Analgesia Database:
Console Driver Program for PalmOS PDA

Version 0.90

J.M. van Schalkwyk

February 27, 2009

Contents

1 The CPP file: osbox.cpp

```
// ============================================================================
//                    OSBOX: THIS IS THE MAIN C++ FILE
//                 Display console on Palm, and allow scrolling
// ============================================================================

#include <PalmOS.h>
#include "fred.hpp"
#include "osbox.h"
#include "../console/CONSOLE.h"


// NO GLOBALS:

// ----------------------------------------------------------------------------
UInt32 PilotMain(UInt16 cmd, void *cmdPBP, UInt16 launchFlags)
{ Boolean more = 1;

  fred * myFred;
  // make instance of fred class:
  myFred = OsboxMakeFred();                  // make instance of Fred without usi
  if (! myFred)
     { return 0;                          // fail
     };

  if (cmd == sysAppLaunchCmdNormalLaunch)
      {  if ( OsbStartApplication(myFred) )
            { while (more)
                { OsbEventLoop (cmd, cmdPBP, launchFlags, myFred);
                  more = OsbStopApplication(myFred);
                };
            }
      };
return 0;                        //
}


// ----------------------------------------------------------------------------
static int OsbStartApplication(fred * myFred)
{ UInt32 romversion;
  Int16  fail;

// --------- check version, if ok then make instance of pgm -----------------
  FtrGet(sysFtrCreator, sysFtrNumROMVersion, &romversion);  // get version.
  if (romversion < sysMakeROMVersion(3,5,0,sysROMStageRelease,0))
     { WinDrawChars("Bad ROM version!", 16, 20, 50);  // *SYSTEM*
       return 0;                          // fail
```

```
     };

UInt16 consolecode = 0;

//  load libraries:
  consolecode = LoadLibrary ("CONSOLE Library", 'CnLi');
  if (consolecode)
     { CONSOLEOpen(consolecode);
     } else
     { DoDebug("? Console lib error, code ",consolecode);
     };

  fail = (myFred->Kickoff(consolecode));
  if (fail)
     { DoDebug("Err ini failed", fail);     // 0 signals success!
     };


  // here intialise... <====================================

  fail = myFred->CreateConsole();
  if (fail)
     { DoDebug("Cons failed", fail);
     };

  return 1;                                 // ok.
}


// ---------------------------------------------------------------------------
static Boolean OsbStopApplication(fred * myFred)
{

  // -----------------------------------------
  // here might confirm that user wants to leave
  // -----------------------------------------

  OsbKillMe(myFred);

  return 0; // signal "QUITTING"..
}



// ===========================================================================
// PALMOS EVENT PROCESSOR:

// ---------------------------------------------------------------------------
static UInt32 OsbEventLoop(UInt16 cmd, void *cmdPBP, UInt16 launchFlags, fred * my
```

```
{   EventType   e;
    UInt16      err;
        // handle events:-
          do { EvtGetEvent(&e, evtWaitForever);            // *SYSTEM*
               if (!SysHandleEvent(&e))                    // *SYSTEM*
                  { if (!MenuHandleEvent(NULL, &e, &err))  // *SYSTEM*
                       { if (! OsbHandleEvent(&e, myFred))       // our own fx.
                            { FrmDispatchEvent(&e);        // *SYSTEM*
                  }    }    }
             } while (e.eType != appStopEvent);            // until termination.
  return 0;
}


// ------------------------------------------------------------------------------
Int16 OsbHandleEvent(EventPtr e, fred * myFred)
{   FormPtr     frm;
    Int16       formId;
    Int16       ctlid;
    Int16       i;
    Int16       fail;
    Boolean     handled = false;

    if (e->eType == frmLoadEvent)
       {    // Load the form resource specified in the event,
            // then activate the form.
          formId = e->data.frmLoad.formID;             // get the form ID number
          frm = FrmGetFormPtr(formId);                 // *SYSTEM*
          if (! frm)
             { // display error?
             };
          FrmSetActiveForm(frm);                       // *SYSTEM*
          FrmSetEventHandler(frm, OsBoxHandleEvent); // *SYSTEM*
          handled = true;
        };

    if (e->eType == frmOpenEvent)
       {
          formId = e->data.frmLoad.formID;             // get the form ID number
          frm = FrmGetFormPtr(formId);                 // *SYSTEM*
          FrmDrawForm(frm);                            // *SYSTEM*
          handled = true;
        };

    // CTRL SELECTION.....................................................
    // The following only works for items designated by Palm as 'controls'. It wil
    // not work for text fields, but does work for buttons, checkboxes,
    // pushbuttons and poplists.
```

```
if (e->eType == ctlSelectEvent)   // =9: better to restyle as select..case?!
   {

     // ==============> HERE TRAP CLICK ON BUTTON (etc) <=====================
     ctlid = e->data.ctlEnter.controlID;

     // check whether Quit button was pressed:
     //
     if (ctlid == But2Quit)
        { e->eType = appStopEvent;    // force 'quit' event!
                                      // don't need to alter other event prope
          return 0;                   // 'unhandled', so now will handle it!
        };

     if (ctlid == But6Kill) // 'delete PAIN DB caching' button
        {
          fail = myFred->KillPainCache();
          if (fail)
             { DoDebug("Deletion failed, code ", fail);
             };
        };

     if (ctlid == But7Clear) // clear console data (delete file CONSOLE.pdb)
        {
          fail = myFred->EradicateConsole();
          if (fail)
             { DoDebug("Error, code ", fail);
             };
        };

     if (ctlid == But1Back)           // [<] back button
        { myFred->BackOneLine();
        };

     if (ctlid == But3Fwd)            // [>] forward button
        { myFred->ForwardOneLine();
        };

     if (ctlid == But4BBack)          // [<<] 7 lines forward
        { i=7;
          while (i > 0)
            { myFred->BackOneLine();  // [>>] 7 back
              i --;
            };
        };
     if (ctlid == But5FFwd)
        { i=7;
          while (i > 0)
```

```
                { myFred->ForwardOneLine();
                  i --;
                };
            };
       };
    // end CTRL SELECTION.....................................................

  if (e->eType == fldEnterEvent)
       {  // for fldenterevent see: http://www.palmos.com/dev/support/docs/palmos/
       };

    return handled;
}


// ---------------------------------------------------------------------------
static Boolean OsBoxHandleEvent(EventPtr event)
{ Boolean     handled = false;

  return(handled);
}



// ===========================================================================

static void DoDebug(const Char * msg, Int16 nmbr)
{
  MemHandle memH;
  MemPtr pStr;
  memH = MemHandleNew(maxStrIToALen);           //*system*. maxStrIToALen is Palm
  if (! memH)
     { return;                                  // fail.
     };
  pStr = MemHandleLock(memH);                   // *system*
  if (! pStr)                                   //
     { return;                                  // fail.
     };
  StrIToA((Char *)pStr, nmbr);                  //
  FrmCustomAlert(DebugAlert,msg,(Char *)pStr,"");//
  if (MemPtrUnlock (pStr) != errNone)           //
     { return;
     };
  if (MemPtrFree (pStr) != errNone)             //
     { return;
     };
 return;
}
```

```
// ------------------------------------------------------------------------------
static UInt16 LoadLibrary (Char * libname, Int32 SILLYCODE)
{
  UInt16 libcode=0;
  LocalID lid;
  UInt32 dbtype=0;
  lid = DmFindDatabase (0, libname);
  if (! lid)
     { DoDebug ("No LIB found",0);
       return 0;
     } else
     { DmDatabaseInfo (0, lid, 0, 0, 0, 0, 0, 0, 0, 0, 0, &dbtype, 0);
     };
// DoDebug("DB type",dbtype);
if (! dbtype)
   { return 0;
   };

Err error;
  error = SysLibFind(libname, &libcode);  // is library already loaded?
  if (error == errNone)
     { DoDebug("Already loaded",libcode);
       return libcode;
     };
  // DoDebug("Installing library", error);
  error = SysLibLoad (dbtype, SILLYCODE, &libcode); //
  if (error != errNone )                          // cumbersome. just 'if (error)
     { DoDebug("Failed to load library", error);
       return 0;
     };
  return libcode;
}


// ------------------------------------------------------------------------------
static fred * OsboxMakeFred ()    // accepts handle of open library!
{ MemHandle   memH=0;
  MemPtr      memP=0;
  fred *      myFred;
                      //
  memH = MemHandleNew(sizeof(fred));
  if (! memH)
     { return 0; // fail
     };
  memP = MemHandleLock(memH); // *SYSTEM*
  if (! memP)
     { return 0;
```

```
     };  //memP pointer to locked new memory.

  myFred = (fred *) memP;
  // end of sql creation section.

return myFred; // success
}


// ----------------------------------------------------------------------------
static Int16 OsbKillMe(fred * myFred)
{ MemPtr killOsb;
  Int16 fail;

  fail = myFred->KillConsole();
  if (fail)
     { DoDebug("? Console not closed",fail);
     };

  fail =  myFred->Cleanup();
  if (fail)
      { DoDebug("? Cannot close console library, code: ", fail);
      };

  killOsb = (MemPtr) myFred;
  if ( (MemPtrUnlock (killOsb) != errNone)
     ||(MemPtrFree (killOsb) != errNone)
     )
   { DoDebug("? Cannot free memory", 0);
     return -1;  // fail
   };

  return 0;  // ok.
}


// ============================================================================
// Static callback fx:

// ----------------------------------------------------------------------------
// 1.




////////////////////////////////////////////////////////////////////////////
// ============================================================================
//                                  end of document
```

2 Header file: osbox.h

```
//============================================================================

#include "osboxrcp.h"

static int OsbStartApplication(fred * myFred);
static Boolean OsbStopApplication(fred * myFred);
static UInt32 OsbEventLoop(UInt16 cmd, void *cmdPBP, UInt16 launchFlags, fred * my
Int16 OsbHandleEvent(EventPtr e, fred * myFred);
static Boolean OsBoxHandleEvent(EventPtr event);
static void DoDebug(const Char * msg, Int16 nmbr);
static fred * OsboxMakeFred ();

static Int16 OsbKillMe(fred * myFred);
static UInt16 LoadLibrary (Char * libname, Int32 SILLYCODE);

//============================================================================
```

3 The Makefile

```
CC = m68k-palmos-gcc
CFLAGS = -Wall -pedantic -g -O2 -fno-exceptions -fno-rtti

all: osbox.prc
ls -l osbox.prc

osbox.prc: osbox.def osbox *.bin
build-prc -o $@ osbox.def *.bin osbox

*.bin: osbox.rcp osbox.h osboxrcp.h
pilrc osbox.rcp

osbox: osbox.cpp osbox.h osboxrcp.h fred.hpp
$(CC) $(CFLAGS) -o $@ osbox.cpp

clean:
rm -f *.grc *.prc *.bin
```

4 The DEF file: osbox.def

```
app { "console" JoVs }
```

5 fred: fred.hpp

```
// ============================================================================
//                                c++ class: fred
// ============================================================================
// We start with a few includes and defines:

#include <PalmOS.h>
#include "../console/CONSOLE.h"
#include "osboxrcp.h"

#define MAXtextsize 512

#define formX 2
#define formY 2
#define formW 156
#define formH 156
#define botLn 144


// ============================================================================
//                                 CLASS DEFINITION:
//----------------------------------------------------------------------------

class fred {
public:
         fred () { }; //INLINE CONSTRUCTOR
         ~fred () { }; // DESTRUCTOR (stubs)

Int16    Kickoff(UInt16 consolecode);
Int16    Cleanup();

Int16    CreateConsole();
Int16    KillConsole();
Int16    BackOneLine ();
Int16    ForwardOneLine ();
Int16    KillPainCache();
Int16    EradicateConsole();


private:

MemHandle   hMEMTEXT;                          // text buffer used by console di
Char *      pMemText;                          // corresponding pointer. Eugh.

UInt16      CONLIBCODE;
FormPtr     CONSOLEFORM;
FieldType * MAINTEXTFIELD;
```

```
Int16       LINEPOSITION;
Boolean     ATSTART;                              // can test if at start of text!

void        ReplaceCharacters(Char * pMemText, Int16 tlen, Char t , Char i);

Int16    InsertControl(  FormPtr * frmP, Int16 ctlcode, Int16 ctlstyle,
                         Char * ctltitl,
                         Int16 x, Int16 y, Int16 w, Int16 h);
Int16    MakeMainText ( FormPtr * frmP); // create main text area

Int16    SetMainText (); // set string to display

// Int16    Redisplay();

// utility functions:

Int16    xCopy (Char * dest, Char * xsrc, Int16 cnt);
Int16    xFill  (Char * p0, Int16 slen, Char c);

};

//-------------------------------------------------------------------------------
//                                 END OF CLASS DEFINITION
// ============================================================================




//-------------------------------------------------------------------------------
// Kickoff: initialise variables, set aside memory buffer(s)
//

Int16 fred::Kickoff(UInt16 consolecode)
{
  CONSOLEFORM=0;
  MAINTEXTFIELD=0;  // clean (good)
  LINEPOSITION=-1;  // number of lines back in console : default is last line
  ATSTART = 0;      // don't know this on entry!

  CONLIBCODE = consolecode;

  hMEMTEXT = MemHandleNew(MAXtextsize);
  if (! hMEMTEXT)
     { return -1;                                // fail.
     };
  pMemText = (Char *) MemHandleLock(hMEMTEXT);

  return 0;
}
```

```
//-------------------------------------------------------------------------
// Cleanup. On exit, do the opposite of Kickoff. Like a destructor.

Int16 fred::Cleanup()
{ Int16 fail;
  UInt16 concnt;

  MemPtrUnlock(pMemText);                       // could check for success ???
  MemHandleFree(hMEMTEXT);                      // free up memory
    // a bit clumsy.

  fail = CONSOLEClose (CONLIBCODE, &concnt);
    // (if other pgms are still talking, then fail will be 1)
  if (fail)
     { return fail;
     };
  if (concnt)     // fail : still instances open ??
     { return -1; // hmm better to have 'unique' error code
     };
  fail = SysLibRemove(CONLIBCODE);  // SysLibRemove returns zero on success
     // ideally this should be done in osbox.cpp, rather than here (ugly).
  if (fail)
     { return fail;
     };

  return 0; // success.
};


// =========================================================================
// Console functions:

//-------------------------------------------------------------------------
// Create PalmOS form that is the 'console'

Int16 fred::CreateConsole ()
{
  Int16     frmcode;
  UInt16    frmid;

  frmcode = 10001;   // arbitrary

  CONSOLEFORM = FrmNewForm( frmcode, "Console", formX, formY, formW, formH,
                    true,         // modal ie. keeps death-like grip on the focus!
                    0, 0, 0);     // no ID for default action, online help, menu r
                                  // later might need to allow for the three, make
  if (! CONSOLEFORM) { return -1; };      // fail
```

```
  // HERE SHOULD ADD text area, BUTTONS ETC...

  InsertControl( &CONSOLEFORM, But1Back, buttonCtl, "<", 3, botLn, 17, 10);
  InsertControl( &CONSOLEFORM, But4BBack,buttonCtl, "<<", 25, botLn, 20, 10);
  InsertControl( &CONSOLEFORM, But2Quit, buttonCtl, "Quit", 55, botLn, 40, 10);
  InsertControl( &CONSOLEFORM, But5FFwd, buttonCtl, ">>", 104, botLn, 20, 10);
  InsertControl( &CONSOLEFORM, But3Fwd,  buttonCtl, ">", 129, botLn, 17, 10);
  InsertControl (&CONSOLEFORM, But6Kill, buttonCtl, "Kill", 3, 1, 20, 10);
  InsertControl (&CONSOLEFORM, But7Clear, buttonCtl, "Clear", 120, 1, 20, 10);

  MakeMainText(&CONSOLEFORM); // create main text area.
  SetMainText();

  // to update text: need : FldDrawField (MAINTEXTFIELD);
  //

  frmid = FrmGetFormId(CONSOLEFORM);
  if (! frmid) { return -2; };    // fail (unlikely)

  FrmGotoForm(frmid); // test-->go, deleting old?!
//  FrmPopupForm(frmid);          // queues frmLoadEvent, BUT leaves current form

  return 0;                                    // ok = 0.
}


//-------------------------------------------------------------------------------
// Get rid of console form when no longer needed:

Int16 fred::KillConsole()
{
  if (! CONSOLEFORM)
     { return -1; // fail
     };

  FldSetText(MAINTEXTFIELD, 0, 0, 0);           // free association between text,
  FrmEraseForm(CONSOLEFORM);    // PalmOS seems to want this first, before FrmDele
  FrmDeleteForm(CONSOLEFORM);   // is void fx!
  return 0; // ok.
}
```

The following little routine substitutes all occurrences of a single byte target character (t) with a given single byte value (i), within an entire text string.

```
void  fred::ReplaceCharacters(Char * pMemText, Int16 tlen, Char t , Char i)
{
  Int16 of = 0;

  while (of < tlen)
    { if ( * (pMemText+of) == t)
         { * (pMemText+of) = i;
         };
      of ++;
    };


}



// ============================================================================
// Form-related functions.

//-----------------------------------------------------------------------------
// Insert a control into a form (Buttons, usually)

Int16 fred::InsertControl(  FormPtr * frmP, Int16 ctlcode, Int16 ctlstyle,
                           Char * ctltitl,
                           Int16 x, Int16 y, Int16 w, Int16 h)
{
   ControlType * ctp = 0;  //
   ctp = CtlNewControl ( (void **)frmP, // pointer to pointer!
         ctlcode,
         (enum controlStyles) ctlstyle, // eg buttonCtl for a button
         ctltitl, x, y, w, h,
         (enum fontID) 0, 0, 0);   // font id, group, resize

  return 0;
}


//-----------------------------------------------------------------------------
// Create the main text 'field': then associate it with our text buffer (hMEMTEXT)

Int16 fred::MakeMainText ( FormPtr * frmP)
{
   if (!hMEMTEXT)
      { return -1;
      };

   MAINTEXTFIELD = FldNewField ((void **) frmP, Fld0,            // id of the fiel
                       1, 10, 155, 134,            // x,y,w,h
                       (enum fontID) 0,
```

```
                      MAXtextsize-1,            // maximum characters, -1 as asci
                      0,                        // not editable
                      0,                        // underlined,
                      0,                        // singleLine,
                      0,                        // dynamicSize,
                      leftAlign,                // justification,
                      true,                     // autoShift,
                      0,                        // hasScrollBar ???,
                      0);                       // numeric (any char is ok thus 0
  // we must cast to (void **) to get the bugger to work!
  if (! MAINTEXTFIELD)
     { return -2; // fail
     };

///  FldSetText(MAINTEXTFIELD, hMEMTEXT, 0, MAXtextsize); // associate field and t

  return 0; //ok.
}


//-------------------------------------------------------------------------------
// Copy text into the main text buffer:
//   the text is obtained from the buffer within the console library (!) using Con
//   (actually, within a file).

Int16  fred::SetMainText ()   // set string to display
{
  Int16 slen=0;
  if (! pMemText)                               // braces+
     { return -1;
     };

  xFill(pMemText, MAXtextsize-1, 0);  // in case of stuffup!

//  slen = ConRead (CONLIBCODE, pMemText, MAXtextsize-1, 0); // get max of MAXtext
          // last 0 arg signals normal mode i.e. read from end.
  // we have a problem here if 0x0 was written to console. One solution would be
  // to here scan for this and replace it with eg a blank!

  slen = ConRead (CONLIBCODE, pMemText, MAXtextsize-1, LINEPOSITION); //
  // alternative and more useful formulation: get maximum of k most recent lines
  //  from console, with max length of MAXtextsize-1. -k signals 'give me k recent
  if (slen < 0)
     { ATSTART = 1;                         // know it's futile to go further back
       slen = -slen;
     };
  *(pMemText+slen) = 0x0;                   // asciiz (eugh)

  // 2005-12-24:
```

```
  // let's try to replace spaces with numeric spaces to disable fancy word wrappin
  // which we really don't want in some apps (but is forced on us)!
  ReplaceCharacters(pMemText, slen, ' ', 0x19);
    // substitute numeric space for conventional space to prevent word wrap(??)

  FldSetTextPtr(MAINTEXTFIELD,pMemText);         // CANNOT use this for editable f
  FldRecalculateField(MAINTEXTFIELD, false);      // recalculate, redraw
  return 0;                                      // 0 = ok.
}

// ------------------------------------------------------------------------------
Int16 fred::BackOneLine ()
{
  if (! ATSTART)                                 // if not futile..
     { LINEPOSITION --;  // move back one line
     };                     // the value should always be < zero.
     // having ATSTART set prevents futile backwards decrements, which in turn
     //  would necessitate moving forward several times before we get anywhere!
  SetMainText();
  FldDrawField(MAINTEXTFIELD);
  return 0; // ok
}
```

In the following addition, we delete the pain caching files, turning off caching and permitting normal log-on:

```
Int16 fred::KillPainCache ()
{
  Int16 err0;
  LocalID   wrid; // usual ugly stuff..

  // 1. identify file
  wrid = DmFindDatabase (0, "CACHESTORE.SQ3");
  if (! wrid)
     {   FrmCustomAlert(MyAlert, "CACHESTORE.SQ3 Not found", "", "");
       return 0;
     };

  // 2. delete
  err0 = DmDeleteDatabase (0, wrid);
  if (err0 != errNone)
     { return err0;
     };

  // 3. exit:
  FrmCustomAlert(MyAlert, "Deleted", "", "");
  return 0; // ok
}
```

On ‘success’ we return zero, otherwise an error code.
Here’s a similar routine to delete the file **CONSOLE.pdb**.

```
Int16 fred::EradicateConsole ()
{
  Int16 err0;
  LocalID   wrid; // usual ugly stuff..

  // 1. identify file
  wrid = DmFindDatabase (0, "CONSOLE.pdb");
  if (! wrid)
     {   // FrmCustomAlert(MyAlert, "CONSOLE.pdb Not found", "", "");
         err0 = (Int16) DmGetLastErr();
         if (! err0) { err0 = 1; };
       return err0;
     };

  // 2. delete
  err0 = DmDeleteDatabase (0, wrid);
  if (err0 != errNone)
     { return err0;
     };

  // 3. exit:
  FrmCustomAlert(MyAlert, "Console cleared", "", "");
  return 0; // ok
}

// ------------------------------------------------------------------------------
Int16 fred::ForwardOneLine ()
{ ATSTART = 0;                                  // indicate NOT at start
  if (LINEPOSITION < -1)
     { LINEPOSITION ++;  // move back one line
     };
  SetMainText();
  FldDrawField(MAINTEXTFIELD);
  return 0; // ok
}



// ==============================================================================
// utility functions:

// ------------------------------------------------------------------------------
// Copy from one pointer to another

Int16 fred::xCopy (Char * dest, Char * xsrc, Int16 cnt)
```

```
{
  if (! dest)
     { return -1;
     };
  if (! xsrc)
     { return -2;
     };
  while (cnt > 0)                                  // permissible to copy NO bytes!
    { * dest++ = * xsrc++;
      cnt --;
    };
  return 0;     //ok
}


//-------------------------------------------------------------------------------
// Fill pointer for specified length with given character:

Int16 fred::xFill (Char * p0, Int16 slen, Char c)
{   if (! p0)
       { return -1;
       };
  while (slen > 0)
    { * p0++ = c;
      slen --;
    };
 return 0;           // success
}


// ===============================================================================
```

6 The RCP file: osbox.rcp

```
#include "osboxrcp.h"

VERSION "1.00"
ICON "osbox.bmp"

ALERT ID DebugAlert
  WARNING
BEGIN
  TITLE "Debug alert!"
  MESSAGE "^3 '^1' (^2)"
  BUTTON "OK"
END

ALERT ID MyAlert
  WARNING
BEGIN
  TITLE "Note!"
  MESSAGE "^1"
  BUTTON "OK"
END

ALERT ID MyConfirm
BEGIN
  TITLE "Are you sure?"
  MESSAGE "^1"
  BUTTONS "NO" "Yes"
END

ALERT ID MyAsk
BEGIN
  TITLE "Enter text"
  MESSAGE "^1"
  BUTTONS "OK" "Cancel"
END
```

7 The bitmap image

Here’s a uuencoded version of the bitmap image required by the RCP file above (for the icon on the PDA).

```
begin 644 osbox.bmp
M0DVˆ‘‘‘‘‘‘‘‘#X‘‘‘H‘‘‘‘(‘‘‘‘"‘‘‘‘‘!‘‘$‘‘‘‘‘‘(‘‘‘‘#>#@‘‘W@X‘
M‘‘(‘‘‘‘"‘‘‘‘‘‘‘‘/___P‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘____XCX<’"F<SCC;&,8QYAC&,>0XQC’@>,8P&-
MC&,1F<SCDD7AX\9__ˆ/ˆ#P>;O!R,RS@X’.<X.!SG.#@<YS@X’.<X.9SG.!F,
*Q[@/!X]P‘‘‘‘‘‘‘‘
‘
end
```

8 The RCP header file: osboxrcp.h

```
#define DebugAlert                  9000
#define MyAlert                     9001
#define MyConfirm                   9002
#define MyAsk                       9003

#define Fld0                        9100
#define But1Back                    9101
#define But2Quit                    9102
#define But3Fwd                     9103
#define But4BBack                   9104
#define But5FFwd                    9105
#define But6Kill                    9106
#define But7Clear                   9107
```