[1] [] 1 [**Generated on Thu Sep 8 16:14:35 2005 for API for WSWUP-DLL by Doxygen**] []**Generated on Thu Sep 8 16:14:35 2005 for API for WSWUP-DLL by Doxygen**

API for WSWUP-DLL Reference Manual

Generated by Doxygen 1.3.3

Thu Sep 8 16:14:35 2005

Contents

Chapter 1

The SWUP-API

1.1 Version-Control

- 2001-09-07, J. Kuenstner:
  - introduced this version-control
  - Added a lot comments about the different distributions of the library
  - Added some details about error-handling
  - Added the error-demo
- 2001-10-21, J. Kuenstner:
  - New function : WSwup_InstallSoftwareInfoCallBackFunction() and MobileSwUpdateInfo-CallBack()
  - Better documentation of the error-struct
- 2001-11-23, J. Kuenstner:
  - New functionality of function : WSwup_CheckUpdateSuccess()
- 2002-02-25, J. Kuenstner:
  - Changes on functionality for : WSwup_InstallSoftwareInfoCallBackFunction() and MobileSw-UpdateInfoCallBack()

1.2 Introduction

The SWUP-API is provided to enable ”everyone” to build it’s own WIN-SWUP-Tool. Mainly it’s used for :

- WINSWUP itself
- Updating in the factory till project-milestone M3
- Updating of SW in I&T
- Updating of Mobile-SW for the Enduser

Depending on the library used and the choosen mobile it is possible to write a more or less secure update even for customers.

But this only applies for the new **K45-Family** together with the complete statically linked version of the library without the Update-SW and with heap-reading, see Library Versions .

All other libraries coverd with this document are **NOT** in the state to generate a trader-tool or worst a enduser-tool.

(Reason : In the moment it consists of several DLLs, which is very usefull for mobile development and for production- purposes, but it is very dangerous for usage at endusers-sites (or hackers) because everyone can ”play” with DLLs).

1.3 Library Versions

Currently there are three different versions of the library available:

- One ”library” that consists of several DLLs to allow update for different projects like winswup. In this library, the reading of the XBI-File is done by the library itself (**wswplibf.lib and wswplibf.dll**)
- One ”library” that consists of several DLLs to allow update for different projects like winswup. In this library the memory-allocation and reading in of the XBI-File must be done be the user of the library (**wswplibh.lib and wswplibh.dll**). This is done to allow some (crypto-) coding of the XBI-File on the way to the enduser and to decode the XBI-File from within the executable (this means : your program)

  So the decode-algorithm is not part of the library and can be changed ”dynamically”
- The third library is a real library, it does not consist of different DLLs (**wswplibh_stat.lib**).

  In this real library the user of the library also is responsible for allocating memory and reading in the XBI-File, this is not done by library (for the same reason like above)

  AND : This library is only for use for the new K45-Family, because it only supports the new upgrade-concept that was introduced with the K45-family. This means that the Library will **NOT** contain any kind of update-SW (the SW that runs in Mobile-RAM and erases and programs the flash during SW-Update) and has no possibility to load one. This means that the user of the library only can run the new update-concept and that he **must** call the function WSwup_SetUpdateConceptMode(enForceNewConcept) !

In future there will be the need for more librarys:

- A completely static linked library containing the update-SW for each possible update-SW

  ex: for SL45-Java - Update from a normal SL45.

  We cant go in this case with the new update-concept introduced with K45 because the ”old GSM-Mobile-SW” must be prepared for this update-concept. Therefore the SL45-Update can’t go with the new Update-Concept and therefore must either call WSwup_SetUpdateConceptMode(enForce-BSLBehaviour) or leave the function-call, because enForceBSLBehaviour is standard in the moment !

1.4 Usage

Here you have to distinguish between the new, more or less enduserproof update-concept introduced with the K45-Family New K45-Library or the old behaviour using boot-strap-loader and update-sw on PC-side

Old Pre-K45-Library

1.4.1 New K45-Library

To perform a complete Software-Update with the new concept the following steps have to be performed:

1. Set update-concept to enForceNewConcept via WSwup_SetUpdateConceptMode()
2. Find out the filesize of the XBI-File, allocate memory for it and read it in your heap
3. Let the library check the contents of ”your” heap and analyse the read in file
4. Check if the Mobile is connected, perform some tests and start the sowftware-update via function WSwup_CheckUpdateVoltage()
5. Continue the whole software-update, using the Software-Update-Functions
6. Check if the Software-Update was successful, also using the BFB-Functions

1.4.2 Old Pre-K45-Library

To perform a complete Software-Update following the old concept (with bootstrap-loader and update-sw on PC-Side) the following steps have to be performed:

1. Set update-concept to enForceBSLBehaviour via WSwup_SetUpdateConceptMode()
2. Find out the filesize of the XBI-File, allocate memory for it and read it in your heap
3. Let the library check the contents of ”your” heap and analyse the read in file
4. Check if the Mobile is connected, perform some tests and switch off the mobile, using the BFB-Functions
5. Perform the whole software-update, using the Software-Update-Functions
6. Check if the Software-Update was successful, also using the BFB-Functions

Most of the functionality contained in the library gives feedback to the user of the library via callbackfunctions. For some functionality of the library the user must install these callbackfunctions first, before the functionality itself can be called. In other cases the user must provide the callbackfunctions direct on call of the functionality itself. For a very basic software-update-functionality nearly all callbackfunctions can be left empty, they are only provided to give the user of the library the possibilty to give some feedback to the enduser (like progress-bars ...) **Never ever give a NULL-Pointer as function-pointer, currently this is not checked in the library !**

1.5 Error-Handling

Dont use the function WSwup_GetLastError() because the function:

- only returns a number
- will be deleted in one of the next versions, it is replaced by WSwup_GiveLastError()

Most funtions in the library return BOOL values, where TRUE says : ”everything okay” and FALSE says: ”an error occured”. Now you have different possibilities to find out more about the error and to show the user a text-string.

1.5.1 You are happy with english strings

If you are happy with english strings, just call the function WSwup_GetLastErrorString().

This function returns an english string containing some description about the error in english language. If you want to create a enduser-tool which only works with english language (who wants to do this ?), then you should call before retrieving the error-string the function WSwup_SetErrorTextBehaviourToEnduser() with TRUE as parameter, so that not all internal details are presented to an enduser. But I assume that someone who wants to build an enduser-tool also must support different languages. So we come to:

1.5.2 You need internationalisation

If you need to generate different languages in your tool, you must deal with errors yourself. This means you must call WSwup_GiveLastError() and handle all the text-formatting yourself. If you want to see how to do this, look at A demonstration how to handle error-strings youself . This means that you have to translate all the error-strings provided in module err_text.h to the different languages and the loop yourself through you new array and do the parameter-substitution yourself like in function SpecialSprintf in the mentioned error-demo.

1.6 A very simple demo

A very simple CMD-Line-Demo could be implemented like the following program. Since nearly all functionality is done in a thread, this demo has to wait in a busy-loop (**Attention : this gives 100% CPU-Load**) until either the MobileSwUpdateSuccessCallBack() - function or the MobileSwUpdateErrorCallBack() - function is called in from inside the library. This looks a little bit strange for a cmd-line-implementation, but is a common way under a gui to react on messages or events. The Demo for the WIN-SWUP-API

1.7 A demonstration how to handle error-strings youself

If you want to show the enduser strings in different languages you have to handle error-strings yourself. This program is a simple demo how to use the also provided files err_text.h and werrenum.h and how to loop through the table and do your parameter-exchangement. The Demo for the ERROR-Handling

1.8 The different function-groups

Init-Function, must be called before usage of the library

Init-Function for the whole Library/DLL

Bfb-Functions, used before and after the update itself to check consistence

BFB-Lib-Functions, used before and after SOFTWARE-Update

Comport-Information-Functions, Checking if comport is available and capable of this speed

Comport-Information-Functions

DLL-Version-Information-Functions, to retrieve information about the different loaded DLLs

DLL-Version-Information-Functions

Synch-Station specific Functions, for dealing with synchstations (only works with old synch-stations)

Synch-Station specific Functions

ERROR-Functions, to get information about the last ERROR

ERROR-Functions, to get information about the last ERROR

FILE-Functions for loading and checking the files with the new software

FILE-Functions

DEBUG-Functions for en/disabling OnLine-Debugging, only works in in the debug-version with an DebugTextViewer

DEBUG-Functions for en/disabling OnLine-Debug

SOFTWARE-Update-Functions, for the Software-Update itself

SOFTWARE-Update-Functions

Chapter 2

API for WSWUP-DLL Module Index

2.1 API for WSWUP-DLL Modules

Here is a list of all modules:

Chapter 3

API for WSWUP-DLL Data Structure Index

3.1 API for WSWUP-DLL Data Structures

Here are the data structures with brief descriptions:

Chapter 4

API for WSWUP-DLL File Index

4.1 API for WSWUP-DLL File List

Here is a list of all files with brief descriptions:

Chapter 5

API for WSWUP-DLL Page Index

5.1 API for WSWUP-DLL Related Pages

Here is a list of all related documentation pages:

Chapter 6

API for WSWUP-DLL Module Documentation

6.1 Init-Function for the whole Library/DLL

Defines

- #define WSWUPLIB_INTERFACE_VERSION_MAJOR 1

  Interface-Version of the whole library (this file), must be increased on interface-changes.

- #define WSWUPLIB_INTERFACE_VERSION_MINOR 37

  Interface-Version of the whole library, minor-number.

Functions

- BOOL SWUPLIBDLLEXIMPORT WSwup_InitLibrary (WORD wMajorNumber, WORD wMinor-Number)

  This function must be called by every user of the library once on starting.

6.1.1 Define Documentation

6.1.1.1 #define WSWUPLIB_INTERFACE_VERSION_MAJOR 1

Interface-Version of the whole library (this file), must be increased on interface-changes.

6.1.1.2 #define WSWUPLIB_INTERFACE_VERSION_MINOR 37

Interface-Version of the whole library, minor-number.

6.1.2 Function Documentation

6.1.2.1 BOOL SWUPLIBDLLEXIMPORT WSwup_InitLibrary (WORD *wMajorNumber*, WORD *wMinorNumber*)

This function must be called by every user of the library once on starting.

This intention of this function is to make a interface-compatibility-check at runtime, so every user of the library has to call this function once before using the library.

The user has to pass the two defines:

- WSWUPLIB_INTERFACE_VERSION_MAJOR
- WSWUPLIB_INTERFACE_VERSION_MINOR

to the function. If the given version is different from the one which the library expects, (the one it had on its own compile-time) the function will return FALSE and the library will not work. This is to prevent library-users from unexpected runtime-errors in their executables due to a new interface and new DLLs but an old Executable !

Parameters:

wMajorNumber : must be the given define WSWUPLIB_INTERFACE_VERSION_MAJOR

wMinorNumber : must be the given define WSWUPLIB_INTERFACE_VERSION_MINOR

Return values:

TRUE -> Library is initialised, version-information is ok

FALSE -> version-check failed, library does not work

6.2 Comport-Information-Functions

Functions

- BOOL SWUPLIBDLLEXIMPORT WSwup_CheckComPortAndSpeed (t_SwupCom WhichCom, DWORD dwBaudrate)

    This function only tries to open the selected comport with the given baudrate.

6.2.1 Function Documentation

6.2.1.1 BOOL SWUPLIBDLLEXIMPORT WSwup_CheckComPortAndSpeed (t_SwupCom *WhichCom*, DWORD *dwBaudrate*)

This function only tries to open the selected comport with the given baudrate.

In all cases (besides comport already open) the comport will be closed after this try. It returns the result from opening the comport, so its possible to determine if the win-system allows this baudrate on this comport. There is no guaranty, that the update with this baudrate will work !

Parameters:

WhichCom a t_SwupCom-enum, which comport should be tried

dwBaudrate a dword meaning the Baudrate at which the comport should be tried

Return values:

TRUE It is possible (from WIN-System-side) to open the Comport at this baudrate

FALSE Either Comport not available or this baudrate not possible

6.3 DLL-Version-Information-Functions

Data Structures

- struct t_VersionInformation

  Version-Information for the different parts of the Library.

Enumerations

- enum t_InfoSelector { enSwupLibDll, enSwupSeriDll, enBootStrapDll, enUpdateSW }

  Different parts of the whole library, for which the user can request version-information.

Functions

- BOOL SWUPLIBDLLEXIMPORT WSwup_GiveVersionInformation (t_VersionInformation ∗p-Information, t_InfoSelector WhichInfo)

  Returns Version Information about the different Parts of WSWUPLIB.

6.3.1 Enumeration Type Documentation

6.3.1.1 enum t_InfoSelector

Different parts of the whole library, for which the user can request version-information.

Give one of this values to function WSwup_GiveVersionInformation() and it will return the information about the requested part.

Enumeration values:

enSwupLibDll information about the whole wrapper

enSwupSeriDll information about the serial DLL

enBootStrapDll information about the BootStrapLoader-DLL itself

enUpdateSW information about the MobileSW-Part that is contained in the BSL-DLL

6.3.2 Function Documentation

6.3.2.1 BOOL SWUPLIBDLLEXIMPORT WSwup_GiveVersionInformation (t_VersionInformation ∗ *pInformation*, t_InfoSelector *WhichInfo*)

Returns Version Information about the different Parts of WSWUPLIB.

Since the SWUPLIB-functionality is contained in 3 different DLLs, and one of them is dynamically loaded for different processor-types, it is possible to retrieve some information about the DLLs.

Parameters:

WhichInfo a t_InfoSelector , about which part the information is to be retrieved

pInformation a ptr to a t_VersionInformation , that will be filled with the requested information. Attention: pInformation->nStructSize must be filled from the caller with sizeof(t_VersionInformation) to avoid conflicts on future enhancements

Return values:

TRUE Information-struct is filled with info

FALSE Information not available, ex. the DLL is not yet loaded

6.4 Synch-Station specific Functions

Functions

- void SWUPLIBDLLEXIMPORT WSwup_PrepareForUpdateWithSynchStation (BOOL fSynch-StationPresent)

  Prepares for performing the update via Synch-Station.

- BOOL SWUPLIBDLLEXIMPORT WSwup_ReloadUpdateDll (void)

  Reloads the DLL that is responsible for performing the Update.

6.4.1 Function Documentation

6.4.1.1 void SWUPLIBDLLEXIMPORT WSwup_PrepareForUpdateWithSynchStation (BOOL *fSynchStationPresent*)

Prepares for performing the update via Synch-Station.

The Bootstrap-Loader for a Synch-Station is different to a normal BSL. Since the Bootstraploader-DLL is dynamically loaded on File-Reading, this function must be called in advance, to force loading a different, synch-station-specific DLL on FileReading

Parameters:

fSynchStationPresent Boolean Flag for presence/absence of Synch-Station

Returns:

void

6.4.1.2 BOOL SWUPLIBDLLEXIMPORT WSwup_ReloadUpdateDll (void)

Reloads the DLL that is responsible for performing the Update.

If - after reading the file - the decision is made, that this update is via a synch-station, the DLL must be relodaded, in order to get the correct Bootstraploader. =>

- call WSwup_PrepareForUpdateWithSynchStation(TRUE)
- call this function

Return values:

TRUE Everything is okay

FALSE something went wrong

6.5 ERROR-Functions, to get information about the last ERROR

Data Structures

- struct t_ErrorStruct

  A structure to allow passing of parameters for error-messages from the Lib to the user of the Lib.

Defines

- #define MAX_ENTRIES 50

  Amount of entries in the error-"stack".

- #define NO_PARAM ""

  an empty parameter-list

- #define SIZE_OF_PINFO 10

  max amount of parameters for errors

- #define SIZE_OF_PSTRING 2000

  maximum size of a error-string-parameter

Functions

- DWORD SWUPLIBDLLEXIMPORT WSwup_GetLastError (t_SwupCom ComPort)

  Returns an specific Error-Code about last Error on this comport.

- void SWUPLIBDLLEXIMPORT WSwup_SetErrorTextBehaviourToEnduser (BOOL fEndUser)

  Switches the behaviour of function WSwup_GetLastErrorString() to a less informative behaviour.

- char SWUPLIBDLLEXIMPORT ∗ WSwup_GetLastErrorString (t_SwupCom ComPort)

  Retrieves the Error-String of the last Error for the given ComPort.

- SWUPLIBDLLEXIMPORT t_ErrorStruct ∗ WSwup_GiveLastError (WORD wUpdateNr)

  Returns the last Error for the indicated updatenumber with its specific parameters.

- char SWUPLIBDLLEXIMPORT ∗ WSwup_GetLastFileError (void)

  Returns a pointer to a string describing the last error occured on FileReading.

6.5.1 Define Documentation

6.5.1.1 #define MAX_ENTRIES 50

Amount of entries in the error-"stack".

6.5.1.2 #define NO_PARAM ""

an empty parameter-list

6.5.1.3 #define SIZE_OF_PINFO 10

max amount of parameters for errors

6.5.1.4 #define SIZE_OF_PSTRING 2000

maximum size of a error-string-parameter

6.5.2 Function Documentation

6.5.2.1 DWORD SWUPLIBDLLEXIMPORT WSwup_GetLastError (t_SwupCom *ComPort*)

Returns an specific Error-Code about last Error on this comport.

Retrieves the last occured Error for the specified COM-Port (almost useless, normally use WSwup_GetLastErrorString) **Dont use this function any longer its obsolete and will be killed in one of the next releases !**

Either use function WSwup_GetLastErrorString() to get an english string or use function WSwup_GiveLastError() and deal with it's information outside the library.

Parameters:

ComPort ComPort on which the Error cccured

Returns:

The Error-Code

6.5.2.2 char SWUPLIBDLLEXIMPORT∗ WSwup_GetLastErrorString (t_SwupCom *ComPort*)

Retrieves the Error-String of the last Error for the given ComPort.

Parameters:

ComPort ComPort on which the Error cccured

Returns:

The Error-Code-String

6.5.2.3 char SWUPLIBDLLEXIMPORT∗ WSwup_GetLastFileError (void)

Returns a pointer to a string describing the last error occured on FileReading.

Currently the error-strings are hardcoded in english language in the library.

Returns:

pointer to a descriptive string about an FileReadError

6.5.2.4 SWUPLIBDLLEXIMPORT t_ErrorStruct* WSwup_GiveLastError (WORD *wUpdateNr*)

Returns the last Error for the indicated updatenumber with its specific parameters.

For a more detailled explanation see a the declaration of t_ErrorStruct or have a look at the also provided err_demo.c<br>

- dwErrorNumber : the Error-number, for corresponding text see err_text.h
- ucParamInfo : indicates the kind of the following parameters, e.g. s for string, x,d,i,o,u for numerals, no l...!

  e.g. "sxsd", maximum SIZE_OF_PINFO Entries (10 at the moment), upper and lower chars allowed

  maximum 5 strings and 5 non-strings

  just like d,X... in printf, without
- ulParamValue : numeral LONG value
- ucParamString : maximum 2000 chars

Parameters:

wUpdateNr : the update-Nr for which you want to retrieve the error-information.

Returns:

t_ErrorStruct -> a pointer to a t_ErrorStruct - structure

6.5.2.5 void SWUPLIBDLLEXIMPORT WSwup_SetErrorTextBehaviourToEnduser (BOOL *fEndUser*)

Switches the behaviour of function WSwup_GetLastErrorString() to a less informative behaviour.

Has to be called in an enduser-update when not dealing yourself with error-strings, this means when you are hapyy with english text. Normally on an enduser-update you should deal with different languages and interpret the errors outside the library by yourself using function : WSwup_GiveLastError( ) and implement something equal to the swupdemo-error-handling.

Parameters:

fEndUser TRUE means enduser FALSE means developper

Return values:

none

6.6 BFB-Lib-Functions, used before and after SOFTWARE-Update

Typedefs

- typedef void(∗ VoltageCheckCallBack)(WORD wUpdateNr, unsigned int unMaxCalls, unsigned int unCurrCall)

  Callbackfunction for VoltageCheck that is performed.

- typedef void(∗ UpdateSuccessCallBack)(WORD wUpdateNr, unsigned int unMaxCalls, unsigned int unCurrCall)

  Callbackfunction for UpdateSuccess that is performed after swup.

Functions

- void SWUPLIBDLLEXIMPORT WSwup_InstallBfbCallBackFunctions (VoltageCheckCallBack pfnVoltageCheck, UpdateSuccessCallBack pfnSuccessCheck)

  Installs the needed callbackfunctions for Pre and Post-Checks via BFB-Library.

- BOOL SWUPLIBDLLEXIMPORT WSwup_CheckUpdateVoltage (WORD wUpdateNumber, t_-SwupCom WhichCom, unsigned short ∗punVoltage, unsigned long ulSpeed)

  Tries to communicate to a "living" mobile to determine if the voltage is ok.

- BOOL SWUPLIBDLLEXIMPORT WSwup_InitCopro (t_SwupCom WhichCom)

  forces mobile into CoproUpdateMode to determine if the voltage is ok.

- BOOL SWUPLIBDLLEXIMPORT WSwup_SipcCheckVoltage (WORD wUpdateNr, unsigned short ∗punVoltage, BOOL ForceFlag)

  forces mobile into CoproUpdateMode to determine if the voltage is ok.

- BOOL SWUPLIBDLLEXIMPORT WSwup_CheckUpdateSuccess (WORD wUpdateNumber, t_-SwupCom WhichCom)

  Checks if Mobile-SW-Update was successfull.

6.6.1 Typedef Documentation

6.6.1.1 typedef void(∗ UpdateSuccessCallBack)(WORD wUpdateNr, unsigned int unMaxCalls, unsigned int unCurrCall)

Callbackfunction for UpdateSuccess that is performed after swup.

On call of function WSwup_CheckUpdateSuccess this callback-function is called several times to allow the user of the library to show the progress.

Parameters:

wUpdateNr First Parameter is the Update-Nr

unMaxCalls Second Parameter is the Max-Amount of Calls of this CallbackFunction

unCurrCall Third parameter is the current Call-Number. It can happen that not all Callbacks are done !

Returns:

void

6.6.1.2 typedef void(∗ VoltageCheckCallBack)(WORD wUpdateNr, unsigned int unMaxCalls, unsigned int unCurrCall)

Callbackfunction for VoltageCheck that is performed.

On call of function WSwup_CheckUpdateVoltage this callback-function is called several times to allow the user of the library to show the progress.

Parameters:

wUpdateNr This parameter is the Update-Nr

unMaxCalls This second Parameter is the Max-Amount of Calls of this CallbackFunction

unCurrCall this parameter is the current Call-Number. It can happen that not all Callbacks are done !

Returns:

void

6.6.2 Function Documentation

6.6.2.1 BOOL SWUPLIBDLLEXIMPORT WSwup_CheckUpdateSuccess (WORD *wUpdateNumber*, t_SwupCom *WhichCom*)

Checks if Mobile-SW-Update was successfull.

Tries to switch on the Mobile and to ping it, to check if the Mobile-SW is okay.

NEWAfterwards the mobile is being switched OFF, to avoid errors with different behaviour between switching on via tool and via key-press.

Problems are: AT-C doesnt work correctly, Carkit doesnt work correctly.

Parameters:

wUpdateNumber Number of the current Update

WhichCom ComPort of this Update

Return values:

TRUE everything is fine

FALSE Some Error occurred, call WSwup_GetLastError() or WSwup_GetLastErrorString()

6.6.2.2 BOOL SWUPLIBDLLEXIMPORT WSwup_CheckUpdateVoltage (WORD *wUpdateNumber*, t_SwupCom *WhichCom*, unsigned short ∗ *punVoltage*, unsigned long *ulSpeed*)

Tries to communicate to a ”living” mobile to determine if the voltage is ok.

Switches the Mobile on, if necessary and afterwards switches it off, so that afterwards the softwareupdate itself can be performed. Calls several times the pfnVoltageCheck CallbackFunction.

In case this update is an update with the new behaviour (introduced for the K45-family) set via WSwup_-SetUpdateConceptMode() this function has a slightly different behaviour:

- this routine MUST be called, because it initiates the update !
- (with BootStrapMode this routine is an option)
- it does not try to switch on the mobile, endusers dont have an ignition line.
- due to this : if connection via ping is not possible, this routine fails.
- the filetype is checked, only a complete Mobile-SW can be updated with this new mode
- partitial SW (Diff/Tegic/Language/...) will be refused.
- after having pinged and checked the voltage, this routine forces the mobile into the new update-mode, without really switching off.
- due to this it's necessary to pass the new baudrate-parameter, which will be used for the following update-procedure itself
- after sucessfully returning from this function, the WSwup_PerformSoftwareUpdate() function must be called, because the mobile is already in update-mode !

Parameters:

wUpdateNumber Number of the current Update

WhichCom ComPort of this Update

punVoltage Pointer to the Voltage in Millivolt

ulSpeed the max speed that can be driven on this PC , (use ADDIDATA-Baudrates even if you have a fastboot-card installed)

Return values:

TRUE everything is fine

FALSE Some Error occurred, call WSwup_GetLastError() or WSwup_GetLastErrorString()

6.6.2.3 BOOL SWUPLIBDLLEXIMPORT WSwup_InitCopro (t_SwupCom *WhichCom*)

forces mobile into CoproUpdateMode to determine if the voltage is ok.

Switches the Mobile on, and afterwards does not switche it off, so that afterwards the softwareupdate itself can be performed. Calls several times the pfnVoltageCheck CallbackFunction.

Parameters:

WhichCom UsbPort of this Update

Return values:

TRUE everything is fine

FALSE Some Error occurred, call WSwup_GetLastError() or WSwup_GetLastErrorString()

6.6.2.4 void SWUPLIBDLLEXIMPORT WSwup_InstallBfbCallBackFunctions (VoltageCheckCallBack *pfnVoltageCheck*, UpdateSuccessCallBack *pfnSuccessCheck*)

Installs the needed callbackfunctions for Pre and Post-Checks via BFB-Library.

If the user of the library wants to perform voltagecheck before the software-update itself and wants to perform the updatesuccess-check after the software-update with the functions WSwup_CheckUpdateVoltage and WSwup_CheckUpdateSuccess he first must install the callbackfunctions that will be called from inside the library to show the progress in the different functionalities.

Parameters:

pfnVoltageCheck Pointer to a CallbackRoutine that is called on Voltage-Check

pfnSuccessCheck Pointer to a CallbackRoutine that is called on Update-Success

Returns:

void

6.6.2.5 BOOL SWUPLIBDLLEXIMPORT WSwup_SipcCheckVoltage (WORD *wUpdateNr*, unsigned short $*$ *punVoltage*, BOOL *ForceFlag*)

forces mobile into CoproUpdateMode to determine if the voltage is ok.

Switches the Mobile on, and afterwards does not switche it off, so that afterwards the softwareupdate itself can be performed. Calls several times the pfnVoltageCheck CallbackFunction.

Parameters:

wUpdateNr UpdateNr = UsbPort of this Update

punVoltage Pointer to the Voltage in Millivolt

Return values:

TRUE everything is fine

FALSE Some Error occurred, call WSwup_GetLastError() or WSwup_GetLastErrorString()

6.7 FILE-Functions

Data Structures

- struct t_SwInformation

  Structure with Information about the SW in the XBI-File or the Mobile itself.

Typedefs

- typedef void($*$ FileReadProgressCallBack)(unsigned long ulAmountOfBytesToRead, unsigned long ulCurrentRead)

  Callbackfunction for FileReading.

Functions

- void SWUPLIBDLLEXIMPORT WSwup_PrepareForUpdateWithUnknownProject (char $*$pszUnknownProjectName, char $*$pszDerivedFromKnownProjectName)

  Prepares the WSWUP-Library for updating an ”unknown” project.

- BOOL SWUPLIBDLLEXIMPORT WSwup_ReadXbiFile (t_SwInformation $*$pSwInformation, FileReadProgressCallBack pfnReadProgress, char $*$pszXbiName)

  Reads the XBI-File with the given name into memory, so a swup can be performed.

- BOOL SWUPLIBDLLEXIMPORT WSwup_ReadXbiFileFromHeap (t_SwInformation $*$pSwInformation, unsigned char $*$pucMem, unsigned long ulBytesInHeap)

  Checks an XBI-File in heap so a swup can be performed.

- unsigned int SWUPLIBDLLEXIMPORT WSwup_ReadXbiHeaderFromHeap (unsigned long ulNumXbiBytes, t_ExtendedInfo $*$ptrWohin, unsigned char $*$pucMem)

  Reads a XBI-Header directly from Heap Just calls the internal function ReadXbiHeaderInHeap, only used in DLL.

- unsigned char SWUPLIBDLLEXIMPORT $*$ WSwup_GivePtrToBinData (void)

  once WSwup_ReadXbiFileFromHeap() is successfully called, this function delivers a ptr to start of bin data in heap.

- void SWUPLIBDLLEXIMPORT WSwup_CloseXbiFile (void)

  Releases all memory allocated on WSwup_ReadXbiFile .

- void SWUPLIBDLLEXIMPORT WSwup_CloseXbiFileHeap (void)

  Releases all memory allocated inside of WSwup_ReadXbiFileFromHeap .

6.7.1 Typedef Documentation

6.7.1.1 typedef void(∗ FileReadProgressCallBack)(unsigned long ulAmountOfBytesToRead, unsigned long ulCurrentRead)

Callbackfunction for FileReading.

This function is called several times from inside the library during reading of the XBI-File to give the user of the library the possibility to show the read-progress to the enduser.

Parameters:

ulAmountOfBytesToRead	First Parameter the is the Amount of Bytes that are to be read
ulCurrentRead	Second Parameter is the current amount of bytes Read

Returns:

void

6.7.2 Function Documentation

6.7.2.1 void SWUPLIBDLLEXIMPORT WSwup_CloseXbiFile (void)

Releases all memory allocated on WSwup_ReadXbiFile .

Must be called before reading another file, before closing the program or if an Error occured on WSwup_-ReadXbiFile() !

Returns:

void

6.7.2.2 void SWUPLIBDLLEXIMPORT WSwup_CloseXbiFileHeap (void)

Releases all memory allocated inside of WSwup_ReadXbiFileFromHeap .

Must be called before:

- closing the program
- before reading another file into your heap
- or if an error occured on WSwup_ReadXbiFileFromHeap() !

Returns:

void

6.7.2.3 unsigned char SWUPLIBDLLEXIMPORT∗ WSwup_GivePtrToBinData (void)

once WSwup_ReadXbiFileFromHeap() is successfully called, this function delivers a ptr to start of bin data in heap.

Usually there is no need to call this function, the heap ptr is used in right manner internally, but maybe there is some interest...

Return values:

NULL	-> WSwup_ReadXbiFileFromHeap() successfully called ?
ptr	to bin data in heap

6.7.2.4 void SWUPLIBDLLEXIMPORT WSwup_PrepareForUpdateWithUnknownProject (char ∗ *pszUnknownProjectName*, char ∗ *pszDerivedFromKnownProjectName*)

Prepares the WSWUP-Library for updating an ”unknown” project.

This function must be called before calling of WSwup_ReadXbiFile() in case the development-project is not known to the library. This only happens during development-phase on introduction of new projects. Normally the Library must know all projects to load the right Update-DLL depending on the read in XBI-File. So if a new project is derived from an already known project, this function can be called to tell the library which update-dll shall be loaded, if an specific unknown project happens to be in the XBI-File.

Parameters:

pszUnknownProjectName Name of the project that is unknown to library up to now.

pszDerivedFromKnownProjectName Name of the project that is known to library and who’s update-DLL should be used.

Returns:

void

6.7.2.5 BOOL SWUPLIBDLLEXIMPORT WSwup_ReadXbiFile (t_SwInformation ∗ *pSwInformation*, FileReadProgressCallBack *pfnReadProgress*, char ∗ *pszXbiName*)

Reads the XBI-File with the given name into memory, so a swup can be performed.

Is NOT done in a THREAD, so if the function returns the data is read. Several times during the read-prcess the via function-ptr given callbackfunction is called so that you can show the readprogress to the enduser.

Parameters:

pszXbiName Name of the XBI-File to be read with complete path and extension

pfnReadProgress Pointer to a callbackfunction to show read progress

pSwInformation Pointer to a t_SwInformation Struct where the Information about the SW will be stored by this function

Return values:

TRUE -> everything is fine

FALSE -> Some Error occurred, call WSwup_GetLastFileError()

6.7.2.6 BOOL SWUPLIBDLLEXIMPORT WSwup_ReadXbiFileFromHeap (t_SwInformation ∗ *pSwInformation*, unsigned char ∗ *pucMem*, unsigned long *ulBytesInHeap*)

Checks an XBI-File in heap so a swup can be performed.

This is to provide a similiar functionality like function WSwup_ReadXbiFile() . The difference is, that the user of the library has to allocate the needed memory by himself. He also has to read in the input-file as a binary file to the allocated heap. Before end of the program this memory must be freed by the user of the library. **Attention:** The library uses this heap, so you are not allowed to free this memory before the end of your program.

Parameters:

pSwInformation Pointer to a t_SwInformation Struct where the Information about the SW will be stored by this function

pucMem Pointer to start of heap with file, read-only access from inside library

ulBytesInHeap bytes in heap

Return values:

TRUE -> everything is fine

FALSE -> Some Error occurred, call WSwup_GetLastFileError()

6.7.2.7 unsigned int SWUPLIBDLLEXIMPORT WSwup_ReadXbiHeaderFromHeap (unsigned long *ulNumXbiBytes*, t_ExtendedInfo $*$ *ptrWohin*, unsigned char $*$ *pucMem*)

Reads a XBI-Header directly from Heap Just calls the internal function ReadXbiHeaderInHeap, only used in DLL.

Parameters:

pucMem ptr to xbi data in heap

(out)plAnzXbiBytes amount of XBI-header-bytes

(out)t_ExtendedInfo ptr to info struct

Return values:

0: XBI-Header successfully read

1: CHK-error reading header

2: HeapLib not initialized, call InitHeapLib first !

3: data len in heap zero, call to InitHeapLib ok ?

5: SwitchCase-Error

7: error in list

6.8 DEBUG-Functions for en/disabling OnLine-Debug

Defines

- #define MAIN_DEBUG_GROUP 0x01

  Enable Main-Functionality Debugging.

- #define FILE_DEBUG_GROUP 0x02

  Enable File-Function Debugging.

- #define COMM_DEBUG_GROUP 0x04

  Enable Debugging on Communciation-Functions.

- #define BFB_DEBUG_GROUP 0x08

  Enable Debugging on BFB-Communciation-Functions.

Enumerations

- enum t_DebugLevel { enNoDebug = 0, enLowDebug = 1, enMidDebug = 2, enHighDebug = 3 }

  Debugging-Level.

Functions

- void SWUPLIBDLLEXIMPORT WSwup_EnableOnlineDebugging (t_DebugLevel Requested-DebugLevel, WORD wRequestedDebugGroups)

  Enables Online Debugging on the requested DebugLevel for the requested Groups.

- void SWUPLIBDLLEXIMPORT WSwup_DisableOnlineDebugging (void)

  Disables Online Debugging.

- void SWUPLIBDLLEXIMPORT WSwup_EnableFileDebugging (t_DebugLevel RequestedDebug-Level, WORD wRequestedDebugGroups, char ∗szFilePrefix)

  Enables File Debugging on the requested DebugLevel for the requested Groups.

- void SWUPLIBDLLEXIMPORT WSwup_DisableFileDebugging (void)

  Disables File Debugging.

6.8.1 Define Documentation

6.8.1.1 #define BFB_DEBUG_GROUP 0x08

Enable Debugging on BFB-Communciation-Functions.

6.8.1.2 #define COMM_DEBUG_GROUP 0x04

Enable Debugging on Communciation-Functions.

6.8.1.3 #define FILE_DEBUG_GROUP 0x02

Enable File-Function Debugging.

6.8.1.4 #define MAIN_DEBUG_GROUP 0x01

Enable Main-Functionality Debugging.

6.8.2 Enumeration Type Documentation

6.8.2.1 enum t_DebugLevel

Debugging-Level.

Enumeration values:

enNoDebug	No debugging.
enLowDebug	Low debugging.
enMidDebug	Mid-Range debugging.
enHighDebug	Debug everything.

6.8.3 Function Documentation

6.8.3.1 void SWUPLIBDLLEXIMPORT WSwup_DisableFileDebugging (void)

Disables File Debugging.

Returns:
void

6.8.3.2 void SWUPLIBDLLEXIMPORT WSwup_DisableOnlineDebugging (void)

Disables Online Debugging.

Returns:
void

6.8.3.3 void SWUPLIBDLLEXIMPORT WSwup_EnableFileDebugging (t_DebugLevel *RequestedDebugLevel*, WORD *wRequestedDebugGroups*, char ∗ *szFilePrefix*)

Enables File Debugging on the requested DebugLevel for the requested Groups.

Parameters:

RequestedDebugLevel	The DebugLevel (a t_DebugLevel enum)
wRequestedDebugGroups	a combination of the above Debug-Groups
szFilePrefix	Name of the files to debug into, without extension !

Returns:
void

6.8.3.4 void SWUPLIBDLLEXIMPORT WSwup_EnableOnlineDebugging (t_DebugLevel *RequestedDebugLevel*, WORD *wRequestedDebugGroups*)

Enables Online Debugging on the requested DebugLevel for the requested Groups.

Parameters:

RequestedDebugLevel The DebugLevel (a t_DebugLevel enum)

wRequestedDebugGroups a combination of the above Debug-Groups

Returns:

void

6.9 SOFTWARE-Update-Functions

The whole softwareupdate is performed with one functioncall : WSwup_PerformSoftwareUpdate() The function itself terminates directly, it will generate a thread.

Defines

- #define CONSISTENCE_PRODUCT_ERROR 0x01

  SW and Mobile are different products.

- #define CONSISTENCE_SWVERSION_ERROR 0x02

  Mobile has newer SVN than the new SW.

- #define CONSISTENCE_SWDATE_ERROR 0x04

  Mobile has newer SW-Date than the new SW.

Typedefs

- typedef void(∗ BootstrapProgressCallBack)(WORD wUpdateNr, WORD unMaxCalls, WORD unCurrCall)

  Callbackfunction for Bootstraploader.

- typedef void(∗ UpdateSwTransmissionStartCallBack)(WORD wUpdateNr, unsigned long ulBytesToTransfer)

  Callbackfunction for UpdateSwTransmissionStart.

- typedef void(∗ UpdateSwTransmissionProgressCallBack)(WORD wUpdateNr, unsigned long ulBytesTransfered)

  Callbackfunction for UpdateSwTransmissionProgress.

- typedef void(∗ FlashTypeCallBack)(WORD wUpdateNr, WORD wFlashId, char ∗pszFlashString)

  Callbackfunction for FlashType.

- typedef void(∗ EraseProgressCallBack)(WORD wUpdateNr, WORD wAmountFlashBlocks, WORD wCurrentBlock)

  Callbackfunction for EraseProgress.

- typedef void(∗ MobileSwTransmissionStartCallBack)(WORD wUpdateNr, unsigned long ulBytesToTransfer)

  Callbackfunction for MobileSwTransmissionStart.

- typedef void(∗ MobileSwTransmissionProgressCallBack)(WORD wUpdateNr, unsigned long ulBytesTransfered)

  Callbackfunction for MobileSwTransmissionProgress.

- typedef void(∗ MobileSwUpdateSuccessCallBack)(WORD wUpdateNr)

  Callbackfunction for SoftwareUpdateSuccess.

- typedef void(∗ MobileSwUpdateErrorCallBack)(WORD wUpdateNr)

  Callbackfunction for SoftwareUpdateError.

- typedef BOOL(∗ MobileSwUpdateConsistenceProblemCallBack)(WORD wUpdateNr, t_SwInformation ∗pMobileInfo, t_SwInformation ∗pNewSwInfo, WORD wErrorComposition)

  Callbackfunction for SoftwareUpdateConsistence-Problem.

- typedef BOOL(∗ MobileSwUpdateInfoCallBack)(WORD wUpdateNr, t_SwInformation ∗pMobileInfo, t_SwInformation ∗pNewSwInfo)

  Callbackfunction for SoftwareUpdateInfo.

Enumerations

- enum t_InfoCallBackTime { enInfoCallBackEarlyWithBfb, enInfoCallBackLateDuringUpdate }

  Callback-Time for MobileSwUpdateInfoCallBack().

- enum t_UpdateMode { enForceBSLBehaviour, enForceNewConcept, enUpdateConceptAutoDetect, enSpecialUsbIbootExe }

  Behaviour of the Library concerning the new Update-Concept (xx45).

Functions

- void SWUPLIBDLLEXIMPORT WSwup_InstallSoftwareUpdateCallBackFunctions (BootstrapProgressCallBack pfnBootstrapProgress, UpdateSwTransmissionStartCallBack pfnUpdateSwTransmissionStart, UpdateSwTransmissionProgressCallBack pfnUpdateSwTransmissionProgress, FlashTypeCallBack pfnFlashType, EraseProgressCallBack pfnEraseProgress, MobileSwTransmissionStartCallBack pfnMobileSwTransmissionStart, MobileSwTransmissionProgressCallBack pfnMobileSwTransmissionProgress, MobileSwUpdateSuccessCallBack pfnMobileSwUpdateSuccess, MobileSwUpdateErrorCallBack pfnMobileSwUpdateError, MobileSwUpdateConsistenceProblemCallBack pfnMobileConsistenceProblem)

  Installs all the needed callbackfunctions for the whole software-update.

- void SWUPLIBDLLEXIMPORT WSwup_InstallSoftwareInfoCallBackFunction (MobileSwUpdateInfoCallBack pfnMobileSwUpdateInfoCallBack, t_InfoCallBackTime WhichCallbackTime)

  Installs the MobileSwUpdateInfoCallBack-Function.

- BOOL SWUPLIBDLLEXIMPORT WSwup_SetUpdateConceptMode (t_UpdateMode WhichMode)

  Tells the library how to perform the Software-Update.

- BOOL SWUPLIBDLLEXIMPORT WSwup_SetUsbParams (WORD wUpdateNr, t_SwupCom WhichUsb, BOOL Force)

  Tells the library how to configure the modem's Software-Update via Usb.

- void SWUPLIBDLLEXIMPORT WSwup_ResetUsbParams (WORD wUpdateNr)

  Tells the library how to configure the modem's Software-Update via Usb.

- BOOL SWUPLIBDLLEXIMPORT WSwup_PerformSoftwareUpdate (WORD wUpdateNr, t_SwupCom WhichCom, unsigned long ulSpeed)

  Performs the Software-Update on the given ComPort with the given Baudrate.

- BOOL SWUPLIBDLLEXIMPORT WSwup_SetBootPIN (WORD wUpdateNr, unsigned int unPinSize, unsigned char ∗pucBootPIN)

  Save a PIN for an Update, This PIN must be provided by data stored in PICS and is sended during BSL connect attempt.

6.9.1 Detailed Description

The whole softwareupdate is performed with one functioncall : WSwup_PerformSoftwareUpdate() The function itself terminates directly, it will generate a thread.

So it is possible to call the function several times to perform software-updates in parallel on different comports. In the generated thread the following steps are performed:

- connection to a switched off mobile with the bootstraploader
- transmission of the update-sw
- erasure of the flash
- transmission of the new mobile-software

To allow communication between the library and the mainprogram, the user has to install several callbackfunctions which will be called from inside the library. The callbackfunctions are called in this order :

- BootstrapProgressCallBack()
- UpdateSwTransmissionStartCallBack()
- UpdateSwTransmissionProgressCallBack()
- FlashTypeCallBack()
- EraseProgressCallBack()
- MobileSwTransmissionStartCallBack()
- MobileSwTransmissionProgressCallBack()
- MobileSwUpdateSuccessCallBack()

In case this is an update following the new concept (introduced for the K45-family) set via WSwup_SetUpdateConceptMode() the following call-back-functions will **NOT** be called, because these steps are not done from inside the library, the mobile will perform this steps by itself automatically:

- BootstrapProgressCallBack()
- UpdateSwTransmissionStartCallBack()
- UpdateSwTransmissionProgressCallBack()

If an error occurs during the update-process the following callbackfunction will be called from inside the library:

- MobileSwUpdateErrorCallBack()

Before erasure of the flash a consistence-check between the new software and the mobile already in the mobile will be performed. If a consistence-error is detected, the following function will be called:

- MobileSwUpdateConsistenceProblemCallBack()

6.9.2 Define Documentation

6.9.2.1 #define CONSISTENCE_PRODUCT_ERROR 0x01

SW and Mobile are different products.

6.9.2.2 #define CONSISTENCE_SWDATE_ERROR 0x04

Mobile has newer SW-Date than the new SW.

6.9.2.3 #define CONSISTENCE_SWVERSION_ERROR 0x02

Mobile has newer SVN than the new SW.

6.9.3 Typedef Documentation

6.9.3.1 typedef void(* BootstrapProgressCallBack)(WORD wUpdateNr, WORD unMaxCalls, WORD unCurrCall)

Callbackfunction for Bootstraploader.

Parameters:

wUpdateNr,First Parameter is the Update-Nr

unMaxCalls Second Parameter is the Max-Amount of Calls of this CallbackFunction

unCurrCall Third parameter is the current Call-Number. It can happen that not all Callbacks are done !

Returns:

void

6.9.3.2 typedef void(* EraseProgressCallBack)(WORD wUpdateNr, WORD wAmountFlashBlocks, WORD wCurrentBlock)

Callbackfunction for EraseProgress.

This callbackfunction is called several times during the flash-erase-process.

Parameters:

wUpdateNr First Parameter is the Update-Nr

wAmountFlashBlocks Second Parameter is the number of flash-blocks to erase

wCurrentBlock Third Parameter is the current number of flashblock that is beeing erased

Returns:
void

6.9.3.3 typedef void(∗ FlashTypeCallBack)(WORD wUpdateNr, WORD wFlashId, char ∗ pszFlashString)

Callbackfunction for FlashType.

This callbackfunction is called once before the flash is beeing erased and it gives the actual flashcode as parameter.

Parameters:

wUpdateNr First Parameter is the Update-Nr

wFlashId Second Parameter is the Flash-Code

pszFlashString Third Parameter is a string to an Description of the Flash

Returns:
void

6.9.3.4 typedef void(∗ MobileSwTransmissionProgressCallBack)(WORD wUpdateNr, unsigned long ulBytesTransfered)

Callbackfunction for MobileSwTransmissionProgress.

This callbackfunction is called from SWUPLIB during the update-process to give the user of the Library the possibility to show the current transmission-progress. Compare the given value against the value given by callbackfunction MobileSwTransmissionStartCallBack() to show the percentage of data beeing transmitted.

Parameters:

wUpdateNr First Parameter is the Update-Nr

ulBytesTransfered Second Parameter is the current amount of sent data

Returns:
void

6.9.3.5 typedef void(∗ MobileSwTransmissionStartCallBack)(WORD wUpdateNr, unsigned long ulBytesToTransfer)

Callbackfunction for MobileSwTransmissionStart.

This callbackfunction is called from SWUPLIB at the beginning of the update-process. If the application wants to show the percentage of completion, it must save the value given by ulBytesToTransfer for later comparison of the actual value (given via MobileSwTransmissionProgressCallBack()) with the stored value.

Parameters:

wUpdateNr First Parameter is the Update-Nr

ulBytesToTransfer Second Parameter is the amount of data to be sent

Returns:

void

6.9.3.6 typedef BOOL(∗ MobileSwUpdateConsistenceProblemCallBack)(WORD wUpdateNr, t_SwInformation ∗ pMobileInfo, t_SwInformation ∗ pNewSwInfo, WORD wErrorComposition)

Callbackfunction for SoftwareUpdateConsistence-Problem.

This function is called by the SWUPLIB if there is a consistence-error between the new SW and the SW that is already in Mobile. The user of the LIB can use the parameters to show the enduser the conflict and let the enduser choose a solution. Depending on this choice the returnvalue of this function must be set to TRUE or FALSE.

Parameters:

wUpdateNr First Parameter is the Update-Nr

pMobileInfo Second Parameter is a pointer to a struct containing the mobile-information

pNewSwInfo Third Parameter is a pointer to a struct containing the new-sw-information

wErrorComposition Fourth Parameter is a composition of the different error-possibilities

Return values:

TRUE -> must be set to TRUE by the user of the LIB , if SW-Update is allowed

FALSE -> must be set to FALSE by the user of the LIB , if SW-Update is NOT allowed

6.9.3.7 typedef void(∗ MobileSwUpdateErrorCallBack)(WORD wUpdateNr)

Callbackfunction for SoftwareUpdateError.

This Callbackfunction is called from inside the SWUPLIB if an error occured inside the Library. Call WSwup_GetLastErrorString() to retrieve an error-description.

Parameters:

wUpdateNr First Parameter is the Update-Nr

Returns:

void

6.9.3.8 typedef BOOL(∗ MobileSwUpdateInfoCallBack)(WORD wUpdateNr, t_SwInformation ∗ pMobileInfo, t_SwInformation ∗ pNewSwInfo)

Callbackfunction for SoftwareUpdateInfo.

This function is called by the SWUPLIB **independent of and before any consistence-errors** if the callbackfunction is installed via function : WSwup_InstallSoftwareInfoCallBackFunction(). This means: If you want to get info about the SW already in Mobile, then you have to install this callbackfuntion. If you want to allow the SW-Update, you must return TRUE to the library. But be prepared: If the library itself detects a conflict between new and old SW, the callbackfunction MobileSwUpdateConsistenceProblemCallBack() will be called, independant of the call of this callbackfunction ! If you want to "deinstall" a previosly registered callbackfunction, call function WSwup_InstallSoftwareInfoCallBackFunction() with a NULL-Parameter.

Attention: If you return FALSE with this callback-function, the Update will be stopped, but neither MobileSwUpdateSuccessCallBack() nor MobileSwUpdateErrorCallBack() will be called, because the library does not know the reason why you stopped the update !

This means you have to generate and show an error to enduser by yourself !

Parameters:

wUpdateNr First Parameter is the Update-Nr

pMobileInfo Second Parameter is a pointer to a struct containing the mobile-information

pNewSwInfo Third Parameter is a pointer to a struct containing the new-sw-information

Return values:

TRUE -> must be set to TRUE by the user of the LIB , if SW-Update is allowed

FALSE -> must be set to FALSE by the user of the LIB , if SW-Update is NOT allowed

6.9.3.9 typedef void($*$ MobileSwUpdateSuccessCallBack)(WORD wUpdateNr)

Callbackfunction for SoftwareUpdateSuccess.

This Callbackfunction is called from inside the SWUPLIB if the complete update was successfull.

Parameters:

wUpdateNr First Parameter is the Update-Nr

Returns:

void

6.9.3.10 typedef void($*$ UpdateSwTransmissionProgressCallBack)(WORD wUpdateNr, unsigned long ulBytesTransfered)

Callbackfunction for UpdateSwTransmissionProgress.

This callbackfunction will be called several times during the transmission of the update-sw. This software is executed in the ram of the mobile and is responsible for the erasure of the flash and the reprogramming of the flash with the new mobile-sw.

Parameters:

wUpdateNr First Parameter is the Update-Nr

ulBytesTransfered Second Parameter is the current amount of sent data

Returns:

void

6.9.3.11 typedef void(∗ UpdateSwTransmissionStartCallBack)(WORD wUpdateNr, unsigned long ulBytesToTransfer)

Callbackfunction for UpdateSwTransmissionStart.

This callbackfunction will be called once before the update-sw is beeing transmitted to mobile. If you want to show the percentage of update-sw beeing transmitted, you have to store the parameter ulBytesToTransfer and compare it against the values given by the calls of the callbackfunction UpdateSwTransmission-ProgressCallBack()

Parameters:

wUpdateNr First Parameter is the Update-Nr

ulBytesToTransfer Second Parameter is the amount of data to be sent

Returns:

void

6.9.4 Enumeration Type Documentation

6.9.4.1 enum t_InfoCallBackTime

Callback-Time for MobileSwUpdateInfoCallBack().

Enumeration values:

enInfoCallBackEarlyWithBfb Callback will be called when Mobile is normal switched on => no timeouts.

enInfoCallBackLateDuringUpdate Callback will be called when mobile is already in update-mode => deal with pc-timeouts !

6.9.4.2 enum t_UpdateMode

Behaviour of the Library concerning the new Update-Concept (xx45).

Enumeration values:

enForceBSLBehaviour force old (Boot-Strap-Loader) behaviour, only for development possible, not for customer

enForceNewConcept force new Update-Concept-Behaviour , must be used for customer-update

enUpdateConceptAutoDetect first try new concept, if mobile doesnt answer, try old BSL-concept, only for development possible, not for customer

enSpecialUsbIbootExe special setting for USB_Iboot update, no button

6.9.5 Function Documentation

6.9.5.1 void SWUPLIBDLLEXIMPORT WSwup_InstallSoftwareInfoCallBackFunction (MobileSwUpdateInfoCallBack *pfnMobileSwUpdateInfoCallBack*, t_InfoCallBackTime *WhichCallbackTime*)

Installs the MobileSwUpdateInfoCallBack-Function.

Parameters:

pfnMobileSwUpdateInfoCallBack -> Pointer to a CallbackRoutine that is called to give info about the SW already in Mobile

pass a NULL if you want to deinstall a previosly installed callbackfunction.

This is the only place in the LIB where NULL as func-ptr is allowed in the moment !

WhichCallbackTime -> a enum of type t_InfoCallBackTime , where you must select, on which time you want this callbackfunction called. There are two possibilities , either you can have this function called during connection-setup with BFB-Communication or you can select to have this function called later, when update is already running.

The reason to have these two different times is : on the one hand you have to deal with timeouts when already really in updatemode, this leads to the early test. On the other hand the early test is less ”hacker-proof” because the mobile is in a totally different mode and there is at least a very little chance to exchange two mobiles between the two mobile-states. So its up to the user of the library at which time he wants the CallBack-Function.

Returns:

void

6.9.5.2 void SWUPLIBDLLEXIMPORT WSwup_InstallSoftwareUpdateCallBackFunctions (BootstrapProgressCallBack *pfnBootstrapProgress*, UpdateSwTransmissionStartCallBack *pfnUpdateSwTransmissionStart*, UpdateSwTransmissionProgressCallBack *pfnUpdateSwTransmissionProgress*, FlashTypeCallBack *pfnFlashType*, EraseProgressCallBack *pfnEraseProgress*, MobileSwTransmissionStartCallBack *pfnMobileSwTransmissionStart*, MobileSwTransmissionProgressCallBack *pfnMobileSwTransmissionProgress*, MobileSwUpdateSuccessCallBack *pfnMobileSwUpdateSuccess*, MobileSwUpdateErrorCallBack *pfnMobileSwUpdateError*, MobileSwUpdateConsistenceProblemCallBack *pfnMobileConsistenceProblem*)

Installs all the needed callbackfunctions for the whole software-update.

Parameters:

pfnBootstrapProgress -> Pointer to a CallbackRoutine that is called on Bootstrap

pfnUpdateSwTransmissionStart -> Pointer to a CallbackRoutine that is called on UpdateSw-TransmissionStart

pfnUpdateSwTransmissionProgress -> Pointer to a CallbackRoutine that is called during Update-SwTransmission

pfnFlashType -> Pointer to a CallbackRoutine that is called for Flash-Type

pfnEraseProgress -> Pointer to a CallbackRoutine that is called on during Erasure of Flash

pfnMobileSwTransmissionStart -> Pointer to a CallbackRoutine that is called on Start of MobileSw-Transmission

pfnMobileSwTransmissionProgress -> Pointer to a CallbackRoutine that is called during Progress of MobileSw-Transmission

pfnMobileSwUpdateSuccess -> Pointer to a CallbackRoutine that is called if SW-Update was successfull

pfnMobileSwUpdateError -> Pointer to a CallbackRoutine that is called if SW-Update had an error

pfnMobileConsistenceProblem -> Pointer to a CallbackRoutine that is called if there is a consistence-problem

Returns:

void

6.9.5.3 BOOL SWUPLIBDLLEXIMPORT WSwup_PerformSoftwareUpdate (WORD *wUpdateNr*, t_SwupCom *WhichCom*, unsigned long *ulSpeed*)

Performs the Software-Update on the given ComPort with the given Baudrate.

Attention: this function creates a thread, so the function itself will terminate very fast, so that further things can be done. The communication with the main-program will be done via the above CallbackFunctions

Parameters:

wUpdateNr The current number for the update, up to four are allowed

WhichCom The Comport on which the update shall be progressed

ulSpeed the max speed that can be driven on this PC , (use ADDIDATA-Baudrates even if you have a fastboot-card installed, the mapping is done internally via the INI-FILE

Return values:

TRUE -> SW-Update is Running (not finished !)

FALSE -> No Callbacks Installed

6.9.5.4 void SWUPLIBDLLEXIMPORT WSwup_ResetUsbParams (WORD *wUpdateNr*)

Tells the library how to configure the modem's Software-Update via Usb.

Parameters:

wUpdateNr identifies updatethread

6.9.5.5 BOOL SWUPLIBDLLEXIMPORT WSwup_SetBootPIN (WORD *wUpdateNr*, unsigned int *unPinSize*, unsigned char ∗ *pucBootPIN*)

Save a PIN for an Update, This PIN must be provided by data stored in PICS and is sended during BSL connect attempt.

This PIN is used for disabled BSL mode devices. If the PIN is correct the mobile will accept the update sw. A PIN can be reseted by calling this function with unPinSize==0 or pucBootPIN==NULL

Parameters:

wUpdateNr The current number for the update, up to four are allowed

unPinSize Number of data bytes of Boot PIN (16 Byte(128Bit) in the moment)

pucBootPIN Pointer to PIN Data

Return values:

TRUE -> PIN was set /reseted

FALSE -> unPINSize or wUpdateNr data to large/ not supported

6.9.5.6 BOOL SWUPLIBDLLEXIMPORT WSwup_SetUpdateConceptMode (t_UpdateMode *WhichMode*)

Tells the library how to perform the Software-Update.

In a development-environment it must be still possible do use the old BSL-behaviour, but for a customer-update its necessary to switch to the new update-mode. For development environment it's also possible to do an autodetect, first the new concept is tried and if this fails (e.x. because its an old project) then the old BSL-Mode is activated. If you want to use the new mode (enForceNewConcept, enAutoDetect) you MUST call this function before WSwup_CheckUpdateVoltage() and you also MUST call function WSwup_CheckUpdateVoltage() !

Parameters:

WhichMode an t_UpdateMode , which tells the library how to behave

Return values:

TRUE -> everything prepared

FALSE ->

6.9.5.7 BOOL SWUPLIBDLLEXIMPORT WSwup_SetUsbParams (WORD *wUpdateNr*, t_SwupCom *WhichUsb*, BOOL *Force*)

Tells the library how to configure the modem's Software-Update via Usb.

Parameters:

wUpdateNr identifies updatethread

WhichUsb UsbPort

Force BOOL, forces updatethread to special UsbPort

Return values:

TRUE -> everything ok

FALSE -> wrong UsbPort

Chapter 7

API for WSWUP-DLL Data Structure Documentation

7.1 t_ComPort Struct Reference

Structure with Serial port Information.

`#include <wswuplib.h>`

Data Fields

- t_SwupCom Nr
- unsigned long ulBaudRate

7.1.1 Detailed Description

Structure with Serial port Information.

7.1.2 Field Documentation

7.1.2.1 t_SwupCom t_ComPort::Nr

7.1.2.2 unsigned long t_ComPort::ulBaudRate

The documentation for this struct was generated from the following file:

- wswuplib.h

7.2 t_ErrorStruct Struct Reference

A structure to allow passing of parameters for error-messages from the Lib to the user of the Lib.

`#include <wswuplib.h>`

Data Fields

- DWORD dwErrorNumber

  the error-enum

- unsigned char ucParamInfo [SIZE_OF_PINFO+1]

  info about the validity of the following params + EOS

- LONG ulParamValue [5]

  variables that will be inserted in the indicated string at d,x,..

- unsigned char ucParamString [5][SIZE_OF_PSTRING]

  variables that will be inserted in the indicated string at s

7.2.1 Detailed Description

A structure to allow passing of parameters for error-messages from the Lib to the user of the Lib.

If you need internationalisation in your application, you have to deal with error-strings yourself outside the library.

To allow the library on errors to pass parameters to the user of the library this structure was introduced.

You will retrieve a pointer to this structure via call of WSwup_GiveLastError() . The first element in the struct is the error-number itself. The second element is a simplified format-string (like in printf but without) that tells the user of the library how many and in which order additional parameters are stored in the third and fourth elements of the struct.

There is only a distinction between strings and numerical integer values (no floats), this means for example if the format-string contains something like ”dsxs” ,that the first and the third error-parameter (d and x) will be stored in t_ErrorStruct::ulParamValue and the second and forth parameter containg two strings are stored in t_ErrorStruct::ucParamString.

If you need more information you should have a look at the also provided err_demo.c.

7.2.2 Field Documentation

7.2.2.1 DWORD t_ErrorStruct::dwErrorNumber

the error-enum

7.2.2.2 unsigned char t_ErrorStruct::ucParamInfo[SIZE_OF_PINFO +1]

info about the validity of the following params + EOS

7.2.2.3 unsigned char t_ErrorStruct::ucParamString[5][SIZE_OF_PSTRING]

variables that will be inserted in the indicated string at s

7.2.2.4 LONG t_ErrorStruct::ulParamValue[5]

variables that will be inserted in the indicated string at d,x,..

The documentation for this struct was generated from the following file:

- wswuplib.h

7.3 t_Ports Struct Reference

Structure with complete port configuration information.

`#include <wswuplib.h>`

Data Fields

- t_USBPort USB_Port
- t_ComPort Com_Port
- t_USB_Serial_Switch Switch

7.3.1 Detailed Description

Structure with complete port configuration information.

7.3.2 Field Documentation

7.3.2.1 t_ComPort t_Ports::Com_Port

7.3.2.2 t_USB_Serial_Switch t_Ports::Switch

7.3.2.3 t_USBPort t_Ports::USB_Port

The documentation for this struct was generated from the following file:

- wswuplib.h

7.4 t_SwInformation Struct Reference

Structure with Information about the SW in the XBI-File or the Mobile itself.

`#include <wswuplib.h>`

Data Fields

- char szProductType [16]

  Official Name of the product , ex: S35 , C3I etc.

- char szProjectName [16]

  Develop-Name of the product , ex: epp35a, epp35ci etc.

- char szProductSVN [10]

  Official SVN-String, ex: "13".

- char szSwGeneration [17]

  SW-Directory where this Software was built (eg.

- char szDevlName [9]

  Name of the developper of this Software (eg.

- char szCompileDate [9]

  Date the SW was compiled, ex: "05.03.00".

- char szOldCompileDate [9]
- char szCompileTime [9]

  Time the SW was compiled, ex: "12:31:47".

- char szOldCompileTime [9]
- char szRbmDate [9]
- char szRbmTime [9]
- unsigned long ulRbmPutCount
- unsigned long ulRbcPutCount
- t_FileFormat stFormatInfo

  what format has the data, a - Value (Bin, Raw, LenCheck, Compressed)

- char szSwType [20]

  Type of the SW , ex: "MobSw".

- char szLangGroup [17]

  string containin the Language-Group (eg

- char szNewSplitIdentifier [16]

  Kind of Split, ex: "SplitOnly".

- char szCommentAboutSplitEntity [20]

  if exist then the header value , ex: "Tegic", if not ??

- char ∗ pcText

  a text the developper wants to show during swup

- unsigned long ulTextLen

  len of the text the developper wants to show during swup

7.4.1 Detailed Description

Structure with Information about the SW in the XBI-File or the Mobile itself.

This Information-Structure will be filled on call of function WSwup_ReadXbiFile and pointers to two of this structs will be given on Consistence-Errors, when eg. the SW already contained in the mobile is newer than the XBI-File that is going to be updated (see MobileSwUpdateConsistenceProblemCallBack()).

7.4.2 Field Documentation

7.4.2.1 char∗ t_SwInformation::pcText

a text the developper wants to show during swup

7.4.2.2 t_FileFormat t_SwInformation::stFormatInfo

what format has the data, a - Value (Bin, Raw, LenCheck, Compressed)

7.4.2.3 char t_SwInformation::szCommentAboutSplitEntity[20]

if exist then the header value , ex: ”Tegic”, if not ??

7.4.2.4 char t_SwInformation::szCompileDate[9]

Date the SW was compiled, ex: ”05.03.00”.

7.4.2.5 char t_SwInformation::szCompileTime[9]

Time the SW was compiled, ex: ”12:31:47”.

7.4.2.6 char t_SwInformation::szDevlName[9]

Name of the developper of this Software (eg.

”beaugear”)

7.4.2.7 char t_SwInformation::szLangGroup[17]

string containin the Language-Group (eg

”lg1”)

7.4.2.8 char t_SwInformation::szNewSplitIdentifier[16]

Kind of Split, ex: ”SplitOnly”.

7.4.2.9 char t_SwInformation::szOldCompileDate[9]

7.4.2.10 char t_SwInformation::szOldCompileTime[9]

7.4.2.11 char t_SwInformation::szProductSVN[10]

Official SVN-String, ex: ”13”.

7.4.2.12 char t_SwInformation::szProductType[16]

Official Name of the product , ex: S35 , C3I etc.

7.4.2.13 char t_SwInformation::szProjectName[16]

Develop-Name of the product , ex: epp35a, epp35ci etc.

7.4.2.14 char t_SwInformation::szRbmDate[9]

7.4.2.15 char t_SwInformation::szRbmTime[9]

7.4.2.16 char t_SwInformation::szSwGeneration[17]

SW-Directory where this Software was built (eg.
”gen7u.z1”)

7.4.2.17 char t_SwInformation::szSwType[20]

Type of the SW , ex: ”MobSw”.

7.4.2.18 unsigned long t_SwInformation::ulRbcPutCount

7.4.2.19 unsigned long t_SwInformation::ulRbmPutCount

7.4.2.20 unsigned long t_SwInformation::ulTextLen

len of the text the developper wants to show during swup
The documentation for this struct was generated from the following file:

- wswuplib.h

7.5 t_USBPort Struct Reference

Structure with USB port Information.

```
#include <wswuplib.h>
```

Data Fields

- t_USBNr Nr
- BOOL Force

7.5.1 Detailed Description

Structure with USB port Information.

7.5.2 Field Documentation

7.5.2.1 BOOL t_USBPort::Force

7.5.2.2 t_USBNr t_USBPort::Nr

The documentation for this struct was generated from the following file:

- wswuplib.h

7.6 t_VersionInformation Struct Reference

Version-Information for the different parts of the Library.

`#include <wswuplib.h>`

Data Fields

- int nStructSize

  must be loaded with sizeof(struct) to avoid Consistency-errors

- char szDescription [100]

  Description, ex: ”BSL-DLL for EGOLD+, Synchstation”.

- char szDeveloper [10]

  the developer, ex: ”kuenstnj” / ”official”

- char szDateTime [18]

  Date and Time, ex: ”30.06.00 20:15:17 ”.

- int nVersionMajor

  the Major-Version-Nr ex: 1

- int nVersionMinor

  the Minor-Version-Nr ex: 25

7.6.1 Detailed Description

Version-Information for the different parts of the Library.

The library itself is split into different DLLs, some of them are loaded on runtime. So the user can call the function WSwup_GiveVersionInformation and will find the information about the different parts in this returned structure.

7.6.2 Field Documentation

7.6.2.1 int t_VersionInformation::nStructSize

must be loaded with sizeof(struct) to avoid Consistency-errors

7.6.2.2 int t_VersionInformation::nVersionMajor

the Major-Version-Nr ex: 1

7.6.2.3 int t_VersionInformation::nVersionMinor

the Minor-Version-Nr ex: 25

7.6.2.4 char t_VersionInformation::szDateTime[18]

Date and Time, ex: ”30.06.00 20:15:17 ”.

7.6.2.5 char t_VersionInformation::szDescription[100]

Description, ex: ”BSL-DLL for EGOLD+, Synchstation”.

7.6.2.6 char t_VersionInformation::szDeveloper[10]

the developer, ex: ”kuenstnj” / ”official”

The documentation for this struct was generated from the following file:

- wswuplib.h

Chapter 8

API for WSWUP-DLL File Documentation

8.1 wswuplib.h File Reference

This is the Export-Header for the WSWUPLIB-Library contains definition groups : -init functions -comport information functions -DLL-Version-Information-Functions -Synch-Station specific Functions -ERROR-Functions, to get information about the last ERROR -BFB-Lib-Functions, used before and after SOFTWARE-Update to check if a Software-Update is possible and if it was successfull ! -FILE-Functions, read XBI-File, release XBI-File, get Information about XBI-File -DEBUG-Functions for enabling and disabling OnLine-Debug I recommend DEBUGVIEW from www.sysinternals.com as viewer Currently only works in the debug-versions of the DLLs ! -SOFTWARE-Update-Functions.

```
#include "xbi_info.h"
```

Data Structures

- struct t_ComPort

  Structure with Serial port Information.

- struct t_ErrorStruct

  A structure to allow passing of parameters for error-messages from the Lib to the user of the Lib.

- struct t_Ports

  Structure with complete port configuration information.

- struct t_SwInformation

  Structure with Information about the SW in the XBI-File or the Mobile itself.

- struct t_USBPort

  Structure with USB port Information.

- struct t_VersionInformation

  Version-Information for the different parts of the Library.

Defines

- #define WSWUPLIB_INTERFACE_VERSION_MAJOR 1

    Interface-Version of the whole library (this file), must be increased on interface-changes.

- #define WSWUPLIB_INTERFACE_VERSION_MINOR 37

    Interface-Version of the whole library, minor-number.

- #define AMOUNT_OF_SWUP_PORTS 12

    general amount of comports supported by winswup

- #define AMOUNT_OF_UPDATES 4

    amount of parallel updates through different comports

- #define WSWUPLIB_USBPORT_OFFSET 100

    offset for USB Ports from the first USB Port (in case more Com-Ports)

- #define MAX_ENTRIES 50

    Amount of entries in the error-"stack".

- #define NO_PARAM ""

    an empty parameter-list

- #define SIZE_OF_PINFO 10

    max amount of parameters for errors

- #define SIZE_OF_PSTRING 2000

    maximum size of a error-string-parameter

- #define SWUPLIBDLLEXIMPORT __declspec(dllimport)
- #define MAIN_DEBUG_GROUP 0x01

    Enable Main-Functionality Debugging.

- #define FILE_DEBUG_GROUP 0x02

    Enable File-Function Debugging.

- #define COMM_DEBUG_GROUP 0x04

    Enable Debugging on Communciation-Functions.

- #define BFB_DEBUG_GROUP 0x08

    Enable Debugging on BFB-Communciation-Functions.

- #define CONSISTENCE_PRODUCT_ERROR 0x01

    SW and Mobile are different products.

- #define CONSISTENCE_SWVERSION_ERROR 0x02

    Mobile has newer SVN than the new SW.

- #define CONSISTENCE_SWDATE_ERROR 0x04

    Mobile has newer SW-Date than the new SW.

- #define WSWUP_SW_TYPE_MOBSW_STRING ”MobSw”
- #define WSWUP_SW_TYPE_EESIMU_STRING ”Eesimu”
- #define WSWUP_SW_TYPE_VOICEMEMO_STRING ”Voice Memo”
- #define WSWUP_SW_TYPE_CODEONLY_STRING ”Code Only”
- #define WSWUP_SW_TYPE_LANGONLY_STRING ”Lang Only”
- #define WSWUP_SW_TYPE_CODEANDLANG_STRING ”Code and Lang”
- #define WSWUP_SW_TYPE_DIFFFILE_STRING ”DiffFile”
- #define WSWUP_SW_TYPE_EXTNEWSPLIT_STRING ”Extended New Split”

Typedefs

- typedef void(∗ VoltageCheckCallBack)(WORD wUpdateNr, unsigned int unMaxCalls, unsigned int unCurrCall)

  Callbackfunction for VoltageCheck that is performed.

- typedef void(∗ UpdateSuccessCallBack)(WORD wUpdateNr, unsigned int unMaxCalls, unsigned int unCurrCall)

  Callbackfunction for UpdateSuccess that is performed after swup.

- typedef void(∗ FileReadProgressCallBack)(unsigned long ulAmountOfBytesToRead, unsigned long ulCurrentRead)

  Callbackfunction for FileReading.

- typedef void(∗ BootstrapProgressCallBack)(WORD wUpdateNr, WORD unMaxCalls, WORD unCurrCall)

  Callbackfunction for Bootstraploader.

- typedef void(∗ UpdateSwTransmissionStartCallBack)(WORD wUpdateNr, unsigned long ulBytesToTransfer)

  Callbackfunction for UpdateSwTransmissionStart.

- typedef void(∗ UpdateSwTransmissionProgressCallBack)(WORD wUpdateNr, unsigned long ulBytesTransfered)

  Callbackfunction for UpdateSwTransmissionProgress.

- typedef void(∗ FlashTypeCallBack)(WORD wUpdateNr, WORD wFlashId, char ∗pszFlashString)

  Callbackfunction for FlashType.

- typedef void(∗ EraseProgressCallBack)(WORD wUpdateNr, WORD wAmountFlashBlocks, WORD wCurrentBlock)

  Callbackfunction for EraseProgress.

- typedef void(∗ MobileSwTransmissionStartCallBack)(WORD wUpdateNr, unsigned long ulBytesToTransfer)

  Callbackfunction for MobileSwTransmissionStart.

- typedef void(∗ MobileSwTransmissionProgressCallBack)(WORD wUpdateNr, unsigned long ulBytesTransfered)

  Callbackfunction for MobileSwTransmissionProgress.

- typedef void(∗ MobileSwUpdateSuccessCallBack)(WORD wUpdateNr)

  Callbackfunction for SoftwareUpdateSuccess.

- typedef void(∗ MobileSwUpdateErrorCallBack)(WORD wUpdateNr)

  Callbackfunction for SoftwareUpdateError.

- typedef BOOL(∗ MobileSwUpdateConsistenceProblemCallBack)(WORD wUpdateNr, t_SwInformation ∗pMobileInfo, t_SwInformation ∗pNewSwInfo, WORD wErrorComposition)

  Callbackfunction for SoftwareUpdateConsistence-Problem.

- typedef BOOL(∗ MobileSwUpdateInfoCallBack)(WORD wUpdateNr, t_SwInformation ∗pMobileInfo, t_SwInformation ∗pNewSwInfo)

  Callbackfunction for SoftwareUpdateInfo.

Enumerations

- enum t_SwupCom {

  SwupCom1 = 0, SwupCom2 = 1, SwupCom3 = 2, SwupCom4 = 3,

  SwupCom5 = 4, SwupCom6 = 5, SwupCom7 = 6, SwupCom8 = 7,

  SwupCom9 = 8, SwupCom10 = 9, SwupCom11 = 10, SwupCom12 = 11,

  SwupUsb0 = 0+WSWUPLIB_USBPORT_OFFSET, SwupUsb1 = 1+WSWUPLIB_USBPORT_OFFSET, SwupUsb2 = 2+WSWUPLIB_USBPORT_OFFSET, SwupUsb3 = 3+WSWUPLIB_USBPORT_OFFSET }

  An enumeration for all possible Comports.

- enum t_USBNr { enUSB0 = 0, enUSB1, enUSB2, enUSB3 }

  An enumeration for all possible USB ports.

- enum t_USB_Serial_Switch { enUsbSerialAutoSwitch = 0, enUsbSerialManualSwitch, enUsbOnly, enSerialOnly }

  An enumeration for all possible serial / USB switch configurations.

- enum t_InfoSelector { enSwupLibDll, enSwupSeriDll, enBootStrapDll, enUpdateSW }

  Different parts of the whole library, for which the user can request version-information.

- enum t_DebugLevel { enNoDebug = 0, enLowDebug = 1, enMidDebug = 2, enHighDebug = 3 }

  Debugging-Level.

- enum t_InfoCallBackTime { enInfoCallBackEarlyWithBfb, enInfoCallBackLateDuringUpdate }

  Callback-Time for MobileSwUpdateInfoCallBack().

- enum t_UpdateMode { enForceBSLBehaviour, enForceNewConcept, enUpdateConceptAutoDetect, enSpecialUsbIbootExe }

  Behaviour of the Library concerning the new Update-Concept (xx45).

Functions

- BOOL SWUPLIBDLLEXIMPORT WSwup_InitLibrary (WORD wMajorNumber, WORD wMinorNumber)

  This function must be called by every user of the library once on starting.

- BOOL SWUPLIBDLLEXIMPORT WSwup_CheckComPortAndSpeed (t_SwupCom WhichCom, DWORD dwBaudrate)

  This function only tries to open the selected comport with the given baudrate.

- BOOL SWUPLIBDLLEXIMPORT WSwup_GiveVersionInformation (t_VersionInformation ∗pInformation, t_InfoSelector WhichInfo)

  Returns Version Information about the different Parts of WSWUPLIB.

- void SWUPLIBDLLEXIMPORT WSwup_PrepareForUpdateWithSynchStation (BOOL fSynchStationPresent)

  Prepares for performing the update via Synch-Station.

- BOOL SWUPLIBDLLEXIMPORT WSwup_ReloadUpdateDll (void)

  Reloads the DLL that is responsible for performing the Update.

- DWORD SWUPLIBDLLEXIMPORT WSwup_GetLastError (t_SwupCom ComPort)

  Returns an specific Error-Code about last Error on this comport.

- void SWUPLIBDLLEXIMPORT WSwup_SetErrorTextBehaviourToEnduser (BOOL fEndUser)

  Switches the behaviour of function WSwup_GetLastErrorString() to a less informative behaviour.

- char SWUPLIBDLLEXIMPORT ∗ WSwup_GetLastErrorString (t_SwupCom ComPort)

  Retrieves the Error-String of the last Error for the given ComPort.

- SWUPLIBDLLEXIMPORT t_ErrorStruct ∗ WSwup_GiveLastError (WORD wUpdateNr)

  Returns the last Error for the indicated updatenumber with its specific parameters.

- void SWUPLIBDLLEXIMPORT WSwup_InstallBfbCallBackFunctions (VoltageCheckCallBack pfnVoltageCheck, UpdateSuccessCallBack pfnSuccessCheck)

  Installs the needed callbackfunctions for Pre and Post-Checks via BFB-Library.

- BOOL SWUPLIBDLLEXIMPORT WSwup_CheckUpdateVoltage (WORD wUpdateNumber, t_SwupCom WhichCom, unsigned short ∗punVoltage, unsigned long ulSpeed)

  Tries to communicate to a "living" mobile to determine if the voltage is ok.

- BOOL SWUPLIBDLLEXIMPORT WSwup_InitCopro (t_SwupCom WhichCom)

  forces mobile into CoproUpdateMode to determine if the voltage is ok.

- BOOL SWUPLIBDLLEXIMPORT WSwup_SipcCheckVoltage (WORD wUpdateNr, unsigned short ∗punVoltage, BOOL ForceFlag)

  forces mobile into CoproUpdateMode to determine if the voltage is ok.

- BOOL SWUPLIBDLLEXIMPORT WSwup_CheckUpdateSuccess (WORD wUpdateNumber, t_SwupCom WhichCom)

  Checks if Mobile-SW-Update was successfull.

- void SWUPLIBDLLEXIMPORT WSwup_PrepareForUpdateWithUnknownProject (char ∗pszUnknownProjectName, char ∗pszDerivedFromKnownProjectName)

  Prepares the WSWUP-Library for updating an "unknown" project.

- BOOL SWUPLIBDLLEXIMPORT WSwup_ReadXbiFile (t_SwInformation ∗pSwInformation, FileReadProgressCallBack pfnReadProgress, char ∗pszXbiName)

  Reads the XBI-File with the given name into memory, so a swup can be performed.

- BOOL SWUPLIBDLLEXIMPORT WSwup_ReadXbiFileFromHeap (t_SwInformation ∗pSwInformation, unsigned char ∗pucMem, unsigned long ulBytesInHeap)

  Checks an XBI-File in heap so a swup can be performed.

- unsigned int SWUPLIBDLLEXIMPORT WSwup_ReadXbiHeaderFromHeap (unsigned long ulNumXbiBytes, t_ExtendedInfo ∗ptrWohin, unsigned char ∗pucMem)

  Reads a XBI-Header directly from Heap Just calls the internal function ReadXbiHeaderInHeap, only used in DLL.

- unsigned char SWUPLIBDLLEXIMPORT ∗ WSwup_GivePtrToBinData (void)

  once WSwup_ReadXbiFileFromHeap() is successfully called, this function delivers a ptr to start of bin data in heap.

- char SWUPLIBDLLEXIMPORT ∗ WSwup_GetLastFileError (void)

  Returns a pointer to a string describing the last error occured on FileReading.

- void SWUPLIBDLLEXIMPORT WSwup_CloseXbiFile (void)

  Releases all memory allocated on WSwup_ReadXbiFile .

- void SWUPLIBDLLEXIMPORT WSwup_CloseXbiFileHeap (void)

  Releases all memory allocated inside of WSwup_ReadXbiFileFromHeap .

- void SWUPLIBDLLEXIMPORT WSwup_EnableOnlineDebugging (t_DebugLevel RequestedDebugLevel, WORD wRequestedDebugGroups)

  Enables Online Debugging on the requested DebugLevel for the requested Groups.

- void SWUPLIBDLLEXIMPORT WSwup_DisableOnlineDebugging (void)

  Disables Online Debugging.

- void SWUPLIBDLLEXIMPORT WSwup_EnableFileDebugging (t_DebugLevel RequestedDebugLevel, WORD wRequestedDebugGroups, char ∗szFilePrefix)

  Enables File Debugging on the requested DebugLevel for the requested Groups.

- void SWUPLIBDLLEXIMPORT WSwup_DisableFileDebugging (void)

  Disables File Debugging.

- void SWUPLIBDLLEXIMPORT WSwup_InstallSoftwareUpdateCallBackFunctions (BootstrapProgressCallBack pfnBootstrapProgress, UpdateSwTransmissionStartCallBack pfnUpdateSwTransmissionStart, UpdateSwTransmissionProgressCallBack pfnUpdateSwTransmissionProgress, FlashTypeCallBack pfnFlashType, EraseProgressCallBack pfnEraseProgress, MobileSwTransmissionStartCallBack pfnMobileSwTransmissionStart, MobileSwTransmissionProgressCallBack pfnMobileSwTransmissionProgress, MobileSwUpdateSuccessCallBack pfnMobileSwUpdateSuccess, MobileSwUpdateErrorCallBack pfnMobileSwUpdateError, MobileSwUpdateConsistenceProblemCallBack pfnMobileConsistenceProblem)

Installs all the needed callbackfunctions for the whole software-update.

- void SWUPLIBDLLEXIMPORT WSwup_InstallSoftwareInfoCallBackFunction (MobileSwUpdateInfoCallBack pfnMobileSwUpdateInfoCallBack, t_InfoCallBackTime WhichCallbackTime)

  Installs the MobileSwUpdateInfoCallBack-Function.

- BOOL SWUPLIBDLLEXIMPORT WSwup_SetUpdateConceptMode (t_UpdateMode WhichMode)

  Tells the library how to perform the Software-Update.

- BOOL SWUPLIBDLLEXIMPORT WSwup_SetUsbParams (WORD wUpdateNr, t_SwupCom WhichUsb, BOOL Force)

  Tells the library how to configure the modem's Software-Update via Usb.

- void SWUPLIBDLLEXIMPORT WSwup_ResetUsbParams (WORD wUpdateNr)

  Tells the library how to configure the modem's Software-Update via Usb.

- BOOL SWUPLIBDLLEXIMPORT WSwup_PerformSoftwareUpdate (WORD wUpdateNr, t_SwupCom WhichCom, unsigned long ulSpeed)

  Performs the Software-Update on the given ComPort with the given Baudrate.

- BOOL SWUPLIBDLLEXIMPORT WSwup_SetBootPIN (WORD wUpdateNr, unsigned int unPinSize, unsigned char ∗pucBootPIN)

  Save a PIN for an Update, This PIN must be provided by data stored in PICS and is sended during BSL connect attempt.

8.1.1 Detailed Description

This is the Export-Header for the WSWUPLIB-Library contains definition groups : -init functions -comport information functions -DLL-Version-Information-Functions -Synch-Station specific Functions -ERROR-Functions, to get information about the last ERROR -BFB-Lib-Functions, used before and after SOFTWARE-Update to check if a Software-Update is possible and if it was successfull ! -FILE-Functions, read XBI-File, release XBI-File, get Information about XBI-File -DEBUG-Functions for enabling and disabling OnLine-Debug I recommend DEBUGVIEW from www.sysinternals.com as viewer Currently only works in the debug-versions of the DLLs ! -SOFTWARE-Update-Functions.

8.1.2 Define Documentation

8.1.2.1 #define AMOUNT_OF_SWUP_PORTS 12

general amount of comports supported by winswup

8.1.2.2 #define AMOUNT_OF_UPDATES 4

amount of parallel updates through different comports

8.1.2.3 #define SWUPLIBDLLEXIMPORT __declspec(dllimport)

8.1.2.4 #define WSWUP_SW_TYPE_CODEANDLANG_STRING "Code and Lang"

8.1.2.5 #define WSWUP_SW_TYPE_CODEONLY_STRING "Code Only"

8.1.2.6 #define WSWUP_SW_TYPE_DIFFFILE_STRING "DiffFile"

8.1.2.7 #define WSWUP_SW_TYPE_EESIMU_STRING "Eesimu"

8.1.2.8 #define WSWUP_SW_TYPE_EXTNEWSPLIT_STRING "Extended New Split"

8.1.2.9 #define WSWUP_SW_TYPE_LANGONLY_STRING "Lang Only"

8.1.2.10 #define WSWUP_SW_TYPE_MOBSW_STRING "MobSw"

8.1.2.11 #define WSWUP_SW_TYPE_VOICEMEMO_STRING "Voice Memo"

8.1.2.12 #define WSWUPLIB_USBPORT_OFFSET 100

offset for USB Ports from the first USB Port (in case more Com-Ports)

8.1.3 Enumeration Type Documentation

8.1.3.1 enum t_SwupCom

An enumeration for all possible Comports.

Enumeration values:

SwupCom1	Comport 1.
SwupCom2	Comport 2.
SwupCom3	Comport 3.
SwupCom4	Comport 4.
SwupCom5	Comport 5.
SwupCom6	Comport 6.
SwupCom7	Comport 7.
SwupCom8	Comport 8.
SwupCom9	Comport 9.
SwupCom10	Comport 10.
SwupCom11	Comport 11.
SwupCom12	Comport 12.
SwupUsb0	UsbPort 0.
SwupUsb1	UsbPort 1.
SwupUsb2	UsbPort 2.
SwupUsb3	UsbPort 3.

8.1.3.2 enum t_USB_Serial_Switch

An enumeration for all possible serial / USB switch configurations.

Enumeration values:

enUsbSerialAutoSwitch	selects automatical switch between USB and serial port
enUsbSerialManualSwitch	selects manual switch between USB and serial port
enUsbOnly	no switch to serial port possible (use USB only)
enSerialOnly	no switch to Usb port possible, use serial only

8.1.3.3 enum t_USBNr

An enumeration for all possible USB ports.

Enumeration values:

- **enUSB0**
- **enUSB1**
- **enUSB2**
- **enUSB3**

8.2 wswupuser.dox File Reference

Chapter 9

API for WSWUP-DLL Page Documentation

9.1 The Demo for the WIN-SWUP-API

```
/*
                          Siemens AG
                    Mobile Radio Terminals
                      Munich,  Germany
                    COMPANY CONFIDENTIAL
                    All Rights Reserved

.AUTHOR         J. Kuenstner, ICP CD MP GSM RD M57

.PROJECT        SWUPDEMO

.SHORT_DESCR    Main-Program for SWUPDEMO


.SW_COMPONENT

.SW_TYPE

.VERSION        1.6

.DATE           2001-09-17

.EXIT_CODES

.STATUS         FIRST_TRY

.CHANGE_CONTROL

Version |  Date    | Changed by | Reason for Change
--------+----------+------------+----------------------
 1.0    |05.03.2000| Jogi       | Modul created
--------+----------+------------+----------------------
 1.1    |20.04.2001| Jogi       | New interface to library because of consistence-check-callback !
--------+----------+------------+----------------------
 1.2    |2001-06-18| Jogi       | New functions for library-init and to to test new update-concept
--------+----------+------------+----------------------
 1.3    |2001-06-29| W. Hettich | testnew heap method,some ugly testdefines, but lintable now
--------+----------+------------+----------------------
 1.4    |2001-08-07| W. Hettich | uses new close up function from heaplib
--------+----------+------------+----------------------
 1.5    |2001-09-07| JKue       | changed to use new error-demo
```

```
--------+----------+------------+----------------------
 1.6    |2001-09-17| PMackenthun| Sleep() inserted


.STILL_TODO

*/

/* INDENT: -nbr -nce -ei -ip -lp -pro -sc -c:33 -cci:3 -cd:33 -ci:3 -cli:3 -cp:0 -d:0 -di:0 -i:3 -l:78 -lc


/*  ***  INCLUDES  ***  ***************************************************** */

#include <windows.h>
// #include <ansi_c.h>    /* cvi inserted */
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <sys/stat.h>     /* stat */
#include <io.h>           /* access() */

#include "wswuplib.h"
#include "err_demo.h"

/*  ***  DEFINES  ***  ******************************************************* */

/*  ***  TYPEDEFS ***  ******************************************************* */

/*  ***  EXTERNALS  ***  ***************************************************** */

/*  ***  CONSTANTS ***  ****************************************************** */

/*  ***  VARIABLES  ***  ***************************************************** */

unsigned long ulAmountUpdateData;
unsigned long ulGlobalMaxVal;
volatile BOOL fErrorOccured = FALSE;   /* must be volatile, because otherwise it will be optimized away in
volatile BOOL fSuccessOccured = FALSE;

/*lint -e528, szErrorMsg: access only by function */
static char szErrorMsg[255] = {'\0'};


/*  ***  PROTOTYPES  ***  **************************************************** */
static BOOL MallocHeapAndReadFile(unsigned char **ppucLocalHeap, unsigned long *pulBytesInHeap, char *pszF

/*  ***  MODUL COMMENTS  ***  ************************************************ */

/*  ***  FUNCTIONS  ***  ***************************************************** */
void ShowFileReadProgress( unsigned long ulMaxVal, unsigned long ulAktVal )
{
   printf("Read %ld Bytes of %ld Bytes\r",ulAktVal,ulMaxVal );
}

/*lint -esym(715, unUpdateNr) */
void ShowVoltageCheckProgress( WORD unUpdateNr, unsigned int unMaxCalls, unsigned int unCurrCall )
{
   printf("." );
   if( unCurrCall == unMaxCalls )
   {
      printf("\n");
   }
}
/*lint +esym(715, unUpdateNr) */

/*lint -esym(715, unDlgBoxNumber) */
```

```
void ShowUpdateCheckProgress( WORD unDlgBoxNumber, unsigned int unMaxCalls, unsigned int unCurrCall )
{
   if( unCurrCall == 1 )
   {
      printf("Testing new SW " );
   }
   printf("." );
   if( unCurrCall == unMaxCalls )
   {
      printf("\n");
   }
}
/*lint +esym(715, unDlgBoxNumber) */


/*lint -esym(715, unDlgBoxNumber) */
void UpdateSwTransmissionStart( WORD unDlgBoxNumber, unsigned long ulMaxVal )
{
   ulAmountUpdateData = ulMaxVal;
   printf("Transmitting Update-SW" );
}
/*lint +esym(715, unDlgBoxNumber) */

/*lint -esym(715, unDlgBoxNumber) */
void ShowUpdateSwTransmissionProgress( WORD unDlgBoxNumber, unsigned long ulAktVal )
{
   printf( "." );
   if( ulAmountUpdateData == ulAktVal )
   {
      printf("\n");
   }
}
/*lint +esym(715, unDlgBoxNumber) */


/*lint -esym(715, wUpdateNr) */
void ShowFlashBlocksEraseProgress( WORD wUpdateNr, WORD wMaxAmount , WORD wAmountBlocksErased  )
{
   printf( "Erasing %3d of %3d Flashblocks\r", wAmountBlocksErased , wMaxAmount );
   if( wAmountBlocksErased == wMaxAmount )
   {
      printf( "\n" );
   }
}
/*lint +esym(715, wUpdateNr) */


/*lint -esym(715, wDlgBoxNumber) */
void ShowBootStrapProgress( WORD wDlgBoxNumber, WORD wMaxAmount , WORD wCurrAmount  )
{
   if( wCurrAmount == 1 )
   {
      printf("Connecting to Mobile " );
   }
   printf( "." );
   if( wCurrAmount == wMaxAmount )
   {
      printf("\n" );
   }
}
/*lint +esym(715, wDlgBoxNumber) */

/*lint -esym(715, unDlgBoxNumber) */
void StartTransmission( WORD unDlgBoxNumber, unsigned long ulBytesToTransfer  )
{
   ulGlobalMaxVal = ulBytesToTransfer;
}
```

```
/*lint +esym(715, unDlgBoxNumber) */


/*lint -esym(715, unDlgBoxNumber) */
void ShowMobileSwTransmissionProgress( WORD unDlgBoxNumber, unsigned long ulAktVal )
{
   printf("Transfering new Mobile-SW %4ld kByte\r" , ulAktVal / 1024 );
   if( ulAktVal == ulGlobalMaxVal )
   {
      printf("\n");
   }
}
/*lint +esym(715, unDlgBoxNumber) */


/*lint -esym(715, wUpdateNr) */
/*lint -esym(715, *pszFlashString) */
void ShowFlashCode( WORD wUpdateNr, WORD wFlashCode , char * pszFlashString )
{
   printf( "Flash-Code %04X\n", wFlashCode  );
}
/*lint +esym(715, *pszFlashString) */
/*lint +esym(715, wUpdateNr) */

/*lint -esym(715, wUpdateNr) */
void UpdateSuccess( WORD wUpdateNr )
{
   printf( "Success occurred \n" );
   fSuccessOccured = TRUE;
}
/*lint +esym(715, wUpdateNr) */


/*lint -esym(715, wUpdateNr) */
void UpdateError( WORD wUpdateNr )
{
   printf( "Error occurred \n" );
   fErrorOccured = TRUE;
}
/*lint +esym(715, wUpdateNr) */

/*lint -esym(715, wUpdateNr) */
BOOL AskForConsistenceProblem( WORD wUpdateNr, t_SwInformation * pMobileInfo, t_SwInformation * pNewSwInfo
{
   char szErrorString[1000];
   int c;
   switch( wErrorComposition )
   {
      case CONSISTENCE_PRODUCT_ERROR:
         (void)wsprintf( szErrorString, "You are trying to update a SW for a %s into a %s",pNewSwInfo->szP
         break;
      case CONSISTENCE_SWVERSION_ERROR:
         (void)wsprintf( szErrorString, "You are trying to update a SW Version %s into a Mobile with %s",p
         break;
      case CONSISTENCE_SWDATE_ERROR:
         (void)wsprintf( szErrorString, "You are trying to update a SW with Date %s into a Mobile with %s"
         break;
      case ( CONSISTENCE_SWDATE_ERROR | CONSISTENCE_SWVERSION_ERROR ) :
         (void)wsprintf( szErrorString, "You are trying to update a SW Version %s with Date %s into a Mobi
                                                                                          pNewSwInfo->szCo
                                                                                                   pMobile
                                                                                                   pMobile
         break;
      default:;
   }
   strcat( szErrorString,"\nDo you want to continue ?");
   printf( "%s\nPress y or j to continue, any other key to abort : ", szErrorString  );
   c = getchar();
   c= toupper( c );
   printf("c=\"%c\"\n",c );
   if(( c == ’J’ ) || (  c == ’Y’ ))
   {
      return TRUE;
   }
   else
   {
      return FALSE;
   }
}
/*lint +esym(715, wUpdateNr) */




void main( int argc, char **argv  )
{
   t_SwupCom MyCom = SwupCom4;
   t_SwInformation InformationAboutSoftware;
   BOOL fLocalErrorOccured ;
   BOOL fLocalSuccessOccured ;
   unsigned long ulBaudRate;
   WORD unUpdateNr = 0;

   unsigned char * pucHeap;
   unsigned long ulBytesInHeap;
   WORD unVoltage;

   if( argc != 4 )
   {
      printf( "Call swupdemo PortNr Baudrate File \n");
      exit( 1 );
   }

   MyCom = (t_SwupCom)(strtoul( argv[1],NULL,10 ) - 1);
   ulBaudRate = strtoul( argv[2], NULL, 10 );

   printf( "Running SWUPDEMO on Port %d with %ld Baud with File %s\n",  (int)MyCom + 1, ulBaudRate, argv[3

   if( WSwup_InitLibrary( WSWUPLIB_INTERFACE_VERSION_MAJOR, WSWUPLIB_INTERFACE_VERSION_MINOR  ) == FALSE )
   {
      printf( "Error on LIB-INIT : %s \n", WSwup_GetLastErrorString( MyCom ));
      return;
   }
    // uncomment the following line, if you want to compile a SWUP with old behaviour
    // ( old behaviour is impossible for consumer-update with the real library  !!! )
    WSwup_SetUpdateConceptMode( enForceNewConcept );

   /* move file to heap, thats what a customer-tool-programmer has to do  */
   if ( FALSE == MallocHeapAndReadFile( &pucHeap, &ulBytesInHeap, argv[3] ))
      exit(1);

   /* check heap contents and prepare SW-Update */
   if ( WSwup_ReadXbiFileFromHeap( &InformationAboutSoftware,
                                   pucHeap,
                                   ulBytesInHeap ) )
   {
      WSwup_InstallBfbCallBackFunctions(  ShowVoltageCheckProgress, ShowUpdateCheckProgress );

      WSwup_InstallSoftwareUpdateCallBackFunctions(   ShowBootStrapProgress,
                                                      UpdateSwTransmissionStart,
                                                      ShowUpdateSwTransmissionProgress,
                                                      ShowFlashCode,
                                                      ShowFlashBlocksEraseProgress,
                                                      StartTransmission,
                                                  ShowMobileSwTransmissionProgress,
                                                  UpdateSuccess,
                                                  UpdateError,
                                                  AskForConsistenceProblem );
      printf("\nThis SW for %s , %s of %s\n", InformationAboutSoftware.szProductType,
                                        InformationAboutSoftware.szProductSVN,
                                        InformationAboutSoftware.szCompileDate );

      printf( "Checking voltage " );
      if( WSwup_CheckUpdateVoltage( 0 , MyCom, &unVoltage, ulBaudRate  ) != TRUE )
      {
         printf( "Error on Voltage-Check: %s\n", ErrDemo_GetLastErrorString( unUpdateNr ));
         free(pucHeap);
         return;
      }
      if( unVoltage < 3400 )
      {
         printf( "Voltage %d.%03d is to low, please charge", unVoltage / 1000, unVoltage % 1000 );
         free(pucHeap);
         return;
      }
      else
      {
         printf( "Voltage : %d.%03d V\n",unVoltage / 1000 ,unVoltage % 1000 );
         if( WSwup_PerformSoftwareUpdate( unUpdateNr, MyCom, ulBaudRate ) == FALSE )
         {
            printf( "Error on Start Update : %s\n", ErrDemo_GetLastErrorString( unUpdateNr ));
            free(pucHeap);
            return ;
         }
         else
         {
            fLocalErrorOccured = FALSE;
            fLocalSuccessOccured = FALSE;
            while( ( fLocalErrorOccured == FALSE ) && ( fLocalSuccessOccured == FALSE ))
            {
               fLocalErrorOccured = fErrorOccured ;
               fLocalSuccessOccured = fSuccessOccured;
               Sleep(100);  // give some processor time to other applications
            }
            if( fErrorOccured )
            {
               printf( "Error on Update : %s\n", ErrDemo_GetLastErrorString( unUpdateNr ));
            }
            else
            {
               if( WSwup_CheckUpdateSuccess( unUpdateNr , MyCom ) != TRUE )
               {
                  printf("\nError on testing new SW : %s\n", ErrDemo_GetLastErrorString( unUpdateNr ));
               }
               else
               {
                  printf( "Mobile-SW works ...\n");
               }
            }
         }
      }
   }
   else
   {
      printf( "Error on Reading : %s\n", ErrDemo_GetLastErrorString( unUpdateNr ));
   }
   WSwup_CloseXbiFileHeap( );
   free( pucHeap );

}
```

```
static BOOL MallocHeapAndReadFile(unsigned char **ppucLocalHeap, unsigned long *pulBytesInHeap, char *pszF
{
   FILE *fp;
   unsigned char * pucXbiInHeap;
   struct stat file_stat;
   unsigned long ulFileSize;
   unsigned long ulReadBytes;

   /* get file size */
   if (stat(pszFileName, &file_stat) == 0)
   {
      ulFileSize = (unsigned long) file_stat.st_size;
   }
   else
   {
      return FALSE;
   }

   /* get heap for file to read */
   pucXbiInHeap = ( unsigned char * ) malloc( (size_t) ulFileSize );
   if( pucXbiInHeap == NULL  )
   {
      printf( "\nCould not allocate %lu Bytes for XBI-File", ulFileSize );
      return FALSE;
   }

   /* open file */
   fp = fopen( pszFileName, "rb"  );

   /* read complete file to heap */
   ulReadBytes = fread( pucXbiInHeap, sizeof( unsigned char ), (size_t) ulFileSize, fp );
   if ( ferror(fp) )
   {
      printf("\nfread error!");
      fclose( fp );
      free( pucXbiInHeap );
      return FALSE;
   }
   /* passed end of file */
   if ( feof(fp) )
   {
      fp = fp; // just for debugging cases
   }
   /* all data now in heap , close file */
   fclose( fp );

   /* init parameter out */
   *ppucLocalHeap = pucXbiInHeap;
   *pulBytesInHeap = ulReadBytes;
   return TRUE;
}
```

9.2 The Demo for the ERROR-Handling

```
/*                        Siemens AG
                     Mobile Radio Terminals
                       Munich,  Germany
                     COMPANY CONFIDENTIAL
                     All Rights Reserved

.AUTHOR         J. Kuenstner, ICP CD MP GSM RD M57

.PROJECT        SWUPDEMO

.SHORT_DESCR    Error-Handling-Demo for SWUPDEMO


.SW_COMPONENT

.SW_TYPE

.VERSION        1.2

.DATE           2002-09-04

.EXIT_CODES

.STATUS         FIRST_TRY

.CHANGE_CONTROL

Version |  Date    | Changed by | Reason for Change
--------+----------+------------+-----------------------
 1.0    |2001-09-06| Jogi       | Modul created
--------+----------+------------+-----------------------
 1.1    |2002-09-03| D. Dittmer | Necessary changes to create the exe-files.
--------+----------+------------+-----------------------
 1.2    |2002-09-04| D. Dittmer | Change some comments for better understanding.
--------+----------+------------+-----------------------

.STILL_TODO

*/

/* INDENT: -nbr -nce -ei -ip -lp -pro -sc -c:33 -cci:3 -cd:33 -ci:3 -cli:3 -cp:0 -d:0 -di:0 -i:3 -l:78 -lc


/*  ***  INCLUDES  ***  ***************************************************** */
#include <windows.h>
#include "wswuplib.h"
#include "werrenum.h"
#include "err_text.h"
#include "err_demo.h"

/*  ***  DEFINES  ***  ****************************************************** */

/*  ***  TYPEDEFS ***  ****************************************************** */

/*  ***  EXTERNALS  ***  **************************************************** */

/*  ***  CONSTANTS ***  ***************************************************** */

/*  ***  VARIABLES  ***  **************************************************** */

static BOOL fEndUserBehaviour =  TRUE;
/*  ***  PROTOTYPES  ***  *************************************************** */

/*  ***  MODUL COMMENTS  ***  *********************************************** */
/*  ***  FUNCTIONS  ***  **************************************************** */





static void SpecialSprintf( char * pszFinalString, char * pszErrorDescription, t_ErrorStruct * pErrStruct
{
    char * pszDestPtr;      // pointer into the final string
    char * pszErrorFmPtr;   // pointer into the given description
    WORD wOutPutCharCount;  // counts the characters in the output-buffer
    char * pOwnFormatString; // points to our own simplified format-string
    WORD wStringBufferIndex = 0;
    WORD wLongBufferIndex = 0;

    char szFormatBufferForSprintf[20];  // to store the format for sprintf
    char * pszFormBufPtr;               // pointer to the above buffer
    char ch;                            // holds one character while detecting the format-string

    pszDestPtr = pszFinalString; // set the destination-ptr


    pOwnFormatString = pErrStruct->ucParamInfo;

    // loop throug the description until a % is found
    for ( pszErrorFmPtr = pszErrorDescription; *pszErrorFmPtr != '\0'; pszErrorFmPtr++ )
    {
        if ( *pszErrorFmPtr != '%')   // if the character it is not a % send it to the final buffer
        {
            wOutPutCharCount = wsprintf( pszDestPtr, "%c", *pszErrorFmPtr );
            pszDestPtr += wOutPutCharCount;
        }
        else
        {
            pszFormBufPtr = szFormatBufferForSprintf;
                                        // now we have the start of a format-string,
                                        // so copy it to a format-buffer
                                        // and let sprintf do the rest ...


            *pszFormBufPtr++ = *pszErrorFmPtr++;  // store the first % to avoid conflict with end-conditio
            do
            {
                ch = (unsigned char ) *pszErrorFmPtr;
                ch = (unsigned char) toupper(ch);
                *pszFormBufPtr++ = *pszErrorFmPtr++;
            }
            // according to K/R 2.nd German Edition , P. 243
            while (  ( ch != 'D') &&
                     ( ch != 'I') &&
                     ( ch != 'O') &&
                     ( ch != 'X') &&
                     ( ch != 'U') &&
                     ( ch != 'C') &&
                     ( ch != 'S') &&
                     ( ch != 'F') &&
                     ( ch != 'E') &&
                     ( ch != 'G') &&
                     ( ch != 'P') &&
                     ( ch != 'N') &&
                     ( ch != '%') );
            pszErrorFmPtr--; // Skip back one char , because we did one ++ to much in the above while-loop
            *pszFormBufPtr = '\0';  // Format-String for sprintf is ready
            // Check for %%, special handling
            if( ch == '%' )
            {
                wOutPutCharCount = wsprintf( pszDestPtr, "%%%%" );
                pszDestPtr += wOutPutCharCount;
            }
            else
            {
                // now go through the simplified own format-buffer and detect where to get the parameter f
                ch = *pOwnFormatString;
                ch = toupper( ch );
                pOwnFormatString++ ;

                switch( ch )
                {
                    case 'S':
                        wOutPutCharCount = wsprintf( pszDestPtr, szFormatBufferForSprintf, pErrStruct->ucP
                        pszDestPtr += wOutPutCharCount;
                        wStringBufferIndex++;
                        break;
                    default:
                        wOutPutCharCount = wsprintf( pszDestPtr, szFormatBufferForSprintf, pErrStruct->ulP
                        pszDestPtr += wOutPutCharCount;
                        wLongBufferIndex++;
                        break;
                }
            } // end special treat for %%
        } // end format-detection via %
    } // end for-loop
} // end function




/*  ------------------------------------------------------------------------
 *
 * ErrDemo_GetLastErrorString
 *
 *  ------------------------------------------------------------------------ */
char  * ErrDemo_GetLastErrorString( WORD wUpdateNr )
{
    t_ErrorStruct * pMyErrStruct;
    WORD wErrorNumber;
    BOOL fErrorFound;
    WORD wErrArrayIndex;

    static char szDecodedErrorText[2000];


    // retrieve a pointer to the Error-Structure
    pMyErrStruct = WSwup_GiveLastError  ( wUpdateNr );

    // extract the Error-Number from the struct
    wErrorNumber =  ( WORD ) pMyErrStruct->dwErrorNumber;

    // Now loop through the Error-Text-Array to find the right error-text
    fErrorFound = FALSE;
    for( wErrArrayIndex = 0; wErrArrayIndex < ( ErrorDescriptionSize /  t_enumErrorDescriptionSize ) ; ++w
    {
        if( ErrorDescription[wErrArrayIndex].ui_enError  == wErrorNumber )
        {
            fErrorFound = TRUE;
            break;
        }
    }
    // if the error is not found, this means eventually that you are running with an old version of err_te
    if( !fErrorFound )
    {
        wsprintf( szDecodedErrorText, "Unknown Error Number %d occured", wErrorNumber );
        return szDecodedErrorText;
    }
    // Now check if there is an alternate text ( but only in enduser-behaviour )
    if( fEndUserBehaviour )
    {
        if( ErrorDescription[wErrArrayIndex].ui_enAltError != 0 )
        {
            // set the new alternate error-number
            wErrorNumber = ErrorDescription[wErrArrayIndex].ui_enAltError;

            // and loop once more through the error-array and find the alternate error with its text
            for( wErrArrayIndex = 0; wErrArrayIndex < ( ErrorDescriptionSize / t_enumErrorDescriptionSize
            {
                if( ErrorDescription[wErrArrayIndex].ui_enError  == wErrorNumber )
                {
                    fErrorFound = TRUE;
                    break;
                }
            }
            // if the error is not found, this means eventually that you are running with an old version o
            if( !fErrorFound )
            {
                wsprintf( szDecodedErrorText, "Unknown Error Number %d occured", wErrorNumber );
                return szDecodedErrorText;
            }
            else
            {
                return ErrorDescription[wErrArrayIndex].pszDescription;
            }
        }
    }
    // now do the special-sprintf, that analyses the format-strings and gets the parameter-info
    SpecialSprintf( szDecodedErrorText, ErrorDescription[wErrArrayIndex].pszDescription, pMyErrStruct );
    return szDecodedErrorText;
}
```

Index