Demonstration Program DateTimeNumbers Listing

```
// *******************************************************************************************
// DateTimeNumbers.c                                                     CLASSIC EVENT MODEL
// *******************************************************************************************
//
// This program, which opens a single modeless dialog, demonstrates the formatting and display
// of dates, times and numbers.
//
// The program utilises the following resources:
//
// •  A 'plst' resource.
//
// •  An 'MBAR' resource, and 'MENU' resources for Apple/Application, File, and Edit menus
//    (preload, non-purgeable).
//
// •  A 'DLOG' resource and associated 'dlgx', 'DITL', 'dfnt', and 'CNTL' resources
//    (purgeable).
//
// •  'hdlg' and 'STR#' resources (purgeable) for balloon help and help tags.
//
// •  A 'SIZE' resource with the acceptSuspendResumeEvents, canBackground,
//    doesActivateOnFGSwitch, and isHighLevelEventAware flags set.
//
// *******************************************************************************************

// .................................................................................. includes

#include <Carbon.h>
#include <string.h>

// ................................................................................... defines

#define rMenubar                128
#define mAppleApplication       128
#define  iAbout                 1
#define mFile                   129
#define  iQuit                  12
#define mEdit                   130
#define  iCut                   3
#define  iCopy                  4
#define  iPaste                 5
#define  iClear                 6
#define rDialog                 128
#define  iStaticTextTodaysDate  2
#define  iStaticTextCurrentTime 4
#define  iEditTextTitle         10
#define  iEditTextQuantity      11
#define  iEditTextValue         12
#define  iEditTextDate          13
#define  iButtonEnter           18
#define  iButtonClear           19
#define  iStaticTextTitle       26
#define  iStaticTextQuantity    27
#define  iStaticTextUnitValue   28
#define  iStaticTextTotalValue  29
#define  iStaticTextDate        30
#define kReturn                 0x0D
#define kEnter                  0x03
#define kTab                    0x09
#define kLeftArrow              0x1C
#define kRightArrow             0x1D
#define kUpArrow                0x1E
#define kDownArrow              0x1F
#define kBackspace              0x08
#define kDelete                 0x7F
#define topLeft(r)              (((Point *) &(r))[0])
#define botRight(r)             (((Point *) &(r))[1])
```

```
// .................................................................................. global variables

Boolean         gRunningOnX = false;
DialogRef       gDialogRef;
DateCacheRecord gDateCacheRec;
Boolean         gDone;
RgnHandle       gCursorRegionHdl;
Boolean         gInBackground;

// ............................................................................... function prototypes

void                   main                (void);
void                   doPreliminaries     (void);
OSErr                  quitAppEventHandler (AppleEvent *,AppleEvent *,SInt32);
void                   eventLoop           (void);
void                   doIdle              (void);
void                   doEvents            (EventRecord *);
void                   doMenuChoice        (SInt32);
void                   doCopyPString       (Str255,Str255);
void                   doTodaysDate        (void);
void                   doAcceptNewRecord   (void);
void                   doUnitAndTotalValue (Str255,Str255);
void                   doDate              (Str255);
void                   doAdjustCursor      (WindowRef);
void                   doClearAllFields    (void);
ControlKeyFilterResult numericFilter       (ControlRef,SInt16 *,SInt16 *,EventModifiers *);
void                   helpTags            (void);

// ************************************************************************************** main

void  main(void)
{
  MenuBarHandle       menubarHdl;
  SInt32              response;
  MenuRef             menuRef;
  Boolean             runningOnX = false;
  ControlKeyFilterUPP numericFilterUPP;
  ControlRef          controlRef;

  // ........................................................................... do preliminaries

  doPreliminaries();

  // ..................................................................... set up menu bar and menus

  menubarHdl = GetNewMBar(rMenubar);
  if(menubarHdl == NULL)
    ExitToShell();
  SetMenuBar(menubarHdl);
  DrawMenuBar();

  Gestalt(gestaltMenuMgrAttr,&response);
  if(response & gestaltMenuMgrAquaLayoutMask)
  {
    menuRef = GetMenuRef(mFile);
    if(menuRef != NULL)
    {
      DeleteMenuItem(menuRef,iQuit);
      DeleteMenuItem(menuRef,iQuit - 1);
      DisableMenuItem(menuRef,0);
    }

    gRunningOnX = true;
  }

  // ........................................................................ open modeless dialog

  if(!(gDialogRef = GetNewDialog(rDialog,NULL,(WindowRef) -1)))
```

```
    ExitToShell();

  // ………… create universal procedure pointers for key filter, attach to two edit text controls

  numericFilterUPP = NewControlKeyFilterUPP((ControlKeyFilterProcPtr) numericFilter);

  GetDialogItemAsControl(gDialogRef,iEditTextQuantity,&controlRef);
  SetControlData(controlRef,kControlEntireControl,kControlEditTextKeyFilterTag,
                 sizeof(numericFilterUPP),&numericFilterUPP);

  GetDialogItemAsControl(gDialogRef,iEditTextValue,&controlRef);
  SetControlData(controlRef,kControlEntireControl,kControlEditTextKeyFilterTag,
                 sizeof(numericFilterUPP),&numericFilterUPP);

  // ……………………………………………………………………………… set help tags, get today's date, and show window

  if(gRunningOnX)
    helpTags();

  doTodaysDate();

  ShowWindow(GetDialogWindow(gDialogRef));

  // ……………………………………………………………………………………………………… initialise date cache structure

  InitDateCache(&gDateCacheRec);

  // ……………………………………………………………………………………………………………………………………… enter eventLoop

  eventLoop();
}

// ************************************************************************** doPreliminaries

void  doPreliminaries(void)
{
  OSErr osError;

  MoreMasterPointers(64);
  InitCursor();
  FlushEvents(everyEvent,0);

  osError = AEInstallEventHandler(kCoreEventClass,kAEQuitApplication,
                          NewAEEventHandlerUPP((AEEventHandlerProcPtr) quitAppEventHandler),
                          0L,false);
  if(osError != noErr)
    ExitToShell();
}

// *************************************************************************** doQuitAppEvent

OSErr  quitAppEventHandler(AppleEvent *appEvent,AppleEvent *reply,SInt32 handlerRefcon)
{
  OSErr    osError;
  DescType returnedType;
  Size     actualSize;

  osError = AEGetAttributePtr(appEvent,keyMissedKeywordAttr,typeWildCard,&returnedType,NULL,0,
                            &actualSize);

  if(osError == errAEDescNotFound)
  {
    gDone = true;
    osError = noErr;
  }
  else if(osError == noErr)
    osError = errAEParamMissed;

  return osError;
```

```
}

// ******************************************************************************** eventLoop

void  eventLoop(void)
{
  EventRecord eventStructure;
  Boolean     gotEvent;
  UInt32      sleepTime;

  gDone = false;
  sleepTime = GetCaretTime();
  gCursorRegionHdl = NewRgn();

  while(!gDone)
  {
    gotEvent = WaitNextEvent(everyEvent,&eventStructure,sleepTime,gCursorRegionHdl);

    if(gotEvent)
      doEvents(&eventStructure);
    else
      doIdle();
  }
}

// *********************************************************************************** doIdle

void  doIdle(void)
{
  UInt32        rawSeconds;
  static UInt32 oldRawSeconds;
  Str255        timeString;
  ControlRef    controlRef;

  if(!gRunningOnX)
    IdleControls(GetDialogWindow(gDialogRef));

  GetDateTime(&rawSeconds);

  if(rawSeconds > oldRawSeconds)
  {
    TimeString(rawSeconds,true,timeString,NULL);

    GetDialogItemAsControl(gDialogRef,iStaticTextCurrentTime,&controlRef);
    SetControlData(controlRef,kControlEntireControl,kControlStaticTextTextTag,timeString[0],
                   &timeString[1]);
    Draw1Control(controlRef);

    oldRawSeconds = rawSeconds;
  }
}

// ********************************************************************************** doEvent

void  doEvents(EventRecord *eventStrucPtr)
{
  WindowPartCode partCode;
  WindowRef      windowRef;
  DialogRef      dialogRef;
  SInt16         itemHit;
  SInt8          charCode;
  ControlRef     controlRef;
  UInt32         finalTicks;

  switch(eventStrucPtr->what)
  {
    case kHighLevelEvent:
      AEProcessAppleEvent(eventStrucPtr);
      break;
```

```
case mouseDown:
  partCode = FindWindow(eventStrucPtr->where,&windowRef);

  switch(partCode)
  {
    case inMenuBar:
      doMenuChoice(MenuSelect(eventStrucPtr->where));
      break;

    case inContent:
      if(IsDialogEvent(eventStrucPtr))
        if(DialogSelect(eventStrucPtr,&dialogRef,&itemHit))
          if(itemHit == iButtonEnter)
          {
            doAcceptNewRecord();
            doClearAllFields();
          }
          else if(itemHit == iButtonClear)
            doClearAllFields();
      doAdjustCursor(windowRef);
      break;

    case inDrag:
      DragWindow(windowRef,eventStrucPtr->where,NULL);
      doAdjustCursor(windowRef);
      break;

    case inGoAway:
      if(TrackGoAway(windowRef,eventStrucPtr->where))
        gDone = true;
      break;
  }
  break;

case keyDown:
  charCode = eventStrucPtr->message & charCodeMask;

  if((charCode == kReturn) || (charCode == kEnter))
  {
    GetDialogItemAsControl(gDialogRef,iButtonEnter,&controlRef);
    HiliteControl(controlRef,kControlButtonPart);
    Delay(8,&finalTicks);
    HiliteControl(controlRef,kControlEntireControl);
    doAcceptNewRecord();
    doClearAllFields();
    return;
  }

  if((eventStrucPtr->modifiers & cmdKey) != 0)
  {
    if(charCode == 'X' || charCode == 'x' ||  charCode == 'C' || charCode == 'c' ||
        charCode == 'V' || charCode == 'v')
    {
      HiliteMenu(mEdit);
      DialogSelect(eventStrucPtr,&dialogRef,&itemHit);
      Delay(4,&finalTicks);
      HiliteMenu(0);
    }
    else
    {
      doMenuChoice(MenuEvent(eventStrucPtr));
    }
    return;
  }

  DialogSelect(eventStrucPtr,&dialogRef,&itemHit);
  if(charCode == kTab)
    doAdjustCursor(GetDialogWindow(gDialogRef));
```

```
      break;

    case autoKey:
      if((eventStrucPtr->modifiers & cmdKey) == 0)
        DialogSelect(eventStrucPtr,&dialogRef,&itemHit);
      break;

    case updateEvt:
    case activateEvt:
      DialogSelect(eventStrucPtr,&dialogRef,&itemHit);
      break;

    case osEvt:
      switch((eventStrucPtr->message >> 24) & 0x000000FF)
      {
        case suspendResumeMessage:
          gInBackground = (eventStrucPtr->message & resumeFlag) == 0;
          if(!gInBackground)
            SetThemeCursor(kThemeArrowCursor);
          break;

        case mouseMovedMessage:
          doAdjustCursor(GetDialogWindow(gDialogRef));
          break;
      }
      break;
  }
}

// ****************************************************************************** doMenuChoice

void  doMenuChoice(SInt32 menuChoice)
{
  MenuID        menuID;
  MenuItemIndex menuItem;

  menuID   = HiWord(menuChoice);
  menuItem = LoWord(menuChoice);

  if(menuID == 0)
    return;

  switch(menuID)
  {
    case mAppleApplication:
      if(menuItem == iAbout)
        SysBeep(10);
      break;

    case mFile:
      if(menuItem == iQuit)
        gDone = true;
      break;

    case mEdit:
      switch(menuItem)
      {
        case iCut:
          DialogCut(gDialogRef);
          break;

        case iCopy:
          DialogCopy(gDialogRef);
          break;

        case iPaste:
          DialogPaste(gDialogRef);
          break;
        case iClear:
          DialogDelete(gDialogRef);
          break;
      }
      break;
  }

  HiliteMenu(0);
}

// ***************************************************************************** doCopyPString

void  doCopyPString(Str255 sourceString,Str255 destinationString)
{
  SInt16 stringLength;

  stringLength = sourceString[0];
  BlockMove(sourceString + 1,destinationString + 1,stringLength);
  destinationString[0] = stringLength;
}

// ****************************************************************************** doTodaysDate

void  doTodaysDate(void)
{
  UInt32     rawSeconds;
  Str255     dateString;
  ControlRef controlRef;

  GetDateTime(&rawSeconds);
  DateString(rawSeconds,longDate,dateString,NULL);

  GetDialogItemAsControl(gDialogRef,iStaticTextTodaysDate,&controlRef);
  SetControlData(controlRef,kControlEntireControl,kControlStaticTextTextTag,dateString[0],
                 &dateString[1]);
}

// ************************************************************************** doAcceptNewRecord

void  doAcceptNewRecord(void)
{
  SInt16     theType;
  Handle     theHandle;
  Rect       theRect;
  Str255     titleString, quantityString, valueString, dateString;
  ControlRef controlRef;

  GetDialogItem(gDialogRef,iEditTextTitle,&theType,&theHandle,&theRect);
  GetDialogItemText(theHandle,titleString);

  GetDialogItem(gDialogRef,iEditTextQuantity,&theType,&theHandle,&theRect);
  GetDialogItemText(theHandle,quantityString);

  GetDialogItem(gDialogRef,iEditTextValue,&theType,&theHandle,&theRect);
  GetDialogItemText(theHandle,valueString);

  GetDialogItem(gDialogRef,iEditTextDate,&theType,&theHandle,&theRect);
  GetDialogItemText(theHandle,dateString);

  if(titleString[0] == 0 || quantityString[0] == 0 || valueString[0] == 0 ||
     dateString[0] == 0)
  {
    SysBeep(10);
    return;
  }

  GetDialogItemAsControl(gDialogRef,iStaticTextTitle,&controlRef);
  SetControlData(controlRef,kControlEntireControl,kControlStaticTextTextTag,titleString[0],
                 &titleString[1]);
  Draw1Control(controlRef);

  GetDialogItemAsControl(gDialogRef,iStaticTextQuantity,&controlRef);
  SetControlData(controlRef,kControlEntireControl,kControlStaticTextTextTag,quantityString[0],
               &quantityString[1]);
  Draw1Control(controlRef);

  doUnitAndTotalValue(valueString,quantityString);

  doDate(dateString);
}

// ************************************************************************ doUnitAndTotalValue

void  doUnitAndTotalValue(Str255 valueString, Str255 quantityString)
{
  Handle           itl4ResourceHdl;
  SInt32           numpartsOffset;
  SInt32           numpartsLength;
  NumberParts      *numpartsTablePtr;
  Str255           formatString = "\p'$'###,###,###.00;'Valueless';'Valueless'";
  NumFormatString  formatStringRec;
  Str255           formattedNumString;
  extended80       value80Bit;
  SInt32           quantity;
  double           valueDouble;
  FormatResultType result;
  ControlRef       controlRef;

  GetIntlResourceTable(smSystemScript,iuNumberPartsTable,&itl4ResourceHdl,&numpartsOffset,
                       &numpartsLength);
  numpartsTablePtr = (NumberPartsPtr) ((SInt32) *itl4ResourceHdl + numpartsOffset);

  StringToFormatRec(formatString,numpartsTablePtr,&formatStringRec);

  StringToExtended(valueString,&formatStringRec,numpartsTablePtr,&value80Bit);
  ExtendedToString(&value80Bit,&formatStringRec,numpartsTablePtr,formattedNumString);

  GetDialogItemAsControl(gDialogRef,iStaticTextUnitValue,&controlRef);
  SetControlData(controlRef,kControlEntireControl,kControlStaticTextTextTag,
               formattedNumString[0],&formattedNumString[1]);
  Draw1Control(controlRef);

  StringToNum(quantityString,&quantity);

  valueDouble = x80tod(&value80Bit);
  valueDouble = valueDouble * quantity;
  dtox80(&valueDouble,&value80Bit);

  result = ExtendedToString(&value80Bit,&formatStringRec,numpartsTablePtr,
                            formattedNumString);

  if(result == fFormatOverflow)
    doCopyPString("\p(Too large to display)",formattedNumString);

  GetDialogItemAsControl(gDialogRef,iStaticTextTotalValue,&controlRef);
  SetControlData(controlRef,kControlEntireControl,kControlStaticTextTextTag,
               formattedNumString[0],&formattedNumString[1]);
  Draw1Control(controlRef);
}

// ************************************************************************************* doDate

void  doDate(Str255 dateString)
{
  SInt32       lengthUsed;
  LongDateRec  longDateTimeRec;
  LongDateTime longDateTimeValue;
  ControlRef   controlRef;
  StringToDate((Ptr) dateString + 1,dateString[0],&gDateCacheRec,&lengthUsed,&longDateTimeRec);

  LongDateToSeconds(&longDateTimeRec,&longDateTimeValue);
  LongDateString(&longDateTimeValue,longDate,dateString,NULL);

  GetDialogItemAsControl(gDialogRef,iStaticTextDate,&controlRef);
  SetControlData(controlRef,kControlEntireControl,kControlStaticTextTextTag,dateString[0],
                 &dateString[1]);
  Draw1Control(controlRef);
}

// ************************************************************************** doAdjustCursor

void  doAdjustCursor(WindowRef windowRef)
{
  GrafPtr   oldPort;
  RgnHandle arrowRegion,iBeamRegion;
  SInt16    currentFocusItem;
  SInt16    theType;
  Handle    theHandle;
  Rect      iBeamRect;
  Point     mouseXY;

  GetPort(&oldPort);
  SetPortWindowPort(windowRef);

  arrowRegion = NewRgn();
  iBeamRegion = NewRgn();

  SetRectRgn(arrowRegion,-32768,-32768,32767,32767);

  currentFocusItem = GetDialogKeyboardFocusItem(gDialogRef);
  GetDialogItem(gDialogRef,currentFocusItem,&theType,&theHandle,&iBeamRect);

  LocalToGlobal(&topLeft(iBeamRect));
  LocalToGlobal(&botRight(iBeamRect));

  RectRgn(iBeamRegion,&iBeamRect);
  DiffRgn(arrowRegion,iBeamRegion,arrowRegion);

  GetMouse(&mouseXY);
  LocalToGlobal(&mouseXY);

  if(PtInRgn(mouseXY,iBeamRegion))
  {
    SetThemeCursor(kThemeIBeamCursor);
    CopyRgn(iBeamRegion,gCursorRegionHdl);
  }
  else
  {
    SetThemeCursor(kThemeArrowCursor);
    CopyRgn(arrowRegion,gCursorRegionHdl);
  }

  DisposeRgn(arrowRegion);
  DisposeRgn(iBeamRegion);

  SetPort(oldPort);
}

// ************************************************************************ doClearAllFields

void  doClearAllFields(void)
{
  SInt16     a;
  ControlRef controlRef;
  Str255     theString = "\p";
  for(a = iEditTextTitle;a <= iEditTextDate;a++)
  {
    GetDialogItemAsControl(gDialogRef,a,&controlRef);
    SetControlData(controlRef,kControlEntireControl,kControlEditTextTextTag,theString[0],
                   &theString[1]);
    Draw1Control(controlRef);

    if(a == iEditTextTitle)
      SetKeyboardFocus(GetDialogWindow(gDialogRef),controlRef,kControlFocusNextPart);
  }
}

// ************************************************************************** numericFilter

ControlKeyFilterResult  numericFilter(ControlRef controlRef,SInt16* keyCode,SInt16 *charCode,
                                      EventModifiers *modifiers)
{
  if(((char) *charCode >= '0') && ((char) *charCode <= '9') || (char) *charCode == '.' ||
     (BitTst(modifiers,15 - cmdKeyBit)))
  {
    return kControlKeyFilterPassKey;
  }

  switch(*charCode)
  {
    case kLeftArrow:
    case kRightArrow:
    case kUpArrow:
    case kDownArrow:
    case kBackspace:
    case kDelete:
      return kControlKeyFilterPassKey;
      break;
  }

  SysBeep(10);
  return kControlKeyFilterBlockKey;
}

// ******************************************************************************* helpTags

void  helpTags(void)
{
  HMHelpContentRec helpContent;
  SInt16           a;
  static SInt16    itemNumber[7] = { 1,3,21,22,23,24,25 };
  ControlRef       controlRef;

  memset(&helpContent,0,sizeof(helpContent));

  HMSetTagDelay(5);
  HMSetHelpTagsDisplayed(true);
  helpContent.version = kMacHelpVersion;
  helpContent.tagSide = kHMOutsideTopCenterAligned;

  helpContent.content[kHMMinimumContentIndex].contentType = kHMStringResContent;
  helpContent.content[kHMMinimumContentIndex].u.tagStringRes.hmmResID = 128;

  for(a = 1;a <= 7; a++)
  {
    helpContent.content[kHMMinimumContentIndex].u.tagStringRes.hmmIndex = a;
    GetDialogItemAsControl(gDialogRef,itemNumber[a - 1],&controlRef);
    HMSetControlHelpContent(controlRef,&helpContent);
  }
}

// ***********************************************************************************************
```

Demonstration Program DateTimeNumbers Comments

When this program is run, the user should enter data in the four edit text fields, using the tab key or mouse clicks to select the required field and pressing the Return key or clicking the Enter Record button when data has been entered in all fields. Note that numeric filters are used in the Quantity and Value edit text controls.

In order to observe number formatting effects, the user should occasionally enter very large numbers and negative numbers in the Value field. In order to observe the effects of date string parsing and formatting, the user should enter dates in a variety of formats, for example: "2 Mar 95", "2/3/95", "March 2 1995", "2 3 95", etc.

Global Variables

gDateCacheRec is used within the function doDate.

main

doTadaysDate is called to get the date and set it in a static text control at the top of the dialog.

In the function doDate, the function which creates the long date-time structure takes an initialised date cache structure as a parameter. The call to InitDateCache initialises a date cache structure.

doIdle

doIdle is called when WaitNextEvent returns with 0. It blinks the insertion point caret and sets the current time in the static text control at top-right in the dialog.

IdleControls is called to ensure that the caret blinks regularly in the edit text control with current keyboard focus (for Mac OS 8/9 only).

GetDateTime retrieves the "raw" seconds value, as known to the system. (This is the number of seconds since 1 Jan 1904.) If that value is greater than the value retrieved the last time doIdle was called, TimeString converts the raw seconds value to a string containing the time formatted according to flags in the numeric format ('itl0') resource. (Since NULL is specified in the resource handle parameter, the appropriate 'itl0' resource for the current script system is used.) This string is then set in the static text control, following which Draw1Control is called to redraw the control. The retrieved raw seconds value is assigned to the static variable oldRawSeconds for use next time doIdle is called.

doEvents

In the case of a mouse-down event in the content region of the dialog, if the Enter Record button is clicked, doAcceptNewRecord, following which doClearAllFields is called to clear all of the edit text controls. The same occurs when the Return or Enter keys are pressed.

doTodaysDate

doTodaysDate sets the date in the static text control at top-left of the dialog.

GetDateTime gets the raw seconds value, as known to the system. DateString converts the raw seconds value to a string containing a date formatted in long date format according to flags in the numeric format ('itl0') resource. (Since NULL is specified in the resource handle parameter, the appropriate 'itl0' resource for the current script system is used.) This string is then set in the static text control.

doAcceptNewRecord

doAcceptNewRecord is called when the Return or Enter key is pressed, or when the Enter Record button is clicked. Assuming each edit text control contains at least one character of text, it calls other functions to format (where necessary) and display strings in the "Last Record Entered" group box area.

The calls to GetDialogItem get the handle in the hText field of each edit text control's TextEdit structure, allowing the calls to GetDialogItemText to get the text into four local variables of type Str255.

If the length of any of these strings is 0, the system alert sound is played and doAcceptNewRecord returns.

The text from the Item Title and Quantity edit text controls are set in the relevant static text controls within the Last Record Entered group box, and DrawIControl is called to draw those controls. doUnitAndTotalValue and doDate are then called.

doUnitAndTotalValue

doUnitAndTotalValue is called by doAcceptNewRecord to convert the string from the Value edit text control to a floating point number, convert that number to a formatted number string, set that string in the relevant static text control, convert that string from the Quantity edit text control to an integer, multiply the floating point number by the integer to arrive at the "Total Value" value, convert the result to a formatted number string, and set that string in the relevant static text control.

A pointer to a number parts table is required by the functions that convert between floating point numbers and strings. Accordingly, the first two lines get the required pointer.

StringToFormatRec converts the number format specification string into the internal numeric representation required by the functions that convert between floating point numbers and strings.

StringToExtended converts the received Value string into a floating point number of type extended (80 bits). ExtendedToString converts that number back to a string, formatted according to the internal numeric representation of the number format specification string. That string is then set in the relevant static text control and Draw1Control is called to draw that control.

The intention now is to multiply the quantity by the unit value to arrive at a total value. The string received in the quantityString formal parameter is converted to an integer value of type SInt32 by StringToNum. The extended80 value is converted to a value of type double before the multiplication occurs. The result of the multiplication is assigned to the variable of type double. This is then converted to an extended80.

The extended80 value is then passed in the first parameter of ExtendedToString for conversion to a formatted string. If ExtendedToString does not return fFormatOverflow, the formatted string is set in the relevant static text control and Draw1Control is called to draw that control.

doDate

doDate is called by doAcceptNewRecord to create a long date-time structure from the string in the "Date" edit text control, format the date as a string (long date format), and set that string in thr relevant static text control.

A pointer to the string containing the date as entered by the user, and the length of that string, are passed in the call to StringToDate. StringToDate parses the input string and fills in the relevant fields of the long date-time structure.

LongDateToSeconds converts the long date-time structure to a long date-time value. The long date-time value is then passed as a parameter to LongDateString, which converts the long date-time value to a long format date string formatted according to the specified international resource. (In this case, NULL is passed as the international resource parameter, meaning that the appropriate 'itl1' resource for the current script system is used.)

The formatted date string is then set in the relevant static text control and Draw1Control is called to draw that control.