TOOLS MANUAL

wavecom

TOOLS MANUAL

Version: 001 / 1.0
Date: October 30th, 2001
Reference: WM_TOO_OAT_UGD_001

wavecom

Contents

List of figures

1 Introduction

1.1 Purpose

This document is the user's guide for the Open AT Development Toolkit.

1.2 References

I. Development Guide
II. Getting Started
III. AT Command Interface

1.3 Glossary

AT commands.....Set of standard modem commands.
Embedded application.....User application sources to be compiled and run on a Wavecom GSM product.
Embedded Core software ....Software that includes the Embedded application and the Wavecom library.
Embedded software.....User application binary: set of Embedded application sources + Wavecom library.
External application .....Application external to the Wavecom product that sends AT commands through the serial link.
Target .....Open AT compatible product supporting an Embedded Application.
Target Monitoring Tool.....Set of utilities used to monitor a Wavecom product.
Remote Application .....Set of libraries with which the User application can be run on a PC.
Wavecom library .....Library delivered by Wavecom to interface Embedded application sources with Wavecom Core Software functions.
Wavecom Core Software .....Set of GSM and open functions supplied to the User.

1.4 Abbreviations

API .....Application Programming Interface
CPU.....Central Processing Unit
FCM .....Flow Control Manager
IR.....Infrared
KB .....Kilobyte
OS.....Operating System
PDU .....Protocol Data Unit
RAM.....Random-Access Memory
ROM .....Read-Only Memory
RTK .....Real-Time Kernel
SMA.....SMall Adapter
SMS.....Short Message Services
SDK.....Software Development Kit

2 Use

2.1 Open AT wizard directories architecture

The typical directory tree structure of an application is shown below. This structure is generated by the Open AT wizard.

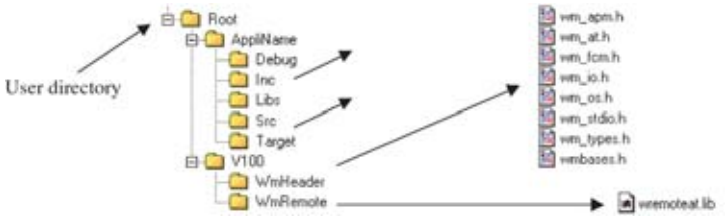

Figure 1: Open AT Wizard Directory Architecture

The **AppliName** directory is specific to each new application. It is made up of the following sub-directories:

- ❑ **Debug :** contains the Remote Application binary, generated by the Wizard,
- ❑ **Target :** contains the Embedded Core Software ready to be downloaded to the Target,
- ❑ **Src :** contains the User Open AT sources,
- ❑ **Inc :** contains the User Open AT headers,
- ❑ **Libs :** contains the Windows libraries related to one part of the target code sources (for Remote Application Execution).

The **V100** directory contains the Wavecom supplied softwares corresponding to this version of Open AT. It includes the following sub-directories:

- ❑ **WmHeader :** contains the Open AT API, header files,
- ❑ **WmRemote :** contains the Open AT library for the Remote Task Environment.

2.2 Open AT SDK Directory Architecture

After the Open AT SDK installation, the following structure is generated :

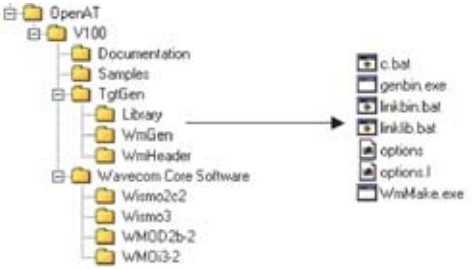

Figure 2 - Open AT SDK Directory Architecture

The current version of Open AT is compatible with the following compiler suite:
ARM suite SDT 2.51.
See the ARM compiler installation procedure in the Getting Started manual (Reference II).
IMPORTANT: Wavecom does not guarantee proper operation with other compiler/linker versions.

The **"C:\OpenAt\V100\TgtGen\WmGen"** directory must contain the minimum set of Wavecom utilities needed by the compiler/linker: ('C:\OpenAT' is an example of Open AT SDK installation directory).

- ❑ **C.BAT**, used to compile C files.
  Syntax : c <name of the file without the ".c" extension>.
- ❑ **LINKLIB.BAT**, used to generate libraries.
  Syntax : linklib <name of the descriptor file without the ".lnk" extension>.
- ❑ **LINKBIN.BAT**, used to generate a binary file and convert it to an Xmodem format file, ready to be downloaded.
  Syntax : linkbin <name of the descriptor file without the ".lnk" extension>.
- ❑ **GENBIN.EXE**, required for linkbin.bat.
- ❑ **OPTIONS**, for compiler option file required for c.bat.
- ❑ **OPTIONS.L**, for linker option file required for linkbin.bat.
- ❑ **WMMAKE.EXE**, used to compile and link a library or the application binary.
  Syntax : WmMake <name of the makefile>.

2.3 Target Binary File Generation

2.3.1 Introduction

All embedded application source files are copied into a directory named '**src**' ; the header files should be copied into a directory named '**inc**' (see figure 3).

Figure 3 : Example of Embedded Application Directory Architecture

2.3.2 Compilation

The compilation of an embedded application source file is performed with the **C.BAT** utility.
For the example, use: "**c appli**" and "**c utils**."
First, the **C.BAT** utility will create the '**Target**' directory if it does not exist.
Then the files appli.obj and utils.obj are generated in the '**Target**' directory.
NOTE: *The "**OPTIONS**" file of "C:\OpenAt\V100\TgtGen\WmGen" directory is also copied into this directory (see Figure 4). The user should add pre-processor flag definitions in this file, using the "**-D__XXX__**" format ("**__XXX__**" is then the defined pre-processor flag).*

Figure 4: Example of a Compilation Result

NOTE: *Compilation log files are generated ('.LST' extension) in the '**src**' directory.*
NOTE: *the "**c**" utility can be called in either the "**src**", "**inc**" or the "**target**" directory. After the "**c**" utility has run, the current path is set to the "**src**" directory.*

2.3.3 Library Link

This phase consists of generating a custom library.
The user has to create a LNK file (e.g. 'Mylib.lnk') in the '**target**' directory, containing the object file list to link in this library.
FOR EXAMPLE : *content of Mylib.lnk :*

appli1.obj

To generate the library, use the LINKLIB.BAT utility.

FOR EXAMPLE : *"**linklib Mylib**"*
LINKLIB.BAT utility will generate the "**Mylib.lib**" file in the '**target**' directory.
NOTE: *the "linklib" utility can be called in either the "**src**", "**inc**" or the "**target**" directory. After the "**linklib**" utility has run, the current path is set to the "**target**" directory.*
NOTE: *the "**WmMake**" utility automaticaly creates the LNK descriptor file.*

2.3.4 Binary File Generation

This phase consists of generating the binary file that will be downloaded into the Wavecom product.
The user has to create a LNK file (e.g. 'MyProject.lnk') in the '**target**' directory, which is typically based on the following files :

- ❑ appli.obj,
- ❑ wmopenat.lib.

For instance, if the user needs to link the "**mylib.lib**" library,
the "**MyProject.lnk**" will contains the following files:

- ❑ appli.obj,
- ❑ wmopenat.lib.
- ❑ mylib.lib.

Then, the binary file is generated with the **LINKBIN.BAT** utility.

FOR EXAMPLE : "linkbin MyProject"
The **LINKBIN.BAT** utility generates the "**MyProject.dwl**" file, ready to be downloaded.
NOTE: *the "**linkbin**" utility can be called either in the "**src**", "**inc**" or the "**target**" directory. After the "**linkbin**" utility has run, the current path is set to the "**target**" directory.*
NOTE: *the "**WmMake**" utility automaticaly creates the LNK descriptor file.*

2.3.5 The "WmMake" utility

This utility is used to compile and link the embedded application source files and libraries.
This program need a makefile (with the "**.mak**" extension), copied into the application root directory (is this sample, the "**MyProject**" directory).

The makefile syntax is given bellow :

TGT=BIN *(to generate the application binary)*
or
TGT=LIB *(to link a customer library)*
appli.c
utils.c *(source file list to compile for this binary/library, with or without the ".c" extension)*
mylib.lib *(library file list to link for the application binary ; **use only with the TGT=BIN option**)*

If the makefile is named "myproject.mak", the "WmMake" utility is used with the command : "**WmMake myproject**".
When the "TGT" value is set to "BIN", WmMake call the "c.bat" utility for each C file of the makefile, creates the ".lnk" descriptor file in the target directory, and call the "linkbin.bat" utility to link the application binary.
When the "TGT" value is set to "LIB", WmMake call the "c.bat" utility for each C file of the makefile, creates the ".lnk" descriptor file in the target directory, and call the "linklib.bat" utility to link the customer library.

NOTE: *the "WmMake" utility may be called either in the root "MyProject" directory, or in each of the project sub-directories.*

NOTE: *for the binary generation, the utility does not care if the "wmopenat.lib" library is included or not in the makefile, and always include it in the ".lnk" descriptor file.*

2.4 Development Toolkit

2.4.1 Overview

The Development Toolkit is a set of 4 tools running under Windows, designed and supplied by Wavecom.
The Development Toolkit environment is presented in Figure 5.

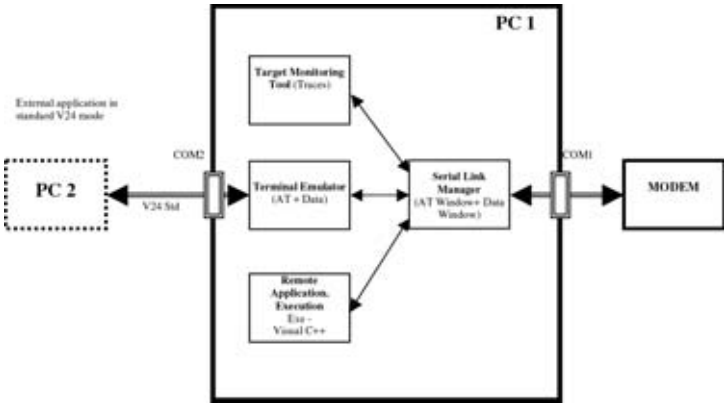

Figure 5: The Development Toolkit Environment

These tools are described here-below.

2.4.2 Serial Link Manager

2.4.2.1 Description

The **Serial Link Manager** uses the PC serial link (COM1) to communicate with Wavecom products. It acts like a server between the different applications.
The other serial port (COM2) can be used to connect another PC, using the serial link in nominal mode (see Figure 5).

2.4.2.2 Getting Started

The Serial Link Manager is automatically launched as soon as the **Target Monitoring Tool**, the **Terminal Emulator** or the **Remote Application** starts: the serial port is opened and the Serial Link Manager icon is displayed on the task bar. Double-clicking on this icon will launch the Serial Link Manager dialog box.
With this Dialog Box, the user can manage the Serial Port (with the "**CommPort**" menu: open/close the port, set the port configuration).
It can be closed with the "**Window**" -> "**Hide Tray**" command. Refer to the Target Monitoring Tool help for more information on the Serial Link Manager.

NOTE: *the Serial Port configuration used in the Serial Link Manager will be also used in the other programs of the Development ToolKit.*

2.4.3 Target Monitoring Tool

2.4.3.1 Description

The **Target Monitoring Tool** is a set of utilities used to monitor a Wavecom product. With Open AT, it is used to display the Embedded application trace messages (from the Target or the Remote Application). The **Target Monitoring Tool** is used together with the Serial Link Manager.

2.4.3.2 Getting Started

In order to use the **Target Monitoring Tool**, please follow these steps:

❶ Download an Embedded Core Software to the target, using one of the samples provided,

❷ Make sure the Com1 port is not used by another application. Otherwise, close this application,

❸ Start the **Target Monitoring Tool** application from the Windows Start menu,

❹ Check the PC serial link settings:
Click "**Settings**" in the "**Preferences**" menu,
Select the "**Com**" tab,
Check Baud rate: 9600
Data bits: 8
Parity: none
Stop bits: 1
This is the default mode.
NOTE: *the serial port settings can be changed using either the Target Monitoring Tool or the Serial Link Manager.*
NOTE: *the serial port speed can be detected by the "Commands" -> "Auto Detect" command of the Target Monitoring Tool.*

❺ Check to see whether the communication with the Target is OK:
Click the "**Terminal Emulator**" button, and select the "**AT1**" window.
Type **AT** and press **ENTER**:
A blue **OK** response must appear. Otherwise, select "**Init Target**" from the "**Commands**" menu of the **Target Monitoring Tool** and type **AT** once again,

❻ Close the **Terminal Emulator** window.

2.4.3.3 Get Traces Sent by the Wavecom Target

❶ Select "**Open**" from the "**Traces**" menu,

❷ Select "**Get Information about Target**" from the "**Commands**" menu,

❸ Select "**Set and Request to the Target…**" from the "**Commands**" menu,
Select **CUS** from the **Parameter** list,
Click on all levels from 0 to 31 in **Bitmap**,
Click on the "**Send Level**" button,
Click on the "**Close**" button,

❹ The Trace window then displays the traces sent by the Wavecom product.

NOTE: *The traces are the strings sent by the embedded application to the Target Monitoring Tool through the "**wm_osDebugTrace()**" function. This function takes the string display level, and the string to be displayed, as its parameters. See the Development Guide for more information.*

NOTE: *With the 9600 bps serial link speed, some traces may not appear. To avoid this, the user should use the 115200 bps serial link speed (see the "**AT+IPR**" command in the AT Interface document).*

NOTE: *In the case of an embedded application using the FCM API, if the Target Monitoring Tool has to display many traces while the Terminal Emulator has to display many data blocks, some display problems may occur. When sending many data on an IO flow, the user should not display too many traces in the Target Monitoring Tool.*

2.4.3.4 Select the Current Embedded Application Symbol File

When an embedded application binary is generated, a symbol file "**s**" is also generated. The Target Monitoring Tool can use this file to display the function's names when the software crashes and leads to a BackTrace. (See the Development Guide for more information on software security.)

❶ Use the "**Preferences**" -> "**Settings...**" menu, or press the F10 key.

❷ Use the "**Browse**" button to select the "**target**" directory of the current downloaded embedded application.

❸ Check to be sure that the "**s**" file is selected in the "**Symbol File**" text field.

❹ Always close and restart the Target Monitoring Tool after a symbol file modification.

2.4.4 Terminal Emulator

2.4.4.1 Description

The **Terminal Emulator** is an AT command terminal. It is used by the User to send/receive string commands to/from a customer task (on the Target or in Remote mode). It can use either the serial link Nominal mode (the AT window is then open) or the Wavecom Debug mode.
The **Terminal Emulator** connects an External application in standard V24 mode. The **Terminal Emulator** transforms a data flow from standard mode to debug mode. An External application can also get information in nominal mode.

2.4.4.2 Getting Started

If the **Target Monitoring Tool** is running, start the **Terminal Emulator** using the corresponding button in the toolbar (check "**Toolbar**" in "**View**" menu), or "**AT cmd terminal**" from the "**Tools**" menu.
You can also launch the **Terminal Emulator** using the shortcut from the Windows Start menu.
The Terminal Emulator provides two windows: one for AT commands and one for the V24 Data flow. Refer to the Terminal Emulator "**help**" menu for more information.

***NOTE**: in case of an embedded application using the FCM API (cf. the Development Guide), the target should be initialized in Debug Mode ("**Commands**" -> "**Init Target**") before any FCM operation; otherwise the Terminal Emulator and the Target Monitoring Tool will not run properly. Moreover, if the external serial port (COM2) is used, the "**External Com**" -> "**Open...**" command can be used before any FCM operation.*

***NOTE**: if there is a data call, or if the serial link is switched into DATA mode with the FCM API, the "+++" sequence does not work in the Terminal Emulator.*

2.4.5 Remote Application Execution

This tool is a set of libraries with which the User application can be executed on a PC by means of Visual C++® while communicating with the target through a serial link.
This tool is detailed in the next paragraph.

2.5 Remote Application Execution Tool

2.5.1 Basic Concepts

An Embedded application can be executed from a PC, while communicating with the rest of the Target software through a serial link.
An Embedded application is implemented in a specific task (customer task). This task can be executed from a PC, provided a Windows version of RTK used on the target is installed. In this case, the customer task is deactivated on the target.
This concept is described in Figure 6.

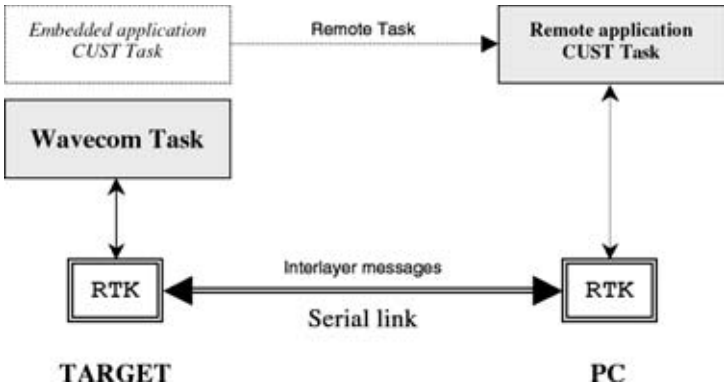

Figure 6: Project Architecture

Without changing the code, an Embedded Application becomes a Remote Application. The Remote process (development, compilation, link, execution) operates in a Visual C++® environment.

2.5.2 Main Environment Features

The list of environment features needed to implement the Remote Application Execution Tool is given below:

- ❑ Visual C++® in Debug mode: breakpoints and step-by-step execution,
- ❑ Wavecom Real Time Kernel (RTK) for Windows, with interrupts supported,
- ❑ Data Flash Objects emulation in local mode (files),
- ❑ Data Flash Objects synchronization between the target and the PC,
- ❑ Target Monitoring Tool,
- ❑ Serial Link Manager,
- ❑ Project creation Wizard.

2.5.3 Project Creation

2.5.3.1 Project Components

A Remote Application environment is actually a Visual C++® project. This environment is set up using DevStudio through a wizard.
Moreover, additional libraries are used to run the Remote Application from the PC. The different components needed to build the Remote Application are described in Figure 7.

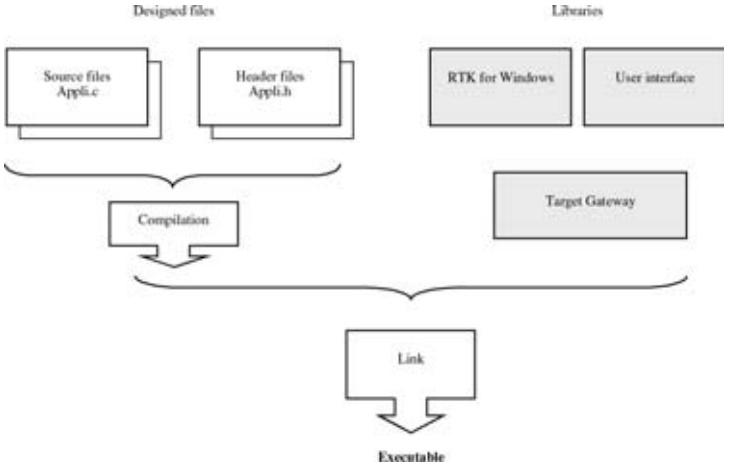

Figure 7: Remote Application Components

2.5.3.2 Project Creation Wizard

This Wizard provides a user-friendly way to create an Open AT project. It is described in the Tutorial document.

2.5.4 Remote Application Compilation

2.5.4.1 Target/Windows Code Compatibility

All the header files and sample codes supplied by Wavecom are fully compatible.
The Target compiler and Visual C++® should use the same files for compilation.

2.5.4.2 Compilation Launch

Before launching the compilation, it is important to check that Visual C++® is in Win32 Debug, instead of Win32 Release, in order to have access to all debug capabilities.
Visual C++® compiler is then called using the following command :

Build -> Build xxx.exe (F7)

where xxx is the name chosen when the project was created.

NOTE: *some "warnings" may occur on the final application link ; they will not cause any trouble in the remote application execution.*

2.5.5 Remote Application Execution

2.5.5.1 Remote Application Launch

2.5.5.1.1 User Interface Launch

To launch the user interface, from the Visual C++® menu, select the following command:

build -> Start Debug -> Go (F5)

The Visual C++® environment then starts in Debug mode, and launches the **Remote Application Execution Tool** (see Figure 8).

Figure 8: Remote Application Controller

2.5.5.1.2 Data Flash Object Synchronization

Data Flash Objects are entities in which the User application can save data such as addresses, phone numbers, etc. An object is referred to by a 32-bit hexadecimal identifier (this is NOT a physical address).

In remote application, these Data Flash objects are emulated through physical files.

The Data Flash Objects directory has the following form:

C:\…\[project path]\objects

The Data Flash object structure (file) is given below:

u32	object ident
u16	object size
u8	data 0
u8	data 1
..	. . .
u8	data (size-1)

In order to have the same Data Flash object configuration as on the Target, the user has to launch a read synchronization before starting the Remote Application.

NOTE: *Because of both the size and number of Data Flash objects present on the Target, read synchronization may take up to one or two minutes.*

2.5.5.1.3 Trace Configuration

Embedded application trace messages are displayed using the **Target MonitoringTool**, by means of a trace window opened from the **Traces** menu.
The Target application traces then appear **in black** and the Remote Application traces appear **in blue**.

Each trace instruction corresponds to a trace level. The user has to select this level using the Remote Application Controller (see Figure 9) before using the **Target Monitoring Tool** to display the traces.

Figure 9: Trace Level Selection

Trace levels may be changed at any time. They are then dynamically taken into account.

2.5.5.1.4 Serial Port Configuration

Using the **Serial Link Manager**, the user must check that the serial port speed is set to the same value as the Target port speed.

The **Auto Detect** command from the **Target Monitoring Tool** can be used to retrieve this speed. In order to get better results, the speed should be set to **115200 bauds**.
When connecting the target to the computer through the serial link, the connection may not work, because the Target and the computer serial interfaces may use different baud rates.

Selecting **Auto Detect** will then make the **Target Monitoring Tool** send frames with different successive baud rates. If the target sends a frame back, it means the target speed and the computer speed match. The **Target Monitoring Tool** then stops scanning.
An **Auto Detect** sequence typically lasts five to ten seconds. The **Target Monitoring Tool** may seem slower during this time.

The Terminal Emulator must be launched before starting the remote application, to check whether the communication with the target is correct or not.

2.5.5.1.5 RTK Launching

The remote process is started as soon as the **Start** button is clicked from the Remote Application Controller (see Figure 8).
The Initialization sequence is described below:

❶ Initialization of the serial link (in Serial Link Manager mode),

❷ Target configuration,

❸ Target RTK Re-initialization,

❹ Remote Application Execution creation and activation.

❺ The **Target Monitoring Tool** should then display the selected trace levels, showing that the application is running correctly.

2.5.6 Notes

During step-by-step execution, trace messages are not displayed at once by means of the **Target Monitoring Tool**. They are only displayed after an RTK scheduling.

WAVECOM S.A. - 12, boulevard Garibaldi - 92442 Issy-les-Moulineaux Cedex - France - Tel: +33 (0)1 46 29 08 00 - Fax: +33 (0)1 46 29 08 08
WAVECOM, Inc. - 4810 Eastgate Mall - Second floor - San Diego, CA 92121 - USA - Tel: +1 858 362 0101 - Fax: +1 858 558 5485
WAVECOM Asia Pacific Ltd - 5/F Shui On Centre - 6/8 Harbour Road - Hong Kong, PRC - Tel: +852 2824 0254 - Fax: +852 2824 0255