www.heimetli.ch

Heimetli Software AG

HSWModule

Senden und Empfangen von SMS-Meldungen mit einem GSM-Modul

Version 1.01

5. August 2005

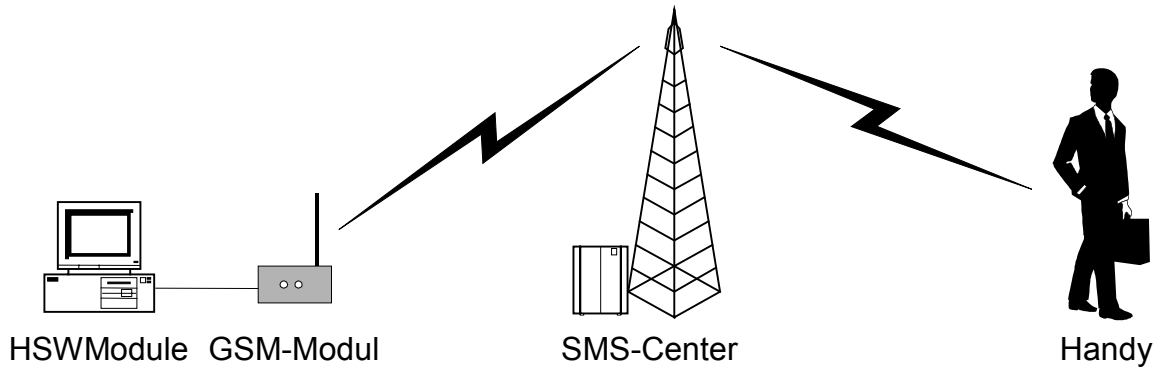
Inhaltsverzeichnis

Übersicht

Was ist HSWModule ?

HSWModule ist eine Dynamic Link Library (DLL) für Windows, die SMS-Meldungen über ein GSM-Modul senden und empfangen kann.

Die DLL verlangt folgende Konfiguration:



Um die DLL richtig einzusetzen, brauchen Sie also zusätzlich:

- Eine freie serielle Schnittstelle
- Ein GSM-Modul von Siemens (oder kompatibel)
- Eine SIM-Karte für das Modul
- Eine Programmiersprache die das Einbinden von DLLs erlaubt

Die Library verwendet AT-Befehle für die Kommunikation mit dem Modul.

Bei der Entwicklung der DLL wurde darauf geachtet, dass das Interface und der Einsatz von HSWModule möglichst einfach wird.

Installation und Interface

Installation

Für die Installation brauchen Sie nichts weiter zu tun als die **HSWModule.dll** aus dem Directory HSWModule auf der CD an einen Ort zu kopieren, wo Ihr Programm sie finden kann. Am besten haben Sie die DLL unter Kontrolle wenn Sie sie im gleichen Directory wie Ihre Applikation ablegen.

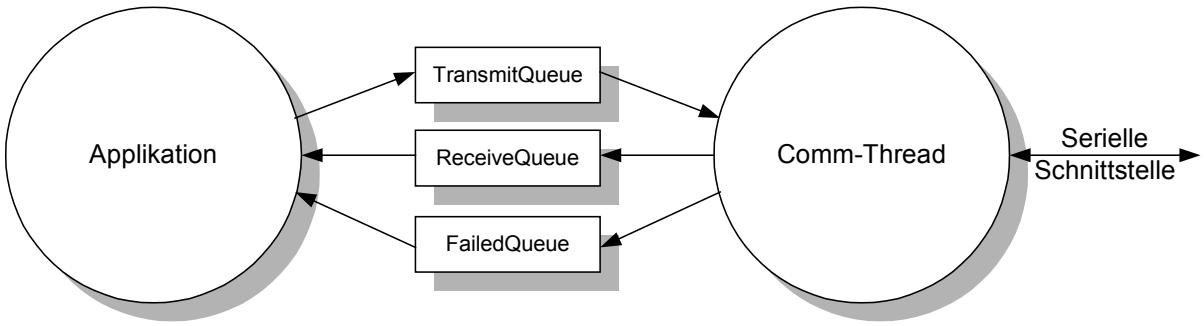
Auf der CD finden Sie ausserdem:

- **HSWModule\HSWModule.h** Das Headerfile für C / C++ - Programme
- **HSWModule\HSWModule.lib** Eine Import-Library die zu C/C++-Programmen gelinkt werden kann
- **C\TestLibrary.cpp** Ein simples Anwendungsbeispiel
- **Delphi\TestModule.pas** Ein Beispielprogramm mit der Deklaration der Library
- **Delphi\TestModule.dfm** Form zum Delphi-Beispiel
- **Delphi\TestLibrary.dpr** Projektfile für Delphi

Interface

Überblick

Um die Applikation so gut wie möglich vom Modul zu entkoppeln, hat die Library folgenden Aufbau:



Ein dedizierter Thread in der DLL hält die Verbindung mit dem GSM-Modul dauernd aufrecht. Wenn das Modul zum Senden bereit ist, holt er eine Meldung aus der TransmitQueue und gibt einen Sendeauftrag ans Modul.

Falls das Modul die korrekte Uebertragung bestätigt, wird der Eintrag in der TransmitQueue gelöscht.

Im Fehlerfalle versucht der Thread drei Mal, die Meldung abzusenden. Nach dem dritten Versuch gibt er auf und betrachtet die Meldung als unzustellbar. Auf Wunsch der Applikation werden solche Meldungen in die Failed-Queue geschrieben oder stillschweigend gelöscht.

Der Thread prüft periodisch, ob das Modul neue Meldungen empfangen hat. Die empfangenen Meldungen werden in die ReceiveQueue geschrieben, wo sie die Applikation abholen kann.

Funktionsaufrufe

Aus diesem Aufbau leiten sich folgende Funktionsaufrufe ab:

- **ActivateModule** Aktiviert die Library und startet den Thread
- **DeactivateModule** Deaktiviert die Library und stoppt den Thread
- **SendTextMessage** Stellt eine Meldung in die TransmitQueue
- **ReceiveTextMessage** Entnimmt eine Meldung aus der ReceiveQueue
- **GetFailedMessage** Entnimmt eine Meldung aus der FailedQueue
- **GetModuleState** Gibt den aktuellen Zustand des Moduls an

ActivateModule

Muss der erste Aufruf der Library sein. Er öffnet und initialisiert die Schnittstelle und konfiguriert und startet den Thread.

Prototype:

```
int ActivateModule( const char *port,        // Name der seriellen Schnittstelle
                    const char *pin,         // PIN-Code der SIM-Karte
                    const char *smsc,        // Nummer des SMS-Centers
                    const char *logfile,     // Name für das Logfile
                    int processfailed ) ;    // Behandlung der Sendefehler
```

Das **port** gibt an, an welcher Schnittstelle das Modul angeschlossen ist, zum Beispiel "COM1". Die DLL erwartet, dass es sich um eine normale Schnittstelle handelt, die mit CreateFile geöffnet, und mit SetCommState konfiguriert werden kann.

Der Parameter **pin** ist der PIN-Code für die SIM-Karte im Modul. Ueblicherweise ist das ein String mit 4 Zahlen, zum Beispiel "3731".

Bei **smsc** geben Sie die Nummer des SMS-Centers an. Dies ist notwendig, da das Modul die SMS nicht direkt an den gewünschten Teilnehmer senden kann, sondern über das SMS-Center gehen muss.

Die Nummer des SMS-Centers ist vom Betreiber des Mobil-Netzes festgelegt. Für die Swisscom ist dieser Parameter "+41794999000".

Ein **logfile** ist optional. Wenn Sie einen Filenamen angeben, schreibt die Library die wichtigsten Events in dieses File. Das Logfile kann Ihnen bei der Inbetriebnahme und der Fehlersuche der Library helfen, herauszufinden was abgelaufen ist.

Im Normalbetrieb können Sie den Wert NULL angeben und es wird kein Logfile geschrieben.

Mit **processfailed** bestimmen Sie, was mit Meldungen passiert, die nicht übertragen werden können. Wenn Sie 0 angeben, werden die Meldungen einfach gelöscht, bei allen anderen Werten wird die Meldung in die Failed-Queue geschrieben, wo Sie sie wieder abholen und analysieren können.

Bei einer erfolgreichen Ausführung des Aufrufes bekommen Sie den Wert HSWMOD_SUCCESS (0) zurück und sonst einen Fehlercode. Bei einem Fehler ist dafür gesorgt dass die Library wieder in den gleichen Zustand wie vor dem Aufruf kommt.

DeactivateModule

Die Hauptfunktion von DeactivateModule besteht darin, den Thread zu stoppen und die Schnittstelle wieder freizugeben. Da es ein paar Sekunden dauern kann, bis der Thread wirklich terminiert, kann diese Funktion die Applikation ein paar Sekunden blockieren.

Prototype:

```
int DeactivateModule( void )
```

Im Erfolgsfalle liefert die Funktion HSWMOD_SUCCESS (0) .

SendTextMessage

Schreibt eine Meldung in die TransmitQueue.

Prototype:

```
int SendTextMessage( const char *to, const char *text )
```

Das Argument **to** ist die Handy-Nummer des Empfängers als String mit bis zu 25 Charactern. Normalerweise sind dies eine Telephonnummer und eventuell ein führendes '+'.

Die Library gibt diesen String unverändert ans Modul weiter, Sie können also alles angeben, was vom Modul als gültige Nummer akzeptiert wird.

Das Argument **text** ist der Meldungstext von maximal 160 Charactern. Dabei gilt es zu beachten, dass das GSM-Alphabet viel weniger Zeichen hat als der ANSI-Zeichensatz von Windows !

Die Library konvertiert den Meldungstext so gut wie möglich, aber viele ANSI-Zeichen können einfach nicht dargestellt werden und werden zu Spaces konvertiert. Sie sind also gut beraten, wenn Sie den Text möglichst auf die druckbaren ASCII-Character beschränken.

Da der Thread asynchron zur Applikation läuft, wird die Meldung nicht sofort gesendet, sondern erst wenn das Modul dafür bereit ist. Wenn das Modul ein permanentes Problem hat, zum Beispiel eine ungültige SIM-Karte, wird die Meldung also gar nie gesendet ...

Aus diesem Grunde könnte es ratsam sein, mit diesem Aufruf zu warten bis **GetModuleState** den Wert STATE_READY zurückgibt.

ReceiveTextMessage

Holt eine Meldung aus der ReceiveQueue.

Prototype:

```
int ReceiveTextMessage( char *from, char *text )
```

Die beiden übergebenen Pointer sind also **OUT**-Parameter, das heisst, die Library schreibt die Handy-Nummer und den Meldungstext an die angegebene Stelle im Speicher.

Die DLL erwartet, dass die Handy-Nummer mindestens 25 Character aufnehmen kann, der Buffer also 26 Bytes gross ist. Der Text kann maximal 160 Character lang sein und benötigt deshalb einen Buffer von mindestens 161 Bytes.

Der Meldungstext beim Empfang vom GSM-Charactersatz in den ANSI-Charactersatz konvertiert.

Wenn keine Meldung in der ReceiveQueue ist, gibt die Funktion 0 zurück.

GetFailedMessage
Holt eine Meldung aus der FailedQueue.

Prototype:

```
int GetFailedMessage( char *to, char *text )
```

Die beiden übergebenen Pointer sind also **OUT**-Parameter, das heisst, die Library schreibt die Handy-Nummer und den Meldungstext an die angegebene Stelle im Speicher.

Die DLL erwartet, dass die Handy-Nummer mindestens 25 Character aufnehmen kann, der Buffer also 26 Bytes gross ist. Der Text kann maximal 160 Character lang sein und benötigt deshalb einen Buffer von mindestens 161 Bytes.

Wenn keine Meldung in der ReceiveQueue ist, gibt die Funktion 0 zurück.

GetModuleState
Da die Applikation nicht direkt mit dem Modul kommuniziert, hat sie auch keine Möglichkeit, Probleme des Moduls zu erkennen.Damit die Applikation nicht ins Leere arbeitet, stellt die Library die Funktion GetModuleState zur Verfügung.

Prototype:

```
int GetModuleState( void )
```

Wenn das Modul ordnungsgemäss arbeitet, gibt dieser Aufruf STATE_READY zurück.

Limiten

GSM-Module
Die aktuelle Version der Library unterstützt nur Module von Siemens oder tatsächlich hundert Prozent kompatible Produkte. Obwohl es eigentlich einen Standard gibt, kann man den offensichtlich verschieden auslegen ...

Windows 95
Die Library läuft nicht unter Windows 95, weil in dieser Version von Windows die CancelIo API-Funktion nicht existiert.

Beispielprogramm

```
#include <windows.h>
#include <stdio.h>

#include "HSWModule.h"

int main()
{
      // Activate the module
      if( ActivateModule("COM2","3731","+41794999000","test.log",TRUE)
                                                        != HSWMOD_SUCCESS )
      {
            printf( "Error: ActivateModule failed\n" ) ;
            return 1 ;
      }

      // Wait until the module is ready
      while( GetModuleState() != STATE_READY )
      {
            Sleep( 500 ) ;
            putchar( '.' ) ;
      }

      putchar( '\n' ) ;


      // Send a text message
      SendTextMessage( "0797790339", "Das ist ein Test" ) ;


      char phone[26] ;
      char text[161] ;

      // Get received messages
      while( ReceiveTextMessage(phone,text) )
            printf( "Received: %s \"%s\"\n", phone, text ) ;

      // Wait a bit for the module to send the message
      Sleep( 15000 ) ;

      // Get failed messages
      while( GetFailedMessage(phone,text) )
            printf( "Failed:   %s: \"%s\"\n", phone, text ) ;

      // Deactivate the module
      DeactivateModule() ;

      return 0 ;
}
```

Lizenzbestimmungen

Eine Lizenz pro Entwickler
Durch die Bezahlung der Lizenzgebühr erhält **ein Entwickler** das Recht, die DLL ohne weitere Kosten in all seinen Projekten einzusetzen. Wenn mehrere Entwickler in einer Firma die DLL einsetzen, muss jeder im Besitze einer eigenen Lizenz sein.

Royalties
Es gibt keine Royalties in irgendwelcher Form, das heisst der Entwickler darf eine beliebige Anzahl von Kopien der DLL für seine Projekte erstellen, und zusammen mit seiner Software weitergeben

Einschränkung
Die DLL darf nur als Teil einer Applikation weitergegeben werden.

Haftung
Wie bei Software üblich, kann die Heimetli Software AG keine Haftung für die Funktion dieses Programmes oder für Folgeschäden übernehmen. Der Anwender der DLL ist dafür verantwortlich, die Software in seiner Applikation so gut wie möglich zu testen. Die Heimetli Software AG haftet höchstens für den Betrag der Lizenzgebühr.

Kontaktadresse

Wir haben uns bemüht, die DLL und das Beschreibung so einfach wie möglich zu machen. Sollten trotzdem noch Fragen oder Probleme auftauchen, dann nehmen Sie doch bitte mit uns Kontakt auf:

Heimetli Software AG
Dorfstrasse 21
4613 Rickenbach

Tel: 062 / 216 29 07

Web: http://www.heimetli.ch

Index