Mobile FBUS

Delphi Components

Version 1.6

? Software Cave
February 2001
All Rights Reserved

FBUS.PAS

Types

TComPort = (COM1,COM2,COM3,COM4);
Used to specify the comport to use

TReset = (rsSoft,rsHard);
Used to specify the type of reset the phone will do

```
TAlarm =
    Record
      AlarmTime : TDateTime;
      AlarmSet  : Boolean;
    end;
```
Record structure of the phone alarm information

```
TPhoneInfo =
    Record
      IMEI      : String;
      SWVersion : String;
      HWVersion : String;
      Model     : String;
      SWDate    : String;
    end;
```
Record Structure of the Phone model information

```
TNetworkInfo =
    Record
      NetworkCode : String;
      Country     : String;
      SMSCenter   : String;
      CellID      : String;
      LAC         : String;
    end;
```
Record Structure of the Phone network settings

Events

TConnectedEvent = procedure(iResult:integer;msg:String) of object;
Raises an event when the phone is connected

TonPlayingTone = procedure(bPlaying : Boolean) of object;
Raises an event when a ringtone is being played

TACDCPower = procedure(ACDCPowered:Boolean) of object;
Raises and event to indicate if the phone is powered by battery or via ACDC power, i.e. a recharger or recharge cradle

TbatteryLevel = procedure(Level:integer) of object;
Raises an event to indicate the level of the batteries charge. This is from 0 to 5. 5 being the highest value, i.e. fully charged

TsignalLevel = procedure(Level:integer) of object;
Raises an event to indicate the level of the signal. This is from 0 to 5. 5 being the highest value, i.e. an excellent signal

TkeyLock = procedure(Locked:Boolean) of object;
Raises an event to indicate if the Phone keyboard is locked. True is locked, false is unlocked

TcallInProgress = procedure(CallInProgress:boolean) of object;
Raises an event to indicate if a call is in progress. True means a call is in progress, false mean there is no call in progress

TPBEntryError = procedure(Sender:String) of object;
Raises an event to indicate that there has been an error adding an entry to the phones phonebook. Sender is a string explanation of the error

TNewSMSMessage = procedure of Object;
Raises an event to indicate thata new SMS message has been received by the phone

TSMS_Msg_Error = procedure(Sender:String) of object;
Raises an event to indicate an error while sending an SMS Message. Sender is a string explanation of the error

TSMSMessageCount = procedure(iSMSCount:integer) of object;
Raises an event to indicate the count if SMS Messages in the Phones Inbox

TSMSDelivMessageCount = procedure(iSMSCount:integer) of object;
Raises an event to indicate the count if SMS Messages in the Phones Delivered messages box

TSMSOutMessageCount = procedure(iSMSCount:integer) of object;
Raises an event to indicate the count if SMS Messages in the Phones out box

TRingToneSent = procedure(Sent:Boolean) of Object;
Raises an event to indicate if a ringtone has been sent via SMS. True means it has been sent successfully, false means there was an error sending it.

TIncomingCall = procedure(Number:String) of Object;
Raises an event to indicate that there is an incoming call on the phone. Number is a string containing the name or number of the caller. If the number is in the phones phonebox, it will display the name of the caller, otherwise it will return the number

Public

procedure Connect;
This procedure initiates a connection to the phone

procedure Disconnect;
This procedure will disconnect the connection to the phone.

procedure SetAlarmTime(value : TDateTime);
This procedure will set the alarm on the phone

procedure SetTime(Value : TDateTime);
This procedure will set the date and time on the phone

function GetTime:TDateTime;
This function will get the current time from the phone. By default, they date may not be set on the phone, and the first time you call GetTime the date may be a random date, but one you set the date and time, the date will be set in the phone, and will it will keep the correct date from there.

function NetMon(sMode:string):string;
This function is used to query the different NetMonitor screens on the phone, and each of the screens is returned as a string.

procedure ResetPhone(ResetType:TReset);
This procedure will reset the phone depending on the TReset Type passed

procedure HangupPhone;
This procedure will hang up a phone call in progress

procedure MakeVoiceCall(sNumber:string);
This procedure will make a voice call to the number passed to it

procedure SendDTMF(sTones:string);
This procedure will send DTMF tones to the phone depending on the tone passed to it.
Valid tones are 0 to 9, # and *

procedure PhoneBeep;
This will cause the phone to make a beep sound

Published

property Comport : TComport read GetComport write SetComPort;
Sets the comport to use to connect to the phone'

property FBUSStatusBar : TMFBUSStatusBar read fFBUSStatusBar write fFBUSStatusBar;
This specifies a connection to the FbusStatusbar to use.

property OnConnected : TConnectedEvent read GetConnectedState write SetConnectedState;
This is were the event to indicate if the phone is connected is handled

property OnPlayingTone : TOnPlayingTone read GetOnPlayingTone write SetOnPlayingTone;
This is were the event to indicate if a ringtone is being played is handled

property OnACDCPower : TACDCPower read GetACDCPowerStatus write SetACDCPowerStatus;
This is were the event to indicate the power source of the phone is handled

property OnBatteryLevel : TBatteryLevel read GetBattLevel write SetBatteryLevel;
This is were the event to indicate the battery charge level is handled

property OnSignalLevel : TSignalLevel read GetSignalLevel write SetSignalLevel;
This is were the event to indicate the signal level is handled

property OnKeyLock : TKeyLock read GetKeyLock write SetKeyLock;
This is were the event to indicate if the phones keyboard is locked or not is handled

property OnCallInProgress : TCallInProgress read GetCallInProgress write setCallInProgress;
This is were the event to indicate if a call is in progress is handled

property OnPBEntryError : TPBEntryError read ePBEntryError write SetPBEntryError;
This is were the event to indicate if there was an error adding a Phonebook entry is handled

property OnNewSMSMessage : TNewSMSMessage read eNewSMSMessage write eNewSMSMessage;
This is were the event to indicate when a new SMS message has been received by the phone is handled

property OnSMSError : TSMS_Msg_Error Read eSMSError write eSMSError;
This is were the event to indicate if there was an error with SMS is handled

property OnSMSSendError : TSMSSendError read eSMSSendError write eSMSSendError;
This is were the event to indicate if there was an error sending an SMS message is handled

property OnSMSSent : TSMSSentOk read eSMSSent write eSMSSent;
This is were the event to indicate if an SMS message was sent successfully or not is handled

property OnInboxCount : TSMSMessageCount read eSMSMessageCount write eSMSMessageCount;
This is were the event to indicate the count of SMS messages in the phones Inbox is handled

property OnOutBoxCount : TSMSOutMessageCount read eSMSOutMessageCount write eSMSOutMessageCount;
This is were the event to indicate the count of SMS messages in the phones OutBox is handled

property OnDelivaryCount : TSMSDelivMessageCount read eSMSDelivMessageCount write eSMSDelivMessageCount;
This is were the event to indicate the count if SMS messages in the Deliverybox is handled

property OnRingToneSent : TRingToneSent read eRingToneSent write eRingToneSent;
This is were the event to indicate if a ringtone has been sent via SMS successfully or not is handled

Property OnIncommingCall : TIncomingCall read eIncomingCall write eIncomingCall;
This is were the event to indicate if the phone is receiving an incoming call is handled

LOGOEDITOR.PAS

TYPES

Type

TLogoType = (fbNoLogo, fbStartupLogo, fbOperatorLogo, fbCallerLogo)

Events

TDownloadErrorEvent = procedure(iError:integer) of object;
This event handler is raised when an error occurs trying to download a logo from the phone

TLogoUploaded = procedure(Uploaded:Boolean) of object;
This event handler is raised when a logo has successfully been sent to the phone

TLogoDownloaded = procedure(Downloaded:Boolean) of object;
This event handler is raised when a logo has successfully been downloaded from the phone

TSavedLogo = Procedure(SavedToFile:Boolean)of object;
This event handler is raised when a logo has successful been saved to file

TSentViaSMS = Procedure(Sent:Boolean)of object;
This event handler is raised when a logo has been successfully sent Via SMS

TGridPos = Procedure(X,Y:integer) of object;

This event returns the cell position of the mouse in the logo editor

PUBLIC

procedure ClearLogo;
This procedure clears the logo editor

procedure Download(iType :Integer);
This procedure downloads a logo of type TlogoType. Use Ord(logotype) to pass the integer value of the logotype, i.e. Ord(fbOperatorLogo)

procedure SendLogoToPhone;
This procedure will send the currently displayed logo to the phone

procedure LoadLogoFromFile(sFilename:string);
This procedure will load a Logo from file. This can be a NOL, NGG or an NLM

procedure LoadBitmapForLogo(sFilename:string);
This procedure will load a graphic from file . This can be a BMP, GIF, JPG

procedure SaveLogoToFile(sFileName:string);
This procedure will save the currently displayed logo to a Logo file

procedure SendLogoViaSMS(sDestination:string);
This procedure will send a logo via SMS. If you are sending a logo to another phone which has a different provider than yours, and you want them to be able to save the logo, you will have to set the providers information before sending the logo. (see SetNewNetCode)

function ExportLogoToFile(sFileName,sType:string):Boolean;
This function will export the currently displayed logo to a specific file type, i.e. BMP, GIF, JPG

function ResetLogo(iType :Integer):Boolean;
This funtction will reset a logo on the phone, i.e. the ORD of TLogoType, eg. ORD(fStartupLogo)

procedure TextToLogo(img:TImage;iThreshold : Integer = 128; lScreenColor :TColor= $FF00FF; dx :integer = 0; dy:integer=0);
This procedure will send Text to the logo editor. (See Demo for an example)

procedure SetNewNetCode(Value:String);
this procedure will set the Netcode of a provider for a Logo to be sent via SMS

procedure SetStatupText(stext:String);
This procedure will set the Startup Text of the phone

Function DistributeLogo(sFileName,sDestination,sNetCode:string; iType:integer):boolean;
This function will send a logo to a destination without previewing the image. If you want to send a logo to multiple recipients, use this function. Sfilename is the logo filename, sDestination is the phone number to send to, sNetcode is the netcode of the recipents provider,and iType is the ORD of the TlogoType.

Properties

Property LogoType : TLogoType read GetLogoType write SetLogoType;
This property will Set the Logo type of the phone, and will resize the logo editor to the right height and width for the logo selected

Property FBUS : TFBUS read GetFBus write SetFBus;
This property sets a reference to the main FBUS component. The logo editor will not work correctly WITHOUT a reference to an FBUS component

Property Display : TImage read fDisplay write SetDisplay;
This property sets a reference to an Image Control that can be used to display the logo I its actual size. This is not required.

Property OnLogoUploaded : TLogoUploaded read GetLogoUploaded write SetLogoUploaded;
This is were the event to indicate when a logo has been uploaded is handled

Property OnLogoDownloadloaded : TLogoDownloaded read GetLogoDownloaded write SetLogoDownloaded;
This is were the event to indicate when a logo has been downloaded is handled

Property OnSavedLogotToFile : TSavedLogo read GetSavedToFile write SetSavedToFile;
This is were the event to indicate when a logo has been saved to file is handled

Property OnSentViaSMS : TSentViaSMS read GetSentViaSMS write SetSentViaSMS;
This is were the event to indicate when a logo has been sent via SMS is handled

Property OnGridPos : TGridPos read GetGridPos write SetGridPos;
This is were the event to indicate the cell position of the mouse on the logo editor is handled

SMS.PAS

Types

TSMSClass = (fbSMSClassFlash,fbSMSClassNormal,fbSMSClass2,fbSMSClass3);

Used to specify the SMSClass type

TSMSError = procedure(ErrorMessage : String) of object;
Raises an event when an SMS message error has occurred

TSMSSendError = procedure of object;
Raises an event when an error occurred sending an SMS Message

TSMSSentOK = Procedure of Object;
Raises an event when an SMS Message has been successfully sent

TSMSMessage =

```
Record
  msg_Date  : TDateTime;
  sSender   : String;
  bRead     : Boolean;
  sMessage  : String;
end;
```

Record Structure of an SMS Message

Constants

```
fbOutgoingMessage = 11;
fbDeliveryReport  = 12;
fbIncommingMessage = 13;
```

Public

procedure RefreshSMSList;
This procedure will refresh the list of SMS messages

Function SMSCount(iType:integer):integer;
This function will return the Count of messages in the Box Type, specified by one of the box type constants.

function GetUnreadCount:integer;
This function returns the count of unread messages in the inbox

function GetMessageTotal:integer;
This function returns the total messages in the inbox

function GetReceivedCount:integer;
This function returns the count of received messages. When a new message has been received by the phone, you can use this function to find out how many new messages were received

Function MessageRead(iType,imsgIndex:integer):Boolean;
This function indicates if a message specified by its index and the box it is in, has been read yet or not

function SMSMessageData(iType,imsgIndex,iProperty:Integer):String;
This function returns an SMS Message and its information based on its index and its box type

function GetSMSMessageDateTime(itype,iMsgIndex:integer):TDateTime;
This function returns the date and time of an SMS Message based on its index and box type

procedure SendSMSMessage(sDestination,sMessage:string;iSet:integer;
smsClass :TSMSClass);
This procedure sends an SMS Message

function RetreiveMessageData(iType,imsgIndex:integer):TSMSMessage;
This function returns the message data based on its index and box type

function DeleteSMSMessage(iType:integer;msg:String):boolean;
This function will delete an SMS message based on its box type.

Published

property OnSMSError : TSMSError read GetSMSError write SetSMSError;
This is were the event to indicate if there has been an SMS error is handled

property OnSMSSendError : TSMSSendError read getSMSSendError write SetSMSSendError;
This is were the event to indicate if an SMS Message had an error when sending is handled

property OnSMSSent : TSMSSentOk read GetSMSSentOk write SetSMSSentOk;
This is were the event to indicate if an SMS Message has been sent successfully is handled

PHONEBOOK.PAS

Types

TMemType = (fbPhoneMemory,fbSimMemory,fbFixedDialMemory,fbOwnNumberMemory,
fbEmergencyNumberMemory,fbDialedNumberMemory,
fbReceivedNumberMemory,fbMissedNumberMemory,fbLastDialedMemory,
fbCombinedPhonebookMemory);
Used to specify the Phonebook memory type

```
TPhoneBookEntry =
    record
      Location : integer;
      MemType  : TMemType;
      Name     : string;
      Number   : String;
      Group    : integer;
    end;
```
Record Structure of a phonebook entry

```
TBook =
    record
      Name : string;
    end;
```
Record structure of a phonebook book

Events

TPhoneBookEntryError = Procedure(msg:String) of Object;
Raises an error when there has been a phonebook entry error

Public

Function GetEntry(iIndex:integer;MemType:TMemType):TPhoneBookEntry;
This function returns a Phonebook entry based on an index to read, and the memory type to read

Procedure AddPhoneBookEntry(MemType:TMemType; sName,sNumber,sGroup:string);
This procedure adds a phonebook entry based on an index to write to, and the memory type to write to, and the group number

Procedure DeleteEntry(MemType:TMemType;sName,sGroup:string);
This procedure deletes a phonebook entry based on an index of the entry, and the memory type and the group number

Function FindSpeedDial(iLocation:integer):TPhoneBookEntry;
This function returns a phonebook entry based on the speeddial location index

Procedure AddSpeedDial(iLocation:integer;sName:String;memtype:TmemType; iPos:integer = 0);
This procedure adds a speeddial based on the speeddial location index, the phonebook entry name, and the memory type

Procedure EditSpeedDial(iLocation:integer;sName:String;memtype:TmemType; iPos:integer = 0);
This procedure edits a speeddial based on the speeddial location index, the phonebook entry name, and the memory type

published

Property OnPhoneBookEntryError : TPhoneBookEntryError read GetPhoneBookEntryError write SetPhoneBookEntryError;
This is were the event to indicate if a Phonebook entry error occurred is handled

RINGTONE.PAS

Types

TMIDIOUTCAPS =
Record
wMid : Integer;
wPid : Integer;
vDriverVersion : dword;
szPname : String[32];
wTechnology : Integer;
wVoices : Integer;
wNotes : Integer;
wChannelMask : Integer;
dwSupport : dword;
end;
Record Structure for MIDI Output

TMIDIHDR =
Record
lpData : String;
dwBufferLength : dword;
dwBytesRecorded : dword;
dwUser : dword;
dwFlags : dword;
lpNext : dword;
Reserved : dword;
End ;
Record Structure for MIDI Header

TColFrequencies =
Record
Value : Double;
Freq : String;
end;
Record Structure for Frequences

EVENTS

TPlayingTone = procedure(bPlaying : Boolean) of object;
Raises an event when the phone is playing a Ringtone

TRingToneSent = procedure(Sent : boolean) of object;
Raises an event when the ringtone has been sent via SMS

Constants

fbPlayRingtoneSpeaker = 0;
fbPlayRingtoneMidi = 1;
fbPlayRingtonePhone = 2;
Specifies the device to use when playing a ringtone

Public

procedure ReadFile(sFileName :String);
This procedure reads a ringtone file into memory

procedure WriteFile(sFileName :String);
This procedure writes a ringtone to a ringtone file

procedure Upload(bDirect : Boolean = False);
This procedure Uploads a new ringtone to a phone
(note: Nokia 51xx phones can not upload new ringtones)

procedure SendAsSMS(sDestination :String);
This procedure sends a ringtone as an SMS message to another cellphone

procedure Play(Device : integer = fbPlayRingtoneMidi);
This procedure will play a ringtone to the specified device.
(Note: the speaker playback only works under Winnt 4 and Win2K, but even though the 51xx phones cant save a ringtone to memory, you can still preview the ringtone through the phone)

Published

property OnPlayingTone : TPlayingTone read GetPlayingTone write SetPlayingTone;
This is were the event to indicate when a ringtone is being played is handled

property OnRingToneSent : TRingToneSent read eRingToneSent write eRingToneSent;
This is were the event to indicate if the ringtone was send via SMS successfully is handled

Extra Components

FBUSOpenPictureDialog

The FBUSOpenPictureDialog is a specially altered OpenDialog that will display a preview of not only BMP, and JPG images, but also GIF images, as well as Nokia Logo files, NOL. NGG and NLM.
It is used only by the LogoEditor, and is opened by calling **FBUSOpenPictureDialog1.execute**.
It returns a modal result, and a filename just like other dialogs.

FBUSStatusBar

The FBUSStatusBar has been specially written to work with the FBUS component and will automatically indicate events like, if the phone is connected, if the keyboard is locked, if the alarm is set, if a call is in progress, the power source of the phone, and the battery and signal levels.
It is designed to just drop on a form, and a property set in the FBUS component properties, and the source code does the rest. Automatically updating each item as its event changes.

FBUSVUMeter

The FBUSVUMeter is designed to work with the FBUSStatusBar, but it can be used on its own. It's a meter that shows levels like the Battery and signal levels, with a color representation of the level value, i.e. Green being good – excellent, Red being low – bad.

It is recommended that the demo be consulted for ideas on how to use all of he functions, procedures, and events in all the above Components and Classes.