DWLWin Download Application User Guide

AirPrime DWLWin

4114140
3.1
March 11, 2013

Important Notice

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. Although significant delays or losses of data are rare when wireless devices such as the Sierra Wireless modem are used in a normal manner with a well-constructed network, the Sierra Wireless modem should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Sierra Wireless accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Sierra Wireless modem, or for failure of the Sierra Wireless modem to transmit or receive such data.

Safety and Hazards

Do not operate the Sierra Wireless modem in areas where cellular modems are not advised without proper device certifications. These areas include environments where cellular radio can interfere such as explosive atmospheres, medical equipment, or any other equipment which may be susceptible to any form of radio interference. The Sierra Wireless modem can transmit signals that could interfere with this equipment. Do not operate the Sierra Wireless modem in any aircraft, whether the aircraft is on the ground or in flight. In aircraft, the Sierra Wireless modem **MUST BE POWERED OFF**. When operating, the Sierra Wireless modem can transmit signals that could interfere with various onboard systems.

Note: *Some airlines may permit the use of cellular phones while the aircraft is on the ground and the door is open. Sierra Wireless modems may be used at this time.*

The driver or operator of any vehicle should not operate the Sierra Wireless modem while in control of a vehicle. Doing so will detract from the driver or operator's control and operation of that vehicle. In some states and provinces, operating such communications devices while in control of a vehicle is an offence.

Limitations of Liability

This manual is provided "as is". Sierra Wireless makes no warranties of any kind, either expressed or implied, including any implied warranties of merchantability, fitness for a particular purpose, or noninfringement. The recipient of the manual shall endorse all risks arising from its use.

The information in this manual is subject to change without notice and does not represent a commitment on the part of Sierra Wireless. SIERRA WIRELESS AND ITS AFFILIATES SPECIFICALLY DISCLAIM LIABILITY FOR ANY AND ALL DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR REVENUE OR ANTICIPATED PROFITS OR REVENUE ARISING OUT OF THE USE OR INABILITY TO USE ANY SIERRA WIRELESS PRODUCT, EVEN IF SIERRA WIRELESS AND/OR ITS AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THEY ARE FORESEEABLE OR FOR CLAIMS BY ANY THIRD PARTY.

Notwithstanding the foregoing, in no event shall Sierra Wireless and/or its affiliates aggregate liability arising under or in connection with the Sierra Wireless product, regardless of the number of events, occurrences, or claims giving rise to liability, be in excess of the price paid by the purchaser for the Sierra Wireless product.

Customer understands that Sierra Wireless is not providing cellular or GPS (including A-GPS) services. These services are provided by a third party and should be purchased directly by the Customer.

Patents

Copyright

Trademarks

Contact Information

Sales Desk:	Phone:	1-604-232-1488
	Hours:	8:00 AM to 5:00 PM Pacific Time
	E-mail:	sales@sierrawireless.com
Post:	Sierra Wireless 13811 Wireless Way Richmond, BC Canada V6V 3A4	
Technical Support:	support@sierrawireless.com	
RMA Support:	repairs@sierrawireless.com	
Fax:	1-604-231-1109	
Web:	www.sierrawireless.com	

Consult our website for up-to-date product descriptions, documentation, application notes, firmware upgrades, troubleshooting tips, and press releases: www.sierrawireless.com

Document History

Version	Date	Updates
001	April 2007	Created
002	August 2008	Revision
3.0	March 08, 2013	Updated document template and naming conventions
3.1	March 11, 2013	Updated specs in section 1.2 DWLWin, updated all instances of WMP100 to WMP1x0

Contents

List of Figures

1. Introduction

1.1. Updating the Flash Memory

There are two ways to update the contents of the embedded module flash memory:

- Xmodem download
- Boot mode download
- Both methods are described in the subsections below.

1.1.1. Xmodem Download

Xmodem download interacts with the downloader embedded in any Sierra Wireless firmware, and is triggered by the AT+WDWL command (please refer to the AT Commands Interface Manual for complete information). This method is very convenient because it requires only a host, which can be a PC or another microcontroller, communicating using the standard Xmodem protocol. But it cannot be used with blank flash memories, without a firmware inside.

1.1.2. Boot Mode Download

The boot mode download is the use of an alternate startup sequence of the baseband processor inside the embedded module. In boot mode, the execution of the code in flash memory is partially or entirely bypassed; the baseband processor runs instead an internal application able to communicate, using a proprietary protocol, with a serially linked remote party for flash upgrade purposes.

1.2. DWLWin

The DWLWin application can communicate and update embedded modules in boot mode. It is a Win32 application compatible with Microsoft Windows 2000, XP, Vista and 7. DWLWin's graphical user interface provides a convenient way to configure and monitor download operations in a single embedded module. For production purposes, its automation interface enables simultaneous downloads in several embedded modules connected on multi-port serial interfaces.

The serial link required between the host and the embedded module must include the following signals:

- GND
- TXD
- RXD
- RTS
- CTS

Note: *"Null-modem" cables cannot be used with DWLWin because they do not connect RTS/CTS signals between the host and embedded module.*

1.2.1. Download Packages

The firmware and its related settings and parameters are provided as a single installation package file for DWLWin (with the “.wpk” extension). This package file replaces all other binary or eeprom configuration files and performs all the necessary download operations to set up the firmware in the flash memory.

Moreover, download packages can be tuned using their specific interface and download option parameters (see section 6.1 Options-Related Functions for more information).

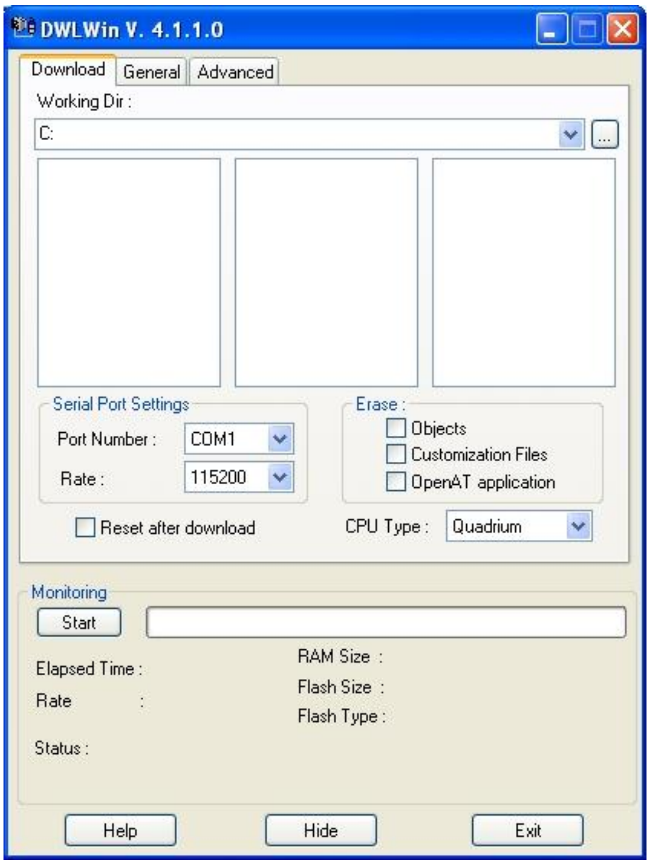
2. Using DWLWin in GUI Mode

The DWLWin GUI is a tabbed interface, with three tabs:

- Download
- General
- Advanced

2.1. Download Tab

This is the main tab of the application, and is shown in the figure below.



Working Dir : C:

Port Number : COM1

Rate : 115200

CPU Type : Quadrium

Figure 1. DWLWin Download Tab

The Download tab is where most of the download parameters are configured, as described in the bulleted list below.

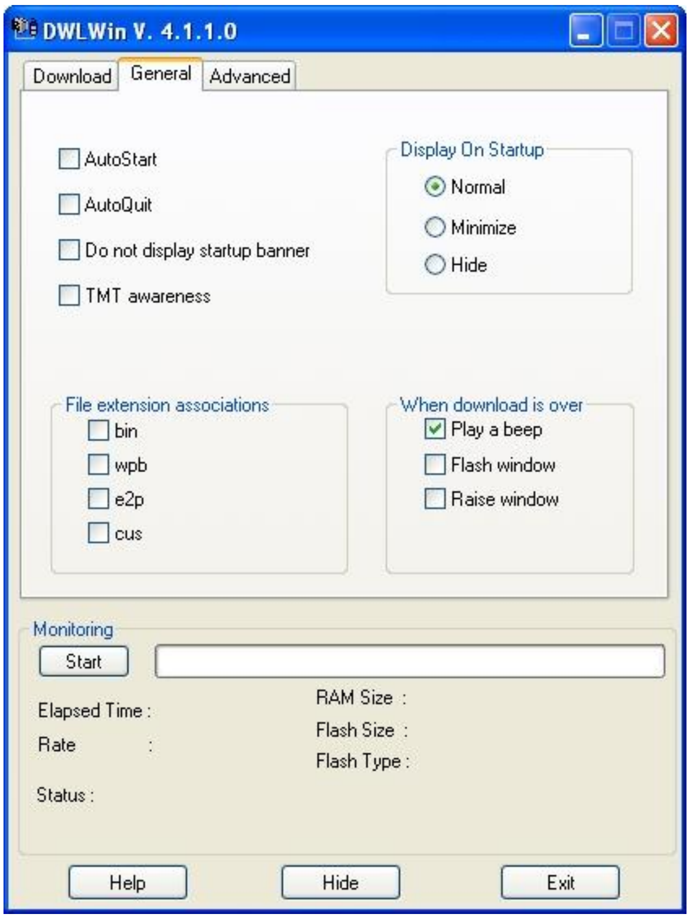
- The Working Directory text box at the top of the window, and the browsing button next to it, set the working directory.
- The three panels list the files that can be selected for download.
- The Serial Port Settings drop-down lists are used to select the serial port and set its rate.
- The Erase checkboxes allow to user to erase objects, customization files, and/or the Open AT application before the firmware is downloaded.
- The CPU Type drop-down list is used to select the embedded module family.

Note: *The proprietary protocols used to communicate with the baseband in boot mode are incompatible from a baseband type to another; the application has to know which protocol it has to use.*

In the Monitoring section, at the bottom part of the tab, a gauge provides information about the download progress and displays messages describing the running operations or to explain any reasons a download might fail.

2.2. General Tab

The majority of the settings in this tab are related chiefly to the GUI itself. See the figure below for what is displayed in the General tab of the DWLWin application.



AutoStart [ ]
AutoQuit [ ]
Do not display startup banner [ ]
TMT awareness [ ]

Display On Startup: [✓ Normal] Minimize Hide

File extension associations: bin [ ], wpb [ ], e2p [ ], cus [ ]

When download is over: Play a beep [✓], Flash window [ ], Raise window [ ]

Elapsed Time :
Rate :
Status :
RAM Size :
Flash Size :
Flash Type :

Figure 2. DWLWin General Tab

The two options that influence the download process within the General tab are "AutoStart" and "AutoQuit".

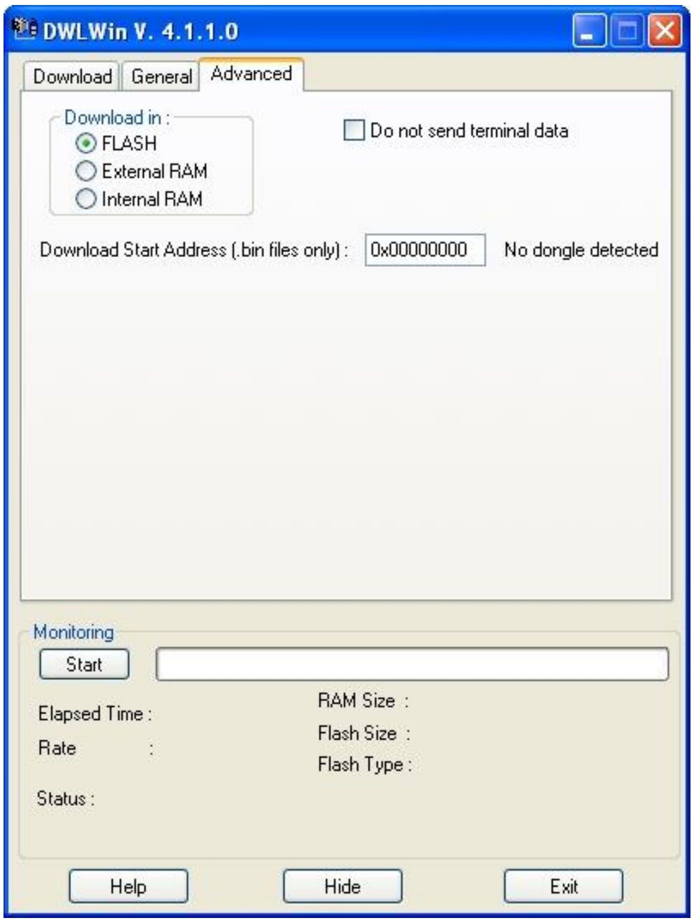
If "AutoStart" is checked, the download starts immediately when the application starts.

If "AutoQuit" is checked, the application quits immediately when the download ends.

Ensure than both options are unchecked unless you want DWLWin to behave in either or both of the ways described above.

2.3. Advanced Tab

The Advanced tab is displayed in the figure below.



Download in :
(● FLASH) External RAM / Internal RAM

☐ Do not send terminal data

Download Start Address (.bin files only) : 0x00000000

No dongle detected

Elapsed Time :

Rate :

Status :

RAM Size :

Flash Size :

Flash Type :

Figure 3. DWLWin Advanced Tab

Ensure that "Flash" radio button is selected in the "Download in" section of the window (in the top-left corner), and that "Download start address" is set to 0.

The main setting in this tab is the choice of the "Initlock file", a ciphered security plugin: note that this setting is useless if a download package is used to setup the firmware in flash memory.

3. Registering DWLWin as a COM Server

Before using DWLWin through its automation interface, you have to register it by running it with the proper command line parameter:

Action	Command
Register	`DWLWin.exe /regserver`
Unregister	`DWLWin.exe /unregserver`

You can browse for DWLWin.exe in the Start Menu/Run… command and then just append the "/regserver" switch before clicking OK.

3.1. Server Identification

User name	"dwlwin.application"
CLSID_DWLWin	`CCEAED01-8F4B-435c-9A5A-F160B183B524`
LIBID_DWLWin	`D3E21A56-9685-4d78-943C-89BB939A2568`
IID_DWLWin	`A25AE82C-5602-429C-9CBD-EC043A474520`

3.2. When Working with Windows Vista

Current versions of DWLWin do not display an error or notification when Vista denies DWLWin the right to modify the registry. DWLWin will display said notification in later versions.

For the current version, commands have to be run with administrative privileges, i.e. run from the Explorer context menu command "Run as Administrator…" One suggested solution is to create and use batch files in the DWLWin installation directory related to registering and unregistering the server, with the following contents:

```
dwlwin /regserver
```

and

```
dwlwin /unregserver
```

Then, when needed, run these batch files from the "Run as admin" context menu command to register/unregister the server.

4. Using DWLWin in Automatic Mode

4.1. Introduction

There are two ways to get an ActiveX automation object: through early binding (for languages and environments supporting it, e.g. C++) or through late binding (for scripting languages like Python). DWLWin supports both bindings. The following subsections present examples in C++ and Visual Basic, but any other language supporting ActiveX/COM interfacing can also be used.

4.2. Automating DWLWin Using C++

The recommended way to automate DWLWin is to use Visual C++ smart pointers (compilers other than Visual C++ v6.0 and higher are not supported).

Smart pointers ease the use of COM interfaces from a C++ application: a special preprocessing option must be added to the client code as shown below:

```
#import “dwlwin.exe”
```

Please refer to MSDN documentation for more information about COM smart pointers and their usage if necessary.

Note: *The DWLWin application must be in the preprocessor’s search path.*

COM interfaces require “OLE Strings” a.k.a. “Basic Strings” rather than plain zero-terminated C strings. Therefore, you have to perform conversions (ATL or MFC provide useful macros or classes to ease these conversions).

4.2.1. Sample Code

The following example (using ATL) downloads a firmware package into the embedded module connected to port COM3.

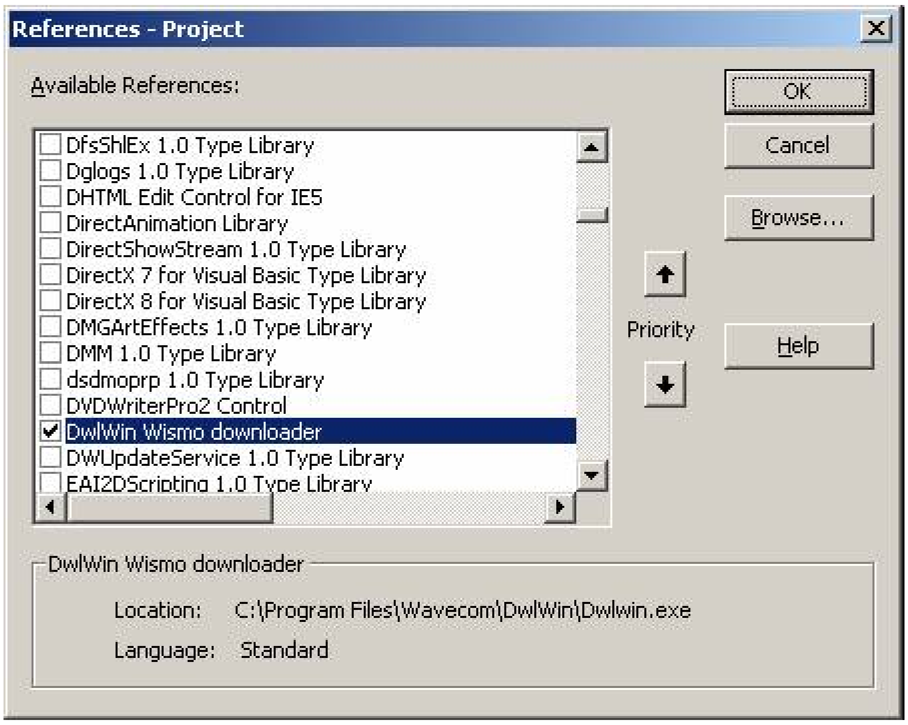
```
#include <objbase.h>
#include <stdio.h>
#import "dwlwin.exe"

int main(int argc, char **argv)
{
   dwlwin::IDwlwinPtr pDWLWin;

   CoInitialize(NULL);
   if (pDWLWin.CreateInstance("dwlwin.application") != NOERROR)
   {
       printf("Cannot launch DWLWin server\n");
       return -1;
   }

   // get the version of DWLWin
```

```
    _bstr_t bsVersion = pDWLWin->getVersion();
    printf("DWLWin Version = %s\n", (LPCTSTR)bsVersion);


    // setup the download options
    pDWLWin->setDownloadOptions(
        3,              // target is COM3
        115200,         // 115200 bps
        "c:\\temp\\wpk\\",      // working directory
        "binary.wpk|",          // download package file name (plus a pipe)
        "",             // no eeprom files
        "",             // no customization files
        1,              // do erase objects
        0,              // do not erase customization files
        0,              // do not erase the Open AT application
        0,              // do not erase the whole flash memory
        0,              // do not reset the firmware after download
        0,              // installation address = don't care
        0,              // autodetection
        0,              // no security plugin
        "");            // no path to security plugin

    // start the download on COM3
    pDWLWin->start(3);
    // Wait (download initialization delay)
    Sleep(500);

    // wait during the download
    bool bConnected = false;
    while (! pDWLWin->isDownloadOver(3))
    {
        ::Sleep(500);
        if (! bConnected && pDWLWin->isBootOk(3))
        {
            printf("DWLWin is connected\n");
            bConnected = true;
        }
    }

    // Display the exit message
    int nExitCode = pDWLWin->getErrorCode(3);
    _bstr_t bsMessage = pDWLWin->translateErrCode(nExitCode);
    printf("Exit message = %s (code %d)\n", (LPCTSTR)bsMessage, nExitCode);

    return 0;
}
```

4.3. Automating DWLWin using Visual Basic

4.3.1. Adding the Reference to the DWLWin COM Object

To use DWLWin as a COM object in a Visual Basic project, its reference has to be added through the “References…” command in the Projects menu as shown in the figure below.



- [ ] DfsShlEx 1.0 Type Library
- [ ] Dglogs 1.0 Type Library
- [ ] DHTML Edit Control for IE5
- [ ] DirectAnimation Library
- [ ] DirectShowStream 1.0 Type Library
- [ ] DirectX 7 for Visual Basic Type Library
- [ ] DirectX 8 for Visual Basic Type Library
- [ ] DMGArtEffects 1.0 Type Library
- [ ] DMM 1.0 Type Library
- [ ] dsdmoprp 1.0 Type Library
- [ ] DVDWriterPro2 Control
- [x] DwlWin Wismo downloader
- [ ] DWUpdateService 1.0 Type Library
- [ ] EAI2DScripting 1.0 Type Library

DwlWin Wismo downloader

Location:	C:\Program Files\Wavecom\DwlWin\Dwlwin.exe
Language:	Standard

Figure 4. Visual Basic References Window

Find the COM object named “DWLWin Wismo downloader” and check it in the list box.

Click OK to register the update and close the References window.

4.3.2. Using the Object from Visual Basic

The object can be created by the New operator using the command below:

```
Dim o As New dwlwin.dwlwin
```

The created object gets all the methods of the DWLWin COM interface.

Note: Visual Basic strings are “Basic Strings” and can readily be used through the COM interface.

4.3.3. Sample Code

The following example downloads a firmware package into the embedded module connected to COM3.

```
rem We need the Win32 Sleep primitive
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

rem Create the DWLWin COM object
Dim o As New dwlwin.dwlwin

Dim i As Integer
Dim exitCode as Integer
Dim exitMessage as String
Dim portNumber As Integer
Dim path As String
Dim package As String

path = "c:\temp\wpk\"
package = "R65.wpk|"
portNumber = 3

rem Configure the download process
With o
    Call .setDownloadOptions(portNumber, 115200, path, package, "", "", 0, 0, 0, 0, 0, 0, 0,
0, "")
    rem Configure the extra options to configure features and the IMEI
    rem Remember that options must be passed after setDownloadOptions
    Call .setStrExtOption(portNumber, "IMEI","11111111111111")
    Call .setStrExtOption(portNumber, "VECTOR_VALUE","xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx")
    rem Launch the download
    Call .start(portNumber)
    rem Wait the download initialization delay
    Sleep(500)
End With

While o.isDownloadOver(portNumber) = 0
    Sleep (100)
    rem ... Do whatever you need, e.g. use "getProgress" to animate a gauge…
Wend

rem Get the exit code and the exit message
exitCode = o.getErrorCode(portNumber)
exitMessage = o.translateErrCode(exitCode)
rem Display the message
rem ...
```

5. Programming Guidelines to Setup an AirPrime WMP1x0

Follow the guidelines below to set up an AirPrime WMP1x0.

Note: *For details regarding all commands listed in this section, please refer to section 6 Functions Reference for complete information about the commands.*

1. Create a DWLWin COM object.
2. Use “setDownloadOptions” to configure the download settings:
   - package paths
   - package filename
   - serial port
   - ensure that the CPU type parameter is set to FAMILY_Q2686 (do not use the autodetection in production code!)
3. Use “setStrExtOption” to configure the needed options variables (IMEI if the WMP1x0 has no IMEI inside, vector configuration value to setup features). Note that this step can be skipped if the IMEI has been programmed in Sierra Wireless factories and if no special additional feature is needed.
4. Launch the download with “start”: versions below 4.1.6 need a delay (about 500 ms) after the “start” function to initialize the download process.
5. DWLWin is able to simultaneously perform downloads on several serial ports.
6. You have to poll DWLWin with the “isDownloadOver” function to check the end of the download. Note that isBootOK returns 1 when DWLWin is successfully connected to its target (repeated failures without such a successful connection could be caused by serial link problems or RAM errors).
7. Meanwhile, functions like “getProgress” can be used to animate gauge bars: this progress percentage does not track the whole process but only the currently downloaded item (there can be many of such items during a full download process).
8. The download can be aborted at any time using the “stop” function.
9. Once “isDownloadOver” returns 1, check the error code and its translated human readable message: the error code is 0 when the process is successful; a non-null value means that an error has occurred.
10. The termination of the last COM client closes DWLWin.

6. Functions Reference

Presented in this section are the functions related to the DWLWin application.

show

Description	Shows the DWLWin main window.
Return	none
See also / Example	hide

Arguments	Type	IN/OUT	Description
None			

hide

Description	Hides the DWLWin main window.
Return	none
See also / Example	show

Arguments	Type	IN/OUT	Description
None			

terminate

Description	Terminates the application (kills DWLWin's process).
Return	none
See also / Example	

Arguments	Type	IN/OUT	Description
None			

getVersion

Description	Returns the DWLWin version number.
Return	DWLWin version number, in "x.y.z.t" format
See also / Example	

Arguments	Type	IN/OUT	Description
None			

setDownloadOptions

Description This function configures all the parameters of the download process.

Note The "setDownloadOptions" function requires "file lists" for some of its arguments; these are strings formed by concatenating all the file names with pipe "|" characters, and appending a final additional pipe.

For example, a single file "appli.wpb" turns into a file list "appli.wpb|"; two files named "file1.e2p" and "file2.e2p" turn into a list "file1.e2p|file2.e2p|".

Return None

See also / Example Configuring a download at 115,200 bps on COM2 onto an AirPrime WMP1x0:

```
DWLWin.setDownloadOptions(
  2,                        // target is COM2
  115200,                   // 115200 bps
  "c:\\wavecom\\FW6.5\\",   // working directory
  "firmware-6.5.wpk|",      // download package file name (plus a pipe)
  "",                       // no eeprom files
  "",                       // no customization files
  1,                        // do erase objects
  0,                        // do not erase customization files
  0,                        // do not erase the Open AT application
  0,                        // do not erase the whole flash memory
  0,                        // do not reset the firmware after download
  0,                        // installation address = don't care
  FAMILY_Q2686,             // required because target is a WMP1x0
  0,                        // no security plugin
  "");                      // no path to security plugin
```

Arguments	Type	Description
nPortNumber	integer	Number of the serial port (COMx) to configure
nSpeed	integer	Serial rate (115200, 230400, 460800 or 921600)
strDirectory	string	Working directory
strBinaries	string	File list (see above) of binary files to download
strEeproms	string	File list (see above) of e2p files to download

Arguments	Type	Description
strCusfiles	string	File list (see above) of cus files to download
bEraseObj	integer	Boolean (0 or 1), to erase or not the objects area
bEraseCus	integer	Boolean (0 or 1), to erase or not the customization files
bEraseOat	integer	Boolean (0 or 1), to erase or not the Open AT application
bEraseAll	integer	WARNING!! Always set to 0, otherwise erases the whole flash memory
bReset	integer	Boolean (0 or 1) to start the code in flash memory after the end of the download process
uDownloadAddress	integer	Installation address of the downloaded .bin file (better off left to 0)
uCpuType	integer	Integer code for the baseband family FAMILY_AUTODETECTION 0 SHOULD NOT BE USED IN PRODUCTION MODE FAMILY_Q2406 1 For Q2406 and Q2403 ranges of embedded modules FAMILY_Q24NG 4 For Q24NG and Q2501 ranges of embedded modules FAMILY_Q2686 7 For Q2686, Q2687 ranges of embedded modules and for WMP1x0
bLockMode	integer	Boolean (0 or 1) to use the initlock security plugin
strInitLock	string	Fully qualified path to initlock security plugin

start

Description	Starts the download previously configured on a serial port.
Return	None
See also / Example	See also stop, setDownloadOptions

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to use

stop

Description	Forcibly stops the download running on a serial port.
Return	None
See also / Example	See also start, setDownloadOptions

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to use

isBootOk

Description	Checks if the host and the remote target on a given serial port are connected. When DWLWin tries to connect to the embedded module in boot mode, it sends connection requests and then installs a special downloader in the embedded module memory; the host and the target are considered connected if this installation stage has succeeded.
Return	Integer value: Boolean (0 or 1); returns 0 if the host and the target are not connected
See also / Example	

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to query

isDownloadOver

Description	Checks if the download on a given serial port is over.
Return	Integer value: Boolean (0 or 1), returns 0 if a download is still running on this port.
See also / Example	

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to query

getFlashName

Description	Gets the name of the flash memory device of the embedded module connected to a port.
Note	This function is not available until the device has completed its boot sequence, i.e. when the function isBootOk returns 1.
Return	String value: name of the flash device
See also / Example	See also getFlashId, getFlashSize

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to query

getFlashId

Description	Gets the manufacturer and device codes of the flash memory device of the embedded module connected to a port.
Note	This function is not available until the device has completed its boot sequence, i.e. when the function isBootOk returns 1.
Return	Integer value. The most significant 16-bit word is the manufacturer code, and the least significant 16-bit word the device code.
See also / Example	See also getFlashName, getFlashSize

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to query

getFlashSize	
Description	Gets the size of the flash memory device of the embedded module connected to a port.
Note	This function is not available until the device has completed its boot sequence, i.e. when the function isBootOk returns 1.
Return	Integer value: size (in bytes) of the flash device
See also / Example	See also getFlashName, getFlashId, getRamSize

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to query

getRamSize	
Description	Gets the size of the RAM device of the embedded module connected to a port.
Note	This function is not available until the device has completed its boot sequence, i.e. when the function isBootOk returns 1.
Return	Integer value: size (in bytes) of the device's RAM
See also / Example	See also getFlashSize

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to query

getProgress	
Description	Gets a progress percentage for the operation running on a serial port.
Note	This is not an overall percentage, but just a progress indicator for the currently downloaded file or executed command.
Return	Integer value: progress percentage between 0 and 100
See also / Example	

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to query

getErrorCode

Description	Gets the error code returned by the download process which has ended on a serial port. Error codes can be translated into a human readable message by the function "translateErrCode".
Return	Integer value: error code, 0 meaning success, any other value means that an error occurred.
See also / Example	See also translateErrCode, getErrorMsg

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to query

getErrorMsg

Description	Gets the error message reported at the end of a download process on a serial port.
Return	String value: error message
See also / Example	See also getErrorCode

Arguments	Type	IN/OUT	Description
nPortNumber	integer		Number of the serial port (COMx) to query

translateErrCode

Description	Translate a download process return code into a readable error message.
Return	String value: error message associated to the given integer code
See also / Example	See also getErrorCode

Arguments	Type	IN/OUT	Description
nErrorCode	integer		Error code to translate

makePersistent	
Description	Renders DWLWin COM server persistent so that it does not terminate when the last COM client disconnects.
Return	None
See also / Example	

Arguments	Type	IN/OUT	Description
None			

6.1. Options-Related Functions

Each running download process keeps an internal pool of string and integer variables called “options”. These options are used internally as a parameter database and serve as an interface between the OLE client and DWLWin to tune the operations performed by a download package.

The download package’s interface therefore specifies options that can be set to modify said package’s behavior, perform additional operations, or configure settings before starting the download.

The functions presented below are those related to setting said options.

setIntExtOption	
Description	Creates an integer-typed option.
Note	This function must be used after setDownloadOptions, because setDownloadOptions resets all the option variables.
Return	None
See also	See also setStrExtOption

Example	Create on COM2 an integer option named “NO_DOWNLOAD” set to 1.
	`DWLWin.setIntExtOption(`
	`   2,`
	`   “NO_DOWNLOAD”,`
	`   1);`

Arguments	Type	Description
nPortNumber	integer	Number of the serial port (COMx) to use
strName	string	Name of the option variable
nValue	integer	Value to set

setStrExtOption

Description	Creates a string-typed option.
Note	This function must be used after setDownloadOptions, because setDownloadOptions resets all the option variables.
Return	None
See also	See also setIntExtOption
Example	Create on COM1 a string option named “VECTOR_VALUE” set to “6BFF3”.

```
DWLWin.setStrExtOption(
   1,
   “VECTOR_VALUE”,
   “6BFF3”);
```

Arguments	**Type**	**Description**
nPortNumber	integer	Number of the serial port (COMx) to use
strName	string	Name of the option variable
strValue	string	Value to set

SIERRA
WIRELESS