UM0794
User manual

GSM interface library for surveillance applications

Introduction

GSM (global system for mobile communications) is an open, digital cellular technology used for transmitting mobile voice and data services. GSM supports voice calls and data transfer speeds of up to 9.6 kbit/s, together with the transmission of SMS (short message service).

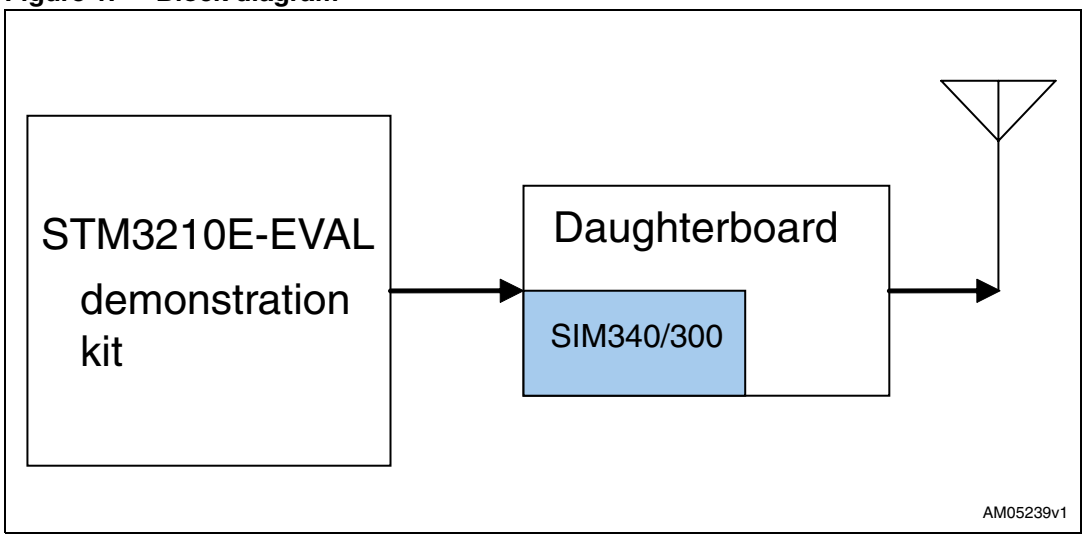
This document explains how to use the GSM interface library to configure and use the GSM module, and use the SMS feature for control and monitoring. This library is developed for the STM32 microcontroller and tested on the STM3210E-EVAL. It uses a UART interface to communicate with the GSM module. The firmware is in C language.

The objective of this library is to show the user the features and capabilities of the STM32 for automation and surveillance applications using the SMS feature of GSM. This library is provided with an application example, and is interfaced with SIM340 and SIM300 (GSM modules from SIMCOM).

This library communicates with the GSM module through AT commands over a UART Interface.

Figure 1 below shows the block diagram of this demonstration board.

Figure 1. Block diagram



Contents

List of tables

1 Getting started

1.1 Package contents

The GSM interface library package includes the following:

- SMS library
- Application example
  - To configure GSM module (SIM300/SIM340)
  - To send/receive SMS,
  - To set SMS parameters
- Documentation:
  - User manual

1.2 Hardware connection to run the application

To connect the GSM daughter board with the STM32 demonstration kit the user must perform the following:

1. Connect pin 1 of J1 (Rx pin of the SIM300 on the daughter board) to PA9 (STM32)
2. Connect pin 3 of J1 (Tx pin of the SIM300 on the daughter board) to PA10 (STM32)
3. Connect pin 2 of J9 (PWRKEY pin of the SIM300 on the daughter board) to PA2 (STM32)

Now common the grounds of both the boards and power-up the daughter board and then the STM32 demonstration kit.

If the DB9 connector is used, the user should first connect the DB9 connector and then the power supply adapter. Now, to turn on the GSM module, the user must press the SW2 switch for 2-3 seconds. Also to turn off the module the user should press the SW2 switch for 1.5 seconds.

The LED D6 shows the status of the GSM module:

Table 1. Network LED status

State	SIM300 function
Off	SIM300 is not running
64 ms on/800 ms off	SIM300 does not find the network
64 ms on/3000 ms off	SIM300 find the network
64 ms on/300 ms off	GPRS communication

2 SMS library

2.1 Getting started

In order to check that the GSM module is initialized properly or not, the user can send “AT” using the SendATCommand (u8 *u8_ATCommand, u8 u8_Length) function defined in the gsm_generic.c file. In response to this the GSM module returns “OK”.

2.2 SMS library architecture

The following architecture is used for the SMS library:

- Application layer (gsm_appli.c)
- SMS library (sms.c,atcommands.h)
- STM32 USART library (stm32f10x_usart.c)
- GSM stack (GSM module)

The files are organized in the following directory structure:

- Application: (user application layer files)
  - main.c
  - gsm_appli.c
  - stm32f10x_it.c
  - stm32f10x_vector.c
  - gsm_appli.h
  - stm32f10x_conf.h
  - stm32f10x_it.h
- SMS library
  - sms.c
  - sms.h
  - atcommands.h
- Library: the STM3210E-EVAL library

2.3 SMS library description

2.3.1 Set SMS mode

The following function can be used to set the SMS mode:

2.3.2 SMS mode initialization

Table 2. SMS mode configuration

Function name	Description	Input parameters	Output parameters
SMS_ModeConfig	This function configures the SMS mode. This may be Test mode or PDU mode	The enum of type SMSFromat_t. The value may be one of the following: – TEXT_MODE – PDU_MODE	The status of the command executed. The function returns true as soon as the mode is set to the desired one

2.3.3 Read SMS service centre number and parameters

Table 3. Read SMS service centre parameters

Function name	Description	Input parameters	Output parameters
SMS_ServiceCentreRead	This function reads the SMS service centre number, service centre address format, and SC alpha ID if enabled	The pointer of the message service centre structure (SMS_SCDetails_t).This routine fills the parameters in this location	u8_MsgSerCntrNo: service centre number in string format
			SMSCNumFormatVal: – NOT_SUPPORTED – UNKNOWN_ISDN – NATIONAL_ISDN – INTNATIONAL_ISDN – NET_SPECIFIC_ISDN
			u8_scaAlpha: alpha ID in string format(1)

1. It also returns the status of the command executed. The function returns true as soon as the parameters are filled in the passed location.

2.3.4 Set SMS service centre parameters

Table 4. Set SMS service centre parameters

Function name	Description	Input parameters	Output parameters
SMS_ServiceCentreSet	This function sets the SMS service centre number, service centre address format, and SC alpha ID if enabled	The pointer of the message service centre structure (SMS_SCDetails_t) that contains the service centre parameters. – u8_MsgSerCntrNo: service centre Number in String format – SMSCNumFormatVal: – NOT_SUPPORTED – UNKNOWN_ISDN – NATIONAL_ISDN – INTNATIONAL_ISDN – NET_SPECIFIC_ISDN – u8_scaAlpha: alpha data in string format (This is not required)	This returns the status of the command executed. The function returns true as soon as the parameters are set

2.3.5 Enable/disable new message alert

Table 5. Enable/disable new message alert

Function name	Description	Input parameters	Output parameters	Comment
SMS_NewSMSAlert	This function enables/disables the alert for new SMS receive.	The value of SetConfig Enum – ENABLE: enables the alert for new message – DISABLE: disables the alert for new message	This returns the status of the command executed. The function returns true as soon as the new configuration is set.	The user can also read and set the configuration parameters for new messages through the following function: bool NewSMSNotificationConfig(NewMsgNotifConfig *ps_NewMsgNitifConfigVal); For details please see the description of the above function

2.3.6 Enable/disable message submit status report

Table 6. Enable/disable message submit status report

Function name	Description	Input parameters	Output parameters
SMS_StausReport	This function enables/disables the SMS submit status report	The value of SetConfig Enum – ENABLE: enables the status report for SMS submit – DISABLE: disables the status report for SMS submit	This returns the status of the command executed. The function returns true as soon as the new configuration is set

2.3.7 Store SMS settings

Table 7. Store SMS settings

Function name	Description	Input parameters	Output parameters
SMS_StoreSettings	This function stores SMS settings for SMS mode (text or PDU), new message indication, detailed information display (+CSDH) from active memory to non-volatile memory	8-bit profile index ranging from 0 to 255 (this should be a number not a character)	This returns the status of the command executed. The function returns true as soon as the configuration is saved

2.3.8 Restore SMS settings

The user must ensure that the mentioned profile index exists.

Table 8. Restore SMS settings

Function name	Description	Input parameters	Output parameters
SMS_RetoreSettings	This function restores SMS settings for SMS mode (test or PDU), new message indication, detailed information display (+CSDH) from non-volatile memory to active memory	8-bit profile index ranging from 0 to 255 (this should be a number not a character) to be restored from the memory	This returns the status of the command executed. The function returns true as soon as the mentioned profile is successfully retrieved

2.3.9 Reading SMS mode

Table 9. Reading SMS mode

Function name	Description	Input parameters	Output parameters
SMS_ModeRead	This function returns the mode of the SMS	None	This returns the SMS mode. – PDU_MODE – TEXT_MODE

2.3.10 Sending SMS

Table 10. Sending SMS

Function name	Description	Input parameters	Output parameters
SMS_Send	This function sends SMS in PDU mode	The following parameters are needed for this function: – Pointer to message string – Destination number in string format	This function returns the message reference number that can be used for message delivery status

Note: 1 If the message string is in ASCII (for extended character set it is ISO-8859-1) format, the user must de-comment the #define ASCII_ISO8859_1 macro in the SMS.h file. In this case the string must be terminated by NULL character.

2 If the message is already in GSM alphabet, the string must be terminated by 0xFF.

2.3.11 Reading SMS

Table 11. Reading SMS

Function name	Description	Input parameters	Output parameters
SMS_Read	This function receives the SMS from the preferred storage	The following parameters are needed to read the SMS: – Index of the SMS to be read (number, not string) – Status change flag – False: normal mode (status change from unread to read if the message is unread) – True: the status of the message remains the same – Pointer of the PDUSMReceiveStruct_t Structure	This function fills the PDUSMReceiveStruct_t type pointer location passed by the user with SMS details: – u8_SMSCLength: service centre number length – b_MoreDataIndic: reserved – u8_SMSCNum: service centre number in string format – u8_PhoneNumLen: sender's number length – u8_SenderName: name of the sender if available in address book – u8_PhoneNum: phone number of the sender – u8_TimeStamp: time Stamp(YY/MM/DD,HH:MM:SS:xx) – u8_UserDataLen: length of the SMS – u8_MsgContent: SMS content in string format

2.3.12 Delete SMS

Table 12. Delete SMS

Function name	Description	Input parameters	Output parameters
SMS_Delete	This function deletes the SMS from the specified storage (Memory1)	The index of the SMS to be deleted (in number format, not string)	This returns the status of the command executed. – True: SMS is deleted – False: SMS is not deleted

2.3.13 Write SMS to memory

Table 13. Write SMS to memory

Function name	Description	Input parameters	Output parameters
SMS_WritetoMemory	This function writes the SMS to memory	The following parameters are needed for this function: – Pointer to message string – Destination number in string format	This function returns SMSSendINdex_t type structure that contains the following 1. u16_RefNum: message reference number (Number format)
			– b_CMSErrOcurred: this flag indicates the following: – True: CMS Error has occurred – False: message written properly and user can use the reference number
			u16_CMSErrIndex: This Indicates the CMS Error Index
			b_CMEErrOcurred: This flag indicates the following:
			– True: CME Error has occurred – False: message written properly and user can use the reference number
			5. u16_CMEErrIndex: this indicates the CME Error Index

2.3.14 Send SMS from memory

Table 14. Send SMS from memory

Function name	Description	Input parameters	Output parameters
SMS_SendFromMemory	This function sets the SMS storage memory	The index of the SMS to be sent (in number format, not string)	This function returns the message reference number that can be used for the message delivery status

2.3.15 Set SMS storage memory

Table 15. Set SMS storage memory

Function name	Description	Input parameters	Output parameters
SMS_StorageMemorySet	SMS_StorageMemoryRead	The pointer to the SMStorageParam_t type structure that contains – u8_Memory1[]: messages to be read and deleted from this memory storage – u8_Memory2[]: message is written and sent from this memory storage – u8_Memory3[]: received message is placed in this memory storage if routing to PC is not set(1)	This function returns the status of the command and also fills the passed location with the number of used locations and total number of available locations

1. Example: u8_Memory2[]=”SM” for SIM card storage

2.3.16 Read SMS storage

Table 16. Read SMS storage

Function name	Description	Input parameters	Output parameters
SMS_StorageMemoryRead	This function sets the SMS storage memory	The pointer to the SMStorageParam_t type Structure	The parameters are filled in the location passed. The details are as follows: – u8_Memory1[]: messages to be read and deleted from this memory storage – u8_Memory2[]: messages are written and sent from this memory storage – u8_Memory3[]: received messages are placed in this memory storage if routing to PC is not set – u16_Memory1Total: total number of locations available in memory1 – u16_Memory1Used: number of locations already used – u16_Memory2Total: total number of locations available in Memory2 – u16_Memory2Used: number of locations already used – u16_Memory3Total: total number of locations available in memory3 – u16_Memory1Used: number of locations already used (1)

1. The status of the execution of the routine is also returned. As soon as the parameters read, this function returns true.

2.3.17 Enable/disable alpha ID

Table 17. Enable/disable alpha ID

Function name	Description	Input parameters	Output parameters
SMS_ConfigAlphaID	This function enables/disables the alpha ID	The value of SetConfig type enum – ENABLE: the alpha ID is enabled – DISABLE: the alpha ID is disabled	The status of the execution of the routine is returned. As soon as the alpha ID is enabled/disabled, this function returns true.

2.3.18 Read alpha ID

Table 18. Read alpha ID

Function name	Description	Input parameters	Output parameters
SMS_ReadAlphaID	This function reads the alpha ID	None	The following is returned by the routine – True: the alpha ID is enabled – False: the alpha ID is disabled

2.3.19 Set SMS validity period

Table 19. Set SMS validity period

Function name	Description	Input parameters	Output parameters
SMS_SetValidityPeriod	This function sets the validity period of the SMS	The SMS validity period in minutes (32 bit)	None

2.3.20 Enable/disable extra information

Table 20. Enable/disable extra information

Function name	Description	Input parameters	Output parameters
SMS_ConfigExtraInfo	This function enables/disables the display of extra information	The value of SetConfig type enum – DISABLE: extra information display is disabled – ENABLE: extra information display is enabled	The status of the routine execution. – True: successful execution – False: execution unsuccessful

2.3.21 Write message to memory

Table 21. Write SMS to memory

Function name	Description	Input parameters	Output parameters
SMS_WritetoMemory	This function writes SMS to memory	– -pu8_Number: Number in String format (This is optional) – -pu8_SMSString: SMS content	– SMSSendIndex_t Structure that contains the following: – 1. u16_RefNum: Message reference Number – 2. b_CMSErrOccurred: This flag indicates the following: – -TRUE: CMS Error has occurred – -FALSE: Message written properly and the user can use the reference number – 3. u16_CMSErrIndex: This indicates the CMS Error Index – 4. b_CMEErrOccurred: This flag indicates the following: – -TRUE: CME Error has occurred – -FALSE: Message written properly and the user can use the reference number – 5. u16_CMEErrIndex: This indicates the CME Error Index

2.3.22 Enable/disable unsolicited messages

Table 22. Enable/disable unsolicited messages

Function name	Description	Input parameters	Output parameters
SMS_ConfigUnsolicitMsg	This function enables/disables the display of unsolicited messages	The value of SetConfig type enum – DISABLE: unsolicited messages display is disabled – ENABLE: unsolicited messages display is enabled	The status of the routine execution. – True: successful execution – False: execution unsuccessful

2.3.23 Read unsolicited messages display status

Table 23. Read unsolicited messages display status

Function name	Description	Input parameters	Output parameters
SMS_ReadUnsolicitMsgStatus	This function returns the status of the display of unsolicited messages status	None	The value of SetConfig type enum – False: extra information display is disabled – True: extra Information display is enabled

2.3.24 Encoding the SMS parameters to PDU

Table 24. Encoding the SMS parameters to PDU

Function name	Description	Input parameters	Output parameters
SMS_EncodePDUString	This routine receives the message structure and converts it into PDU string	The pointer to the PDUSMSendStruct _t type structure and string into which the PDU is assigned	The decoded PDU is assigned to the string pointer passed in the function. Also the status of the routine execution is returned. – True: successful execution – False: execution unsuccessful

2.3.25 Copying strings from one to another

Table 25. Copying strings from one to another

Function name	Description	Input parameters	Output parameters
UserStrcpy	This routine copies the string argument 1 into string argument 2	The pointer to the strings	The string argument 1 is copied into string argument 2. It also returns the length of the string

2.3.26 Hexadecimal to character string conversion

Table 26. Hexadecimal to character string conversion

Function name	Description	Input parameters	Output parameters
HexNoToCharConverter	This routine converts the 16-bit hex digit into string	The 16-bit digit and the pointer to the string	The converted string is copied into the string passed

2.3.27 Character string to hexadecimal conversion

Table 27. Character string to hexadecimal conversion

Function name	Description	Input parameters	Output parameters
CharToHexDigitConverter	This routine converts the string up to 5-character-long 16-bit hex digit	The pointer to the string to be converted into hexadecimal digit	The converted hexadecimal digit is returned

2.3.28 Character string to octet conversion

Table 28. Character string to octet conversion

Function name	Description	Input parameters	Output parameters
CharToOctat	This routine converts the string up to 2-character-long octet	The pointer to the string to be converted into octet	The converted hexadecimal digit is returned

2.3.29 Encoding of SMS content to octets

Table 29. Encoding of SMS content to octets

Function name	Description	Input parameters	Output parameters
SMS_EncodeSMSContent	This routine encodes the text message content in to octets of the data section of the PDU	The pointer to the message string and the length of the string	The decoded PDU octet is available in the same string and also the function returns the length of the data octets

2.3.30 Calculation of length of string

Table 30. Calculation of length of string

Function name	Description	Input parameters	Output parameters
StringLength	This routine calculates the length of the string up to 255 characters long	The pointer to the string	The calculated length of the string is returned back

2.3.31 Decoding the data section of PDU

Table 31. Decoding the data section of PDU

Function name	Description	Input parameters	Output parameters
SMS_DecodeSMSContent	This routine decodes the TPDU to string	The pointer to the TPDU string and the length	The decoded string is copied into the same pointer and the length of the decoded string is returned

2.3.32 Decoding the SMS deliver status PDU

Table 32. Decoding the SMS deliver status PDU

Function name	Description	Input parameters	Output parameters
SMS_DecodeStatusPDU	This routine decodes the SMS status PDU	The pointer to the location of the PDU and the length of the PDU	The decoded status report is filled in the status buffer (SMSStatusReportBuff[u8_gStatusBuff Front-1]). The status of the SMS is also returned. – True: the SMS is delivered – False: the SMS is in pending state

2.3.33 Conversion between GSM alphabet and the ASCII format

Table 33. Conversion between GSM alphabet and the ASCII format

Function name	Description	Input parameters	Output parameters
SMS_GSMAlbhabettoISO88591	This routine converts GSM alphabet to ASCII character string	The pointer to the string to be converted to ASCII	The converted string is available in the same string. The length of the string after conversion is also returned back

Table 34. Conversion from ASCII (for extended ISO-8859-1) to GSM alphabet

Function name	Description	Input parameters	Output parameters
SMS_ISO88591toGSMAlbhabet	This routine converts the string from ASCII to GSM alphabet	The pointer to the string to be converted to GSM alphabet	The converted string is available in the same string. The length of the string after conversion is also returned back

2.3.34 The routines for definition of delay

Table 35. SMS_Delay

Function name	Description	Input parameters	Output parameters
SMS_Delay	This function offers delay in the execution	The delay count (16-bit). For determining the exact value delay see the routines below	The successful completion of delay returns true

Table 36. Decrement_TimingDelay

Function name	Description	Input parameters	Output parameters
Decrement_TimingDelay	This routine decrements the delay counter	None	None

Table 37. SysTick_configuration

Function name	Description	Input parameters	Output parameters
SysTick_Config	This routine configures the SysTick for delay. For example, if the micro is running at 72 MHz and for 1ms delay counter decrement period, the value of SYSTIC_COUNTER is 72000	None	None

3 Application example

3.1 Sending an SMS

To send an SMS the user must do the following:

1. Set the SMS-PDU mode
2. Enable the status report
3. Send the SMS and read the SMS reference number returned by the SMS send routine
4. Check the status report for the above reference number

The example code for SMS send with status report enabled is below. For details see the gsm_appli.c, the send SMS section in the ApplicationRun() routine.

```
    /*Set the PDU Mode*/
   b_TestStatus=SMS_ModeConfig(PDU_MODE);
    if(TRUE==b_TestStatus)
    {
      u8 u8_Temp=0;
      u16 u16_SMSRefNo=0;/*SMS Send Reference Number*/
      u8 u8_ATC12[]="AT+CSMS=0";
      u8 u8_PhNo[]="+xxxxxxxxxxxx";/*SMS Destination Number*/
      u8 puc8_SMS_String[]="Test SMS";/*Message Content*/
      b_gCurrentSMSMode=FALSE;/*Current SMS Mode indicator Flag*/
      /*Enable Status Report*/
      SMS_StausReport(ENABLE);
      u16_SMSRefNo=SMS_Send(u8_PhNo, puc8_SMS_String);
      while(b_gStatusReportReceived == FALSE);//Wait Until Status
Report Received
      if
(SMSStatusReportBuff[u8_gStatusBuffRear].u8_SMReferenceNo==u16_SMSR
efNo)
      {
        if(SMSStatusReportBuff[u8_gStatusBuffRear].u8_MSGStatus==0)
        {
          /*Messsage is delivered*/
        }
        else
        {
          /*Message pending*/
```

```
        }
        /*Clear the SMSStatusReportBuff[u8_gStatusBuffRear]
structure members*/
        SMSStatusReportBuff[u8_gStatusBuffRear].u8_MSGStatus=0;
        SMSStatusReportBuff[u8_gStatusBuffRear].u8_SMReferenceNo=0;
        for (u8_Temp=0;u8_Temp<14;u8_Temp++)
        {

SMSStatusReportBuff[u8_gStatusBuffRear].u8_SMSCTimeStamp[u8_Temp]=0
;

SMSStatusReportBuff[u8_gStatusBuffRear].u8_SMSCDelieverTime[u8_Temp
]=0;

SMSStatusReportBuff[u8_gStatusBuffRear].u8_SMRcvrNo[u8_Temp]=0;
        }
        /*Now point the Rear pointer of the circular buffer to the
next element*/
        u8_gStatusBuffRear++;
        if(u8_gStatusBuffRear>=STATUS_PDU_QUEUE_LEN)
        {
          u8_gStatusBuffRear=0;
        }
        /*Clear the Flag if there is no SMS Status Report to Read*/
        if(u8_gStatusBuffRear==u8_gStatusBuffFront)
        {
          b_gStatusReportReceived=FALSE;
        }
      }
    }/******* SMS Send with Status Report Ends Here
******************/
```

3.2 Receiving an SMS

To receive an SMS the user must do the following:

1. Set the SMS-PDU mode
2. Enable the new message indication
3. Read SMS

The example code is given below, for details refer to the gsm_appli.c file

```
/**********************Receiving new Message
******************/
     /*  Set the PDU Mode   */
    b_TestStatus=SMS_ModeConfig(PDU_MODE);
    if(TRUE==b_TestStatus)
    {
      /*  Wait Until new message received */
      while(b_gNewMsgReceived == FALSE);
      {
        e_SendorIdentityVal=UNKNOWN_NUM;

SMS_Read(u16_NewMessageLocBuffer[u8_gBuffRear],FALSE,&s_RcvStruct1)
;
        /*Clear the location in the New mesage Index buffer*/
        u16_NewMessageLocBuffer[u8_gBuffRear]=0;
        u8_gBuffRear++;
        if(u8_gBuffRear>=NEW_MSG_BUFFLEN)
        {
          u8_gBuffRear=0;
        }
        /* Clear the Flag if there is no other new message to read*/
        if(u8_gBuffRear==u8_gBuffFront)
        {
          b_gNewMsgReceived=FALSE;
        }
      /*   Check for Message Sendor Authorization
*/
        if (TRUE ==
UserStrcompare(u8_AdminNumber,s_RcvStruct1.u8_PhoneNum))
        {
          e_SendorIdentityVal=ADMIN_NUM;
          /*Message Sendor is authorised and user can decode and
process the command here*/
          ApplicationSMSContentDecode();
        }
```

```
        else if(TRUE
==UserStrcompare(u8_UserNumber1,s_RcvStruct1.u8_PhoneNum) )
        {
          e_SendorIdentityVal=USER1_NUM;
          /*Message Sendor is authorised and user can decode and
process the command here*/
          ApplicationSMSContentDecode();
        }
        else if(TRUE
==UserStrcompare(u8_UserNumber1,s_RcvStruct1.u8_PhoneNum) )
        {
           e_SendorIdentityVal=USER2_NUM;
          /*Message Sendor is authorised and user can decode and
process the command here*/
          ApplicationSMSContentDecode();
        }
        else
        {
           e_SendorIdentityVal=UNKNOWN_NUM;
          /*User is not authorised and if needed then system can
inform the admin*/
          ApplicationUnauthorizedNotification();
        }
      }
    }
```

Note 1: The Data length more than 159 is not supported in this version of library.

3.3 Deleting an SMS

To delete an SMS the user must do the following:

1. Set the delete SMS index
2. Call delete SMS routine

If the delete SMS routine returns TRUE the SMS is deleted. The following is the example code to show how to use the SMS_Delete routine:

```
b_SMDeleteFlag=SMSDelete(u16_SMDeleteIndex);

if(TRUE==b_SMDeleteFlag)

 {

     //Message is Deleted Successfully

 }
```

4 Revision history

Table 38. Document revision history

Date	Revision	Changes
03-Jun-2010	1	Initial release.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com