
Char * sMakeBufFile="Failed to create buffer Palmos file";
Char * sOpenBufFile="Failed to open buffer Palmos file";
Char * sMakeBufRecord="Failed to make buffer record";
Char * sMakeStack ="Failed to create STACK";
Char * sOpenStack ="Failed to open STACK";
Char * sFailSTACKUnlock ="Failed to unlock stack";
Char * sFailSTACKFree ="Failed to free stack";
Char * sFailSTACKUnlock2 ="Failed unlock stackstrings";
Char * sFailSTACKFree2 ="Failed free stackstrings";
Char * sFailSTACKClose ="Failed close stack";
Char * sLibDb="Library not found";
Char * sLibLoad="Failed to load library";

Char * sSQLbadStmt="SQL too short";
Char * sSQLbadStmt2="Resolved SQL statement bad";
Char * sSQLbadStmt3="Resolved Non-sel SQL bad";
Char * sErSQLbadStmtSmall="Non-sel SQL too short";

Char * sSQLmultipleMatches="Warn: SQL SELECT >1 match";
Char * sSQLunknownStmt="Unknown SQL stmt";
Char * sNoScriptStack="Failed script stack init. Baad!";
Char * sSQLinsertNoColumn="SQL column not in INSERT";
Char * sNotWidget="Failed to make widget";
Char * sMenuFound="Menu not found";
Char * sTopMenu="Cannot return from top menu!";
Char * sMakeDynamicForm="?";
Char * sNoWidgetData="No data found for widget";
Char * sBadWidgetName="Bad widget name";
Char * sFloatPop="Pop of float failed";
Char * sIntegerPop="Failed to pop integer";
Char * sStringPop="Failed to pop string";
Char * sNoFunction="Function not found";
Char * sBottomReturn="Warning: return from bottom fx";
Char * sScriptUnknown="Unknown option in script";
Char * sFunctionBuffer="Failed to open function buffer";
Char * sFunctionMemory="Insufficient function memory";
Char * sNoFunctionDB="Function database not found";
Char * sOpenFunctionDB="Failed to open function database";
Char * sInsortFailed="Two-byte insertion failed (sort?)";
Char * sLowLeftCompare="Bad comparison: left item";
Char * sLowRightCompare="Bad comparison: right item";
```

```
  Char * sHiLeftCompare="Bad compare: hi left";
  Char * sHiRightCompare="Bad compare: hi right";
  Char * sFreeFxPtr="Failed to free function ptr";
  Char * sCloseFx="Failed to close function file";
  Char * sFreeFxBuffer="Failed to free function buffer";
  Char * sFreeFxBufrec="Failed fx buffer record release";
  Char * sCloseFxBuffer="Failed to close function buffer";
  Char * sHexNumber="Bad hexadecimal number";
  Char * sColourLength="Bad colour length";
  Char * sColour="Bad colour";
  Char * sColourPop="Colour pop failed";
  Char * sColourObject="Bad ID of object in setting colour";
  Char * sObjectTypeColour="Bad object in colouring";
  Char * sObjHackID="Bad object ID in form";
  Char * sEnableObj="Can't en/disable bad object";
  Char * sLabel="Label text failed";
  Char * sLabelObject="Unknown object in label attempt";
  Char * sFxStackOver="Function stack overflow";
  Char * sBadMenuName="Bad menu name";
  Char * sNoMenuTitle="No menu title";
  Char * sNoXPush="Push of X failed. Full?";
//  Char * sPopForceX="Bad pop for ForceX!";
  Char * sRunRecursion="RUN not allowed within RUN";
  Char * sPopSetX="Bad pop for SetX";
  Char * sNoV="V value not found";
  Char * sFailEnterMenu="Failed to enter menu";
  Char * sPopInAsk="Pop failed in ASK stmt";
  Char * sPopTitleAsk="Title pop failed in ASK stmt";
  Char * sScriptFx="Script function failed";
  Char * sFailSQL="SQL statement failed";
  Char * sAlertFail="Alert fx failed";
  Char * sFailResolve="RESOLVE fx failed";
  Char * sFldTxt="Failed to insert field text";
  Char * sSqlTranslation="SQL translation failed";
  Char * sPackFail="SQL pack failed";
  Char * sSqlTableList="Bad SQL table list";
  Char * sRecFind="Record not found";
  Char * sNoDataTable="Data table not found";
  Char * sFailFormat="Formatting failed";
  Char * sWarnLongDat="Warn. Formatted datum too long. Trunc";
  Char * sMakeLocals="Failed make local variable file";
  Char * sOpenLocals="Failed open local variables";
  Char * sFailLocalsUnlock="On closing locals unlock failed";
  Char * sFailLocalsFree="On closing locals free failed";
  Char * sFailLocalsClose="Failed to close local variables";

//  Char * sPopMenuStack="POPMENU no menus?";
  Char * sPopMenuStack1="POPMENU bad argument";
```

```
Char * sPopMenuStack2="POPMENU bad index";
Char * sErBadMenuInteger="Bad integer as argument for MENU";
Char * sErBadColumnName2="Defective column name?";
Char * sErBadColumnItemCode="Bad column item code";
Char * sErBadColEnabling="Bad enabling integer";
Char * sErWidgetNoDbCode="No key response: no db widget code";
Char * sErPopIntConvertFail="Failed Numeric->Integer in Int POP";
Char * sErFailPopStringLen="Could not get length in string pop";
Char * sErLocalVariableName="Failed to MAKE local var";
Char * sErLocalVariableSet="Failed to SET local var";
Char * sErGetLocalVariable="Couldn't get local variable";
Char * sErFailFindFx="Fn not found";
Char * sErRunPushScript="push script failed in RUN";
Char * sErIniFxHandle="Problem init fx handle. AAgh";
Char * sErFunctionInitialisation="Bad Fx init";
Char * sErFailIniLocals="Failed to init local variables";
Char * sErDeepRecurStkShow="Deep recursion in stack display";
Char * sErBadLengthPeek="Problem peeking at length of stack item";
Char * sErFailSysPtrArray="System failed to make list array";
Char * sErFailLockPtrArray="System failed to lock list array";
Char * sErFailPostProcess="Post-processing of SQL failed";
Char * siColumnBeingSought = "Column being sought";
Char * siNodeDetails = "Node Details";

Char * sErWidgetMake = "Widget stack error";
Char * sErWidgetMake2 = "Widget stack err2";
Char * sErBadUser = "Bad user ID?";
Char * sErShortBytecode = "Bad @code";
Char * sErBadListInt = "Non-integer list code";

Char * sErNoFrom                = "SQL bad near FROM";
Char * sErNoWhere               = "SQL bad WHERE stmt";
Char * sErSqlPackFail           = "SQL Pack to IF failed";
Char * sErDbNotFound            = "SQL database not found";
Char * sErCannotOpenDb          = "SQL cannot open DB";
Char * sErDbNotMakeLink         = "SQL cannot make link";
Char * sErDbNoColumnLink        = "SQL bad column association";
Char * sErDbBadColumnOffset     = "SQL bad column offset";
Char * sErFailLocateColumn      = "SQL cant find column";
Char * sErFailMakeColumnNode    = "SQL failed to make column node";
Char * sErTruncateMaxColSize    = "SQL column truncated";
Char * sErRecursionInQuery      = "SQL odd recursion in query";
Char * sErStmtTooShort          = "SQL stmt too short";
Char * sErBadStmtLogic          = "SQL stmt bad logic";
Char * sErBadColNameLen         = "SQL bad col name length";
Char * sErBadColumnNodeCreation = "SQL column node failed";
Char * sErBadTranslation        = "SQL bad translation";
Char * sErTranslationFailed     = "";
```

```
Char * sErBadDatumType          = "SQL bad datum type";
Char * sErDatumFormatFailed     = "SQL datum format failed";
Char * sErBadItemQuote          = "SQL bad quotes";
Char * sErBadJoinNode           = "SQL bad join";
Char * sErBadQueryList          = "SQL bad query list";
Char * sErQueryItmFailed        = "SQL query item dud";
Char * sErBadRecordCount        = "SQL bad record count";
Char * sErBadDataTable          = "SQL bad data table";
Char * sErJoinConditionOmitted  = "SQL join condition left out";
Char * sErFailedDatumPush       = "SQL failed datum push";
Char * sErNoJoinUp              = "SQL no join up";
Char * sErComparisonFailed      = "SQL comparison failed";
Char * sErNumberSeekFailed      = "SQL number seek failed";
Char * sErGenericSeekFailure    = "SQL generic seek failure";
Char * sErBadLogicalOp          = "SQL bad logical operand";
Char * sErFailMakeDbList        = "SQL failed to make db list";
Char * sErRecQNotFound          = "SQL record not found";
Char * sErCollectionFailed      = "SQL query clean-up failed";
Char * sErDbFreeFailed          = "SQL query failed to free Db";
Char * sErNoSeparator           = "SQL separator not found";
Char * sErDudTableNames         = "SQL bad table names";
Char * sErPreformatFailed       = "SQL preprocess failed";
Char * sErStackMarkFailed       = "SQL stack mark failed";
Char * sErQueryFailed           = "SQL query failed";
Char * sErPostProcessFailed     = "SQL postprocess failed";
Char * sErFailedPackConditions  = "SQL pack failed";
Char * sErBadColumnIndex        = "SQL bad column index";
Char * sErWontFit               = "SQL update won't fit";
Char * sErBadColOffst           = "SQL update bad column offset";
Char * sErDefectiveRecord       = "SQL update dud record";
Char * sErBufferFailed          = "SQL update buffer failed";
Char * sErBadTableName          = "SQL update bad table name";
Char * sErNoSetCmd              = "SQL update no SET command";
Char * sErBadTableUp            = "SQL update bad table ptr";
Char * sErBadWhere              = "SQL update bad WHERE";
Char * sErBadQueryNodes         = "SQL update bad query node(s)";
Char * sErNoNodeInUpdate        = "SQL update no node";
Char * sErBadFormatting         = "SQL update bad format";
Char * sErBadUpdWhere           = "SQL update WHERE problem";
Char * sErFailRowUpd            = "SQL update fail row";
Char * sErSQLleftSpace          = "SQL bad operand no L space!";
Char * sErSqTinyBuffer          = "SQL buffer too small";
Char * sErSqBadIcomparator      = "SQL bad comparator I..";
Char * sErSqEqSpace             = "SQL '=' not followed by space";
Char * sErSqLtBad               = "SQL '<' followed by bad syntax";
Char * sErSqGtBad               = "SQL '>' followed by bad syntax";
Char * sErSqBadOperator         = "SQL bad operator";
Char * sErSqNo2ndOp             = "SQL no second operand";
```

```
Char * sErSq2ManyRpar           = "SQL too many ))s";
Char * sErLogicTooDeep          = "SQL logic too deep!";
Char * sErSqlWhile2Long         = "SQL WHILE stmt component too big";
Char * sErSqImbalParens         = "SQL Imbalanced parentheses";
Char * sErSqImbalQuotes         = "SQL Imbalanced 'quotes'";
Char * sErSqBadLogic            = "SQL Baaad logic";
Char * sErSqDudSort             = "SQL bad sort order";
Char * sErSqPreFull             = "SQL full preprocessing buffer";
Char * sErSqPreLogic            = "SQL preprocessing: bad (' logic";
Char * sErSqJoinFail            = "SQL Failure of relational integrity in DB:jo

Char * sScErAddEmpty            = "Addition err: no args";
Char * sScErAddArgs1            = "Addition err: bad integer";
Char * sScErAddArgs2            = "Addition err: bad float";
Char * sScErAddArgs3            = "Addition err: unsupported type";
Char * sScErAddIOver            = "Addition err: our overflow";

Char * sScErSubEmpty            = "Subtract err: no args";
Char * sScErSubArgs1            = "Subtract err: bad integer";
Char * sScErSubArgs2            = "Subtract err: bad float";
Char * sScErSubArgs3            = "Subtract err: unsupported type";
Char * sScErSubIUnder           = "Subtract err: underflow";


Char * sScErMulEmpty            = "Multiply err: no args";
Char * sScErMulArgs1            = "Multiply err: bad integer arg";
Char * sScErMulArgs2            = "Multiply err: bad float";
Char * sScErMulArgs3            = "Multiply err: unsupported type";
Char * sScErMulOver1            = "Multiply err: overflow";
Char * sScErMulOver2            = "Multiply err: our overflow";

Char * sScErNegEmpty            = "Negation err: no args";
Char * sScErNegArgs             = "Negation err: bad arguments";


Char * sScErDivEmpty            = "Divide err: no args";
Char * sScErDivArgs1            = "Divide err: bad integer dividend";
Char * sScErDivArgs2            = "Divide err: bad float dividend";
Char * sScErDivArgs3            = "Divide err: unsupported numeric";
Char * sScErDivArgs4            = "Divide err: bad numeric arg(s)";
Char * sScErDivOver             = "Divide err: divisor zero";


Char * sScErModEmpty            = "Modulo err: no args";
Char * sScErModArgs1            = "Modulo err: no args";
Char * sScErModArgs2            = "Modulo err: unsupported args";
Char * sScErModOver             = "Modulo err: divide by 0";
```

```
Char * sErPushNoStack           = "Push: no stack!";
Char * sErPushBadLength         = "Push: bad length";
Char * sErPushStackFull         = "Push: stack full";
Char * sErPushFailed            = "Push: failed";
Char * sErLongPushFull          = "Long push: full";
Char * sErLongPushFail          = "Long push: failed";

Char * sErEncodeIntegerEmpty     = "Int: empty";
Char * sErEncodeIntegerType      = "Int: bad type";
Char * sErEncodeIntegerOverflow  = "Int: bad size";

Char * sSqErEncodeNil             = "no datum lgth";
Char * sSqErUnquoted              = "varchar unquoted";
Char * sSqErOneQuote              = "unbalanced quote'";
Char * sSqErTruncated             = "truncated varc";
Char * sSqErIntegerSpace          = "int bad destination";
Char * sSqErIntegerBad            = "SQL bad integer";
Char * sSqErNumericScale          = "numeric bad scale";
Char * sSqErBadNumeric1           = "num char over 9";
Char * sSqErBadNumeric2           = "num char under 0";
Char * sSqErBadNumeric3           = "num double dot";
Char * sSqErBadNumeric4           = "num dest small";
Char * sSqErDateSpace             = "date dest small";
Char * sSqErDate                  = "date unquoted";
Char * sSqErDateTerminal          = "'date end quote";
Char * sSqErDateFormat            = "date bad format";
Char * sSqErTimeSpace             = "time dest small";
Char * sSqErTime                  = "bad time";
Char * sSqErTimeTerminal          = "'time end quote";
Char * sSqErTimeFormat            = "bad time format";
Char * sSqErStampSpace            = "stamp dest small";
Char * sSqErStamp                 = "bad stamp";
Char * sSqErStampTerminal         = "'stamp end quote";
Char * sSqErStampFormat           = "bad stamp format";
Char * sSqErFloatSpace            = "float dest small";
Char * sSqErFloatFormat           = "bad float format";
Char * sSqErEncode                = "encode unknown type";

Char * sScErSplitEmpty            = "Split <2 args";
Char * sScErSplitSeparator        = "Bad split separator";
Char * sScErSplitLen              = "Bad split len";
Char * sScErSplitResolve          = "Split resolve failed";
Char * sScErCutEmpty              = "Cut <2 args";
Char * sScErCut                   = "Cut error";

Char * sScErResolveEmptyStack      = "Resolve few args";
Char * sScErResolveSpace           = "Resolve too small";
Char * sScErResolveCount           = "Resolve bad count";
```

```
Char * sScErResolveInsertions       = "Resolve too few";
Char * sScErResolveFull             = "Resolve no space";

Char * sScErDecodeNoSpace           = "Decode space?";
Char * sScErDecodeCompound          = "Decode X?";
Char * sScErDecodeType              = "Decode type?";
Char * sScErDecodeDate              = "Decode date?";
Char * sScErDecodeFloat             = "Decode float len?";
Char * sScBadDecode                 = "Decode bad";

Char * sScErDecodeNumeric           = "Dud numeric";
Char * sScErDecodeTimestamp         = "Dud timestamp";
Char * sScErDecodeTime              = "Dud time";
Char * sScErDecodeInteger           = "Dud Int";


Char * sScErCopyEmpty                = "Copy empty";
Char * sScErCopyOverflow             = "Copy STK full";
Char * sScErCopyFull                 = "Copy full";
Char * sScErEncodeFloatEmpty         = "Float: empty";
Char * sScErFloatLength              = "Float too long";
Char * sScErEncodeFloatType          = "Float arg?";
Char * sScErEncodeTsEmpty          = "Ts: empty";
Char * sScErTsLength               = "Ts: dud";
Char * sScErEncodeTsType           = "Ts: arg?";

Char * sScErEncodeDateEmpty          = "Date: empty";
Char * sScErDateLength               = "Date: dud";
Char * sScErEncodeDateType           = "Date: arg?";
Char * sScErEncodeTimeEmpty          = "Time: empty";
Char * sScErTimeLength               = "Time: dud";
Char * sScErEncodeTimeType           = "Time: arg?";

Char * sScErJoinEmpty                = "Bad join(few args)";
Char * sScErRepairFailed             = "Fatal. PDB repair failed";
Char * sScErReplace                  = "Bad REPLACE";
```

Whew! Here's the meat of the routine, which is no less cumbersome. All we really use is a large switch statement to switch between the options. As always with C, it's important to remember to put a break statement after every case. From here you can jump to the end of the boring switch statement.

```
Char * es=0;
Char * keepes;
Int16 outlen;
outlen = 0;
```

```
switch (errnum)
  {
    case ErTestingError:
      es = sTestingError;
    break;

    case ErNullMemoryRequest:
      es = sNullMemoryRequest;
    break;

    case ErNoListMemory:
      es = sNoListMemory;
    break;

    case ErFailInsertList:
      es = sFailInsertList;
    break;

    case ErListCreationFailed:
      es = sListCreationFailed;
    break;

    case ErNoScratchBuffer:
      es = sNoScratchBuffer;
    break;

    case ErPalmDbNotFound:
      es = sPalmDbNotFound;
    break;

    case ErPalmDbNotOpen:
      es = sPalmDbNotOpen;
    break;

    case ErPalmLinkDbNotOpen:
      es = sPalmLinkDbNotOpen;
    break;

    case ErNotClosePalmDb:
      es = sNotClosePalmDb;
    break;

    case ErBadDbSizeCount:
      es = sBadDbSizeCount;
    break;

    case ErNulInt16Write:
      es = sNulInt16Write;
```

```
      break;

      case ErNulInt32Write:
        es = sNulInt32Write;
      break;

      case ErScratchOverflow:
        es = sScratchOverflow;
      break;

      case ErScratchOverflow2:
        es = sScratchOverflow2;
      break;

      case ErCannotCreateScratch:
        es = sCannotCreateScratch;
      break;

      case ErTextComparisonMode:
        es = sTextComparisonMode;
      break;

      case ErFloatComparison:
        es = sFloatComparison;
      break;

      case ErFloatComparison2:
        es = sFloatComparison2;
      break;

      case ErQueryNoIFdelimiter:
        es = sQueryNoIFdelimiter;
      break;

      case ErQueryNodeNoMake:
        es = sQueryNodeNoMake;
      break;

      case ErIntComparisonMode:
        es = sIntComparisonMode;
      break;

      case ErSQLTranslation1:
        es = sSQLTranslation1;
      break;

      case ErSQLNoWhere:
        es = sSQLNoWhere;
```

```
      break;

      case ErSQLBadWhere:
        es = sSQLBadWhere;
      break;

      case ErSQLselect:
        es = sSQLselect;
      break;

      case ErSQLleftSpaceCondition:
        es =sSQLleftSpaceCondition;
      break;

      case ErSQLBadCondition:
        es = sSQLBadCondition;
      break;

      case ErUnknownSQLTest:
        es = sUnknownSQLTest;
      break;

      case ErSQLComparator:
        es = sSQLComparator;
      break;

      case ErSQLComparator2:
        es = sSQLComparator2;
      break;

      case ErSQLComparator3:
        es = sSQLComparator3;
      break;

      case ErSQLComparatorStop:
        es = sSQLComparatorStop;
      break;

      case ErSQLNoCfSpace:
        es = sSQLNoCfSpace;
      break;

      case ErKillLongUpQueryNode:
        es = sKillLongUpQueryNode;
      break;

      case ErDefectNextPtr:
        es = sDefectNextPtr;
```

```
      break;

      case ErQueryTooShort:
        es = sQueryTooShort;
      break;

      case ErQueryNoSpace:
        es = sQueryNoSpace;
      break;

      case ErSQLnoColumn:
        es = sSQLnoColumn;
      break;

      case ErSQLBadDatum:
        es = sSQLBadDatum;
      break;

      case ErBadSQLtype:
        es = sBadSQLtype;
      break;

      case ErDatumXlateFailed:
        es = sDatumXlateFailed;
      break;

      case ErSQLNoCol:
        es = sSQLNoCol;
      break;

      case ErSQLNoQuote:
        es = sSQLNoQuote;
      break;

      case ErSQLlogic:
        es = sSQLlogic;
      break;

      case ErNoMemory:
        es = sNoMemory;
      break;

      case ErNoMemLock:
        es = sNoMemLock;
      break;


      case ErMemoryNoUnlock:
```

```
  es = sMemoryNoUnlock;
break;

case ErMemoryNoFree:
  es = sMemoryNoFree;
break;

case ErBadControlStyle:
  es = sBadControlStyle;
break;

case ErPalmDbCantClose:
  es = sPalmDbCantClose;
break;

case ErFailNewRecord:
  es = sFailNewRecord;
break;

case ErFailWriteRecord:
  es = sFailWriteRecord;
break;

case ErFailUnlockRecord:
  es = sFailUnlockRecord;
break;

case ErFailReleaseRecord:
  es = sFailReleaseRecord;
break;

case ErLockNullHandle:
  es = sLockNullHandle;
break;

case ErUnlockNullHandle:
  es = sUnlockNullHandle;
break;

case ErFailUnlockNulHandle:
  es = sFailUnlockNulHandle;
break;

case ErNoFreeNulPtr:
  es = sNoFreeNulPtr;
break;

case ErFailFreeMemPtr:
```

```
  es = sFailFreeMemPtr;
break;

case ErCannotUnlockNulPtr:
  es = sCannotUnlockNulPtr;
break;

case ErFailUnlockMemPtr:
  es = sFailUnlockMemPtr;
break;

case ErCantReleaseNulRec:
  es = sCantReleaseNulRec;
break;

case ErFailReleaseRec:
  es = sFailReleaseRec;
break;

case ErCloseNulDb:
  es = sCloseNulDb;
break;

case ErFailCloseDb:
  es = sFailCloseDb;
break;

case ErNulDbName:
  es = sNulDbName;
break;

case ErFailMakeDb:
  es = sFailMakeDb;
break;

case ErDelNulDb:
  es = sDelNulDb;
break;

case ErDelDbFail:
  es = sDelDbFail;
break;

case ErWriteNulRec:
  es = sWriteNulRec;
break;

case ErWriteNulSrc:
```

```
      es = sWriteNulSrc;
    break;

    case ErFailRecCheck:
      es = sFailRecCheck;
    break;

    case ErFailRecWrite:
      es = sFailRecWrite;
    break;

    case ErSortNulDb:
      es = sSortNulDb;
    break;

    case ErSortNoRec:
      es = sSortNoRec;
    break;

    case ErSortNoFx:
      es = sSortNoFx;
    break;

    case ErNotRemoveNulRec:
      es = sNotRemoveNulRec;
    break;

    case ErFailRecRemove:
      es = sFailRecRemove;
    break;

    case ErNulCopyDestin:
      es = sNulCopyDestin;
    break;

    case ErNulCopySrc:
      es = sNulCopySrc;
    break;

    case ErNoScratch:
      es = sNoScratch;
    break;

    case ErFailScratchUnlock:
      es = sFailScratchUnlock;
    break;

    case ErFailScratchFree:
```

```
    es = sFailScratchFree;
  break;

  case ErFailScratchClose:
    es = sFailScratchClose;
  break;

  case ErDelNulPtr:
    es = sDelNulPtr;
  break;

  case ErBadAscIntLen:
    es = sBadAscIntLen;
  break;

  case ErLongAscInt:
    es = sLongAscInt;
  break;

  case ErBadAscIntString:
    es = sBadAscIntString;
  break;

  case ErSizeInt32:
    es = sSizeInt32;
  break;

  case ErFloatTooLong:
    es = sFloatTooLong;
  break;

  case ErDateLen:
    es = sDateLen;
  break;

  case ErDateSeparator:
    es = sDateSeparator;
  break;

  case ErDateSeparator2:
    es = sDateSeparator2;
  break;

  case ErImbalancedPars:
    es = sImbalancedPars;
  break;

  case ErBadConditional:
```

```
  es = sBadConditional;
break;

case ErBadConditional2:
  es = sBadConditional2;
break;

case ErLogicOverflow:
  es = sLogicOverflow;
break;

case ErWhileOverflow:
  es = sWhileOverflow;
break;

case ErFewParentheses:
  es = sFewParentheses;
break;

case ErQuoteImbalance:
  es = sQuoteImbalance;
break;

case ErBadLogic:
  es = sBadLogic;
break;

case ErStoreCondition:
  es = sStoreCondition;
break;

case ErSelCleanFail:
  es = sSelCleanFail;
break;

case ErDbCleanFail:
  es = sDbCleanFail;
break;

case ErNoCol:
  es = sNoCol;
break;

case ErBadTable:
  es = sBadTable;
break;

case ErNoDbTbl:
```

```
  es = sNoDbTbl;
break;

case ErBadDbMatch:
  es = sBadDbMatch;
break;

case ErBadColumnName:
  es = sBadColumnName;
break;

case ErColNotFound:
  es = sColNotFound;
break;

case ErNoQueryOp:
  es = sNoQueryOp;
break;

case ErBadNumericType:
  es = sBadNumericType;
break;

case ErBadType:
  es = sBadType;
break;

case ErScratchWrite:
  es = sScratchWrite;
break;

case ErScratchTerminal:
  es = sScratchTerminal;
break;

case ErInScratch:
  es = sInScratch;
break;

case ErScratchDest:
  es = sScratchDest;
break;

case ErBadScratchDatum:
  es = sBadScratchDatum;
break;

case ErPullScratch:
```

```
  es = sPullScratch;
break;

case ErScratchDest2:
  es = sScratchDest2;
break;

case ErSizeInt32A:
  es = sSizeInt32A;
break;

case ErBadTblCREATE:
  es = sBadTblCREATE;
break;

case ErCreateTermRpar:
  es = sCreateTermRpar;
break;

case ErCREATENoMem:
  es = sCREATENoMem;
break;

case ErCREATEfailed:
  es = sCREATEfailed;
break;

case ErBadTblNameCreate:
  es = sBadTblNameCreate;
break;

case ErCreateTableExists:
  es = sCreateTableExists;
break;

case ErBadColNameCreate:
  es = sBadColNameCreate;
break;

case ErCreateBadColType:
  es = sCreateBadColType;
break;

case ErCreateBadCol:
  es = sCreateBadCol;
break;

case ErCreateNoComma:
```

```
  es = sCreateNoComma;
break;

case ErCreateNoPrecision:
  es = sCreateNoPrecision;
break;

case ErCreateTooManyCols:
  es = sCreateTooManyCols;
break;

case ErCreateFailed:
  es = sCreateFailed;
break;

case ErCreateOpenFail:
  es = sCreateOpenFail;
break;

case ErFailMakeCreateHdr:
  es = sFailMakeCreateHdr;
break;

case ErFailWriteCreateHdr:
  es = sFailWriteCreateHdr;
break;

case ErCreateNotCloseHdr:
  es = sCreateNotCloseHdr;
break;

case ErInsertNoTblname:
  es = sInsertNoTblname;
break;

case ErNoInsertTbl:
  es = sNoInsertTbl;
break;

case ErInsertNoHdrAccess:
  es = sInsertNoHdrAccess;
break;

case ErInsColsNoRpar:
  es = sInsColsNoRpar;
break;

case ErInsNoVALUES:
```

```
  es = sInsNoVALUES;
break;

case ErInsRemainingColData:
  es = sInsRemainingColData;
break;

case ErInsBadKey:
  es = sInsBadKey;
break;

case ErBadInsDatum:
  es = sBadInsDatum;
break;

case ErInsExtraCol:
  es = sInsExtraCol;
break;

case ErRowInsFailed:
   es = sRowInsFailed;
break;

case ErDuplicateKey:
  es = sDuplicateKey;
break;

case ErRowInsFailed2:
   es = sRowInsFailed2;
break;

case ErBadStringQuotes:
   es = sBadStringQuotes;
break;

case ErString1Quote:
  es = sString1Quote;
break;

case ErStringTrunc:
  es = sStringTrunc;
break;

case ErIntNoSpace:
  es = sIntNoSpace;
break;

case ErSizeInt32B:
```

```
  es = sSizeInt32B;
break;

case ErNumericNoSpace:
  es = sNumericNoSpace;
break;

case ErNumericTooLong:
  es = sNumericTooLong;
break;

case ErNumericTooLong2:
  es = sNumericTooLong2;
break;

case ErNoDateSpace:
  es = sNoDateSpace;
break;

case ErBadDateFormat:
  es = sBadDateFormat;
break;

case ErTimeNoSpace:
  es = sTimeNoSpace;
break;

case ErTimeBadFormat:
  es = sTimeBadFormat;
break;

case ErTimestampNoSpace:
  es = sTimestampNoSpace;
break;

case ErTimestampFormat:
  es = sTimestampFormat;
break;

case ErFloatNoSpace:
  es = sFloatNoSpace;
break;

case ErFloatFailedConversion:
  es = sFloatFailedConversion;
break;
```

```
case ErAdjustFail1:
  es = sAdjustFail1;
break;

case ErAdjustFail2:
  es = sAdjustFail2;
break;

case ErAdjustFail3:
  es = sAdjustFail3;
break;

case ErFailNewRecord2:
  es = sFailNewRecord2;
break;

case ErFailReleaseRecord2:
  es = sFailReleaseRecord2;
break;

case ErMakeBufFile:
  es = sMakeBufFile;
break;

case ErOpenBufFile:
  es = sOpenBufFile;
break;

case ErMakeBufRecord:
  es = sMakeBufRecord;
break;

case  ErMakeStack:
  es = sMakeStack;
break;

case  ErOpenStack:
  es = sOpenStack;
break;

case  ErFailSTACKUnlock:
  es = sFailSTACKUnlock;
break;

case  ErFailSTACKFree:
  es = sFailSTACKFree;
break;
```

```
case  ErFailSTACKUnlock2:
  es = sFailSTACKUnlock2;
break;

case  ErFailSTACKFree2:
  es = sFailSTACKFree2;
break;

case  ErFailSTACKClose:
  es = sFailSTACKClose;
break;


case ErLibDb:
  es = sLibDb;
break;

case ErLibLoad:
  es = sLibLoad;
break;

case ErSQLbadStmt:
  es = sSQLbadStmt;
break;

case ErSQLbadStmt2:
  es = sSQLbadStmt2;
break;

case ErSQLbadStmt3:
  es = sSQLbadStmt3;
break;

case ErSQLmultipleMatches:
  es = sSQLmultipleMatches;
break;

case ErSQLunknownStmt:
  es = sSQLunknownStmt;
break;

case ErNoScriptStack:
  es = sNoScriptStack;
break;

case ErSQLinsertNoColumn:
  es = sSQLinsertNoColumn;
break;
```

```
case ErNotWidget:
  es = sNotWidget;
break;

case ErMenuFound:
  es = sMenuFound;
break;

case ErTopMenu:
  es = sTopMenu;
break;

case ErMakeDynamicForm:
  es = sMakeDynamicForm;
break;

case ErNoWidgetData:
  es = sNoWidgetData;
break;

case ErBadWidgetName:
  es = sBadWidgetName;
break;

case ErFloatPop:
  es = sFloatPop;
break;

case ErIntegerPop:
  es = sIntegerPop;
break;

case ErStringPop:
  es = sStringPop;
break;

case ErNoFunction:
  es = sNoFunction;
break;

case ErBottomReturn:
  es = sBottomReturn;
break;

case ErScriptUnknown:
  es = sScriptUnknown;
break;
```

```
case ErFunctionBuffer:
  es = sFunctionBuffer;
break;

case ErFunctionMemory:
  es = sFunctionMemory;
break;

case ErNoFunctionDB:
  es = sNoFunctionDB;
break;

case ErOpenFunctionDB:
  es = sOpenFunctionDB;
break;

case ErInsortFailed:
  es = sInsortFailed;
break;

case ErLowLeftCompare:
  es = sLowLeftCompare;
break;

case ErLowRightCompare:
  es = sLowRightCompare;
break;

case ErHiLeftCompare:
  es = sHiLeftCompare;
break;

case ErHiRightCompare:
  es = sHiRightCompare;
break;

case ErFreeFxPtr:
  es = sFreeFxPtr;
break;

case ErCloseFx:
  es = sCloseFx;
break;

case ErFreeFxBuffer:
  es = sFreeFxBuffer;
break;
```

```
case ErFreeFxBufrec:
  es = sFreeFxBufrec;
break;

case ErCloseFxBuffer:
  es = sCloseFxBuffer;
break;

case ErHexNumber:
  es = sHexNumber;
break;

case ErColourLength:
  es = sColourLength;
break;

case ErColour:
  es = sColour;
break;

case ErColourPop:
  es = sColourPop;
break;

case ErColourObject:
  es = sColourObject;
break;

case ErObjectTypeColour:
  es = sObjectTypeColour;
break;

case ErObjHackID:
  es = sObjHackID;
break;

case ErEnableObj:
  es = sEnableObj;
break;

case ErLabel:
  es = sLabel;
break;

case ErLabelObject:
  es = sLabelObject;
break;
```

```
      case ErFxStackOver:
        es = sFxStackOver;
      break;

      case ErBadMenuName:
        es = sBadMenuName;
      break;

      case ErNoMenuTitle:
        es = sNoMenuTitle;
      break;

      case ErNoXPush:
        es = sNoXPush;
      break;

//      case ErPopForceX:
//        es = sPopForceX;
//      break;

      case ErRunRecursion:
        es = sRunRecursion;
      break;

      case ErPopSetX:
        es = sPopSetX;
      break;

      case ErNoV:
        es = sNoV;
      break;

      case ErFailEnterMenu:
        es = sFailEnterMenu;
      break;

      case ErPopInAsk:
        es = sPopInAsk;
      break;

      case ErPopTitleAsk:
        es = sPopTitleAsk;
      break;

      case ErScriptFx:
        es = sScriptFx;
      break;
```

```
case ErFailSQL:
  es = sFailSQL;
break;

case ErAlertFail:
  es = sAlertFail;
break;

case ErFailResolve:
  es = sFailResolve;
break;

case ErFldTxt:
  es = sFldTxt;
break;

case ErSqlTranslation:
  es = sSqlTranslation;
break;

case ErPackFail:
  es = sPackFail;
break;

case ErSqlTableList:
  es = sSqlTableList;
break;

case ErRecFind:
  es = sRecFind;
break;

case ErNoDataTable:
  es = sNoDataTable;
break;

case ErFailFormat:
  es = sFailFormat;
break;

case ErWarnLongDat:
  es = sWarnLongDat;
break;

case  ErMakeLocals:
  es = sMakeLocals;
break;
```

```
      case  ErOpenLocals:
        es = sOpenLocals;
      break;

      case  ErFailLocalsUnlock:
        es = sFailLocalsUnlock;
      break;

      case  ErFailLocalsFree:
        es = sFailLocalsFree;
      break;

      case  ErFailLocalsClose:
        es = sFailLocalsClose;
      break;

//      case ErPopMenuStack:
//        es = sPopMenuStack;
//      break;

      case ErPopMenuStack1:
        es = sPopMenuStack1;
      break;

      case ErPopMenuStack2:
        es = sPopMenuStack2;
      break;

      case iColumnBeingSought:
        es = siColumnBeingSought;
      break;

      case iNodeDetails:
        es = siNodeDetails;
      break;

      case ErBadMenuInteger:
        es = sErBadMenuInteger;
      break;

      case ErBadColumnName2:
        es = sErBadColumnName2;
      break;

      case ErBadColumnItemCode:
        es = sErBadColumnItemCode;
      break;
```

```
case ErBadColEnabling:
  es = sErBadColEnabling;
break;

case ErWidgetNoDbCode:
  es = sErWidgetNoDbCode;
break;

case ErFailPopStringLen:
  es = sErFailPopStringLen;
break;

case ErLocalVariableName:
  es = sErLocalVariableName;
break;

case ErLocalVariableSet:
  es = sErLocalVariableSet;
break;

case ErGetLocalVariable:
  es = sErGetLocalVariable;
break;

case ErFailFindFx:
  es = sErFailFindFx;
break;

case ErRunPushScript:
  es = sErRunPushScript;
break;

case ErSQLbadStmtSmall:
  es = sErSQLbadStmtSmall;
break;

case ErIniFxHandle:
  es = sErIniFxHandle;
break;

case ErFunctionInitialisation:
  es = sErFunctionInitialisation;
break;

case ErFailIniLocals:
  es = sErFailIniLocals;
break;
```

```
    case ErDeepRecurStkShow:
      es = sErDeepRecurStkShow;
    break;

    case ErBadLengthPeek:
      es = sErBadLengthPeek;
    break;

    case ErFailSysPtrArray:
      es = sErFailSysPtrArray;
    break;

    case ErFailLockPtrArray:
      es = sErFailLockPtrArray;
    break;

    case ErFailPostProcess:
      es = sErFailPostProcess;
    break;

    case ErPopIntConvertFail:
      es = sErPopIntConvertFail;
    break;

    case ErWidgetMake:
     es = sErWidgetMake;
    break;

    case ErWidgetMake2:
     es = sErWidgetMake2;
    break;

    case ErBadUser:
     es = sErBadUser;
    break;

    case ErBadListInt:
     es = sErBadListInt;
    break;

    case ErShortBytecode:
     es = sErShortBytecode;

case  ErNoFrom:  es = sErNoFrom; break;
case  ErNoWhere:  es = sErNoWhere; break;
case  ErSqlPackFail:  es = sErSqlPackFail; break;
case  ErDbNotFound:  es = sErDbNotFound; break;
```

```
case  ErCannotOpenDb:  es = sErCannotOpenDb; break;
case  ErDbNotMakeLink:  es = sErDbNotMakeLink; break;
case  ErDbNoColumnLink:  es = sErDbNoColumnLink; break;
case  ErDbBadColumnOffset:  es = sErDbBadColumnOffset; break;
case  ErFailLocateColumn:  es = sErFailLocateColumn; break;
case  ErFailMakeColumnNode:  es = sErFailMakeColumnNode; break;
case  ErTruncateMaxColSize:  es = sErTruncateMaxColSize; break;
case  ErRecursionInQuery:  es = sErRecursionInQuery; break;
case  ErStmtTooShort:  es = sErStmtTooShort; break;
case  ErBadStmtLogic:  es = sErBadStmtLogic; break;
case  ErBadColNameLen:  es = sErBadColNameLen; break;
case  ErBadColumnNodeCreation:  es = sErBadColumnNodeCreation; break;
case  ErBadTranslation:  es = sErBadTranslation; break;
case  ErTranslationFailed:  es = sErTranslationFailed; break;
case  ErBadDatumType:  es = sErBadDatumType; break;
case  ErDatumFormatFailed:  es = sErDatumFormatFailed; break;
case  ErBadItemQuote:  es = sErBadItemQuote; break;
case  ErBadJoinNode:  es = sErBadJoinNode; break;
case  ErBadQueryList:  es = sErBadQueryList; break;
case  ErQueryItmFailed:  es = sErQueryItmFailed; break;
case  ErBadRecordCount:  es = sErBadRecordCount; break;
case  ErBadDataTable:  es = sErBadDataTable; break;
case  ErJoinConditionOmitted:  es = sErJoinConditionOmitted; break;
case  ErFailedDatumPush:  es = sErFailedDatumPush; break;
case  ErNoJoinUp:  es = sErNoJoinUp; break;
case  ErComparisonFailed:  es = sErComparisonFailed; break;
case  ErNumberSeekFailed:  es = sErNumberSeekFailed; break;
case  ErGenericSeekFailure:  es = sErGenericSeekFailure; break;
case  ErBadLogicalOp:  es = sErBadLogicalOp; break;
case  ErFailMakeDbList:  es = sErFailMakeDbList; break;
case  ErRecQNotFound:  es = sErRecQNotFound; break;
case  ErCollectionFailed:  es = sErCollectionFailed; break;
case  ErDbFreeFailed:  es = sErDbFreeFailed; break;
case  ErNoSeparator:  es = sErNoSeparator; break;
case  ErDudTableNames:  es = sErDudTableNames; break;
case  ErPreformatFailed:  es = sErPreformatFailed; break;
case  ErStackMarkFailed:  es = sErStackMarkFailed; break;
case  ErQueryFailed:  es = sErQueryFailed; break;
case  ErPostProcessFailed:  es = sErPostProcessFailed; break;
case  ErFailedPackConditions:  es = sErFailedPackConditions; break;
case  ErBadColumnIndex:  es = sErBadColumnIndex; break;
case  ErWontFit:  es = sErWontFit; break;
case  ErBadColOffst:  es = sErBadColOffst; break;
case  ErDefectiveRecord:  es = sErDefectiveRecord; break;
case  ErBufferFailed:  es = sErBufferFailed; break;
case  ErBadTableName:  es = sErBadTableName; break;
case  ErNoSetCmd:  es = sErNoSetCmd; break;
case  ErBadTableUp:  es = sErBadTableUp; break;
```

```
case  ErBadWhere:  es = sErBadWhere; break;
case  ErBadQueryNodes:  es = sErBadQueryNodes; break;
case  ErNoNodeInUpdate:  es = sErNoNodeInUpdate; break;
case  ErBadFormatting:  es = sErBadFormatting; break;
case  ErBadUpdWhere:  es = sErBadUpdWhere; break;
case  ErFailRowUpd:  es = sErFailRowUpd; break;
case  ErSQLleftSpace: es = sErSQLleftSpace; break;
case  ErSqTinyBuffer: es = sErSqTinyBuffer; break;
case  ErSqBadIcomparator: es = sErSqBadIcomparator; break;
case  ErSqEqSpace: es = sErSqEqSpace; break;
case  ErSqLtBad: es = sErSqLtBad; break;
case  ErSqGtBad: es = sErSqGtBad; break;
case  ErSqBadOperator: es = sErSqBadOperator; break;
case  ErSqNo2ndOp: es = sErSqNo2ndOp; break;
case  ErSq2ManyRpar: es = sErSq2ManyRpar; break;
case  ErLogicTooDeep: es = sErLogicTooDeep; break;
case  ErSqlWhile2Long: es = sErSqlWhile2Long; break;
case  ErSqImbalParens: es = sErSqImbalParens; break;
case  ErSqImbalQuotes: es = sErSqImbalQuotes; break;
case  ErSqBadLogic: es = sErSqBadLogic; break;
case  ErSqDudSort: es = sErSqDudSort; break;
case  ErSqPreFull: es = sErSqPreFull; break;
case  ErSqPreLogic: es =sErSqPreLogic; break;
case  ErSqJoinFail: es = sErSqJoinFail; break;


case  ScErAddEmpty : es = sScErAddEmpty; break;
case  ScErAddArgs1 : es = sScErAddArgs1; break;
case  ScErAddArgs2 : es = sScErAddArgs2; break;
case  ScErAddArgs3 : es = sScErAddArgs3; break;
case  ScErAddIOver : es = sScErAddIOver; break;

case  ScErSubEmpty : es = sScErSubEmpty; break;
case  ScErSubArgs1 : es = sScErSubArgs1; break;
case  ScErSubArgs2 : es = sScErSubArgs2; break;
case  ScErSubArgs3 : es = sScErSubArgs3; break;
case  ScErSubIUnder: es = sScErSubIUnder; break;

case  ScErMulEmpty : es = sScErMulEmpty; break;
case  ScErMulArgs1 : es = sScErMulArgs1; break;
case  ScErMulArgs2 : es = sScErMulArgs2; break;
case  ScErMulArgs3 : es = sScErMulArgs3; break;
case  ScErMulOver1 : es = sScErMulOver1; break;
case  ScErMulOver2 : es = sScErMulOver2; break;

case  ScErNegEmpty : es = sScErNegEmpty; break;
case  ScErNegArgs  : es = sScErNegArgs; break;
```

```
case  ScErDivEmpty : es = sScErDivEmpty; break;
case  ScErDivArgs1 : es = sScErDivArgs1; break;
case  ScErDivArgs2 : es = sScErDivArgs2; break;
case  ScErDivArgs3 : es = sScErDivArgs3; break;
case  ScErDivArgs4 : es = sScErDivArgs4; break;
case  ScErDivOver  : es = sScErDivOver; break;

case  ScErModEmpty : es = sScErModEmpty; break;
case  ScErModArgs1 : es = sScErModArgs1; break;
case  ScErModArgs2 : es = sScErModArgs2; break;
case  ScErModOver  : es = sScErModOver; break;

case  ErPushNoStack   : es = sErPushNoStack  ; break;
case  ErPushBadLength : es = sErPushBadLength; break;
case  ErPushStackFull : es = sErPushStackFull; break;
case  ErPushFailed    : es = sErPushFailed   ; break;
case  ErLongPushFull  : es = sErLongPushFull ; break;
case  ErLongPushFail  : es = sErLongPushFail  ; break;

case  ErEncodeIntegerEmpty: es = sErEncodeIntegerEmpty; break;
case  ErEncodeIntegerType: es = sErEncodeIntegerType; break;
case  ErEncodeIntegerOverflow: es = sErEncodeIntegerOverflow; break;

case SqErEncodeNil           : es = sSqErEncodeNil    ; break;
case SqErUnquoted            : es = sSqErUnquoted     ; break;
case SqErOneQuote            : es = sSqErOneQuote     ; break;
case SqErTruncated           : es = sSqErTruncated    ; break;
case SqErIntegerSpace        : es = sSqErIntegerSpace ; break;
case SqErIntegerBad          : es = sSqErIntegerBad   ; break;
case SqErNumericScale        : es = sSqErNumericScale ; break;
case SqErBadNumeric1         : es = sSqErBadNumeric1  ; break;
case SqErBadNumeric2         : es = sSqErBadNumeric2  ; break;
case SqErBadNumeric3         : es = sSqErBadNumeric3  ; break;
case SqErBadNumeric4         : es = sSqErBadNumeric4  ; break;
case SqErDateSpace           : es = sSqErDateSpace    ; break;
case SqErDate                : es = sSqErDate         ; break;
case SqErDateTerminal        : es = sSqErDateTerminal ; break;
case SqErDateFormat          : es = sSqErDateFormat   ; break;
case SqErTimeSpace           : es = sSqErTimeSpace    ; break;
case SqErTime                : es = sSqErTime         ; break;
case SqErTimeTerminal        : es = sSqErTimeTerminal ; break;
case SqErTimeFormat          : es = sSqErTimeFormat   ; break;
case SqErStampSpace          : es = sSqErStampSpace   ; break;
case SqErStamp               : es = sSqErStamp        ; break;
case SqErStampTerminal       : es = sSqErStampTerminal; break;
case SqErStampFormat         : es = sSqErStampFormat  ; break;
case SqErFloatSpace          : es = sSqErFloatSpace   ; break;
case SqErFloatFormat         : es = sSqErFloatFormat  ; break;
```

```
  case SqErEncode             : es = sSqErEncode       ; break;

  case ScErSplitEmpty         : es = sScErSplitEmpty    ; break;
  case ScErSplitSeparator     : es = sScErSplitSeparator; break;
  case ScErSplitLen           : es = sScErSplitLen      ; break;
  case ScErSplitResolve       : es = sScErSplitResolve  ; break;
  case ScErCutEmpty           : es = sScErCutEmpty      ; break;
  case ScErCut                : es = sScErCut           ; break;

  case ScErResolveEmptyStack  : es = sScErResolveEmptyStack ; break;
  case ScErResolveSpace       : es = sScErResolveSpace      ; break;
  case ScErResolveCount       : es = sScErResolveCount      ; break;
  case ScErResolveInsertions  : es = sScErResolveInsertions ; break;
  case ScErResolveFull        : es = sScErResolveFull       ; break;

  case ScErDecodeNoSpace     : es = sScErDecodeNoSpace     ; break;
  case ScErDecodeCompound    : es = sScErDecodeCompound    ; break;
  case ScErDecodeType        : es = sScErDecodeType        ; break;
  case ScErDecodeDate        : es = sScErDecodeDate        ; break;
  case ScErDecodeFloat       : es = sScErDecodeFloat       ; break;
  case ScBadDecode           : es = sScBadDecode           ; break;

  case ScErDecodeNumeric     : es = sScErDecodeNumeric ; break;
  case ScErDecodeTimestamp   : es = sScErDecodeTimestamp; break;
  case ScErDecodeTime        : es = sScErDecodeTime; break;
  case ScErDecodeInteger     : es = sScErDecodeInteger; break;

  case ScErCopyEmpty         : es = sScErCopyEmpty     ; break;
  case ScErCopyOverflow      : es = sScErCopyOverflow  ; break;
  case ScErCopyFull          : es = sScErCopyFull      ; break;
  case ScErEncodeFloatEmpty  : es = sScErEncodeFloatEmpty; break;
  case ScErFloatLength       : es = sScErFloatLength; break;
  case ScErEncodeFloatType   : es = sScErEncodeFloatType; break;

  case ScErEncodeTsEmpty  : es = sScErEncodeTsEmpty; break;
  case ScErTsLength       : es = sScErTsLength; break;
  case ScErEncodeTsType   : es = sScErEncodeTsType; break;

  case ScErEncodeDateEmpty  : es = sScErEncodeDateEmpty; break;
  case ScErDateLength       : es = sScErDateLength; break;
  case ScErEncodeDateType   : es = sScErEncodeDateType; break;

  case ScErEncodeTimeEmpty  : es = sScErEncodeTimeEmpty; break;
  case ScErTimeLength       : es = sScErTimeLength; break;
  case ScErEncodeTimeType   : es = sScErEncodeTimeType; break;

  case ScErReplace          : es = sScErReplace; break;
  case ScErJoinEmpty        : es = sScErJoinEmpty; break;
```

```
case ScErRepairFailed     : es = sScErRepairFailed; break;
```

...and here we are at the end. The default (failure mode) is to return zero:

```
    default:
    return 0;
  };

if (! es)
   { return 0;
   }; // just in case

keepes = es;
if (dest) // just in case (2)
   { while (  (lgth > 0)
            &&(* dest ++ = * es ++)
           )
           { lgth --;
             outlen++;
           };
   } else
   { outlen = e_StrLen(es); // [wrap me!]
   };

XWriteConsoleText (refnum,"\x0A" "*error* ", 9);
XWriteConsoleText (refnum, keepes, outlen); // may clip.

return (outlen);   }
```

Otherwise we copy the string into the destination buffer. We check that the destination buffer still has space to accept the data (counting down using the submitted length; if this is incorrect, a buffer overflow will result), doing this until we reach the end of the ASCIIZ string in `es`. The sneaky C code above checks whether a NUL (zero character) was transferred using the `&&(* dest ...)` condition.

We then write the error message to the console. Ultimately, we return the length of the copied string, contained in `outlen`.

5 Error header file: ERRDEBUG.h

This is a fairly long file, mainly because of the need to define all of the numeric constants corresponding to the various error message strings. We make sure that ERRDEBUG_H hasn't been defined already (a standard precaution), so the first `#ifndef` ... `#endif` brackets the whole file. Here are all of those boring defines, which you will want to skip.

```
#ifndef ERRDEBUG_H
#define ERRDEBUG_H

#define ErTestingError           999
#define ErNullMemoryRequest      1000
#define ErNoListMemory           1001
#define ErFailInsertList         1002
#define ErListCreationFailed     1003
#define ErNoScratchBuffer        1004
#define ErPalmDbNotFound         1005
#define ErPalmDbNotOpen          1006
#define ErPalmLinkDbNotOpen      1007
#define ErNotClosePalmDb         1008
#define ErBadDbSizeCount         1009
#define ErNulInt16Write          1010
#define ErNulInt32Write          1011
#define ErScratchOverflow        1012
#define ErScratchOverflow2       1013
#define ErCannotCreateScratch    1014
#define ErTextComparisonMode     1015
#define ErFloatComparison        1016
#define ErFloatComparison2       1017
#define ErQueryNoIFdelimiter     1018
#define ErQueryNodeNoMake        1019
#define ErIntComparisonMode      1020
#define ErSQLTranslation1        1021
#define ErSQLNoWhere             1022
#define ErSQLBadWhere            1023

#define ErSQLselect              1025
#define ErSQLleftSpaceCondition  1026
#define ErSQLBadCondition        1027
#define ErUnknownSQLTest         1028
#define ErSQLComparator          1029
#define ErSQLComparator2         1030
#define ErSQLComparator3         1031
#define ErSQLComparatorStop      1032
#define ErSQLNoCfSpace           1033
#define ErKillLongUpQueryNode    1034
#define ErDefectNextPtr          1035
```

```
#define ErQueryTooShort          1036
#define ErQueryNoSpace           1037
#define ErSQLnoColumn            1038
#define ErSQLBadDatum            1039
#define ErBadSQLtype             1040
#define ErDatumXlateFailed       1041
#define ErSQLNoCol               1042
#define ErSQLNoQuote             1043
#define ErSQLlogic               1044
#define ErNoMemory               1045
#define ErNoMemLock              1046
#define ErMemoryNoUnlock         1047
#define ErMemoryNoFree           1048
#define ErBadControlStyle        1049
#define ErPalmDbCantClose        1050
#define ErFailNewRecord          1051
#define ErFailWriteRecord        1052
#define ErFailUnlockRecord       1053
#define ErFailReleaseRecord      1054
#define ErLockNullHandle         1055
#define ErUnlockNullHandle       1056
#define ErFailUnlockNulHandle    1057
#define ErNoFreeNulPtr           1058
#define ErFailFreeMemPtr         1059
#define ErCannotUnlockNulPtr     1060
#define ErFailUnlockMemPtr       1061
#define ErCantReleaseNulRec      1062
#define ErFailReleaseRec         1063
#define ErCloseNulDb             1064
#define ErFailCloseDb            1065
#define ErNulDbName              1066
#define ErFailMakeDb             1067
#define ErDelNulDb               1068
#define ErDelDbFail              1069
#define ErWriteNulRec            1070
#define ErWriteNulSrc            1071
#define ErFailRecCheck           1072
#define ErFailRecWrite           1073
#define ErSortNulDb              1074
#define ErSortNoRec              1075
#define ErSortNoFx               1076
#define ErNotRemoveNulRec        1077
#define ErFailRecRemove          1078
#define ErNulCopyDestin          1079
#define ErNulCopySrc             1080
#define ErNoScratch              1081
#define ErFailScratchUnlock      1082
#define ErFailScratchFree        1083
```

```
#define ErFailScratchClose       1084
#define ErDelNulPtr              1085
#define ErBadAscIntLen           1086
#define ErLongAscInt             1087
#define ErBadAscIntString        1088
#define ErSizeInt32              1089
#define ErFloatTooLong           1090
#define ErDateLen                1091
#define ErDateSeparator          1092
#define ErDateSeparator2         1093
#define ErImbalancedPars         1094
#define ErBadConditional         1095
#define ErBadConditional2        1096
#define ErLogicOverflow          1097
#define ErWhileOverflow          1098
#define ErFewParentheses         1099
#define ErQuoteImbalance         1100
#define ErBadLogic               1101
#define ErStoreCondition         1102
#define ErSelCleanFail           1103
#define ErDbCleanFail            1104
#define ErNoCol                  1105
#define ErBadTable               1106
#define ErNoDbTbl                1107
#define ErBadDbMatch             1108
#define ErBadColumnName          1109
#define ErColNotFound            1110
#define ErNoQueryOp              1111
#define ErBadNumericType         1112
#define ErBadType                1113
#define ErScratchWrite           1114
#define ErScratchTerminal        1115
#define ErInScratch              1116
#define ErScratchDest            1117
#define ErBadScratchDatum        1118
#define ErPullScratch            1119
#define ErScratchDest2           1120
#define ErSizeInt32A             1121
/* do we need the above? No! */
#define ErBadTblCREATE           1122
#define ErCreateTermRpar         1123
#define ErCREATENoMem            1124
#define ErCREATEfailed           1125
#define ErBadTblNameCreate       1126
#define ErCreateTableExists      1127
#define ErBadColNameCreate       1128
#define ErCreateBadColType       1129
#define ErCreateBadCol           1130
```

```
#define ErCreateNoComma          1131
#define ErCreateNoPrecision      1132
#define ErCreateTooManyCols      1133
#define ErCreateFailed           1134
#define ErCreateOpenFail         1135
#define ErFailMakeCreateHdr      1136
#define ErFailWriteCreateHdr     1137
#define ErCreateNotCloseHdr      1138
#define ErInsertNoTblname        1139
#define ErNoInsertTbl            1140
#define ErInsertNoHdrAccess      1141
#define ErInsColsNoRpar          1142
#define ErInsNoVALUES            1143
#define ErInsRemainingColData    1144
#define ErInsBadKey              1145
#define ErBadInsDatum            1146
#define ErInsExtraCol            1147
#define ErRowInsFailed           1148
#define ErDuplicateKey           1149
#define ErRowInsFailed2          1150
#define ErBadStringQuotes        1151
#define ErString1Quote           1152
#define ErStringTrunc            1153
#define ErIntNoSpace             1154
#define ErSizeInt32B             1155
#define ErNumericNoSpace         1156
#define ErNumericTooLong         1157
#define ErNumericTooLong2        1158
#define ErNoDateSpace            1159
#define ErBadDateFormat          1160
#define ErTimeNoSpace            1161
#define ErTimeBadFormat          1162
#define ErTimestampNoSpace       1163
#define ErTimestampFormat        1164
#define ErFloatNoSpace           1165
#define ErFloatFailedConversion  1166

#define ErAdjustFail1            1176
#define ErAdjustFail2            1177
#define ErAdjustFail3            1178
#define ErFailNewRecord2         1179
#define ErFailReleaseRecord2     1180
#define ErMakeBufFile            1181
#define ErOpenBufFile            1182
#define ErMakeBufRecord          1183
#define ErMakeStack              1184
#define ErOpenStack              1185
#define ErFailSTACKUnlock        1186
```

```
#define ErFailSTACKFree          1187
#define ErFailSTACKUnlock2       1188
#define ErFailSTACKFree2         1189
#define ErFailSTACKClose         1190
#define ErLibDb                  1200
#define ErLibLoad                1201
#define ErNoScriptStack          1202
#define ErSQLbadStmt             1300

#define ErSQLmultipleMatches     1301
#define ErSQLunknownStmt         1302
#define ErSQLinsertNoColumn      1303
#define ErNotWidget              1304
#define ErMenuFound              1305
#define ErTopMenu                1306
#define ErMakeDynamicForm        1307
#define ErNoWidgetData           1308
#define ErBadWidgetName          1309
#define ErFloatPop               1310
#define ErIntegerPop             1311
#define ErStringPop              1312
#define ErNoFunction             1313
#define ErBottomReturn           1314
#define ErScriptUnknown          1315
#define ErFunctionBuffer         1316
#define ErFunctionMemory         1317
#define ErNoFunctionDB           1318
#define ErOpenFunctionDB         1319
#define ErInsortFailed           1320
#define ErLowLeftCompare         1321
#define ErLowRightCompare        1322
#define ErHiLeftCompare          1323
#define ErHiRightCompare         1324
#define ErFreeFxPtr              1325
#define ErCloseFx                1326
#define ErFreeFxBuffer           1327
#define ErFreeFxBufrec           1328
#define ErCloseFxBuffer          1329
#define ErHexNumber              1330
#define ErColourLength           1331
#define ErColour                 1332
#define ErColourPop              1333
#define ErColourObject           1334
#define ErObjectTypeColour       1335
#define ErObjHackID              1336
#define ErEnableObj              1337
#define ErLabel                  1338
#define ErLabelObject            1339
```

```
#define ErFxStackOver           1340
#define ErBadMenuName           1341
#define ErNoMenuTitle           1342
#define ErNoXPush               1343
// #define ErPopForceX             1344
#define ErRunRecursion          1345
#define ErPopSetX               1346
#define ErNoV                   1347
#define ErFailEnterMenu         1348
#define ErPopInAsk              1349
#define ErPopTitleAsk           1350
#define ErScriptFx              1351
#define ErFailSQL               1352
#define ErAlertFail             1353
#define ErFailResolve           1354
#define ErFldTxt                1355
#define ErSqlTranslation        1356
#define ErPackFail              1357
#define ErSqlTableList          1358
#define ErRecFind               1359
#define ErNoDataTable           1360
#define ErFailFormat            1361
#define ErWarnLongDat           1362
#define ErMakeLocals            1363
#define ErOpenLocals            1364
#define ErFailLocalsUnlock      1365
#define ErFailLocalsFree        1366
#define ErFailLocalsClose       1367
// #define ErPopMenuStack          1368
#define ErBadMenuInteger        1369
#define ErBadColumnName2        1370
#define ErBadColumnItemCode     1371
#define ErWidgetNoDbCode        1372
#define ErPopIntConvertFail     1373
#define ErFailPopStringLen      1374
#define ErLocalVariableName     1375
#define ErLocalVariableSet      1376
#define ErGetLocalVariable      1378
#define ErFailFindFx            1379
#define ErRunPushScript         1380
#define ErSQLbadStmtSmall       1381
#define ErIniFxHandle           1382
#define ErFunctionInitialisation 1383
#define ErFailIniLocals         1384
#define ErDeepRecurStkShow      1385
#define ErBadLengthPeek         1386
#define ErFailSysPtrArray       1387
#define ErFailLockPtrArray      1388
```

```
#define ErFailPostProcess        1389

#define ErSQLbadStmt2            1390
#define ErSQLbadStmt3            1391
#define ErPopMenuStack1          1395
#define ErPopMenuStack2          1396

#define ErWidgetMake             1400
#define ErWidgetMake2            1401
#define ErBadUser                1402
#define ErShortBytecode          1403
#define ErBadListInt             1404

#define ErBadColEnabling         1410

#define ErNoFrom                 2000
#define ErNoWhere                2001
#define ErSqlPackFail            2002
#define ErDbNotFound             2003
#define ErCannotOpenDb           2004
#define ErDbNotMakeLink          2005
#define ErDbNoColumnLink         2006
#define ErDbBadColumnOffset      2007
#define ErFailLocateColumn       2008
#define ErFailMakeColumnNode     2009
#define ErTruncateMaxColSize     2010
#define ErRecursionInQuery       2011
#define ErStmtTooShort           2012
#define ErBadStmtLogic           2013
#define ErBadColNameLen          2014
#define ErBadColumnNodeCreation  2015
#define ErBadTranslation         2016
#define ErTranslationFailed      2017
#define ErBadDatumType           2018
#define ErDatumFormatFailed      2019
#define ErBadItemQuote           2020
#define ErBadJoinNode            2021
#define ErBadQueryList           2022
#define ErQueryItmFailed         2023
#define ErBadRecordCount         2024
#define ErBadDataTable           2025
#define ErJoinConditionOmitted   2026
#define ErFailedDatumPush        2027
#define ErNoJoinUp               2028
#define ErComparisonFailed       2029
#define ErNumberSeekFailed       2030
#define ErGenericSeekFailure     2031
#define ErBadLogicalOp           2032
```

```
#define ErFailMakeDbList         2033
#define ErRecQNotFound           2034
#define ErCollectionFailed       2035
#define ErDbFreeFailed           2036
#define ErNoSeparator            2037
#define ErDudTableNames          2038
#define ErPreformatFailed        2039
#define ErStackMarkFailed        2040
#define ErQueryFailed            2041
#define ErPostProcessFailed      2042
#define ErFailedPackConditions   2043
#define ErBadColumnIndex         2044
#define ErWontFit                2045
#define ErBadColOffst            2046
#define ErDefectiveRecord        2047
#define ErBufferFailed           2048
#define ErBadTableName           2049
#define ErNoSetCmd               2050
#define ErBadTableUp             2051
#define ErBadWhere               2052
#define ErBadQueryNodes          2053
#define ErNoNodeInUpdate         2054
#define ErBadFormatting          2055
#define ErBadUpdWhere            2056
#define ErFailRowUpd             2057
#define ErSQLleftSpace           2058
#define ErSqTinyBuffer           2059
#define ErSqBadIcomparator       2060
#define ErSqEqSpace              2061
#define ErSqLtBad                2062
#define ErSqBadOperator          2063
#define ErSqNo2ndOp              2064
#define ErSqGtBad                2065
#define ErSq2ManyRpar            2066
#define ErLogicTooDeep           2067
#define ErSqlWhile2Long          2068
#define ErSqImbalParens          2069
#define ErSqImbalQuotes          2070
#define ErSqBadLogic             2071
#define ErSqDudSort              2072
#define ErSqPreFull              2073
#define ErSqPreLogic             2074
#define ErSqJoinFail             2075



#define ErrVarNotFound           231
```

```
#define ScErStackIsNull             1
#define ScErStackstringIsNull       2
#define ScErFailedWriteStackStart   3
#define ScErFailedWriteStackTop     4
#define ScErFailedWriteStackMax     5
#define ScErFailedWriteStackFlags   6
#define ScErFailedWriteSSStart      7
#define ScErFailedWriteSSTop        8
#define ScErFailedWriteSSMax        9
#define ScErFailedWriteSSFlags      10
#define ScErShortPushFailed         16
#define ScErPushLengthRewrite       18
#define ScErPushoffsetRewrite       19

#define ScErLongPushFailed          17
#define ScErPushLong                20

#define ScErPushLongTop             21
#define ScErPushLength              22
#define ScErPushTop                 23
#define ScErPopNoStack              24
#define ScErPopDestination          25
#define ScErPopStackEmpty           26
#define ScPopFull                   27
#define ScPopFullLong               28
#define ScErPopStack                29
#define ScErCompoundNoFit           30
#define ScErCompoundBadSsTop        31
#define ScErCompoundStackWrite      32
#define ScErCompoundStackWriteBig   33
#define ScErCompoundBadLen          34
#define ScErCompoundBadOffset       35
#define ScErCompoundInsertions      36
#define ScErCompoundStart           37
#define ScErCompoundTop             38
#define ScErFloat0                  39
#define ScErNumericSpace            42
#define ScErTimestamp1              43
#define ScErTimestamp2              44

#define ScErEncodeIntegerMax        59
#define ErScPushNumberBad           60
#define ErScPushNumberFull          61
#define ErScPushNumberFullLong      62
#define ScErTerminalQuote           71
#define ScErLeftParenthesis         72
#define ScErAx                      73
#define ScErBx                      74
```

```
#define ScErCx                    75
#define ScErDx                    76
#define ScErEx                    77
#define ScErFx                    78
#define ScErGx                    79
#define ScErHx                    80
#define ScErIx                    81
#define ScErJx                    82
#define ScErKx                    83
#define ScErLx                    84
#define ScErMx                    85
#define ScErNx                    86
#define ScErOx                    87
#define ScErPx                    88
#define ScErQx                    89
#define ScErRx                    90
#define ScErSx                    91
#define ScErTx                    92
#define ScErUx                    93
#define ScErVx                    94
#define ScErWx                    95
#define ScErXx                    96
#define ScErYx                    97
#define ScErZx                    98
#define ScErOther                 99
#define ScErUnTime               100
#define ScErPeek                 101

#define ScErSkipEmpty            125
#define ScErBooleanEmpty         126
#define ScErAndEmpty             127
#define ScErAndArgs              128
#define ScErAndArgs2             129
#define ScErOrEmpty              130
#define ScErOrArgs               131
#define ScErOrArgs2              132
#define ScErNotEmpty             133
#define ScErNotArgs              134
#define ScErNotArgs2             135

#define ScErBuryEmpty            145
#define ScErBurySpace            146
#define ScErBuryFull             147
#define ScErDigEmpty             148
#define ScrErDigFull             149
#define ScrErDigSpace            150
#define ScErDiscardEmpty         151
#define ScErGtEmpty              152
```

```
#define ScErGtArgs                  153
#define ScErInEmpty                 154
#define ScErInType                  155
#define ScErTestnumEmpty            156
#define ScErLenEmpty                157

#define ScErNowFull                 160
#define ScErTestNullEmpty           161
#define ScErSecEmpty                162
#define ScErSecType                 163
#define ScErSecEarly                164
#define ScErSwopFull                165
#define ScErSwopEmpty               166
#define ScErSwopLong                167
#define ScErListFull                168
#define ScErListFull2               169
#define ScErMap                     170
#define ScErRegexEmpty              171
#define ScErBool                    172
#define ScErReplace                 173

#define ScErBadHexDigit             179
#define ScErBadReadMode             180
#define ScErHexColour               181
#define ScErBadUReadLen             182
#define ScErNumSillyScale           183
#define ScErDoFloatMode             184
#define ScErNegativeInteger         187

#define ScErBadDepth                215
#define ScErStrangeBottom           216
#define ScErPopNoDest               217
#define ScErNotEnoughData           219

#define  ScErLowLeftCompare         220
#define  ScErLowRightCompare        221
#define  ScErHiLeftCompare          222
#define  ScErHiRightCompare         223
#define ErScIniFx1                  225
#define ErScIniFx2                  226
#define ErScIniFx3                  227
#define ErScIniFxHandle             228

#define ErNoLocalBuffer             229
#define ErrGetVarName               230
#define ErrGetFullStack             232
#define ErrVarBadName               233
#define ErrVarAlreadyExists         234
```

```
#define ErrVarsFull                235
#define ErrLocalStringFull         236
#define ScErMakePop                237

#define ScERnotX                   257
#define ScErNoSpace                258
#define ScErBadInsertionCount      259
#define ScErInsertImbalance        260

#define ScListEmpty                261
#define ScListSeparator            262

#define ScErVarCheckEmpty          267
#define ScErVarType                268
#define ScErIntegerWrite0           40
#define ScErIntegerWrite1           41
#define ScErIntegerWrite2          269
#define ScErIntegerWrite4          270



#define ScErEncode                 124
#define ScErEncodeNil              102
#define ScErUnquoted               103
#define ScErOneQuote               104
#define ScErTruncated              105
#define ScErIntegerSpace           106
#define ScErIntegerBad             107
#define ScErNumericScale           108
#define ScErBadNumeric1            175
#define ScErBadNumeric2            176
#define ScErBadNumeric3            177
#define ScErBadNumeric4            178
#define ScErDateSpace              110
#define ScErDate                   111
#define ScErDateTerminal           112
#define ScErDateFormat             113
#define ScErTextComparisonMode     194
#define ScErIntComparisonMode      195
#define ScErWarnSillyEqual         196
#define ScErWarnNotEqual           197
```

```
#define ScErMarkEmpty               9173
#define ScErMarkArg                 9174
#define ScErMarkMax                 9239
#define ScErMarkMin                 9240
#define ScErNoMax                   9241
#define ScErNoMin                   9243
#define ScErNoDistinct              9244
#define ScErBadDistinct             9245

#define ScErBadSortItems            9246
#define ScErNoSort                  9247
#define ScErBadSortMode             9248
#define ErSortOhBugger              9249
#define ScErSortMajor               9250
#define ScErSortCorruptStack        9251

#define ScErBypass                  9252
#define ScErSQLsortorder            9253
#define ScErFullSQL                 9254
#define ScErSQLimbalance            9255
#define ScErBadParens               9256

#define ScErSQLleftSpaceCondition   9198
#define ScErSQLBadCondition         9199
#define ScErUnknownSQLTest          9200
#define ScErSQLComparator           9201
#define ScErSQLComparator2          9202
#define ScErSQLComparator3          9203
#define ScErSQLComparatorStop       9204
#define ScErSQLNoCfSpace            9205
#define ScErImbalancedPars          9206
#define ScErBadConditional          9207
#define ScErBadConditional2         9208
#define ScErLogicOverflow           9209
#define ScErWhileOverflow           9210
#define ScErFewParentheses          9211
#define ScErQuoteImbalance          9212
#define ScErBadLogic                9213
#define ScErStoreCondition          9214

#define ScErBadTypeCompare          9242
#define  ScErInsortFailed           9224

#define ScErrDebugFull              9225
#define ScErrDebugCopy              9226
#define ScErrRubbish1               9227
```

```
#define ErBaadFloat                 9271



#define ScErAddEmpty                3000
             // "Addition err: no args"
#define ScErAddArgs1                3001
             // "Addition err: bad integer"
#define ScErAddArgs2                3002
             // "Addition err: bad float"
#define ScErAddArgs3                3003
             // "Addition err: unsupported type"
#define ScErAddIOver                3004
             // "Addition err: our overflow"

#define ScErSubEmpty                3005
             // "Subtract err: no args"
#define ScErSubArgs1                3006
             // "Subtract err: bad integer"
#define ScErSubArgs2                3007
             // "Subtract err: bad float"
#define ScErSubArgs3                3008
             // "Subtract err: unsupported type"
#define ScErSubIUnder               3009
             // "Subtract err: underflow"


#define ScErMulEmpty                3010
   // "Multiply err: no args"
#define ScErMulArgs1                 3011
             // "Multiply err: bad integer arg"
#define ScErMulArgs2                 3012
             // "Multiply err: bad float"
#define ScErMulArgs3                 3013
             // "Multiply err: unsupported type"
#define ScErMulOver1                 3014
             // "Multiply err: overflow"
#define ScErMulOver2                 3015
             // "Multiply err: our overflow"

#define ScErNegEmpty                3016
             // "Negation err: no args"
#define ScErNegArgs                 3017
   // "Negation err: bad arguments"
```

```
#define ScErDivEmpty                3018
       // "Divide err: no args"
#define ScErDivArgs1                3019
             // "Divide err: bad integer dividend"
#define ScErDivArgs2                3020
             // "Divide err: bad float dividend"
#define ScErDivArgs3                3021
             // "Divide err: unsupported numeric"
#define ScErDivArgs4                3022
            // "Divide err: bad numeric arg(s)"
#define ScErDivOver                 3023
             // "Divide err: divisor zero"


#define ScErModEmpty                3024
             // "Modulo err: no args"
#define ScErModArgs1                3025
             // "Modulo err: no args"
#define ScErModArgs2                3026
             // "Modulo err: unsupported args"
#define ScErModOver                 3027
             // "Modulo err: divide by 0"

#define ErPushNoStack              3028
#define ErPushBadLength            3029
#define ErPushStackFull            3030
#define ErPushFailed               3031
#define ErLongPushFull             3032
#define ErLongPushFail             3033

#define ErEncodeIntegerEmpty       3034
#define ErEncodeIntegerType        3035
#define ErEncodeIntegerOverflow    3036

#define SqErEncodeNil             3102
#define SqErUnquoted              3103
#define SqErOneQuote              3104
#define SqErTruncated             3105
#define SqErIntegerSpace          3106
#define SqErIntegerBad            3107
#define SqErNumericScale          3108
#define SqErBadNumeric1           3175
#define SqErBadNumeric2           3176
#define SqErBadNumeric3           3177
#define SqErBadNumeric4           3178
#define SqErDateSpace             3110
#define SqErDate                  3111
#define SqErDateTerminal          3112
```

```
#define SqErDateFormat            3113
#define SqErTimeSpace             3114
#define SqErTime                  3115
#define SqErTimeTerminal          3116
#define SqErTimeFormat            3117
#define SqErStampSpace            3118
#define SqErStamp                 3119
#define SqErStampTerminal         3120
#define SqErStampFormat           3121
#define SqErFloatSpace            3122
#define SqErFloatFormat           3123
#define SqErEncode                3124

#define ScErSplitEmpty            3130
#define ScErSplitSeparator        3131
#define ScErSplitLen              3132
#define ScErSplitResolve          3133

#define ScErDecodeNumeric          3134
#define ScErDecodeTimestamp        3135
#define ScErDecodeTime             3136
#define ScErDecodeInteger          3137

#define ScErCutEmpty              3138
#define ScErCut                   3139

#define ScErResolveEmptyStack      3150
#define ScErResolveSpace           3151
#define ScErResolveCount           3152
#define ScErResolveInsertions      3153
#define ScErResolveFull            3154

#define ScBadDecode                3144
#define ScErDecodeNoSpace          3145
#define ScErDecodeCompound         3146
#define ScErDecodeType             3147
#define ScErDecodeDate             3148
#define ScErDecodeFloat            3149


#define ScErCopyEmpty              3155
#define ScErCopyOverflow           3156
#define ScErCopyFull               3157

#define ScErEncodeFloatEmpty       3158
#define ScErFloatLength            3159
#define ScErEncodeFloatType        3160
```

```
#define ScErEncodeTsEmpty       3161
#define ScErTsLength            3162
#define ScErEncodeTsType        3163

#define ScErEncodeDateEmpty       3164
#define ScErDateLength            3165
#define ScErEncodeDateType        3166

#define ScErEncodeTimeEmpty       3167
#define ScErTimeLength            3168
#define ScErEncodeTimeType        3169

#define ScErJoinEmpty             3170
#define ScErRepairFailed          3171


#define iColumnBeingSought       5000
#define iNodeDetails             5001
```

Okay, now down to nuts and bolts. The following library stuff is all swiped from Ian Goldberg's examples which come with GCC for PalmOS, so you may wish to consult the file *shlib.html* for a better description of what is going on.[2] The basic idea is this: the pre-defined constants sysLibTrapOpen and sysLibTrapClose are used to associate our functions contained in the initial C code (Section 2) with system calls on library startup. The use of ERRDEBUG_TRAP is a frill. We might simply replace:

```
Int16 ErrorString (UInt16 refnum, Char * dest, Int16 lgth,
                   Int16 errnum)
ERRDEBUG_TRAP(ERRDEBUGString_trapno);
```

...with:

```
Int16 ErrorString (UInt16 refnum, Char * dest, Int16 lgth,
                    Int16 errnum)
  SYS_TRAP(sysLibTrapCustom+0)
```

Here is the code referring to the various functions fleshed out in the library C code:

```
#define ERRDEBUGString_trapno sysLibTrapCustom+0
#define ERRDEBUGPassConsole_trapno sysLibTrapCustom+1
```

[2] There's a lot of devious and confusing stuff on the Internet!

```
#define ERRDEBUGErrorWrite_trapno sysLibTrapCustom+2

#ifndef ERRDEBUG_TRAP
#define ERRDEBUG_TRAP(trapno) SYS_TRAP(trapno)
#endif

Err ERRDEBUGOpen (UInt16 refNum)
ERRDEBUG_TRAP(sysLibTrapOpen);

Err ERRDEBUGClose (UInt16 refNum, UInt16 *numappsP)
ERRDEBUG_TRAP(sysLibTrapClose);

Int16 ErrorString (UInt16 refnum, Char * dest, Int16 lgth,
                   Int16 errnum)
ERRDEBUG_TRAP(ERRDEBUGString_trapno);

Int16 ErrPassConsole (UInt16 refnum, UInt16 cons)
ERRDEBUG_TRAP(ERRDEBUGPassConsole_trapno);

void ErrorWrite(UInt16 refnum, Int16 e)
ERRDEBUG_TRAP(ERRDEBUGErrorWrite_trapno);
#endif
```

We still need some bits and bobs to create the library.

6 The Makefile

Here's the all-important makefile. Makefiles for GCC for PalmOS libraries are a bit of a black art, and things have changed considerably over the past few years.[3] The new program m68k-palmos-stubgen is used to turn the def file (Section 7) into something which can be handled by GCC. The `-nostartfiles` option is important for compilation. Heck, I'm not sure that I understand all of this, but it works!

```
CC = m68k-palmos-gcc -Wall -g -O2 -mdebug-labels
AS = m68k-palmos-as

all: ERRDEBUG-syslib.prc

ERRDEBUG-syslib.prc: ERRDEBUG.def ERRDEBUG
build-prc -o $@ ERRDEBUG.def ERRDEBUG

ERRDEBUG_objs = ERRDEBUG.o ERRDEBUG-dispatch.o
ERRDEBUG: $(ERRDEBUG_objs)
$(CC) -nostartfiles -o $@ $(ERRDEBUG_objs)

ERRDEBUG.o: ERRDEBUG.c ERRDEBUG.h

ERRDEBUG-dispatch.o: ERRDEBUG-dispatch.s

ERRDEBUG-dispatch.s: ERRDEBUG.def
m68k-palmos-stubgen ERRDEBUG.def

clean:
rm -f *.o *.prc *-dispatch.? ERRDEBUG
```

In the final section we examine the tiny DEF file on which the above is based.

[3] Previously you had to tweak nasty raw assembler code!

7 The DEF file: ERRDEBUG.def

Here we simply list the various routines *in order*, without any arguments. Before we do so, we have to state the name of the library, and the creation identifer (we use the arbitrary ‘SqEr’, which is bound to cause problems sooner or later).

The first *four* functions always have the same functionality (RTM), so if you don’t define any of them you *must* put in a stub — now we understand where the `nothing` routine fits in! After the first four, come the user-defined functions corresponding to sysLibTrapCustom+0, sysLibTrapCustom+1 and so on.

```
syslib { "ERRDEBUG Library" ErLi }

export {
  ERRDEBUGOpen ERRDEBUGClose nothing nothing
  ErrorString ErrPassConsole ErrorWrite
  }
```

And that’s it for the error library.