Open AT® GTi User Manual

Revision: 004
Date: November 2006

Make it wireless

Operating Systems | Integrated Development Environments | Plug-Ins | Wireless CPUs | Services

Open AT® GTi User Manual

Reference: WA_DEV_GTI_UGD_001

Revision: 004

Date: November 21, 2006

Trademarks

Copyright

Document History

Revision	Date	List of revisions	Writer
001	March 31, 2006	Initial version of Open AT® GTi User Manual	RAU
002	September 25, 2006	Updated version for OS 6.61	RAU
003	October 19, 2006	Addition of Q2686	RAU
004	November 21, 2006	Addition of New Q24 Series	RAU

Table of Contents

List of Figures

1 Introduction

1.1 Overview

Wavecom is a worldwide leader in pre-packaged wireless communication solutions for automotive, industrial and mobile professional applications. Wavecom's solutions include all the software and hardware elements that are necessary to develop innovative wireless devices, as well as tools and necessary associated services needed to bring them to market quickly and easily.

One such solution is the Wavecom Open AT® Software framework. The Open AT® Software is at the heart of a growing number of wireless products in the automotive, M2M and mobile professional market segments. The Open AT® software development kit provided by Wavecom helps developers shorten time to market by reducing hardware design, software development, and test and validation cycles. Eliminating extra components and using existing software building blocks helps reduce development costs and shrink hardware footprints. Furthermore, the Open AT® Application Development Layer provides simple, ready-to-use templates for managing communications, messaging, hardware interfaces and more.

In order to considerably reduce the time to market for customers, Wavecom has further provided an evolution of the existing Open AT® Software solution. The solution is referred to as an Open AT® GTi. The Open AT® GTi is an innovative development platform that enables developers of wireless devices to create customized, display-centric products. Differentiating products quickly and at the same time keeping costs down are the key success factors for our customers. Wavecom has built Open AT® GTi on four cornerstone elements:

- RapidPLUS™ a proven, user-friendly development environment from Wavecom's partner e-SIM Ltd., allowing for quick application building and customization, computer testing and automatic code generation;
- An industrial reference application to get started quickly;
- A wide range of Wavecom's Wireless CPUs to choose from;
- Professional training services and technical support to ease development.

By employing e-SIM's market-proven Man-Machine Interface (MMI) technology, already widely used in the mobile handset and automotive markets, development and customization of Open AT® GTi-based products becomes easy and fast, allowing maximum flexibility at minimum risk.

In brief, Open AT® GTi simplifies a very complex process and unlocks the real potential of creative companies who want to quickly differentiate their products by providing display layout flexibility and allowing for other graphic elements such as color images to be incorporated.

1.2 Objectives

The objectives of this document are to provide:

- An overview of the Wavecom Open AT® Software Architecture
- An overview of the e-SIM MMI development tools used for MMI customization
- An understanding of how the Wavecom Sample MMI has been integrated to Wavecom Open AT® Software to create the Wavecom Open AT® GTi Architecture
- Help on the customization of the Open AT® GTi MMI using e-SIM MMI customization tools
- Development of a sample application using e-SIM and Wavecom development tools.

1.3 Software Installation

1.3.1 System Requirements

To work on the Open AT® GTi framework, your system must support one of the following operating systems:

- WINDOWS 98
- WINDOWS NT
- WINDOWS Me
- WINDOWS 2000
- WINDOWS XP

Note	Even though the GCC compiler is provided with the Open AT® Software installation, it is not used with the Open AT ® GTi.

1.3.2 Hardware Requirements

The main requirements for the PC supporting the Open AT® Software IDE are:

- Pentium 300MHz (or higher) processor with at least 128 MB of RAM and 500 MB free hard disk space.
- A CD ROM drive.
- At least one free COM (serial) port for the communication with the target product.

Currently, the Open AT® GTi application is supported only on Wavecom new Q24 series, Q2686H and Q2687H Wireless CPUs.

1.3.3 Software Requirements

The Wavecom Open AT® GTi Architecture requires installation of the following software for application development on the PC:

i. Open AT® Software IDE

ii. e-SIM RapidPLUS™

The Wavecom Open AT® GTi requires the following software on the Wavecom CPUs to run the applications:

i. For New Q24 Series
   a. Open AT® Software V3.12
   b. Wavecom OS 6.57

ii. For Q2686H
   a. Open AT® Software V4.10
   b. Wavecom OS 6.61

iii. For Q2687H
   a. Open AT® Software V4.10
   b. Wavecom OS 6.61

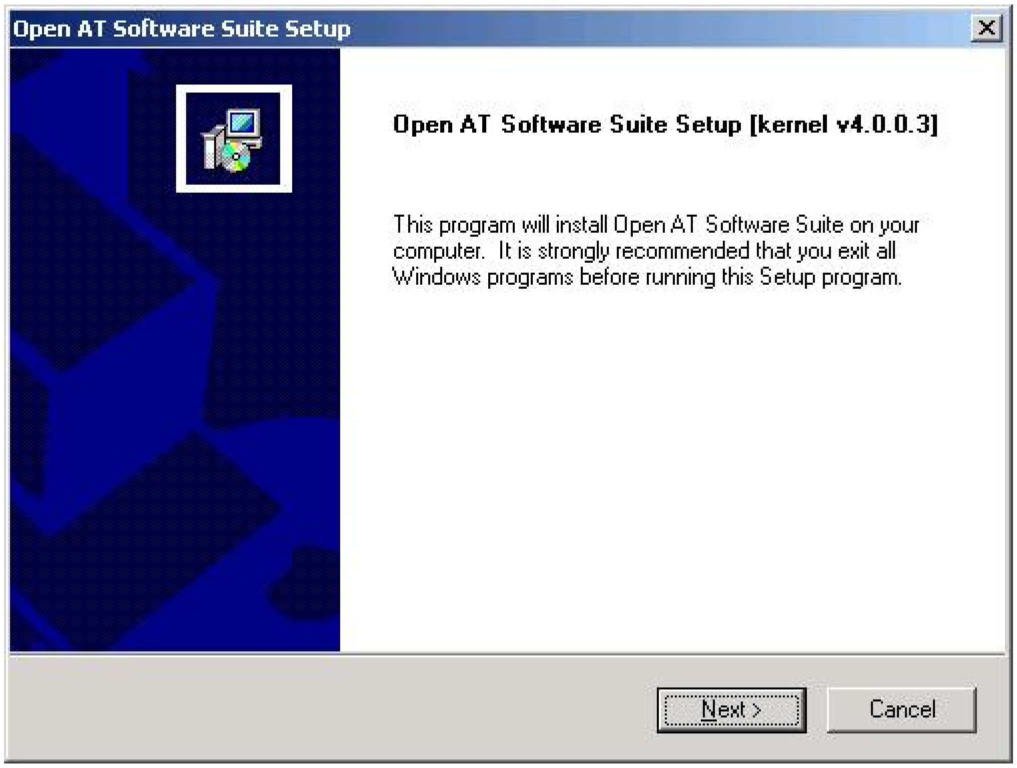
1.3.3.1 Installation of the Open AT® Software IDE

The steps listed below will guide you through the installation of the Open AT® Software IDE. Refer to the screenshots for references made to specific field labels.

i. Insert the Open AT® Software IDE CD-ROM in the CD drive on your PC. If the Autostart option is enabled on your PC, the installation starts automatically.

ii. If the Autostart option is disabled, and if automatic installation does not start, you can launch the executable file directly from the root folder of the CD-ROM using Windows Explorer. The path for this executable file is: \autorun\autorun.exe

iii. The Open AT® Software Setup screen is displayed. Click **Next** to continue.



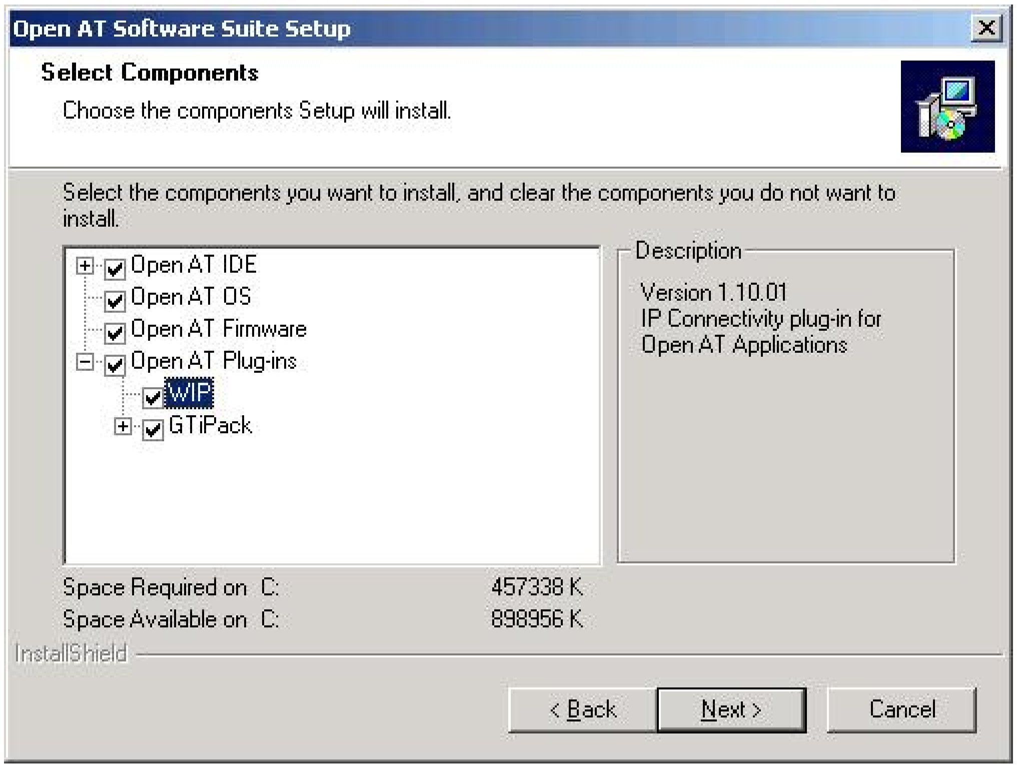
Open AT Software Suite Setup [kernel v4.0.0.3]

This program will install Open AT Software Suite on your computer. It is strongly recommended that you exit all Windows programs before running this Setup program.

Figure 1: Open AT® Software Installation – Startup Screen

iv. The following components can be installed either together or separately:

- Open AT® Software development environment, with libraries, header files, and documentation in PDF format
- Open AT® Software sample applications, with source and binary files with ADS,GCC or a Visual C++ compiler
- Wavecom software generation toolkit, used to build Open AT® Software applications in the target mode
- Cygwin(optional) - this tool is a Linux environment running under Windows
- Select the required components from the Setup options and click **Next** to continue



Open AT Software Suite Setup

Select Components

Choose the components Setup will install.

Select the components you want to install, and clear the components you do not want to install.

- ✓ Open AT IDE
- ✓ Open AT OS
- ✓ Open AT Firmware
- ✓ Open AT Plug-ins
  - ✓ WIP
  - ✓ GTiPack

Description

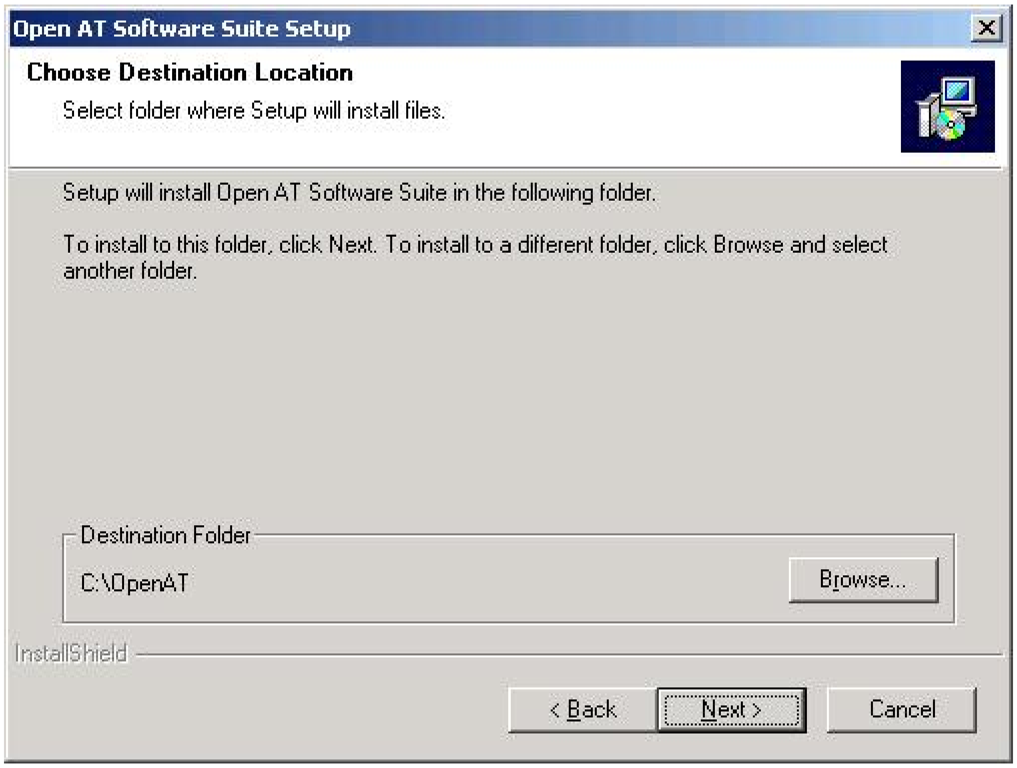
Version 1.10.01
IP Connectivity plug-in for Open AT Applications

Space Required on C: 457338 K
Space Available on C: 898956 K

Figure 2: Open AT® Software Installation – Setup Options

v. The Open AT® Software IDE can be installed in any directory on the hard disk.

vi. By default, files are installed on the C drive.



Choose Destination Location

Select folder where Setup will install files.

Setup will install Open AT Software Suite in the following folder.

To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

Destination Folder

C:\OpenAT

Figure 3: Open AT® Software Installation – Destination Folder

vii. Click **Next** to proceed. The files will be copied under the specified destination folder.

Note	The folder path must not include any space.

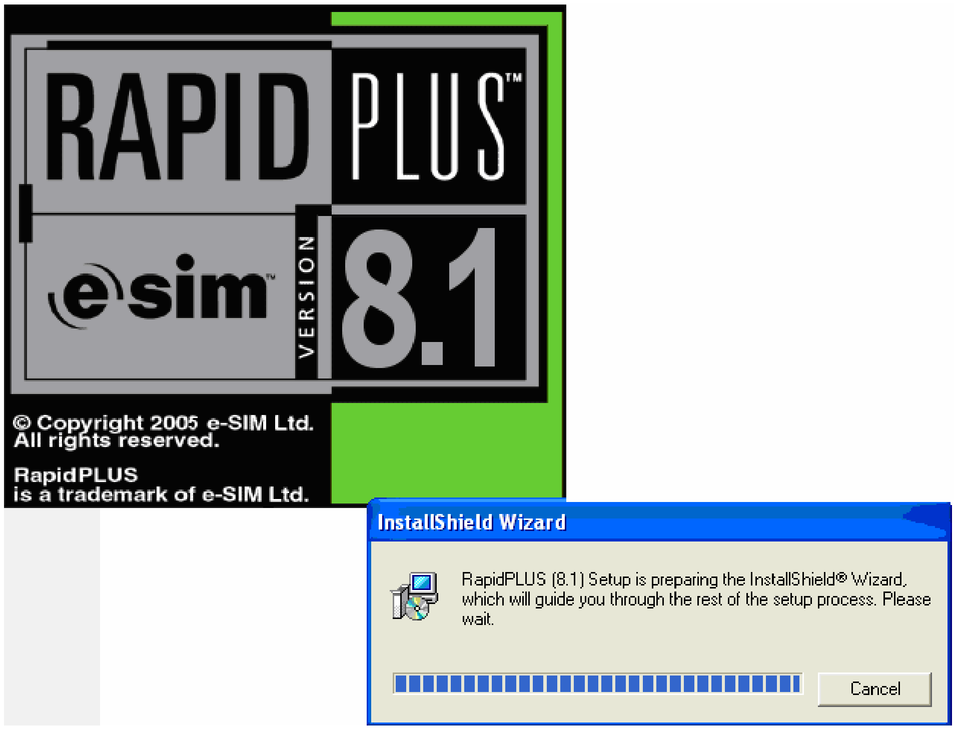
viii. The default location (as shown in the example) is the C drive. However, you can click **Browse** and change this location to suit your requirements. Click **Next** to proceed.

1.3.3.2 Installation of e-SIM RapidPLUS™ 8.1

To install the RapidPLUS™ Application:

i. This launches an install shield wizard, which guides the user through the various installation steps; as shown in the figure below.

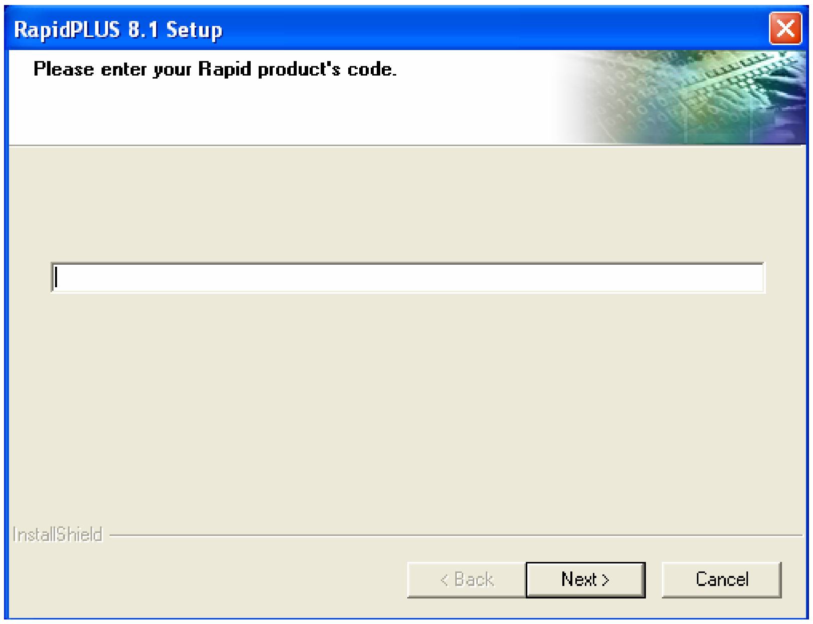
.



RapidPLUS (8.1) Setup is preparing the InstallShield® Wizard, which will guide you through the rest of the setup process. Please wait.

Figure 4: RapidPLUS™ Installation

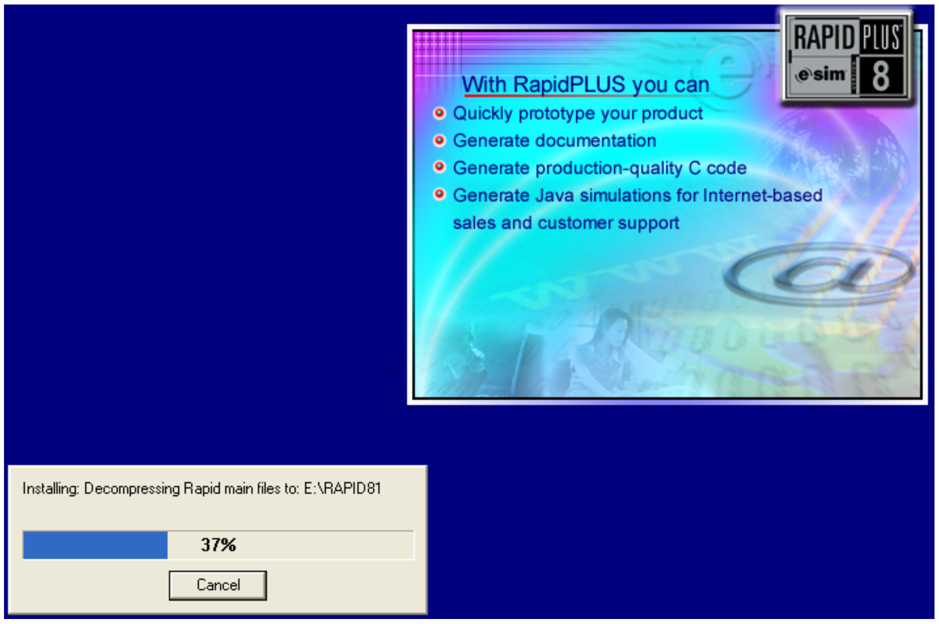
ii. Once installation has started, the setup program prompts the user to enter the product key for the RapidPLUS™. The product key is provided with the setup of RapidPLUS™ and must be entered in the text area provided, as shown in the figure below.


RapidPLUS 8.1 Setup

Please enter your Rapid product's code.

Figure 5: Dialog Box to Enter the Rapid Product Code.

iii. Once the Product key has been supplied, the installation continues as shown in the next figure before it is completed.


RAPID PLUS
e sim
8
With RapidPLUS you can
Quickly prototype your product
Generate documentation
Generate production-quality C code
Generate Java simulations for Internet-based sales and customer support

Installing: Decompressing Rapid main files to: E:\RAPID81

37%

Figure 6: RapidPLUS™ Installation Continues

Installing the RapidPLUS™ License

Once the RapidPLUS™ Application has been installed on your system, proceed with the following steps to retrieve the license from e-SIM.

i. Run the executable ...\RAPID81\FLEXlm\lmtools.exe. This saves the Host ID settings in a file.

ii. This file must be sent to e-SIM for the generation of the Rapid License.

iii. E-SIM sends back the license.DAT file to the user. This file must be copied in the ...\RAPID81\FLEXlm folder.

iv. Once the License file has been copied at the specified location, all the features of RapidPLUS™ such as code generation, are activated.

2 e-SIM Development Tools

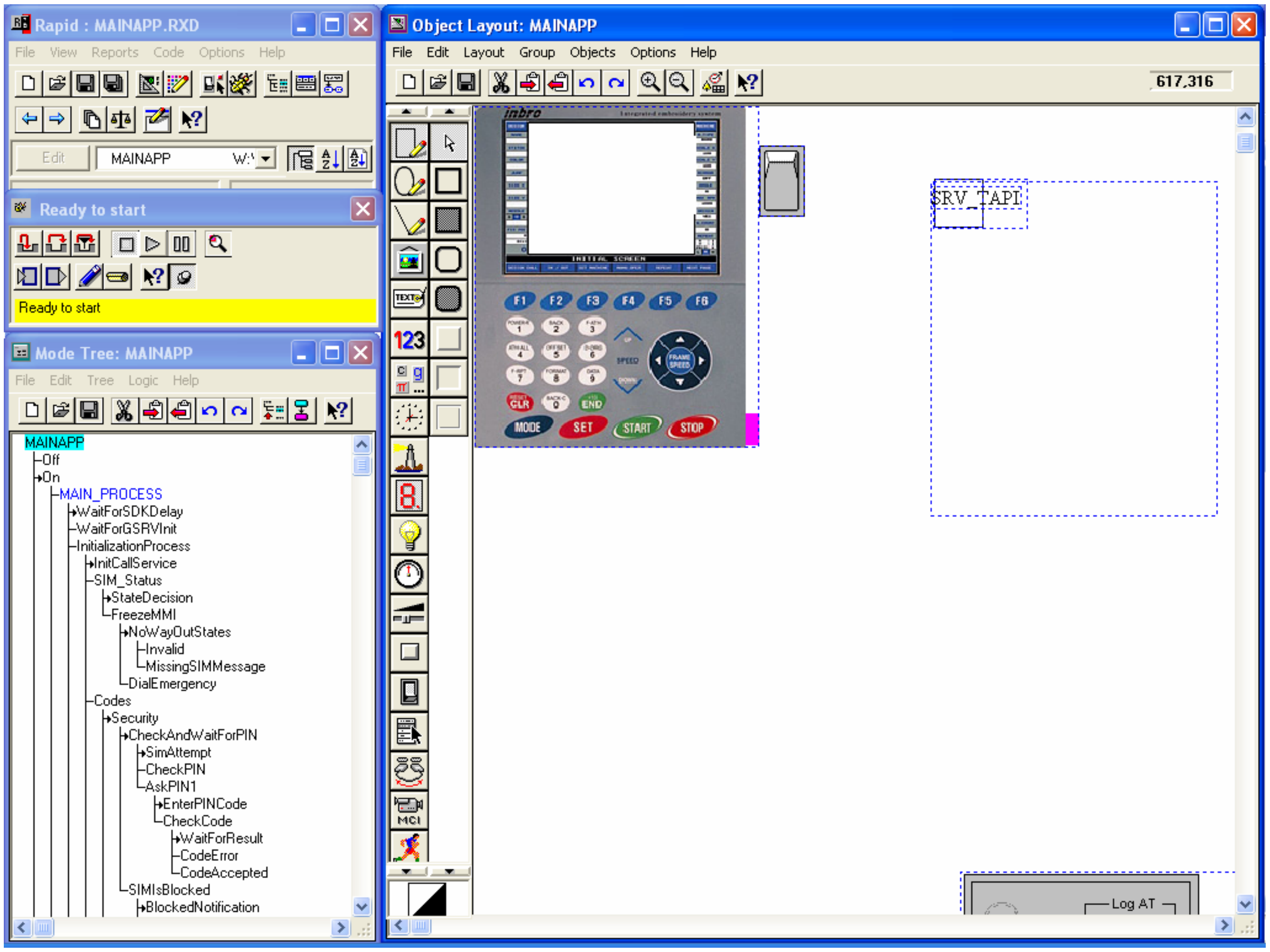
2.1 e-SIM Overview

e-SIM (www.e-Sim.com) is the leading provider of complete MMI solutions to the mobile handset industry. Incorporated in millions of handsets and vertical applications on the market and used by major companies, e-SIM's proven MMI technology enables handset manufacturers to create state-of-the-art products and bring them to market quickly and cost-effectively. e-SIM facilitates the attainment of this goal by providing the necessary elements for quick, high-quality MMI development using e-SIM's MMI Reference Design, RapidPLUS™ development and customization tools.

2.2 RapidPLUS™ 8.1

What is RapidPLUS™?

RapidPLUS™, a product of e-SIM Ltd., is a comprehensive software package for the generation of simulations and prototypes of electronic systems. Simulations developed with RapidPLUS™ can be linked with external programs to obtain the ultimate computer based training (CBT) solution. With ANSI-C code generation, RapidPLUS™ enables transformation of a virtual prototype into an executable application that can run on genuine embedded systems. In a typical multitasking embedded system, the generated RapidPLUS™ application, or the RapidPLUS™ task, runs the system's man-machine interface.



```
MAINAPP
  Off
  On
    MAIN_PROCESS
      WaitForSDKDelay
      WaitForGSRVInit
      InitializationProcess
        InitCallService
        SIM_Status
          StateDecision
          FreezeMMI
            NoWayOutStates
              Invalid
              MissingSIMMessage
            DialEmergency
        Codes
          Security
            CheckAndWaitForPIN
              SimAttempt
              CheckPIN
              AskPIN1
                EnterPINCode
                CheckCode
                  WaitForResult
                  CodeError
                  CodeAccepted
            SIMIsBlocked
              BlockedNotification
```

Figure 7: RapidPLUS™ 8.1 – Snapshot

In general terms, the RapidPLUS™ is an Integrated Development Environment (IDE) for the development of Man-Machine Interfaces (MMI). The primary use of the RapidPLUS™ includes:

- Building simulations
- Generating documentation
- Generating ANSI-C code for embedded systems

The RapidPLUS™ draws its power from being based on a robust state-machine. The RapidPLUS™ language semantics may seem familiar to experienced software developers, but the concept of state-machine oriented development needs getting used to. The RapidPLUS™ state-machine is defined by modes and by transitions between these modes. The most important rule of RapidPLUS™ application development is building a good set of modes and defining the correct transitions between them.

RapidPLUS™ Code Generator

The RapidPLUS™ Code Generator produces a header (*.h) and a program (*.c) files for the currently loaded RapidPLUS™ application and each of its user objects. These C source code files translate the RapidPLUS™ objects, modes, transitions, triggers, and activities from their native RapidPLUS™ syntax into C data structures and functions.

RapidPLUS™ User Objects

User objects are RapidPLUS™ applications that have a clearly defined interface. They can be used as encapsulated objects within another RapidPLUS™ application (called the parent application). The parent application sees the user object like any other RapidPLUS™ object. A user object's interface to the parent application consists of exported properties, exported events, exported functions, and messages (structure unions). In the code generation context, the parent application represents the embedded system's RapidPLUS™ task, while its user objects represent 'real-life' components or modules, such as keypads, switches, displays or protocol stacks.

While generating the application, objects can be looked upon in two ways:

- A user object can be generated in full (including its objects and internal logic)
- A user object can be generated as an interface

In the latter case, the Code Generator ignores the objects and internal logic of the user object and only generates the user object's interface to the parent application.

User Defined Objects (UDOs)

A UDO is a discrete unit of functionality, developed as a separate module. It can be used in different applications. A UDO can be seen as a complete lower level RapidPLUS™ application, used by a higher-level RapidPLUS™ application. It provides functions, properties, and events that can be used in the parent application. It can send and receive messages from the parent application. UDOs can be reused in different applications, and hence expand the RapidPLUS™ object library with customized objects.

User Defined Interfaces (UDIs)

An interface-only UDO, or UDI, is a UDO used to communicate with external modules in an embedded system. Such systems often contain hardware and software modules that must interact with the RapidPLUS™ application, such as keypad or AT module. Since these are independent modules, the RapidPLUS™ application provides connection points in order to communicate with them. These connection points are known as UDIs. A UDI does not export any implementation, but exports the component interface when the code is generated. The resulting generated code of a UDI contains empty functions. These are the connection points to the different independent modules. During integration, these empty functions are filled with the relevant API calls.

User Defined Data Objects (UDDs)

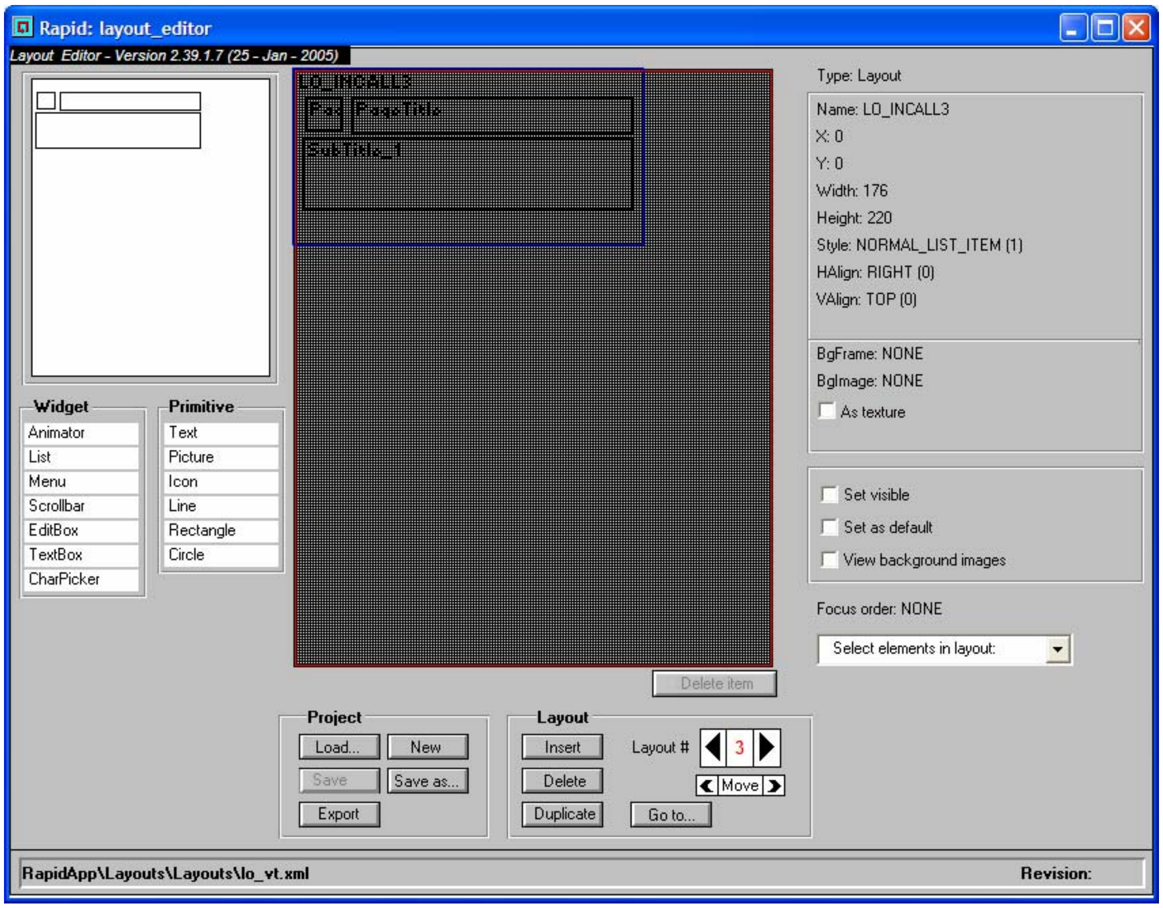
A data container UDO, or UDD, is essentially a UDO that only exports a message and is used for data exchange between two or more other UDOs. The UDD itself contains no object or logic. Each one of the UDOs that exchange the data has an access to the UDD to set the values of different fields in the UDD's message.

2.3 Other MMI Customization Tools

2.3.1 Layout Editor

As the name suggests, the Layout Editor is a tool to create and modify MMI layouts. The Layout Editor itself is developed using the RapidPLUS™.

The Layout Editor is part of the Open AT® GTi build and can be accessed from the path: Reference_Design\libs\ESIM\LayoutEditor\ LAYOUT_EDITOR.RPD.



Type: Layout
Name: LO_INCALL3
X: 0
Y: 0
Width: 176
Height: 220
Style: NORMAL_LIST_ITEM (1)
HAlign: RIGHT (0)
VAlign: TOP (0)
BgFrame: NONE
BgImage: NONE
As texture
Set visible
Set as default
View background images
Focus order: NONE

RapidApp\Layouts\Layouts\lo_vt.xml
Revision:

Figure 8: Layout Editor

The Layout Editor provides options to create new layouts, as well as to modify and delete existing layouts. Some predefined shapes and widgets[1] are also provided to speed up the layout development process. Layouts, once developed, are saved in an xml file, which is later used by RapidPLUS™ for MMI development.

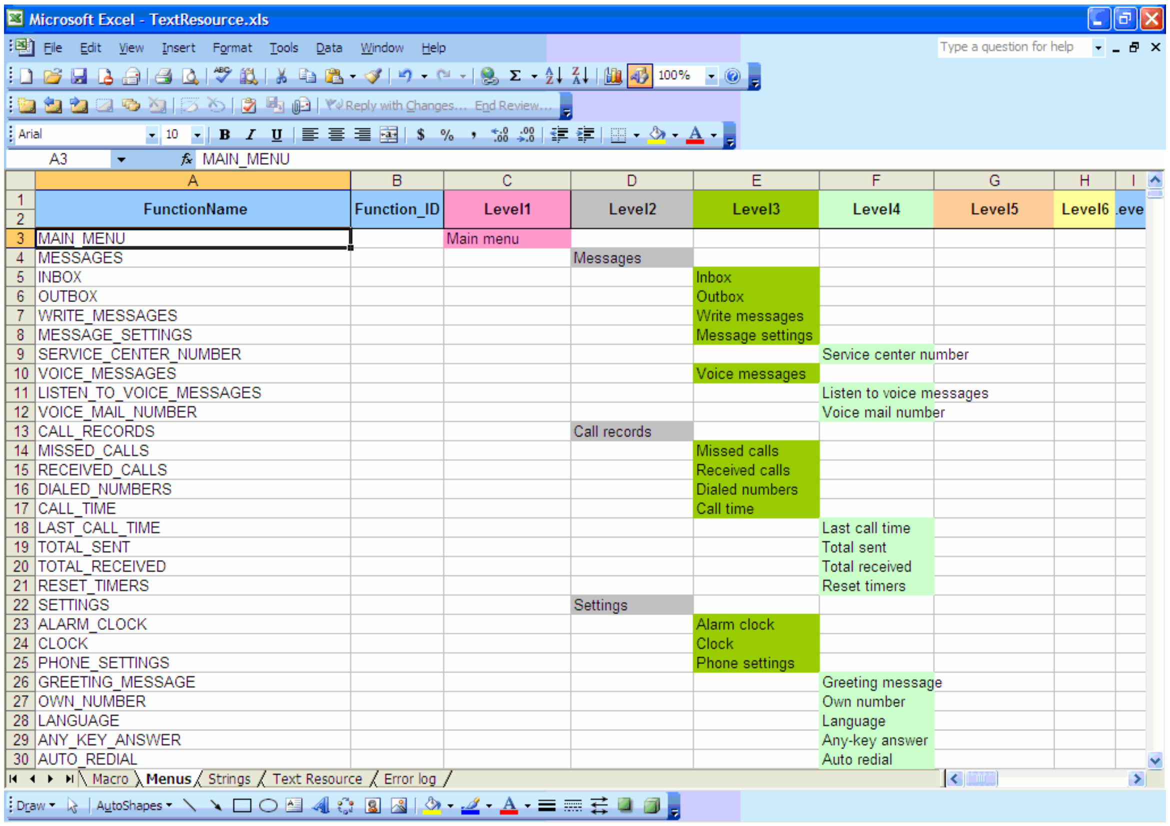
2.3.2 Text Resource Manager

The Text Resource Manager is a tool provided by e-SIM to add, delete, and modify the text string used in the MMI. This tool is also provided along with the Open AT® GTi build.

The Text Resource Manager is an excel file with embedded macros used to perform the desired operations. It can be accessed from the following path: Reference_Design\libs\ESIM\RapidApp\TextResource.xls in the Open AT® GTi build. The TextResource.xls file includes different worksheets for different operations. The worksheets in this excel file are: Macro, Menus, Strings, Text Resource, and Error log.

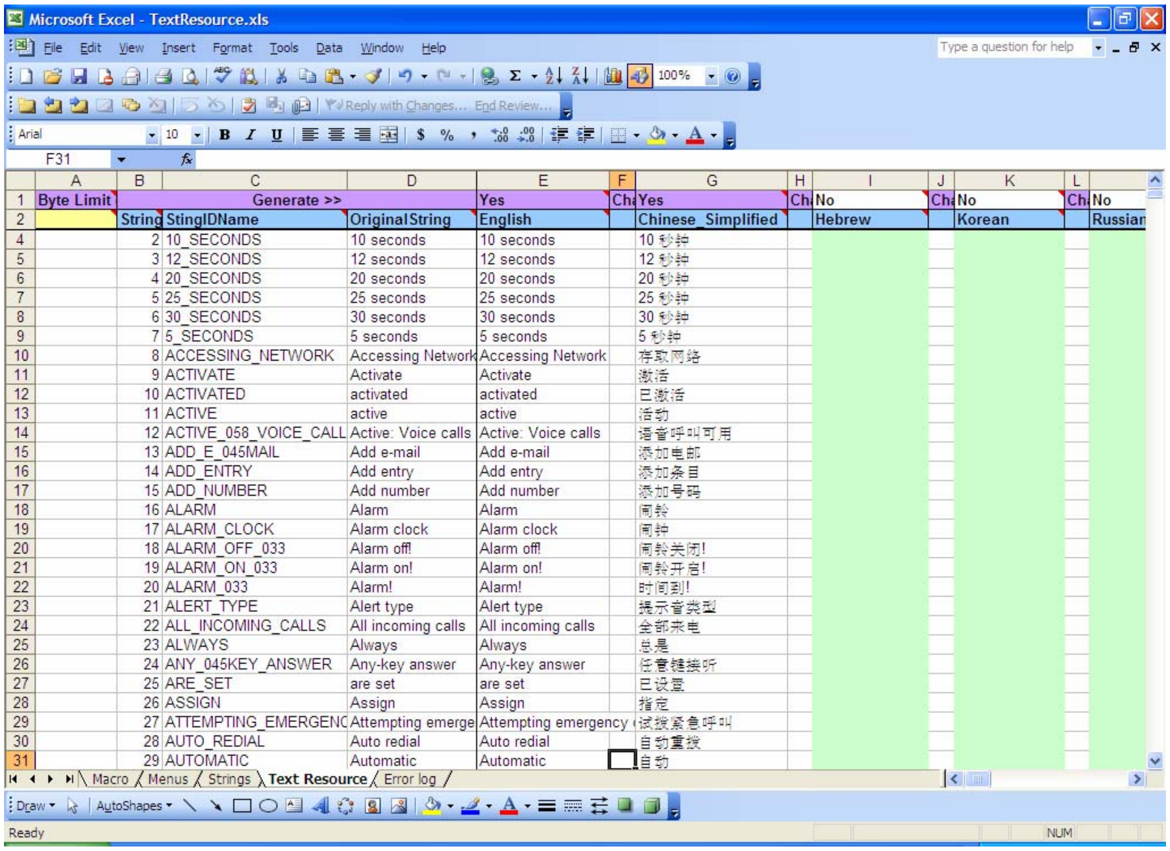
The 'Menus' worksheet contains the menu strings shown in the MMI menu in the same order and sequence as shown. The Level 1 column contains the highest level menu, followed by level 2 child menus and so on. Each unique menu item string is added to the 'Text Resource' worksheet, which contains all the strings used in the MMI along with their translations in all the supported languages.

[1] The Widgets are reusable MMI controls provided by e-SIM along with the reference MMI. They are detailed in section 5.1.4


Microsoft Excel - TextResource.xls
File Edit View Insert Format Tools Data Window Help
Type a question for help
Reply with Changes... End Review...
Arial 10
100%
A3
MAIN_MENU
Macro
Menus
Strings
Text Resource
Error log
Draw
AutoShapes

	A	B	C	D	E	F	G	H	I
1–2	FunctionName	Function_ID	Level1	Level2	Level3	Level4	Level5	Level6	.eve
3	MAIN_MENU		Main menu						
4	MESSAGES			Messages					
5	INBOX				Inbox				
6	OUTBOX				Outbox				
7	WRITE_MESSAGES				Write messages				
8	MESSAGE_SETTINGS				Message settings				
9	SERVICE_CENTER_NUMBER					Service center number			
10	VOICE_MESSAGES				Voice messages				
11	LISTEN_TO_VOICE_MESSAGES					Listen to voice messages			
12	VOICE_MAIL_NUMBER					Voice mail number			
13	CALL_RECORDS			Call records					
14	MISSED_CALLS				Missed calls				
15	RECEIVED_CALLS				Received calls				
16	DIALED_NUMBERS				Dialed numbers				
17	CALL_TIME				Call time				
18	LAST_CALL_TIME					Last call time			
19	TOTAL_SENT					Total sent			
20	TOTAL_RECEIVED					Total received			
21	RESET_TIMERS					Reset timers			
22	SETTINGS			Settings					
23	ALARM_CLOCK				Alarm clock				
24	CLOCK				Clock				
25	PHONE_SETTINGS				Phone settings				
26	GREETING_MESSAGE					Greeting message			
27	OWN_NUMBER					Own number			
28	LANGUAGE					Language			
29	ANY_KEY_ANSWER					Any-key answer			
30	AUTO_REDIAL					Auto redial			

Figure 9: Text Resource Manager - Menus



Microsoft Excel - TextResource.xls

F31

	A	B	C	D	E	F	G	H	I	J	K	L	
1	Byte Limit	Generate >>			Yes	Cha	Yes	Ch	No	Ch	No	Ch	No
2		String	StingIDName	OriginalString	English		Chinese_Simplified		Hebrew		Korean		Russian
4		2	10_SECONDS	10 seconds	10 seconds		10 秒钟						
5		3	12_SECONDS	12 seconds	12 seconds		12 秒钟						
6		4	20_SECONDS	20 seconds	20 seconds		20 秒钟						
7		5	25_SECONDS	25 seconds	25 seconds		25 秒钟						
8		6	30_SECONDS	30 seconds	30 seconds		30 秒钟						
9		7	5_SECONDS	5 seconds	5 seconds		5 秒钟						
10		8	ACCESSING_NETWORK	Accessing Network	Accessing Network		存取网络						
11		9	ACTIVATE	Activate	Activate		激活						
12		10	ACTIVATED	activated	activated		已激活						
13		11	ACTIVE	active	active		活动						
14		12	ACTIVE_058_VOICE_CALL	Active: Voice calls	Active: Voice calls		语音呼叫可用						
15		13	ADD_E_045MAIL	Add e-mail	Add e-mail		添加电邮						
16		14	ADD_ENTRY	Add entry	Add entry		添加条目						
17		15	ADD_NUMBER	Add number	Add number		添加号码						
18		16	ALARM	Alarm	Alarm		闹铃						
19		17	ALARM_CLOCK	Alarm clock	Alarm clock		闹钟						
20		18	ALARM_OFF_033	Alarm off!	Alarm off!		闹铃关闭!						
21		19	ALARM_ON_033	Alarm on!	Alarm on!		闹铃开启!						
22		20	ALARM_033	Alarm!	Alarm!		时间到!						
23		21	ALERT_TYPE	Alert type	Alert type		提示音类型						
24		22	ALL_INCOMING_CALLS	All incoming calls	All incoming calls		全部来电						
25		23	ALWAYS	Always	Always		总是						
26		24	ANY_045KEY_ANSWER	Any-key answer	Any-key answer		任意键接听						
27		25	ARE_SET	are set	are set		已设置						
28		26	ASSIGN	Assign	Assign		指定						
29		27	ATTEMPTING_EMERGENC	Attempting emerge	Attempting emergency (		试拨紧急呼叫						
30		28	AUTO_REDIAL	Auto redial	Auto redial		自动重拨						
31		29	AUTOMATIC	Automatic	Automatic		自动						

Figure 10: Text Resource Manager - Multilingual Support

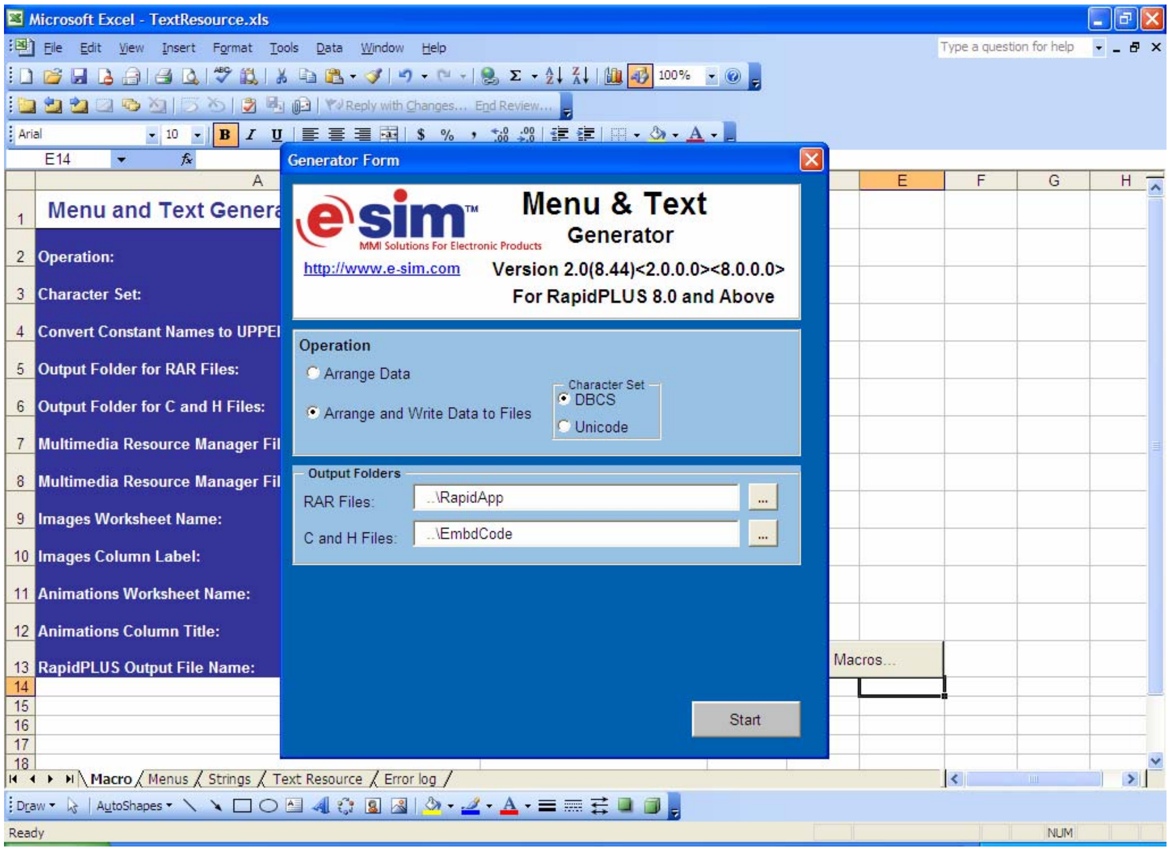

Microsoft Excel - TextResource.xls
Menu and Text Genera
Operation:
Character Set:
Convert Constant Names to UPPE
Output Folder for RAR Files:
Output Folder for C and H Files:
Multimedia Resource Manager Fil
Multimedia Resource Manager Fil
Images Worksheet Name:
Images Column Label:
Animations Worksheet Name:
Animations Column Title:
RapidPLUS Output File Name:
Generator Form
Menu & Text Generator
http://www.e-sim.com
Version 2.0(8.44)<2.0.0.0><8.0.0.0>
For RapidPLUS 8.0 and Above
Operation
Arrange Data
Arrange and Write Data to Files
Character Set
DBCS
Unicode
Output Folders
RAR Files: ..\RapidApp
C and H Files: ..\EmbdCode
Start
Macros...
Macro
Menus
Strings
Text Resource
Error log

Figure 11: Text Resource Manager - Menu & Text Generator

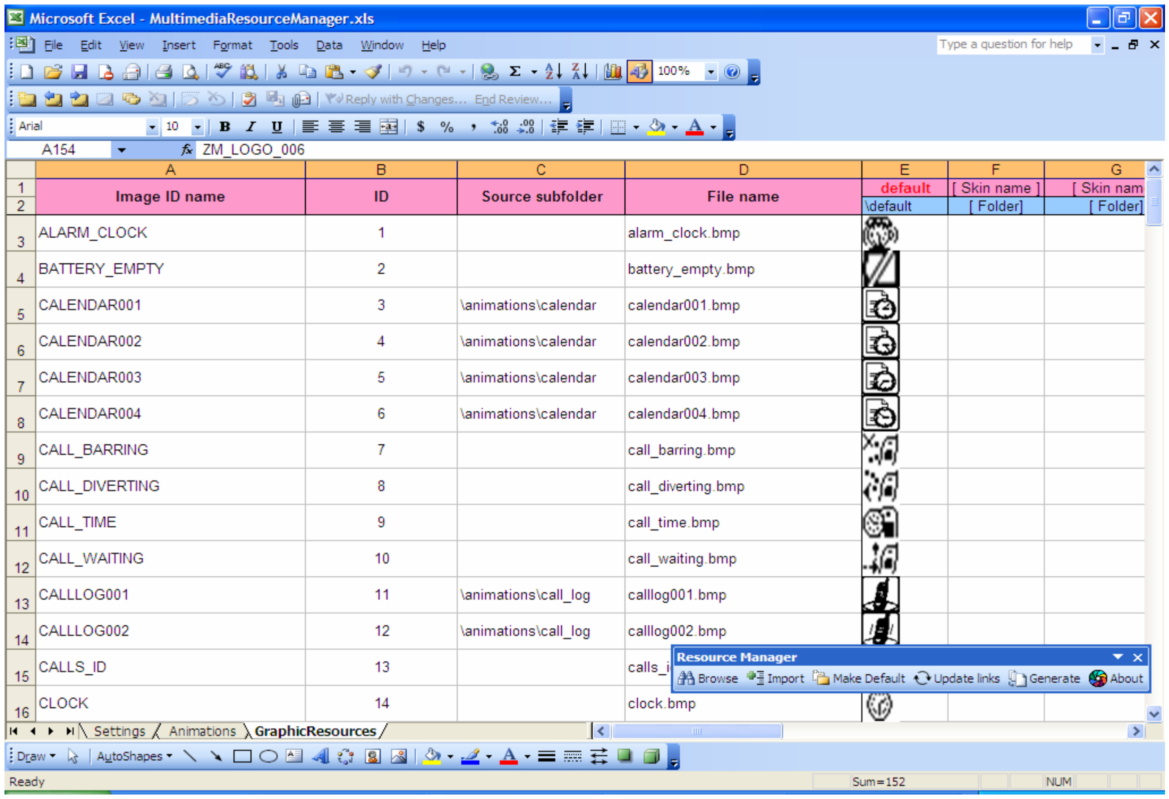
The 'Macro' worksheet contains the paths for the generated output folders as well as options related to the character set used for the MMI. For Open AT® GTi, the DBCS character set is used. The Text Resource Manager is not only responsible for text strings, but it is also the source for the generation of the corresponding .c and .h files for the Multimedia Resource Manager.

Note	DBCS stands for *double-byte character set*, a character set that uses two-byte (16-bit) characters rather than one-byte (8-bit) characters. Some languages, such as Chinese, Japanese and Korean, have writing schemes with many different characters that cannot be represented with single-byte codes such as ASCII and EBCDIC. In a single-byte character set, the possible number of binary combinations is 256; this number increases to 65,536 in a double byte character set.

2.3.3 Multimedia Resource Manager

The Multimedia Resource Manager is a tool provided by e-SIM to manage the multimedia resources such as graphics and animations. It is provided along with the Open AT® GTi build and can be accessed from the following path: Reference_Design\libs\ESIM\RapidApp\Images\MultimediaResourceManager.xls.

This tool is an MS excel file with some predefined macros to perform the required operations. This tool provides options to add, delete, and modify the graphics elements and animations in the MMI. All the graphics used in the MMI are added to the GraphicsResources worksheet of this excel file. The contents of the Multimedia Resource Manager are translated to the .c and .h files using the macros available in Text Resource Manager (TextResource.xls). The Text Resource Manager is detailed in section 2.3.2. These .c and .h files are used by the RapidPLUS™ to include all the graphics present in the MMI.


Microsoft Excel - MultimediaResourceManager.xls
File Edit View Insert Format Tools Data Window Help
Type a question for help
Reply with Changes... End Review...
Arial 10
100%
Resource Manager
Browse Import Make Default Update links Generate About
Settings Animations GraphicResources
Draw AutoShapes
Ready Sum=152 NUM

A154 ZM_LOGO_006

	Image ID name	ID	Source subfolder	File name	default \default	[Skin name] [Folder]	[Skin nam [Folder]
3	ALARM_CLOCK	1		alarm_clock.bmp			
4	BATTERY_EMPTY	2		battery_empty.bmp			
5	CALENDAR001	3	\animations\calendar	calendar001.bmp			
6	CALENDAR002	4	\animations\calendar	calendar002.bmp			
7	CALENDAR003	5	\animations\calendar	calendar003.bmp			
8	CALENDAR004	6	\animations\calendar	calendar004.bmp			
9	CALL_BARRING	7		call_barring.bmp			
10	CALL_DIVERTING	8		call_diverting.bmp			
11	CALL_TIME	9		call_time.bmp			
12	CALL_WAITING	10		call_waiting.bmp			
13	CALLLOG001	11	\animations\call_log	calllog001.bmp			
14	CALLLOG002	12	\animations\call_log	calllog002.bmp			
15	CALLS_ID	13		calls_i			
16	CLOCK	14		clock.bmp			

Figure 12: Multimedia Resource Manager - GraphicResources

Note	Before opening the MultimediaResourceManager.xls file, run a batch file, namely the RegisterActiveX.bat file, stored in the same folder. This batch file registers the ActiveX components used by the macros in the excel file.

3 Wavecom Open AT® Software IDE

3.1 Open AT® Software Overview

The Open AT® Software is an application development framework that helps in the design, development, and execution of self-reliant and complete applications, which runs on the Open AT® Firmware in the Wavecom Wireless CPUs. With the Open AT® Software, wireless application developers have the flexibility and autonomy of creating innovative wireless applications that can be *embedded* in a Wavecom Wireless CPU. This application development environment provides various APIs with which you can easily create applications to perform complex tasks; for example, an automatic meter reading system.

The following image represents a microcontroller-based application without the Open AT® Software. In such a configuration, all applications are external to the Wireless CPU.

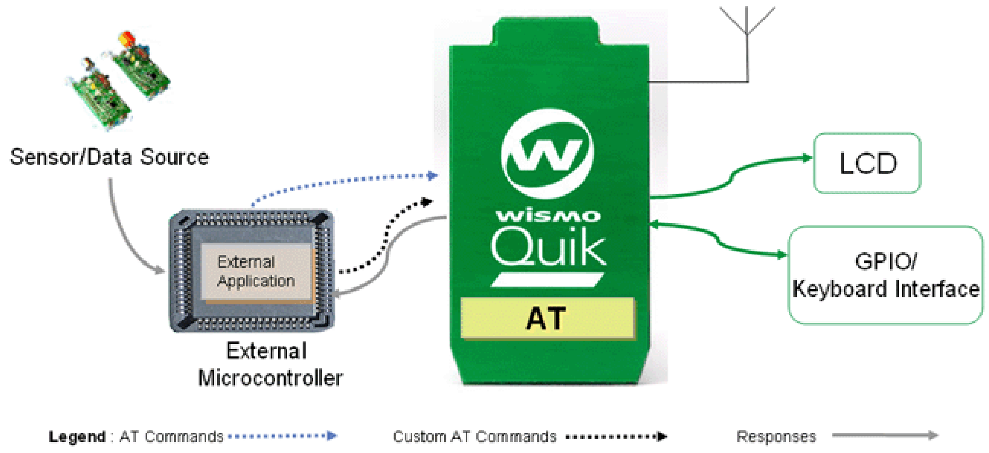


Figure 13: WITHOUT Open AT® Software

In such a scenario, without the Open AT® Software, you must have interfaces for the hardware, and develop custom applications to embed in the external micro-controller, so as to interact with WISMO and the external devices (either via a GPIO or BUS interface).

With the Open AT® Software, the external micro-controller is not required. Therefore, the embedded application takes control of the Wireless CPU and interacts with external devices. This is shown in the following image.

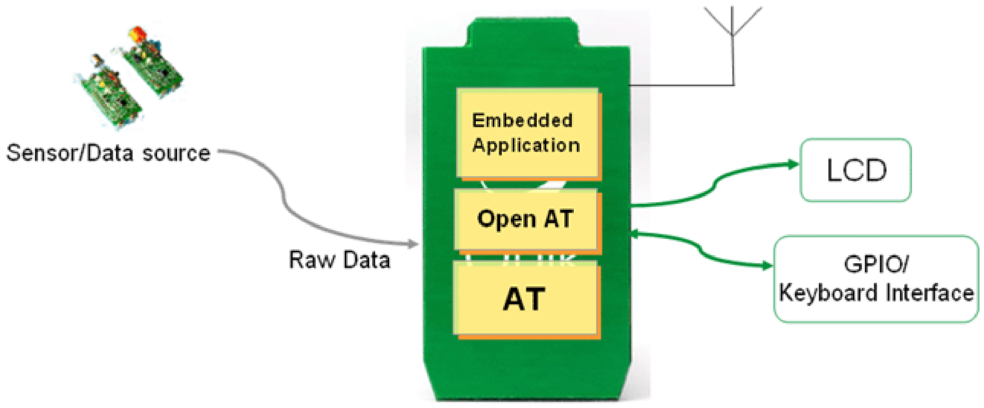


Figure 14: WITH Open AT® Software

The Open AT® Software application ensures better flexibility. With the Open AT® Software, you can modify the response to basic AT commands and therefore, customize the system according to the required functionality. In addition to basic AT commands, you can also develop custom commands and add them to the available command set. The required functionality can be implemented in these commands.

This provides more power to the wireless developer and eases the programming task. The following figure illustrates this advantage with a scenario in which custom commands are used.

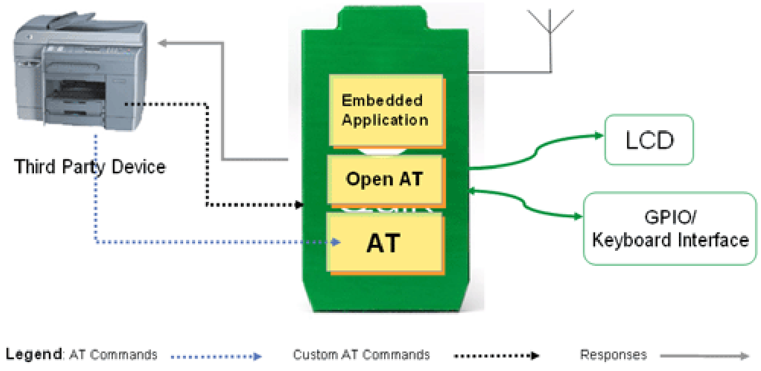


Figure 15: Open AT® Software with Custom Commands

3.2 Open AT® Software Areas of Application

The Open AT® Software can provide functionalities in a wide variety of applications, such as:

- Systems that use standard GSM and GPRS services for status reporting

- Systems that provide internet connectivity and access to Web services such as FTP, SMTP, and PING
- Systems that act as TCP/UDP client/server machines and provide services to other nodes
- Systems that provide location information in automotive systems using GPS (Global Positioning System) services
- Applications that send sensor information (read through GPIOs) to remote base locations
- Applications that store sensor data or other vital information in flash (ROM) and help protect critical information for state-of-the-art information processing systems

Domains where the Open AT applications can be used include Automotive, Fleet Management, Remote Asset Monitoring, Telemetry and Telematics, etc.

The Open AT® Software Automotive Application figure shown below is an example of the Open AT® Software usage. In case of a road traffic accident, the sensors fitted in the vehicle, send information to the base server, which in turn informs the emergency helpline, prompting it to send help to the accident site. The Open AT® Software application runs on the Wavecom Wireless CPU handles this process of collecting and sending information to the base server.

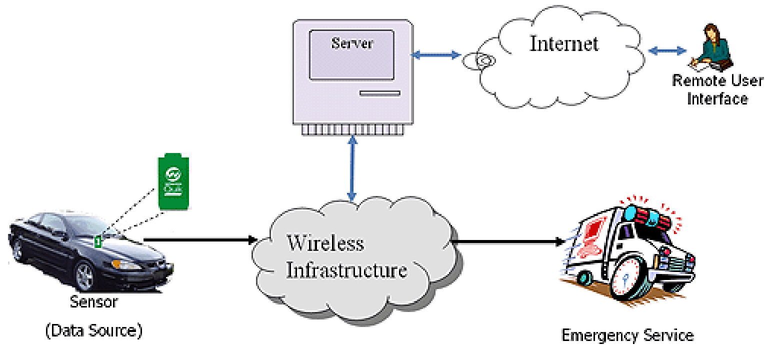


Figure 16: Open AT® Software Automotive Application

3.3 Advantages of Using the Open AT® Software

The advantages of using the Open AT® Software are:

- The Wavecom Open AT® Software library considerably helps in reducing development and testing time. You can concentrate on the application functionalities, instead of lower level technical details

- The Open AT® Software reduces the product cost. Without the Open AT® Software, a considerable amount of interfacing hardware and drivers are used to provide external stimuli (information) to the Wireless CPU. The Open AT® Software provides APIs to adjust the Wireless CPU power, without any external interfacing hardware or software
- The Open AT® Software application is firmly integrated within the core AT software. As the Wavecom Open AT® Software library (on which the application is built) is extensively tested to work with the core AT software, the level of integration and reliability of the embedded application meets user expectations. With the Wavecom Open AT® Software library, the embedded application can interact efficiently with the core software
- The Open AT® Software application provides more flexibility. In addition to basic AT commands, you can also develop custom commands and add them to the available command set. The required functionality can be implemented in these commands. Hence, the overall flexibility of programming is improved

3.4 The Open AT® Software ADL Library

The Application Development Library (ADL) was introduced in the V2 release of the Open AT® Software. This library corrects the previous version drawbacks of the Open AT Software and is highly recommended for all kinds of applications. The ADL mode offers the following advantages:

- High-level event processing interface
- Highly modular capabilities
- Plug-in libraries are available only for ADL interfaces (such as TCPIP library-Wavecom WIP)
- Event processing using user-defined callback functions
- Easily maintainable code
- Programming easiness

Note	The Open AT® GTi is available only when the ADL runs on an Open AT® Operating System.

3.5 Mandatory Embedded Application Code

To develop an Open AT® Software application, certain mandatory APIs must be implemented. The following sections describe the mandatory ADL APIs.

3.5.1 ADL API

To define the application call stack size follow the prototype below:

For Open AT 3.12:

```
u32 wm_apmCustomStack [256];     /* 256 is an example */
const u16 wm_apmCustomStackSize = sizeof (wm_apmCustomStack);
```

For Open AT 4.10:

```
const u16 wm_apmCustomStackSize = 256;    /* 256 is an example */
```

The entry point function called during the application initialization is:

```
void adl_main ( adl_InitType_e   InitType )
{
}
```

3.6 Open AT® Software Running Modes

There are two execution modes for any Open AT® Software application. The following section describes the target and remote modes of execution.

3.6.1 Target Mode

The application is actually downloaded in the Wireless CPU (programmed in Flash) and executed in the target mode. The application uses the Wireless CPU resources to perform the required operation. This mode is used when a standalone application must be deployed in the Wireless CPU. The application is self-reliant and can run without user intervention.

3.6.2 Remote Mode

In the remote mode, the application is executed on the development environment (PC) and uses its own resources such as memory and clock. This mode provides the facility of line-by-line debugging. This mode can be used to debug the application by inserting various breakpoints and traces. Dynamic changes in the contents of the memory can be viewed either using the tools available in Visual Studio 6 or Visual Studio .NET.

Wavecom provides the RTE (Remote Task Execution) tool (detailed in section 4.4) to debug applications in the remote mode.

The following chapters of this document provide more information on the target and remote modes of application development.

3.7 Open AT® Software Application Size Restrictions

The size of the Open AT® Software application when built must not exceed the following limits as described for the following Wavecom OS and CPUs.

OS 6.57:

ROM: up to 1.5 MB for Open AT ® 3.12

RAM: 256 KB

OS 6.61:

ROM: up to 1.5 MB for Open AT ® 4.10

RAM: 256 KB

The RAM allocated to the application in the Wireless CPU is not only for the application, but also shared with the ADL library and the plug-in (Internet, TCP/IP), if any.

Note	If the Open AT® GTi feature is not activated on the Wireless CPU, then the Open AT® GTi application will not run.

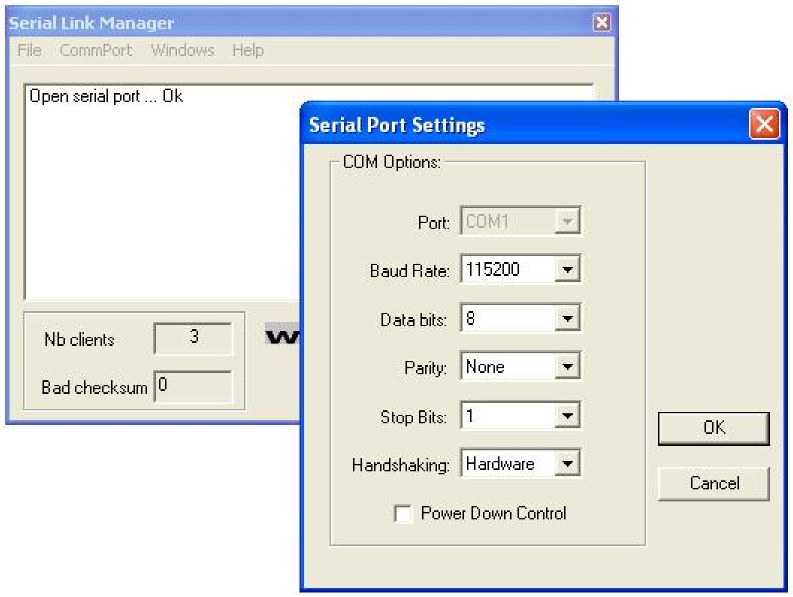
4 Wavecom Development Tools

4.1 SELIMA

The Serial Link Manager (SELIMA) is a tool used to communicate between the PC and Wavecom Wireless CPU. SELIMA acts as a mediator for the different applications on the PC, talking to the Wavecom Wireless CPU through the serial port (COM1).

SELIMA dispatches the data received from the serial link to the Wavecom PC applications (TMT, TE, and RTE). It also multiplexes the frames sent by these applications and forwards them to the Wireless CPU.

The Serial Link Manager is launched when you start the Target Monitoring Tool (TMT), the Terminal Emulator or the Remote Application. This opens the serial port and the SELIMA icon is displayed on the task bar. Click the **SELIMA** icon on the task bar; proceed to configure the serial port settings.



Open serial port ... Ok

Nb clients 3
Bad checksum 0

Port: COM1
Baud Rate: 115200
Data bits: 8
Parity: None
Stop Bits: 1
Handshaking: Hardware
☐ Power Down Control

Figure 17: Serial Link Manager Port Settings

Use the Comport option in the main menu to manage the serial port. Serial port settings are shown in the Serial Link Manager Port Settings image.

4.2 Target Monitoring Tool

The Target Monitoring Tool (TMT) is used to display the application's trace messages either from the Target or Remote Application. To use the TMT, follow the steps below:

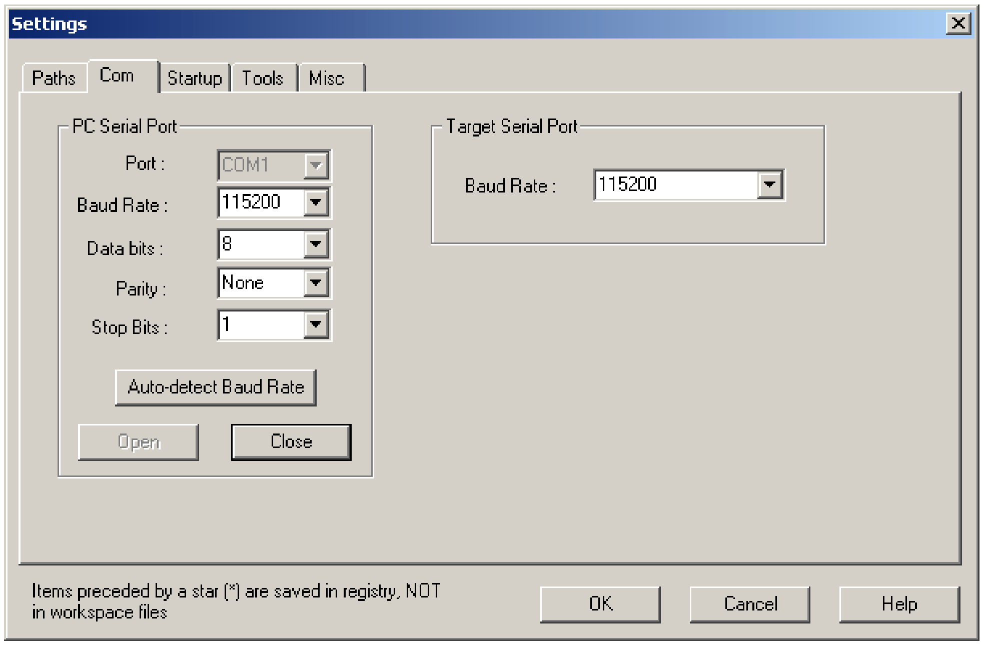
i. Switch on the Wireless CPU. Ensure the target is not in the initialization stage.

ii. Make sure that the selected COM port is not being used by any other application.

iii. Select the Start → Wavecom → Development Toolkit → Target Monitoring Tool shortcut icon.

iv. Check the PC serial link settings. Select the Preferences > Settings menu option. Select the COM tab.

v. Click the **Com** tab. Check default mode parameters. Default values are shown in the Default PC Serial Link Settings image.



Port : COM1
Baud Rate : 115200
Data bits : 8
Parity : None
Stop Bits : 1

Target Serial Port Baud Rate : 115200

Items preceded by a star (*) are saved in registry, NOT in workspace files

Figure 18: Default PC Serial Link Settings

vi. Select the Commands → Init Target menu option to initialize the target in the remote mode.

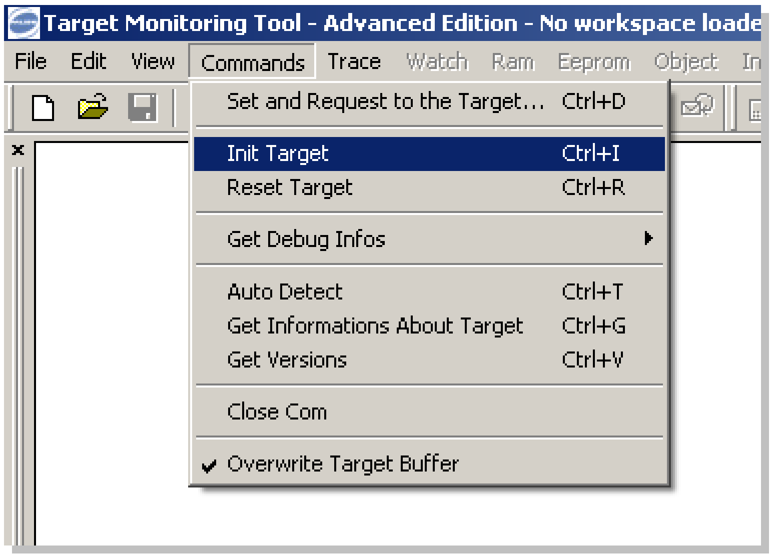

Target Monitoring Tool - Advanced Edition - No workspace loade
File Edit View Commands Trace Watch Ram Eeprom Object In
Set and Request to the Target... Ctrl+D
Init Target Ctrl+I
Reset Target Ctrl+R
Get Debug Infos
Auto Detect Ctrl+T
Get Informations About Target Ctrl+G
Get Versions Ctrl+V
Close Com
✓ Overwrite Target Buffer

Figure 19: Init Target Command Window

vii. To load the workspace, select the File → Open Workspace menu option.

viii. Run the Auto Detect command to get the same baud rate on your PC and on your target (typically: 115200 bps or 9600 bps).

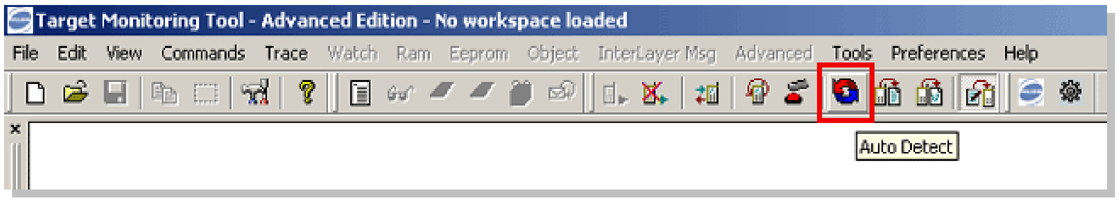

Target Monitoring Tool - Advanced Edition - No workspace loaded
File Edit View Commands Trace Watch Ram Eeprom Object InterLayer Msg Advanced Tools Preferences Help
Auto Detect

Figure 20: Auto Detect Icon

ix. The TMT will send frames at different successive baud rates when you select the Auto Detect option. If the target sends a frame back, target speed and computer speed match.

x. If the Auto Detect operation is successful, the values in the Baud Rate and Target Speed combo boxes in the Com tab of the Settings dialog box must be identical.

xi. Select the Trace → Set the Trace levels menu options to set trace levels.

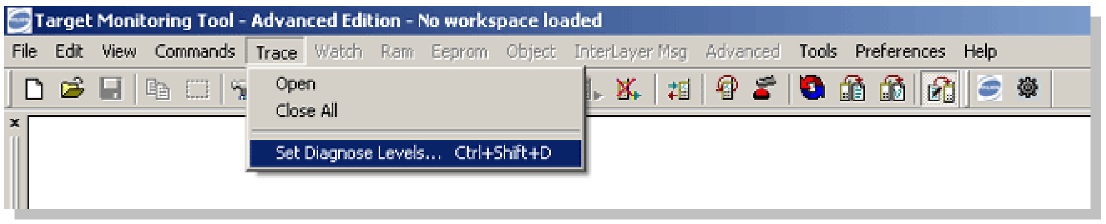

Target Monitoring Tool - Advanced Edition - No workspace loaded
File Edit View Commands Trace Watch Ram Eeprom Object InterLayer Msg Advanced Tools Preferences Help
Open
Close All
Set Diagnose Levels... Ctrl+Shift+D

Figure 21: Set Trace Level

xii. The Set and Request to the target window is displayed, from where you can enable traces. Select the trace parameter and click **Send Level**.

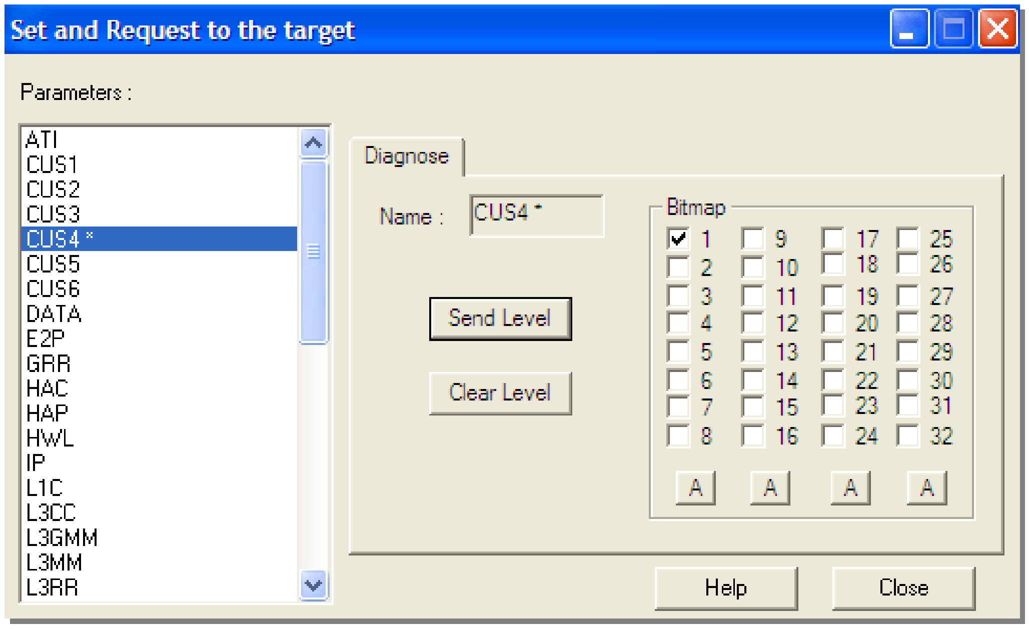


Figure 22: Set and Request to the Target Window

xiii. To check communication with the target:

a. Select the Terminal Emulator option on the TMT toolbar. Select the AT1 window.

b. Type AT and press Enter on your keyboard. A confirmation response must be displayed, otherwise, select the Commands → Init Target menu option (TMT toolbar) and type AT again.

c. Close the Terminal Emulator window.

xiv. Select the Traces → Open menu option in TMT to open the trace window.

xv. Run the application. The required trace levels are displayed in the Target Monitoring window.

xvi. The application can be run either in the Debug or Target mode.

xvii. To run the application in the remote mode (Remote Task Environment):

a. Run the application.

b. Select the CUS trace levels from the Remote Task Monitor.

c. Click **Start** on the Remote Task Monitor to start the application.

xviii. To run the application in the target mode:

a. Download the application on the Wireless CPU using HyperTerminal.

b. Follow the directions given in step 11 to set trace levels.

c. Run the application using the AT+WOPEN command (AT+WOPEN=1).

4.3 Terminal Emulator

The Terminal Emulator is an AT command terminal. It is used to send and receive string commands to or from a customer task (target or remote mode).

The terminal emulator connects an external application using the standard V.24 protocol.

The Terminal Emulator can be launched in one of the following ways:

i. If the TMT is running, start the Terminal Emulator using the corresponding button in the toolbar. The appropriate toolbar must be selected from the View option in the main menu.

ii. Select the Start → Wavecom → Development Toolkit → Terminal Emulator shortcut.

The Terminal Emulator provides two windows: one for AT commands, and the other for V24 Data flow. For more details, refer to the online help of the Terminal Emulator as shown in the figure below.

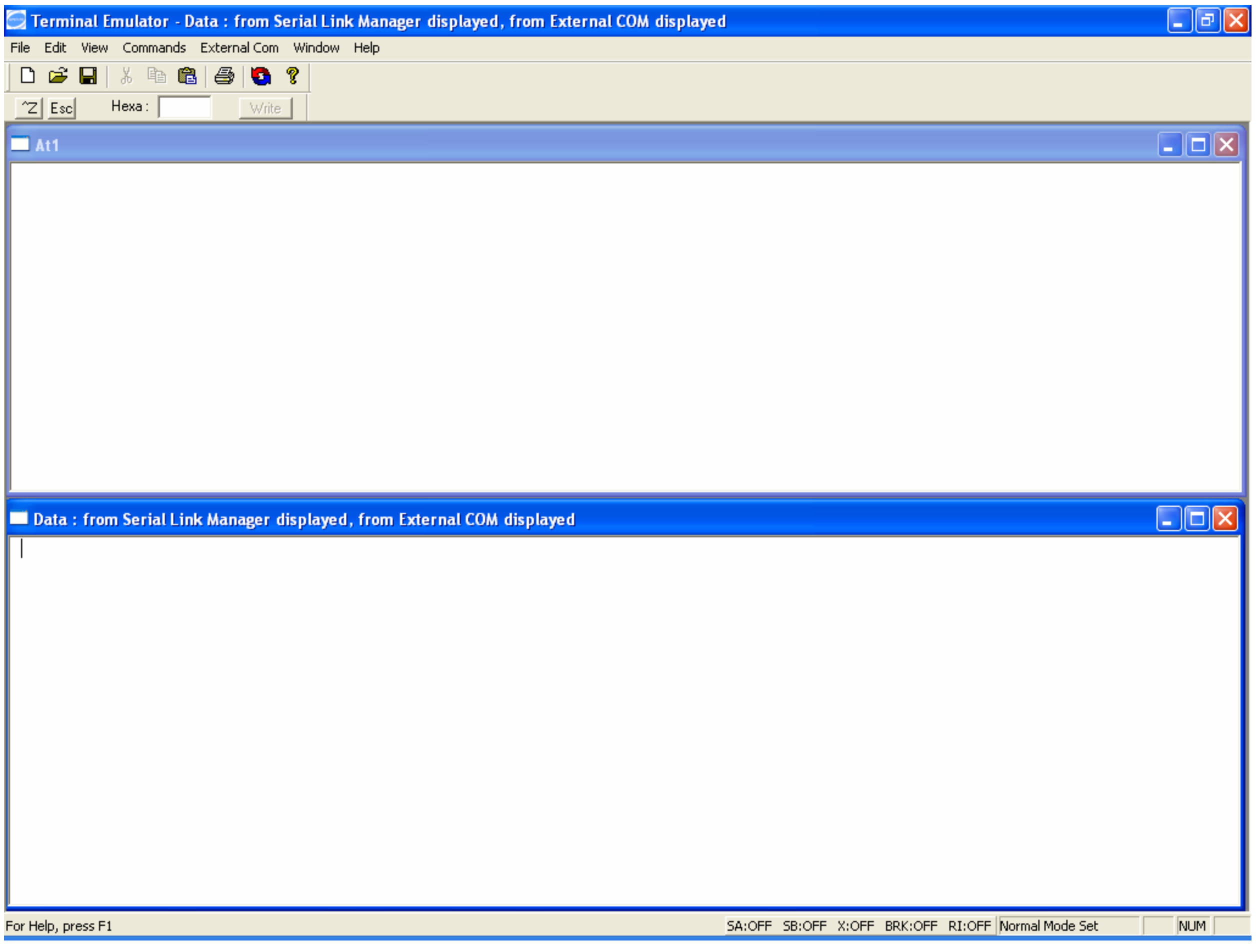

Terminal Emulator - Data : from Serial Link Manager displayed, from External COM displayed
File Edit View Commands External Com Window Help
^Z Esc Hexa : Write
At1
Data : from Serial Link Manager displayed, from External COM displayed
For Help, press F1
SA:OFF SB:OFF X:OFF BRK:OFF RI:OFF Normal Mode Set NUM

Figure 23: Terminal Emulator

4.4 Debugger: RTE Based on VC++

In the remote mode, the application is executed on the development environment (PC) and uses its own resources such as memory and clock. This mode, as the name indicates, is used to debug applications.

The application runs using the Real Time Execution Environment (RTE), which integrates itself either with Visual Studio 6 or Visual Studio .NET during the installation of the Open AT® Software. This mode can be used to debug the application by inserting various breakpoints and traces. Moreover, dynamic changes in the contents of the memory can also be seen using the tools available either in Visual Studio 6.00 or Visual Studio .NET. This mode can be used during the application development before producing the final binary (which will be downloaded to the Wireless CPU).

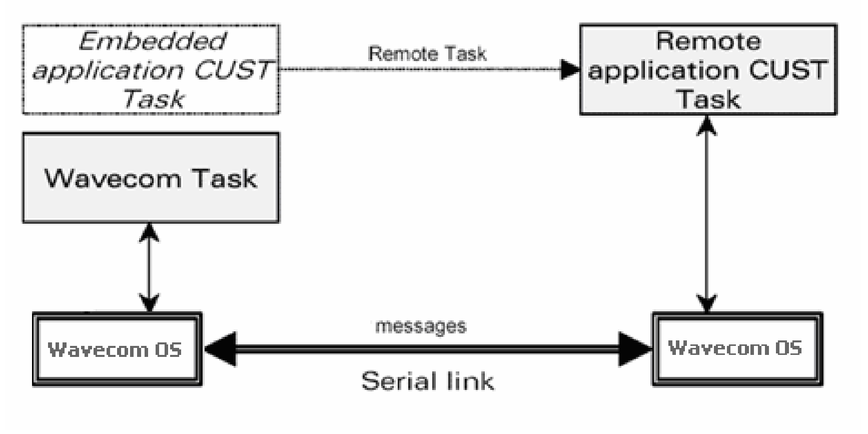


Figure 24: Remote Task Environment

The RTE is used to send instructions directly to the Wireless CPU, which then executes these instructions and performs the required operation. The Remote Task Execution (Remote Mode) image shows a Remote Task Monitor screenshot, which is displayed when the Open AT® Software program runs in the remote mode.

Before starting the remote application, set the appropriate CUS level traces from the Remote Task Monitor.

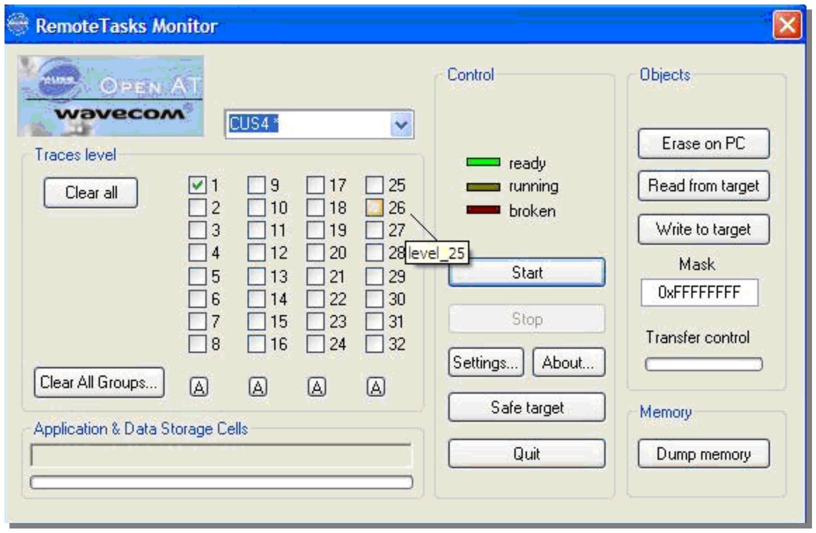


Figure 25: Remote Task Execution (Remote Mode)

5 Wavecom Sample MMI

5.1 Sample MMI Architecture

Wavecom provides the sample MMI implementation that has been developed using Wavecom and e-SIM MMI development tools. The Wavecom sample MMI is an embedded Open AT® Software application and runs on the Wavecom Open AT® Software ADL interface. Once integrated in the Open AT® Software, the ADL ensures connection with the underlying protocol stack.

The Wavecom MMI can be used as reference for development of customized MMI in the Wavecom Open AT® GTi architecture.

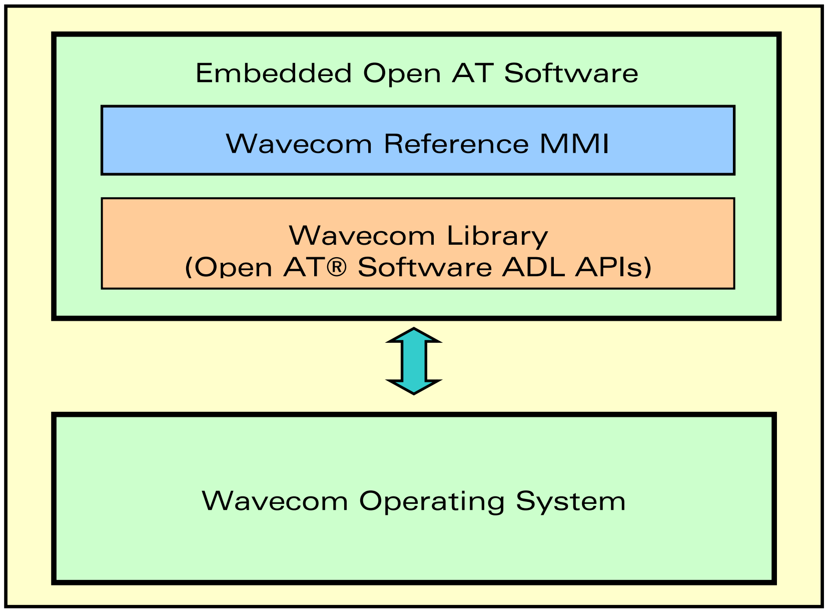


Figure 26: Sample MMI integration with Wavecom Open AT® Software

The Wavecom sample MMI consists of three main layers. The layers extend from high-level functionality (layer 1) to low-level functionality (layer 3). The main application, runs on the embedded AT application, manages the entire project by activating and deactivating the various components, according to priorities. It is the umbrella application that contains all other components.

The various types of functional components are:

- MMI Modules
- Service Components
- Data Containers
- Widgets
- Embedded Interface Component

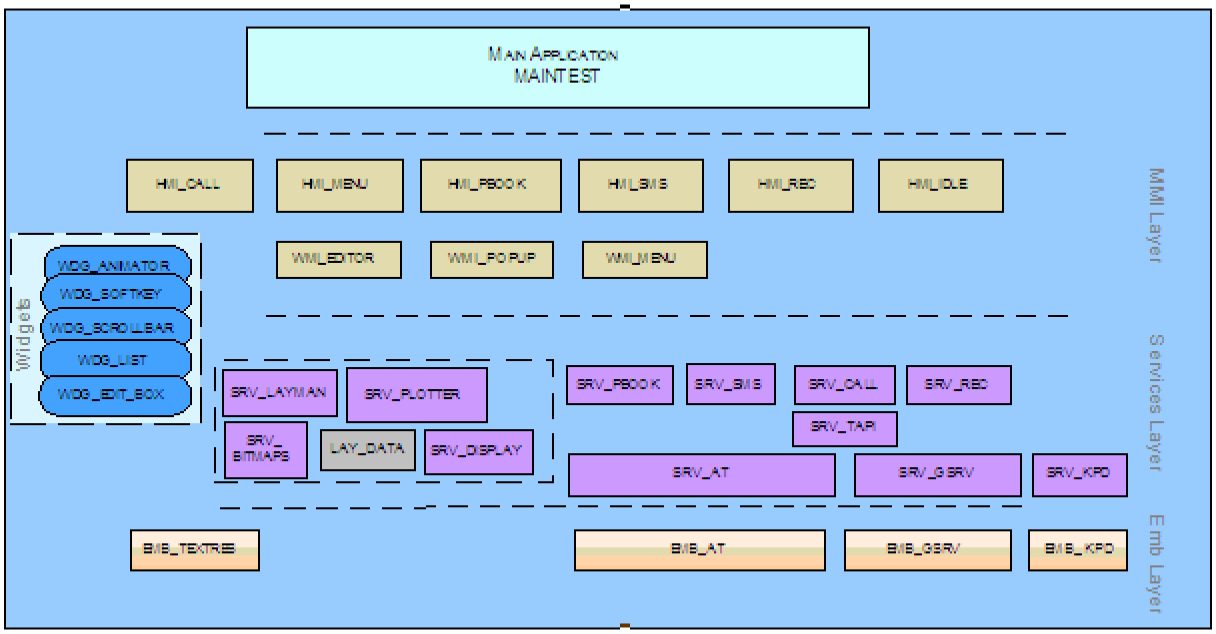


Figure 27: Wavecom Sample MMI Design

5.1.1 MMI Modules

MMI modules:

i. Control the main functionalities of the reference design.

ii. Handle the MMI: screens, menus, texts, and soft keys.

Only one MMI module can be active at a time. It controls displays on the screen.

There are two hierarchical levels of MMI modules:

- WMI modules: reusable MMI components that can control widgets, services, and embedded components
- HMI modules: each module handles a MMI project sub-section

The WMI modules provided along with the Wavecom MMI are:

- WMI_ADD_ENTRY - Encapsulates the adding entry logic. It recognizes if there is already a name similar to the one to add, and requests a different one if it is the case
- WMI_EDITOR - Collects the user numeric key inputs and generates the text on the phone, to dial, to send data or to create records of phonebook entries. It uses the edit box widget (WDG_EDIT_BOX) to add text to the display and the character picker widget (WDG_CHARPICKER) to select special characters
- WMI_MENU - Implements navigation in the menu tree. It is responsible for opening the correct layout, presenting soft keys, titles, icons, and status strings. It has two instances in the project: WMI_MENU and WMI_MENU_PRIVATE
- WMI_MENU_PRIVATE – The HMI_MENU is the only module that uses the WMI_MENU_PRIVATE as a private instance. This is done to distinguish between navigation on the main menu tree and navigation in small menus, requested when implementing functions. This distinction is required in order to enable function implementation resuming on the main menu
- WMI_PB_BROWSE – Used as a browser in the active phonebook. It receives an index of a record and starts browsing from that point. Pressing the alphanumeric keys may cause a search for a matching name and may change the list, if a match is found
- WMI_PB_DETAIL – Receives an index of an entry in the current phonebook and displays its details (phone numbers, email names, etc.,)
- WMI_POPUP – Supplies popup functionality for the entire project

The HMI modules provided with the Wavecom MMI are:

- HMI_CALL – Manages dialing, outgoing calls, active calls, conference calls, on hold calls and swapping calls
- HMI_IDLE – Handles the activity of the phone in idle mode e.g. missed call, new message and time & date information
- HMI_MENU – Handles menus, starting at the main menu. This module controls all leaf functions
- HMI_PBOOK – Handles the user interface for the phonebook functionality. It can be started in two modes – one to browse the phonebook and the other to open the phonebook menu
- HMI_REC – Handles the user interface for the call recording functionality. It uses UDD_REC to retrieve data
- HMI_SMS – Implements the MMI for the Messages menu as defined in the file, TextResources.xls

5.1.2 Service Components

Service components:

i. Provide specialized functionality to the MMI modules
ii. Create a flexible buffer between the MMI modules and embedded interface layer
iii. Manage the project's state and data, such as phonebook list, messages list, active calls, and call recordings using data containers called UDD. UDD's are detailed further down in the document
iv. Are active throughout the project lifetime

The service components provided with the Wavecom MMI are:

- SRV_CALL – Implements call functionality. Uses SRV_TAPI to interact with the network
- SRV_PBOOK – This service component provides access to the various phonebooks of the device
- SRV_TAPI – Provides AT implementation for call and many other network related operations. Uses SRV_AT to send AT commands
- SRV_SMS – Provides AT implementation for the SMS functionality. Uses SRV_AT to send AT commands
- SRV_REC – Maintains call recordings. Uses EMB_GSRV to read and write all call recordings and store them permanently
- SRV_KPD – Provides keyboard service to the MMI. Uses EMB_KPD to interact with the platform keypad
- SRV_AT – All other services uses this module to send AT commands and receive responses. SRV_AT uses EMB_AT to communicate with the platform
- SRV_PDU – This is used by SRV_SMS to decode and encode SMSs in PDU format
- SRV_LAYMAN – This service component controls the layouts on the screen
- SRV_PLOTTER – SRV_LAYMAN uses SRV_PLOTTER to create layouts on the display
- SRV_DISPLAY – This component controls the display elements on the screen
- SRV_BITMAPS – This service component manages all the bitmaps used in the MMI
- SRV_ICON – This is the drawing component of the status icons, mainly visible on the screen when idle

5.1.3 Data Containers

A data container UDO, or UDD, is used for data exchange between two or more other UDOs. It contains no objects or logic.

The UDOs that use UDDs, include an access on the UDD, through which they set the different field values in the UDD's message.

The UDDs used in the Wavecom sample MMI are:

- UDD_SMSDATA – Used for data exchange between HMI_SMS and SRV_SMS
- UDD_CALLS – Holds the data of all active calls in its message interface. It is updated only by SRV_CALL
- UDD_NET – Holds data of all available operators in a network
- UDD_REC – Holds call time and duration
- UDD_SET – Holds user settings
- UDD_SMS_ILIST – Holds the maximum possible number of SMS and the count of SMS actually stored in the list
- UDD_PDU – Used for data exchange between SRV_SMS and SRV_PDU. It also contains buffers, in which SRV_PDU keeps the temporary results of encoding and decoding PDU strings

5.1.4 Widgets

Widgets:

i. Provide basic user interface functionalities, such as the scroll bar and editing functionality
ii. Are easily reusable

Some of the frequently used widgets in the sample MMI are:

- WDG_ANIMATOR – This component provides animation services for the entire project. The animation data is retrieved by using a nested component, ANIM_DATA.UDO. Currently, the timing of the animation is hard coded in this widget
- WDG_SOFTKEY – This component handles two soft keys. It receives the labels to display and triggers exported events when some keys are pressed. It is the only component that monitors soft key events on the keypad. All other components monitor this widget's events when dealing with soft keys
- WDG_SCROLLBAR – This component provides two types of scrolling behavior; it scrolls the main menu and page mode lists (bitmap) and scrolls simple lists (text). The first mode displays a scroll bar element on the display and calculates its position according to the displayed list item. The second mode displays the number of displayed list items out of the total number of items in the list

- WDG_LIST – This component provides a navigation functionality list for other components in the project. It is initialized with items to display. The number of items it receives depends on the number of display items that are defined in the Layout Editor. Therefore, if the total number of list items is larger than the number of displayed items, the user must initialize the list with the next items when required
- WDG_EDIT_BOX – This component generates text on the display, inside the widget rectangle. It is used mainly by the editor (WMI_EDITOR) and also by other MMIs that uses it directly. WDG_EDIT_BOX has two generation orientations; from bottom right or from top left. It can generate different cursor types. It also has a special view-only mode. Each time the drawText function is called, the entire text is generated

5.1.5 Embedded Interface Components

Embedded interface components:

i. Provide an interface between the RapidPLUS™ components and the functions supplied by Wavecom

ii. There is usually (but not necessarily) a one-to-one relationship to functions with similar names to the IDE functions they call

iii. Any part of the embedded system, hardware or software that communicates with the RapidPLUS™ application must be represented by an embedded interface component

The embedded components provided in the Wavecom MMI are:

- EMB_KPD – Provides interface methods to communicate with the platform keypad
- EMB_AT – Provides an interface to send AT commands to the underlying AT modem provided by the platform and receives responses
- EMB_GSRV – Provides an interface for general services such as battery, audio, backlight, call recordings, charger etc.
- EMB_TEXTRES – Manages different language text strings used in the MMI

5.1.6 Reference MMI Set up

Couple of files are required to copy to the computer, in order to have the reference MMI to display the fonts properly.

- The reference MMI uses fonts provided by e-SIM. The Fonts files can be installed by copying from the \Reference_Design\libs\ESIM\fonts folder to the \Windows\Fonts folder
- The LangAPI.DLL file is copied from \Reference_Design\libs\ESIM\fd_dll folder to \Windows\System32 folder

5.2 MMI Integration in the Wavecom Open AT® Software

5.2.1 Basic Services Integration

This involves integration of the following components:

- Display
- Keypad
- Timer

5.2.1.1 Display

The display components in the MMI layer of the sample MMI architecture are responsible to handle the screen display.

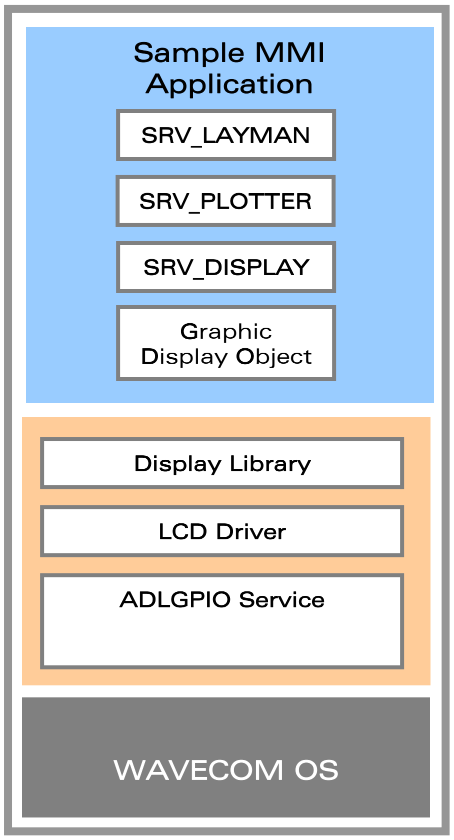


Figure 28: Display Integration

Each screen shot in the MMI conforms to one of the pre-created layouts. The layout development tool: 'Layout Editor' can be used to design and edit the layout elements used in the MMI. The HMI module that is active is responsible for managing and updating the display. All the HMI modules use the services from SRV_LAYMAN to define components and their appearance on the screen (size, position, style, etc.). For every layout that exists in the MMI, there exists an *initLO_<LayoutName>* method in SRV_LAYMAN. The MMI calls a suitable method to display a specific layout.

The *VideoDriver_BitBlit* method is responsible for the integration of the RapidPLUS™ GDO with the platform LCD. The platform LCD device driver is implemented using the Open AT® Software ADL API. The ADL GPIO Service interface and ADL Bus Service interface are used to implement the target display driver.

Every time the display changes, the MMI application, using the service layer components updates the display buffer maintained by the RapidPLUS™ GDO. The RapidPLUS™ GDO then calls *VideoDriver_BitBlit* to update the platform display.

5.2.1.2 Keypad

All MMI components use services of SRV_KEYPAD to provide the keypad functionality. SRV_KEYPAD provides a more convenient interface to the MMI by supplying events for each of the possible key events, including long press.

SRV_KEYPAD encapsulates the single instance of EMB_KPD that simulates the embedded keypad, and generates the keypad user events. EMB_KPD contains the interface between the hardware and the MMI keypad.

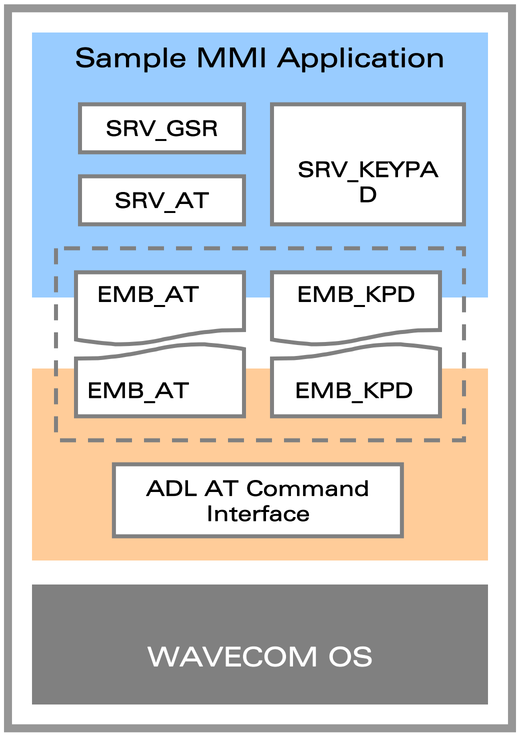


Figure 29: Keypad Integration

The ADL interface is used to capture keypad events from the hardware. The embedded application subscribes for keypad events using *AT+CMER* command. This command is used for mobile equipment event reporting. To capture the keypad events, the application handles unsolicited responses with *"+CKEV"*. Along with the key press event, this response contains the ID of the key pressed.

These key press events received from the AT are used to trigger EMB_KPD events. The key ID received in the AT response is used to set the key code property of EMB_KPD. EMB_KPD, in turn, updates SRV_KEYPAD accordingly.

SRV_KEYPAD uses services of SRV_GSRV to switch on the platform keypad backlight when any key is pressed. SRV_GSRV uses SRV_AT → EMB_AT to send the AT command to change the state of a specific GPIO that switches on the keypad backlight.

5.2.1.3 Timer

The ADL library method, *adl_tmrSubscribe,* is used to obtain the timer ticks from the hardware. Every time the ADL timer expires, MMI timers are updated using the RapidPLUS™ interface method *rpd_PrivUpdateTimer.* The RapidPLUS™ internally is responsible to update all timers used in various MMI components.

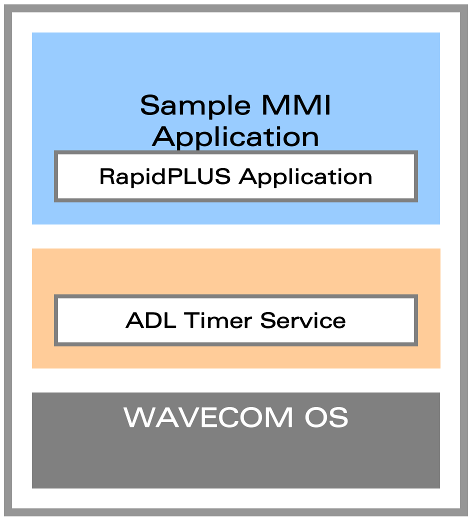


Figure 30: Timer Integration

The Method *rpd_PrivUpdateTimer* is implemented within the e-SIM micro kernel. This provides the interface to synchronize the RapidPLUS™ clock using the system clock. ADL library functions are used for this synchronization. The ADL timer function, *adl_tmrSubscribe* is used to subscribe a timer proc with the ADL. From within the timer proc, a call to *rpd_PrivUpdateTimer* is made. Every time the timer expires, the ADL passes the control to the timer proc which updates the RapidPLUS™ clock with the same time interval used in adl_tmrSubscribe. The Wavecom platform provides two units for the timer expiry. These are 18.5 ms and 100 ms.

5.2.2 Embedded AT Integration

The embedded AT component serves as the AT command channel UDI. Its sole purpose is to connect the AT service, SRV_AT, to the embedded system. The embedded AT component does not perform any processing on the AT command itself, it acts as a dispatcher for AT commands.

In RapidPLUS™, the embedded AT is implemented as EMB_AT.UXO. During code generation, this component is generated as a UDI, i.e., Interface only. The actual implementation of the interface is achieved by the Open AT® Software ADL APIs provided by Wavecom.

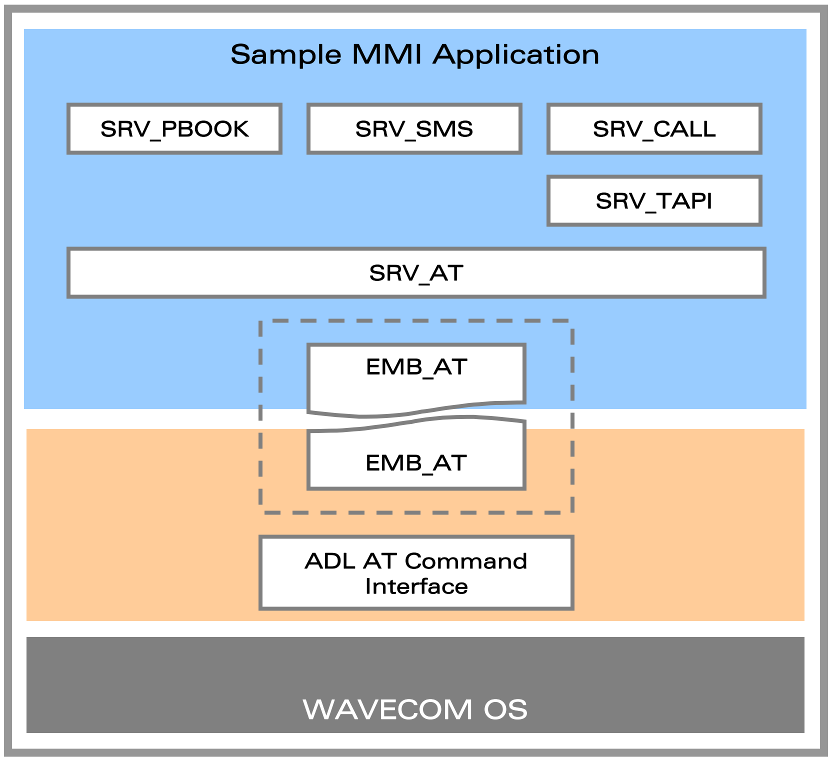


Figure 31: Embedded AT Integration

SRV_AT & EMB_AT, the RapidPLUS™ User Data Objects are responsible for AT integration of sample MMI over the Wavecom platform and actually is the most important aspect of the design.

SRV_AT

This component acts as an interface between application services and the underlying Wavecom layer. SRV_AT itself does not process anything from AT commands. It cannot parse the AT commands or responses passing through it. Its main purpose is to provide the service interface of the underlying EMB_AT module, which actually connects to the protocol stack to deliver AT commands and receive AT responses. All traffic to and from the stack must pass through this service, which operates as a single instance.

The following services are provided by SRV_AT:

- Registering unsolicited responses
- Sending commands
- Receiving responses

EMB_AT

This component is implemented as a RapidPLUS™ User Interface Object. Its sole purpose is to connect SRV_AT to the underlying protocol stack. It has almost same interface as SRV_AT which implies they present a one to one mapping. Since Phonebook, SMS and Calls are implemented using AT commands, EMB_AT serves as a single point of integration for these modules.

5.2.3 General Services Integration

The general services integration controls the integration of the following services:

- Battery
- Charger
- Backlight
- Clock
- Tone
- Melody
- Mute
- Volume
- Settings
- Call Records

AT commands and ADL APIs are used for the integration of these services.

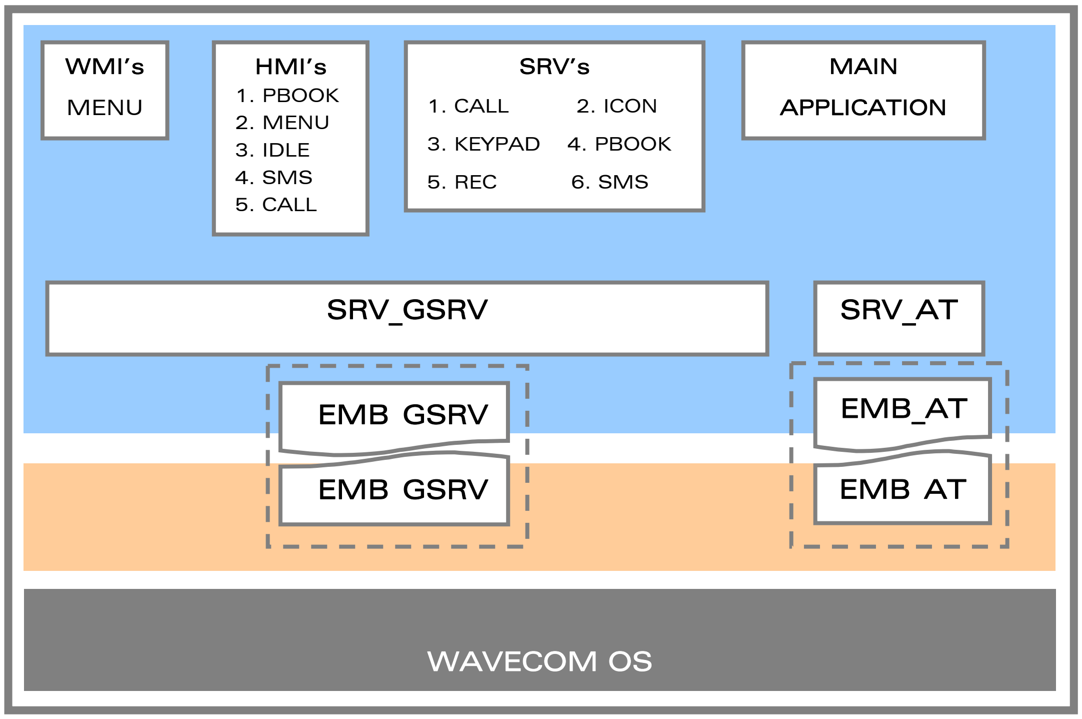


Figure 32: General Services Integration

SRV_GSRV

The general services component SRV_GSRV provides an interface to the Service and MMI components above it to interface hardware services such as battery, charger, backlight, clock etc. SRV_GSRV uses the services of SRV_AT to send various different AT commands to the platform.

EMB_GSRV

Apart from other services, this component controls reading and writing on the flash memory. Flash read/write operations are required to read/write User Settings and Call Records. Flash read/write operations are performed using the ADL flash read/write APIs provided by the Open AT® Software. SVR_GSRV calls the UDI component EMB_GSRV, which calls the ADL flash APIs for the implementation of its interface.

5.2.4 Service Component Integration

Service component integration includes integration of the following services:

- Phonebook
- SMS
- Calls
- Call Recordings
- Network Management

As discussed in section 5.2.2, the integration of these services depends mainly on the integration of the Embedded AT Interface and the Embedded General Service Interface.

Referring to Figure 31, the Phonebook, SMS and Call Services internally use AT implementation. The required AT commands are sent to the AT service component, SRV_AT, which passes the AT command to the embedded AT component. The embedded AT component, EMB_AT, once integrated using the Wavecom Open AT® Software ADL APIs, will send the AT command to the underlying Wavecom AT parser, ATI. The AT response is redirected to respective modules using the same path. EMB_AT will receive all the responses from the underlying layer. SRV_AT will receive the responses from EMB_AT in the form of a message and it will pass it on to the services connected to SRV_AT.

Call Recordings use EMB_GSRV (refer to Figure 32) to read and write recordings to the flash memory. Since SRV_GSRV provides general purpose services, it is used by other components such as phonebook, SMS and also calls.

5.2.4.1 Phonebook

The sample MMI Phonebook uses the SRV_AT/EMB_AT module to communicate with the Wavecom OS. The Phonebook implementation is done with the following modules (refer to Figure 33):

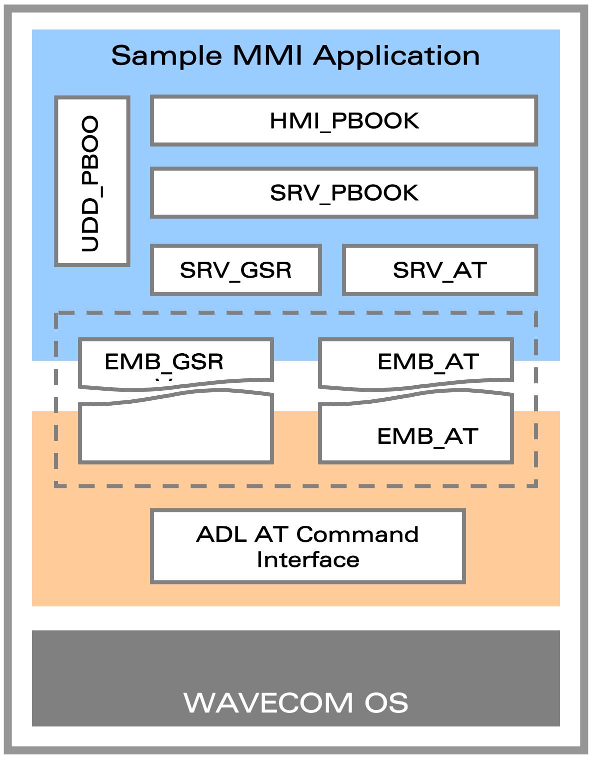


Figure 33: Phonebook Integration

HMI_PBOOK

As presented above, HMI modules provide the human machine interface and are responsible for updating the display when they are active. HMI_PBOOK is called by HMI_IDLE when the phonebook option is selected. HMI_PBOOK handles the user interface for the phonebook functionality. It can be started in two modes, defined in MODE_Cs:

- Browsing mode - the entries list of the current phonebook is displayed
- Menu mode - the phone book menu is opened (pressing the right soft key when in idle position)

HMI_PBOOK uses the services of SRV_PBOOK for all operations related to the Phonebook.

SRV_PBOOK

SRV_PBOOK provides the interface used by HMI_PBOOK for all processes related to the Phonebook. This service component is responsible for implementing the phonebook functionality in terms of AT commands. According to the option selected in the menu, SRV _AT is responsible for selecting the type of phonebook to use. Selection is done using AT commands only. Currently, both SIM and memory phonebooks are supported, but they cannot be used simultaneously. As presented above, this module is used for services. SRV_AT sends the commands to the underlying protocol stack. Apart from sending the commands, this module monitors all types of responses, both AT command responses and unsolicited responses, from the network. It can filter and act on the responses valid for Phonebook.

UDD_PBOOK

UDD_PBOOK is implemented as a RapidPLUS™ Data Container Object. Being a data container, it does not provide any interface except to insert and retrieve data. UDD_PBOOK is used by SRV_PBOOK to store the date related to the Phonebook.

SRV_GSRV

SRV_PBOOK uses the services of SRV_GSRV to save speed dial entries in phone settings. SRV_GSRV internally uses EMB_GSRV for implementation.

5.2.4.2 SMS

SMS Services are also integrated with the Wavecom platform by using SRV_AT/EMB_AT module. The sections below briefly explain the RapidPLUS™ modules used by SMS. (refer to Figure 34).

Note	In contrast to the way other AT commands are sent, SMS text is sent using the adl API adl_atCmdSendText in the emb_at.c file.

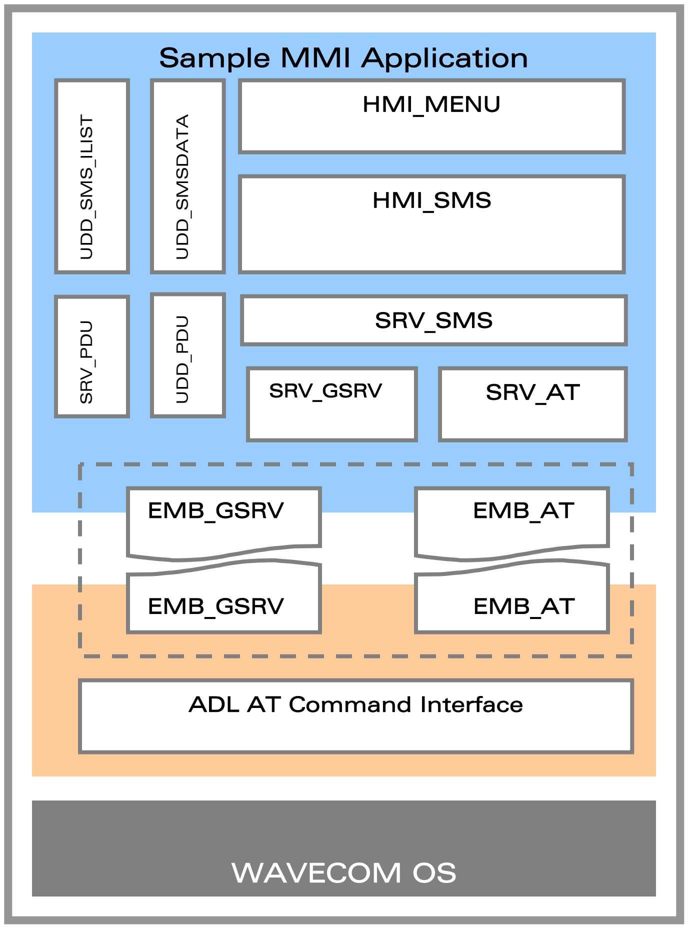


Figure 34: SMS Integration

HMI_MENU

This is an HMI module responsible for providing the Menu display. HMI_MENU handles the menus starting from the main menu. When the user selects an SMS option from the menu, HMI_MENU launches HMI_SMS, which in turn is responsible for showing the SMS menu.

HMI_SMS

This module implements the MMI for the Messages Menu and is responsible for updating the display when the SMS menu is active. HMI_SMS gets/sets messages data from SRV_SMS using UDD_SMS_DATA, a data container that holds data in a structure.

UDD_SMSDATA

This is a data holder component dedicated to data exchange between HMI_SMS and SRV_SMS. SRV_SMS gets the message from the underlying layer and updates it in this data container from where HMI_SMS takes the inputs.

SRV_SMS

This service component transforms requests from HMI_SMS into AT command form. It sends, stores, and indicates incoming short messages. Most of the operations that SRV_SMS performs are asynchronous; therefore, a following operation cannot be requested before Ready_evt is triggered. SRV_SMS uses SRV_AT module to send and receive the SMSs to and from the network. Apart from interacting with HMI_SMS and UDD_SMSDATA discussed above, it stores the SMS list in the UDD_SMS_ILIST which acts as a temporary storage of SMS data. SRV_SMS also interacts with SRV_PDU to convert and decode a message in PDU format.

UDD_SMSILIST

This data container is used to store the list and the content of the SMS received. This is used by SRV_SMS.

SRV_PDU

SRV_PDU provides an interface for PDU creation and decoding when sending and receiving SMS. It is solely used by SRV_SMS. SRV_PDU uses services of SRV_GSRV for the methods related to Unicode character conversion.

UDD_PDU

This component is dedicated to data exchange between SRV_SMS and SRV_PDU. It also contains buffers, in which SRV_PDU keeps the temporary results of encoding and decoding PDU strings.

5.2.4.3 Call Control

Similar to Phonebook and SMS, Call Control also uses services of the SRV_AT/EMB_AT module to communicate with the Wavecom OS. The sections below briefly detail the RapidPLUS™ modules used for Call Control Services (refer to Figure 35).

HMI_CALL

This component is responsible for the MMI during active calls. It manages dialing, outgoing calls, active calls, conference calls (multi-party), on-hold calls, and swapping between a held call and an active call. It also manages all in-call options. This module is activated by HMI_IDLE when the user starts dialing a number or when HMI_IDLE receives an incoming call event from SRV_CALL. HMI_CALL, when active, is responsible for showing all the information of the display. This module communicates with SRV_CALL for all the information to show on the display.

SRV_CALL

This service component is responsible for managing all active calls. It manages calls data using UDD_CALLS. SRV_CALL is mainly used by HMI_CALL and also by other HMI modules that initiate the calls. As seen in the design, the call functionality is split into two modules, SRV_CALL and SRV_TAPI. This design keeps the AT implementation out of SRV_CALL and avoids complicating the SRV_CALL module. Thus, SRV_CALL module acts as a service module for the display component, HMI_CALL and internally it communicates with the network using the interface provided by SRV_TAPI. SRV_TAPI is the module that actually implements AT commands related to call handling.

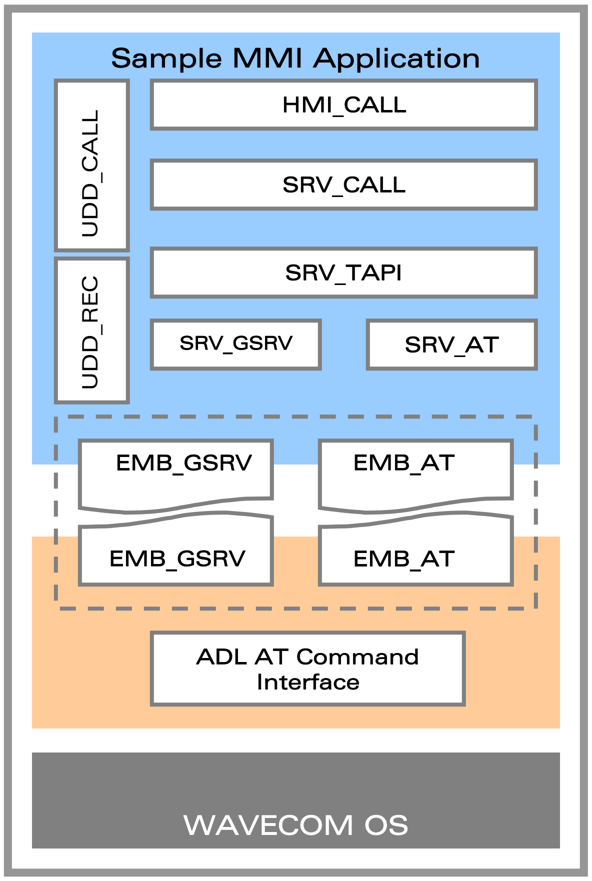


Figure 35: Call Control Integration

UDD_REC

This component is used to transfer data of a single call record. There are two instances of UDD_REC in the application:

- To send call information from SRV_REC to HMI_REC
- To send call Information from SRV_CALL to SRV_REC

Apart from being used by SRV_CALL, SRV_REC uses this module to show the call records when the Records menu is active.

UDD_CALL

This component holds the data of all active calls in its message interface. It is updated only by SRV_CALL.

SRV_TAPI

This service component provides AT implementation of several network related functions:

- Modem initialization functions
- Dialing and call handling functions
- PIN code functions
- Network services function

SRV_TAPI uses the interface provided by SRV_AT to send AT commands and receive the different types of responses. SRV_TAPI can process several kinds of AT responses related to Call handling, network services etc., and then inform the upper services as well as HMI layer about it.

SRV_GSRV

Apart from reading and writing settings on the flash memory, SRV_GSRV is also responsible for providing updated date and time to the HMI_IDLE and HMI_MENU components. SRV_GSRV uses SRV_AT to send the command to get the current date and time. SRV_GSRV uses EMB_GSRV to save the settings to flash.

5.2.4.4 Call Records

As presented in section 5.2.3, Call Recordings are implemented using flash read/write methods provided by SRV_GSRV. SRV_GSRV internally calls EMB_GSRV for the implementation of the flash read/write functions. EMB_GSRV implements these methods using Open AT® Software ADL flash API. The sections below briefly explain the RapidPLUS™ modules used for Call Recordings. (refer to Figure 36)

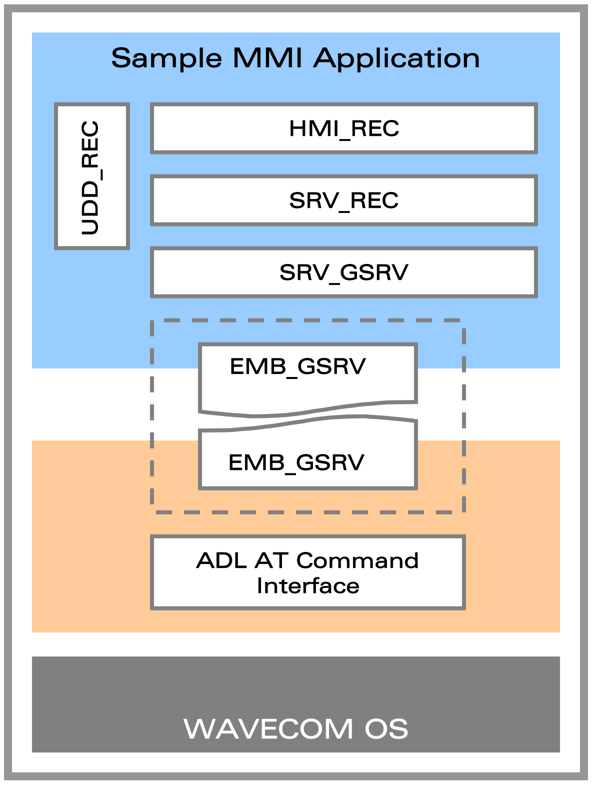


Figure 36: Call Records Integration

HMI_REC

This component handles the user interface for the Call Recording functionality. It handles all Call Recording interaction and all menu functionality under the Call Records menu. The call recording data is held in SRV_REC. HMI_REC uses UDD_REC to retrieve this data.

SRV_REC

SRV_REC provides the interface which allows HMI_REC to process all features related to call records. This service component provides data to the call records menu in HMI_REC. This data includes:

- The number, date, and time of the 10 most recent incoming, outgoing, and missed calls
- Total time of all received calls
- Total time of all calls sent
- Last call time

SRV_REC receives the data from SRV_CALL, sorts the data by time and date, and saves the data using SRV_GSRV. Call recording data is not held by this service component. The service only holds indexes and manages the data that is stored in the flash memory.

SRV_GSRV

SRV_GSRV functions are called by SRV_REC to read and save call recordings. This component further calls EMB_GSRV functions. EMB_GSRV is used to read and write data to the flash memory. For the implementation of this interface, the Open AT® Software flash API will be used.

UDD_REC

UDD_REC is implemented as a RapidPLUS™ data container object, so it only holds the data, but does not provide any interface to read and write data.

UDD_REC holds call related data. It is used to transfer data of a single call recording.

There are two instances of UDD_REC in the application:

- To send call information from SRV_REC to HMI_REC
- To send call Information from SRV_CALL to SRV_REC

5.2.4.5 Network Management

Network Management integration (refer to Figure 37) is similar to what has been detailed above on Call Control Integration. The network related functionality is implemented in the SRV_TAPI component which is also responsible for the call control functionality (refer to section 1.1.1.1).

HMI_IDLE

HMI_IDLE is responsible for showing all the network related indications on the idle screen. This includes network strength indicator as well as operator name. HMI_IDLE calls methods on SRV_TAPI to retrieve this information and then updates the idle screen accordingly.

HMI_MENU

This is an HMI module responsible for providing the menu display. HMI_MENU handles the menus starting from the main menu. All the menus related to Network management and network settings are handled by HMI_MENU. For all network related menu options such as set/cancel call diverts, set/cancel call barring, set/cancel call waiting, PIN On/Off/change, network selection, HMI_MENU calls SRV_TAPI component. For reading/writing all these settings in the flash memory, the SRV_GSRV component is used.

SRV_TAPI

This service component provides AT implementation of several network related functions:

- Modem initialization functions – under init mode
- Dialing and call handling functions

- PIN code functions
- Network services function

SRV_CALL uses interface methods provided by SRV_TAPI to communicate with the network. SRV_TAPI uses the interface provided by SRV_AT to send AT commands and receive different types of responses. SRV_TAPI can process several kinds of AT responses related to call handling, network services etc., and then inform the upper services as well as the HMI layer about it.

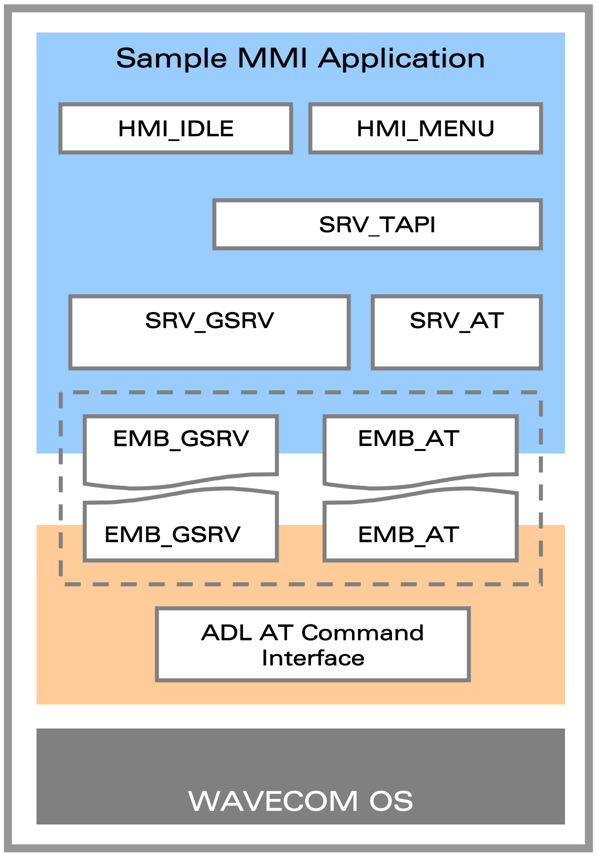


Figure 37: Network Management Integration

5.2.5 Embedded Component Integration

Integration of all these components is performed through AT commands only. SRV_GSRV contains the logic of sending the corresponding AT commands and receiving responses for the integration of these components. SRV_GSRV hides the details of AT commands and provides direct interface methods that can be used to interact with any of these components.

5.2.5.1 Battery

The battery implementation is achieved using the AT commands in the SRV_GSRV component. In order to implement the check status of battery, there is a mode in the SRV_GSRV component which is activated whenever the main application calls the SRV_GSRV component to check the battery status. The AT command, AT+CIND, gets the battery status as its response. This response is then captured into a local variable and displayed on the target screen. There are five battery levels (0-5) that can be returned as response.

5.2.5.2 Synthesizer

The Melody synthesizer is implemented in the SRV_GSRV user object. It is implemented using the AT command AT+CRMP. Various melodies available on the target can be played using this AT command. The main application provides the index of the melody to play.

Similarly, there is a separate mode to play the tones on the press of keys. This is implemented in the tone mode using the AT command AT+WDTMF. The parameters for this command are provided by the main application component.

5.2.5.3 Backlight

Backlight implementation is done in the SRV_GSRV component. Similar to the other embedded components, it is implemented using the AT commands. The command used to start and stop the backlight is AT+WIOM.

5.2.5.4 Charger

Charger implementation is provided in the main application startup file oat_appli.c. The charger and charger events are subscribed in this file using the adl APIs,

`adl_atUnSoSubscribe, adl_ioEventSubscribe, adl_ioSubscribe.`

Once the charger is plugged, a charger plugged event is triggered. This causes the control of the application in the SRV_GSRV component to go to the StartCharging mode and vice versa when the charger is unplugged.

5.3 Sample Use Case - Outgoing Call Set up

When the user starts dialing a number on the idle screen, the HMI_IDLE calls the *startDialing_string* method on HMI_CALL component to handle the dialed number on the screen. HMI_CALL, after checking the emergency number, waits for the press of SEND key.

On the press of the SEND key, HMI_CALL extracts the number from the editor widget handling number input and calls the SRV_CALL:*dialTo* method with the dialed number as parameter for the method. The SRV_CALL:*dialTo(..)* method sets the CallNumber_Str property on the SRV_TAPI and then calls the *tapi_dialTo* method on the SRV_TAPI module (refer to Figure 38).

SRV_TAPI is responsible for the conversion of the dialed number request to an AT command and sends this command to SRV_AT for execution. To send AT commands and receive the responses synchronously, SRV_TAPI implements the AT dispatcher.

The AT dispatcher is implemented as a concurrent node to the *"Functions"* node in SRV_TAPI. When a command is sent to the AT dispatcher, it goes into the *"WaitForDispatcher"* state until SRV_AT is *ready*. When *ready*, it goes to the *"TryToSend"* state from where it calls a SendAT method on SRV_AT. If it returns the error code as *busy*, it goes back to the *"WaitForDispatcher"* state and repeats the process until the error code is *not equal to busy*. Once the command is sent, the AT dispatcher goes to the *"WaitForResponse"* state.

SRV_AT calls *sendAT* on EMB_AT. Since EMB_AT is generated as a UDI, it holds the key to the entire AT integration. The sendAT maps to c-code function which calls the Open AT® Software ADL API to send the AT Command, *atdXXXXXXXXXX;*, to the Wavecom AT parser for execution.

On receiving the response from the underlying AT parser, EMB_AT informs SRV_AT using the *send message* interface. EMB_AT encapsulates the response in a message structure and sends it to SRV_AT. SRV_AT extracts the response from the message and triggers the response events, *response_Evt* and *final_Evt* to inform SRV_TAPI. In this case, SRV_TAPI waits for the *final_Evt* response event. On receiving the indication for *final_Evt*, the AT dispatcher in SRV_TAPI goes back to idle state and SRV_TAPI sends the *OK_Ev* response to SRV_CALL.

Apart from responses to AT commands, the AT parser also sends some unsolicited responses. When the call connects, the underlying AT parser sends an unsolicited response with string "%CPI". To get the indication on call connection, SRV_TAPI has subscribed for this message. On receiving an unsolicited response with "%CPI", it sends CallConnected_Ev to SRV_CALL. SRV_CALL passes the same event to HMI_CALL to enable any changes on the display, if needed.

SRV_CALL maintains the list of active calls in the data container, UDD_CALLS. On getting *CallConnected_Ev* from SRV_TAPI, it updates the list and sends *CallStatusUpdate_Ev* to HMI_CALL to update the list of active calls on the display.

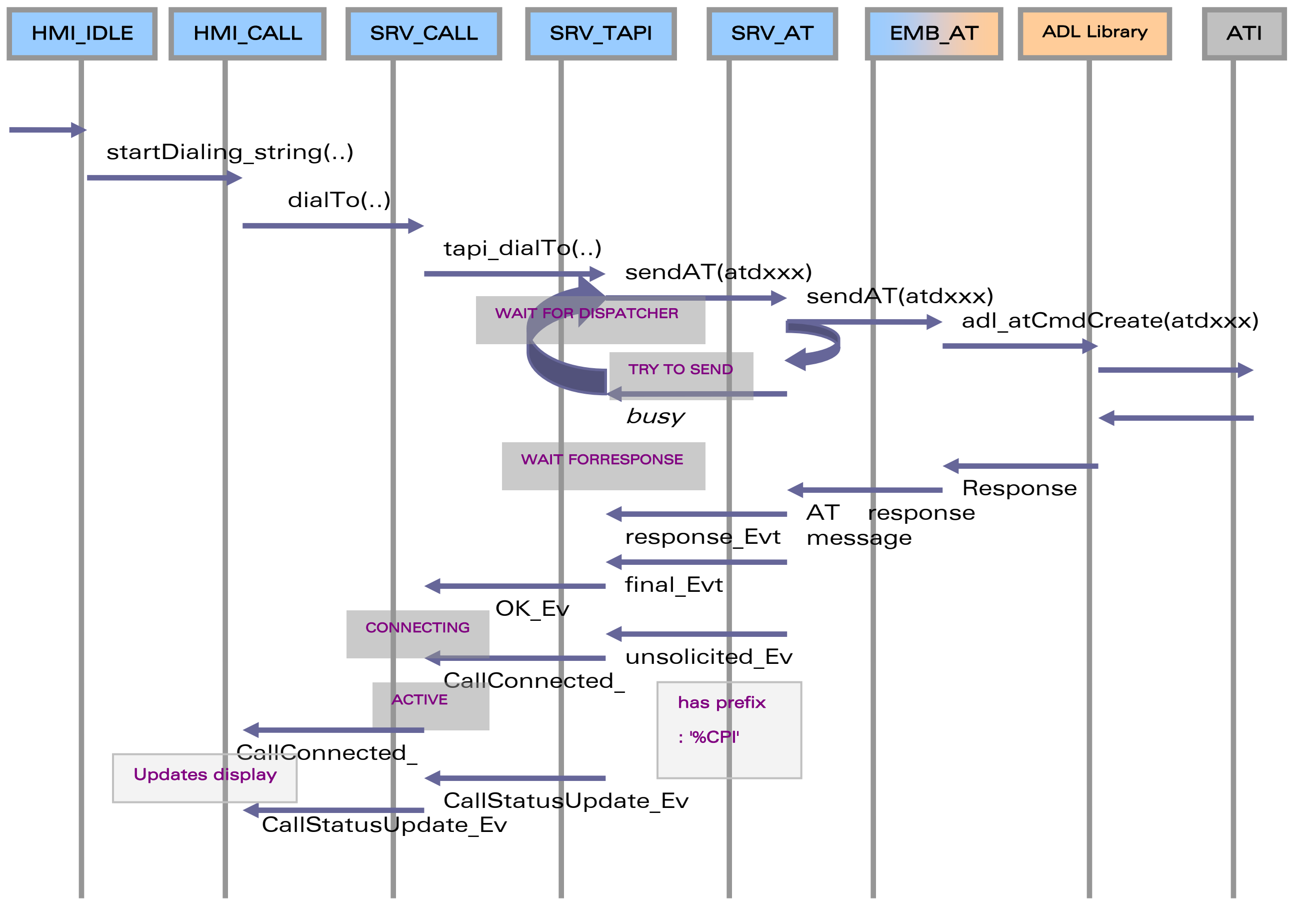


Figure 38: Sample Use case Realization – Outgoing Call

5.4 Sample MMI Menu Structure

Sample MMI consists of various menus and submenus. The menu structure is specified in the textresource.xls file. A snapshot of the main menu for the sample application is shown in the image Figure 39.

In order to add either a menu or submenu to the existing menu, the developer can add a row in the excel file in the level required. E.g. a user wants to add another submenu say "NewMenu" to the main menu. This menu can be added by inserting another row in the spreadsheet and adding a menu in the second column.

After adding the menu, the user must save the excel file and then generate the Text & Menu Generator macro. This updates the necessary user objects in the application and the new menu is shown in the application.

Level1	Level2	Level3
Main menu		
	Messages	
		Inbox
		Outbox
		Write messages
		Message settings
		Voice messages
	Call records	
		Missed calls
		Received calls
		Dialed numbers
		Call time
	Settings	
		Alarm clock
		Clock
		Phone settings
		Sound settings
		Security settings
	Network	
		Call diverting
		Call barring
		Network selection
		Caller ID
	Lock keypad	
	Phonebook	

Figure 39: Sample MMI Menu Structure

5.5 Sample MMI Workspace Architecture

The highest directory level in the structure is Reference_Design. Under Reference_Design, the Open AT® GTi workspace architecture is divided into three main directories:

- **inc** - This directory contains the Wavecom header files for the LCD device and display drivers
- **src** - This directory contains the Wavecom source files for the LCD device and display drivers and also the main source file of the Open AT® GTi project
- **libs** – This directory contains three main folders, ESIM, Device, and Display. The ESIM contains the customized code for the application; device and display include the code for the custom device and display of LCD drive

Note	Before starting with the application development, make sure that the GTi folder containing the libraries for the Open AT® GTi development is copied to the **Plug-in** subfolder in the Open AT® Software Framework folder.

5.5.1 Overall Directory Structure

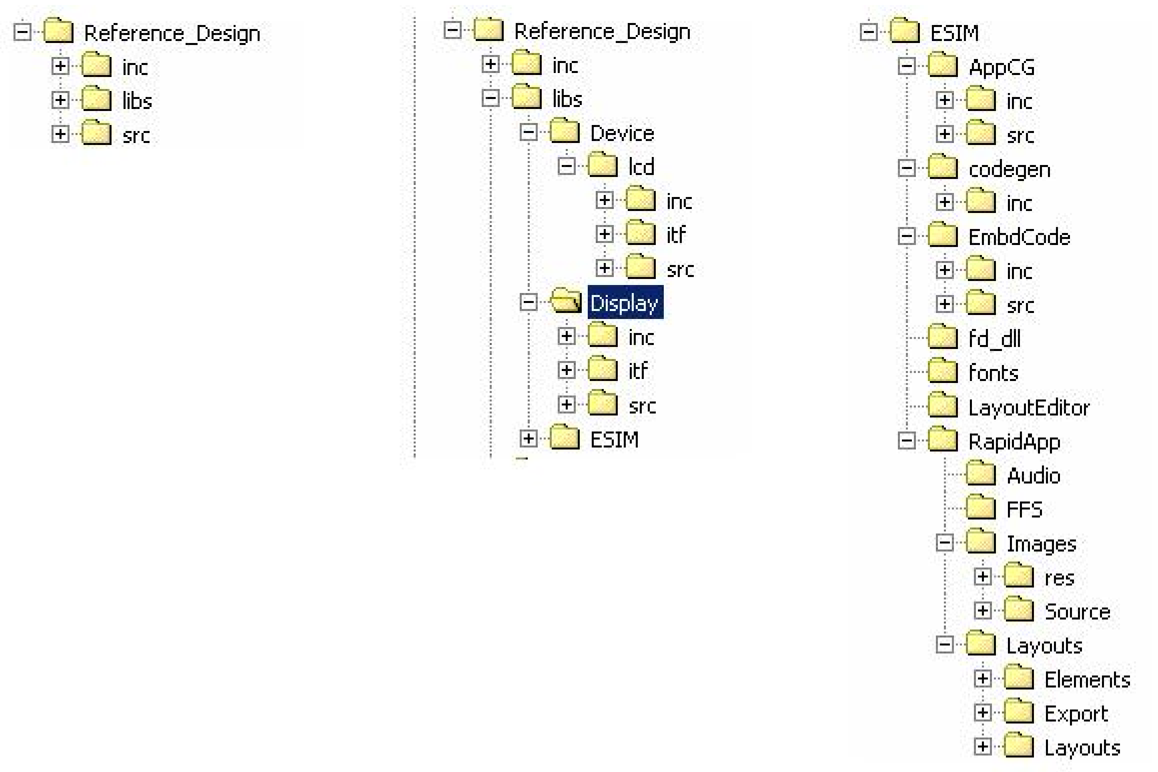


Figure 40: Overall Directory Structure

5.5.2 inc - Folder Containing Device and Display Driver Header Files

The inc folder contains the header files related to the Wavecom specific LCD display and device drivers.

5.5.3 src –Folder Containing Device and Display Source Files

The src folder contains the source files related to the Wavecom specific LCD display and device drivers.

It also contains the main application file which contains the entry point of the application.

Note	**Src** folder contains the ***oat_appli.c*** which includes the start point *adl_main* for the Open AT® Software embedded application

All the projects contained in the Reference_Design.dsw have two variants: ***<ProjectName>.dsp*** and ***<ProjectName>_rte.dsp***. <ProjectName>.dsp contains the files required for the target build (ARM build) of the application and <ProjectName>_rte.dsp contains the files required for compiling the build using MS Visual Studio 6.0. The rte build is used for debugging the entire application. The projects included in Reference_Design.dsw are:

- Reference_Design.dsp
- Reference_Design_rte.dsp
- lcd.dsp
- lcd_rte.dsp
- display.dsp
- display_rte.dsp
- ESIM.dsp
- ESIM_rte.dsp

To make the target build, the Reference_Design.dsp project must be compiled and must detail dependencies set for all other projects. Reference_Design.dsp uses the ***Reference_Design.mak*** file to compile the build using ARM. To create the debug build, Reference_Design_rte.dsp must be compiled.

The target build binary is generated in.the .Reference_Design**ads\out** folder. An out folder is created when the target build is compiled for the first time. The Debug build is generated in the **Reference_Design\rte\vc6\debug** folder. The rte folder and all its subfolders are created when the debug build is compiled for the first time.

libs: The libs folder contains the following folders:

- Device
- Display
- ESIM

Device: The device folder contains an LCD folder which further contains the following folders:

- inc
- itf
- src

Note	▪ The **Device** folder contains subfolders with drivers for the various devices used by the application. Currently, it only contains the **lcd** folder for the lcd driver files ▪ The **lcd\src, lcd\inc and lcd\itf** folders contain required the .c and .h files requested for the lcd driver to work. The LCD driver must be written using the Open AT® Software ADL APIs only

The project files required to open this project, lcd.dsp and lcd_rte.dsp exist in the device\lcd folder. The lcd.dsp file uses the ***lcd.mak*** file kept in the same folder to make the target build. The lcd.dsp and lcd_rte.dsp files are already included in the Reference_Design workspace: Reference_Design.dsw.

Display: The display folder contains the following folders:

- inc
- itf
- src

Note	▪ The **Display** folder contains the files required for the display module. The Display module provides a layer of glue between the MMI application and the underlying LCD driver ▪ The **src, inc and itf** folders contain the .c and .h files required for the implementation of the display module

The project files required to open this project, display.dsp and display_rte.dsp, are present in the display folder. The display.dsp file uses the ***display.mak*** file kept in the same folder to make the target build. The display.dsp and display_rte.dsp files are already included in the reference design workspace: Reference_Design.dsw.

5.5.4 ESIM Specific Code Directory

The **ESIM** folder includes:

RapidApp: This folder contains the following folders:

- Audio
- Images
- Layouts
- FFS

Note	▪ All the MMI files containing RapidPLUS™ objects are placed in the RapidApp folder. All these files are xml files. MainApp.RXD is the project file which can be used to open the project in RapidPLUS™ ▪ The Audio folder contains all the midi files used by the MMI ▪ The Images folder contains all the graphics used by the sample MMI ▪ The Layouts folder contains the information related to the layouts used in the MMI ▪ The FFS folder contains a phonebook data file and phone memory data file. These files are of no relevance for the target build, but are used in the simulation build

appCG: The appCG folder contains the following folders:

- inc
- src

Note	▪ The appCG folder contains the code generated from the sample MMI application ▪ The RapidPLUS™ code generator generates .c and .h files in the Reference_Design\libs\ESIM \appCG folder. The Source and Include files are separated using the post build script **post-build.bat** kept in the same folder. The Source and Include files are separated in the folders appCG\src and appCG\inc respectively

CodeGen: The CodeGen folder contains only the **inc** folder.

Note	▪ The CodeGen\inc contains the .h files required by RapidPLUS™ for code generation

Embdcode: The Embdcode folder contains the following folders:

- inc
- src

Note	▪ The Embedded code folder contains the .c and .h files required by the RapidPLUS™ for code generation. This folder also contains the .c and .h files generated for language strings

fd_dll and lib: These folders contain dll files and lib files required by e-SIM RapidPLUS™.

fonts: This folder contains the fonts used by the Sample MMI application.

LayoutEditor: This contains a tool, 'Layout Editor', used to modify screen layouts. The process to modify the screen layouts is detailed in section 5.8.

ads: This folder contains the make folder containing all make files required to build the generated code.

5.6 Build and Deployment Procedure

The Build procedure involves the following steps:

i. Generating code from MMI xml files using e-SIM RapidPLUS™.

ii. Building the code using the ARM compiler.

iii. Downloading the image to the target platform using the 1K Xmodem protocol in the HyperTerminal.

5.6.1 Code Generation Using RapidPLUS™

The steps for RapidPLUS™ Code Generation are:

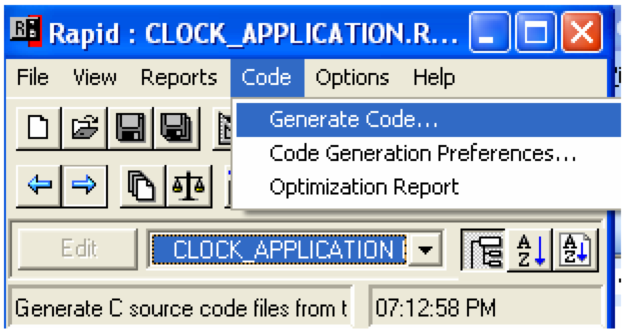
i. Open the Wavecom Reference MMI workspace: Reference_Design\libs\ESIM \RapidApp\mainapp.rxd using the RapidPLUS™.

ii. Select the MAINAPP project from the list of projects in the drop-down list.

iii. On the Menu bar, open the generate code dialog from the Code → Generate Code menu shown in Figure 41.

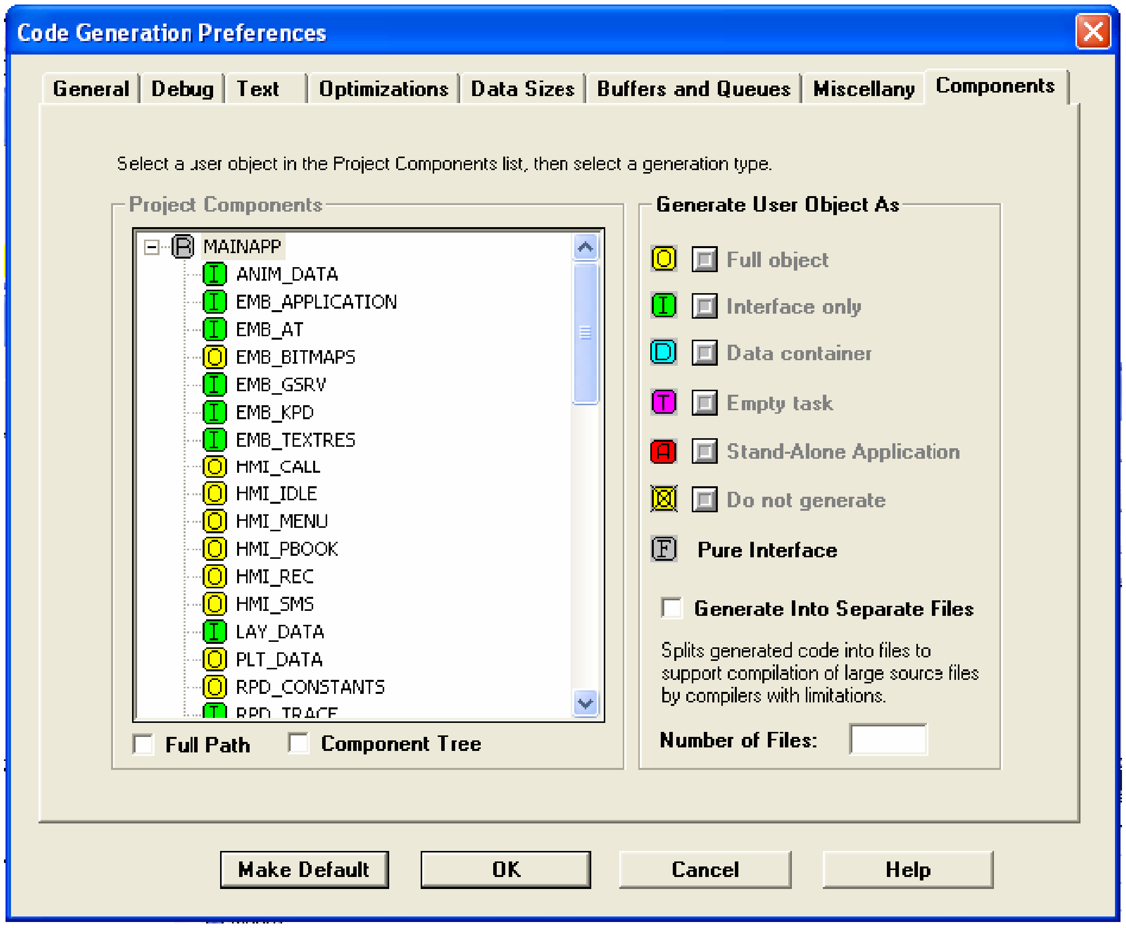
iv. As shown in Figure 42, the user can change various preferences for code generation such as the user objects that must be generated as interfaces only and those that must be generated either as full objects or data objects. Other settings such as application queue size and path for the .c and .h files are specified in the preferences section.

v. Once the user clicks on the 'Start' button to generate code, the code generation status dialog box appears on the screen and the percentage completion progress bar is displayed on the screen as shown in Figure 42.



Generate C source code files from t
07:12:58 PM

Figure 41: Code Generation Menu



Select a user object in the Project Components list, then select a generation type.

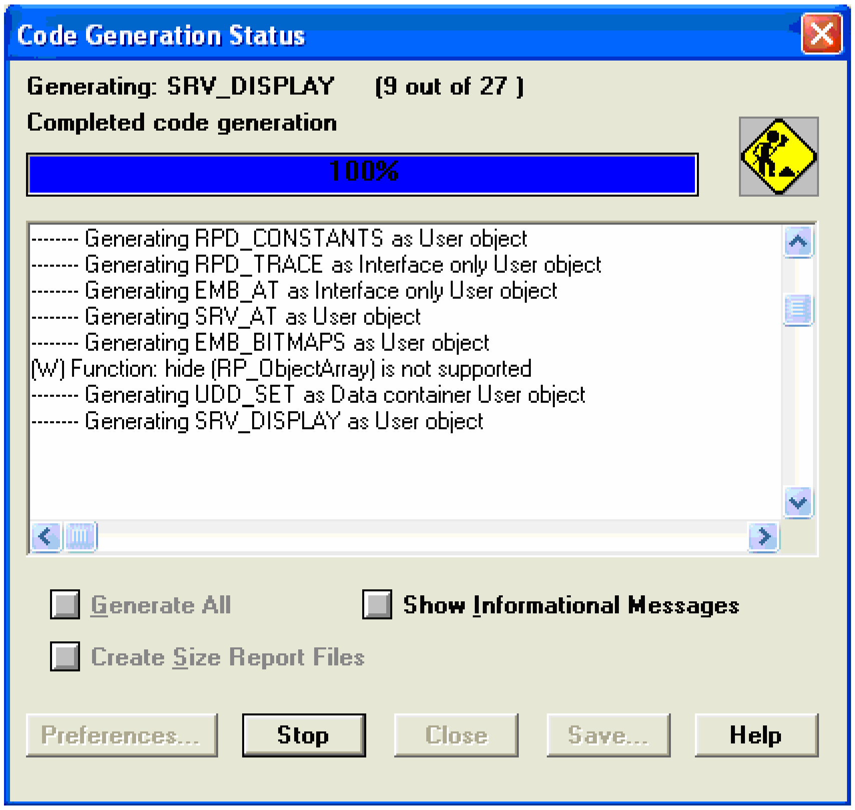
MAINAPP
- ANIM_DATA
- EMB_APPLICATION
- EMB_AT
- EMB_BITMAPS
- EMB_GSRV
- EMB_KPD
- EMB_TEXTRES
- HMI_CALL
- HMI_IDLE
- HMI_MENU
- HMI_PBOOK
- HMI_REC
- HMI_SMS
- LAY_DATA
- PLT_DATA
- RPD_CONSTANTS
- RPD_TRACE

Generate Into Separate Files

Splits generated code into files to support compilation of large source files by compilers with limitations.

Number of Files:

Figure 42: Code Generation - Preferences



Generating: SRV_DISPLAY (9 out of 27)
Completed code generation
100%

```
-------- Generating RPD_CONSTANTS as User object
-------- Generating RPD_TRACE as Interface only User object
-------- Generating EMB_AT as Interface only User object
-------- Generating SRV_AT as User object
-------- Generating EMB_BITMAPS as User object
(W) Function: hide (RP_ObjectArray) is not supported
-------- Generating UDD_SET as Data container User object
-------- Generating SRV_DISPLAY as User object
```

Figure 43: Code Generation - In progress

5.6.2 Build using ARM Compiler

The steps to make a target build are:

i. Open the project workspace using MS Visual Studio.

ii. Make a default project: Reference_Design.

iii. From the Menu, go to Build → Rebuild All. This builds the target build using the make file Reference_Design.mak kept in the same folder.

iv. Once the structure is completed, the image file 'ads_Reference_Design.wpb.dwl', is created in the ...\ads\out folder.

Note	The ARM compiler used is ARM® Developer Suite (ADS) Version 1.2 http://www.arm.com/support/downloads/ads12.html Build 837 is recommended for new Q24 series. Build 848 is recommended for Q2686H and Q2687H.

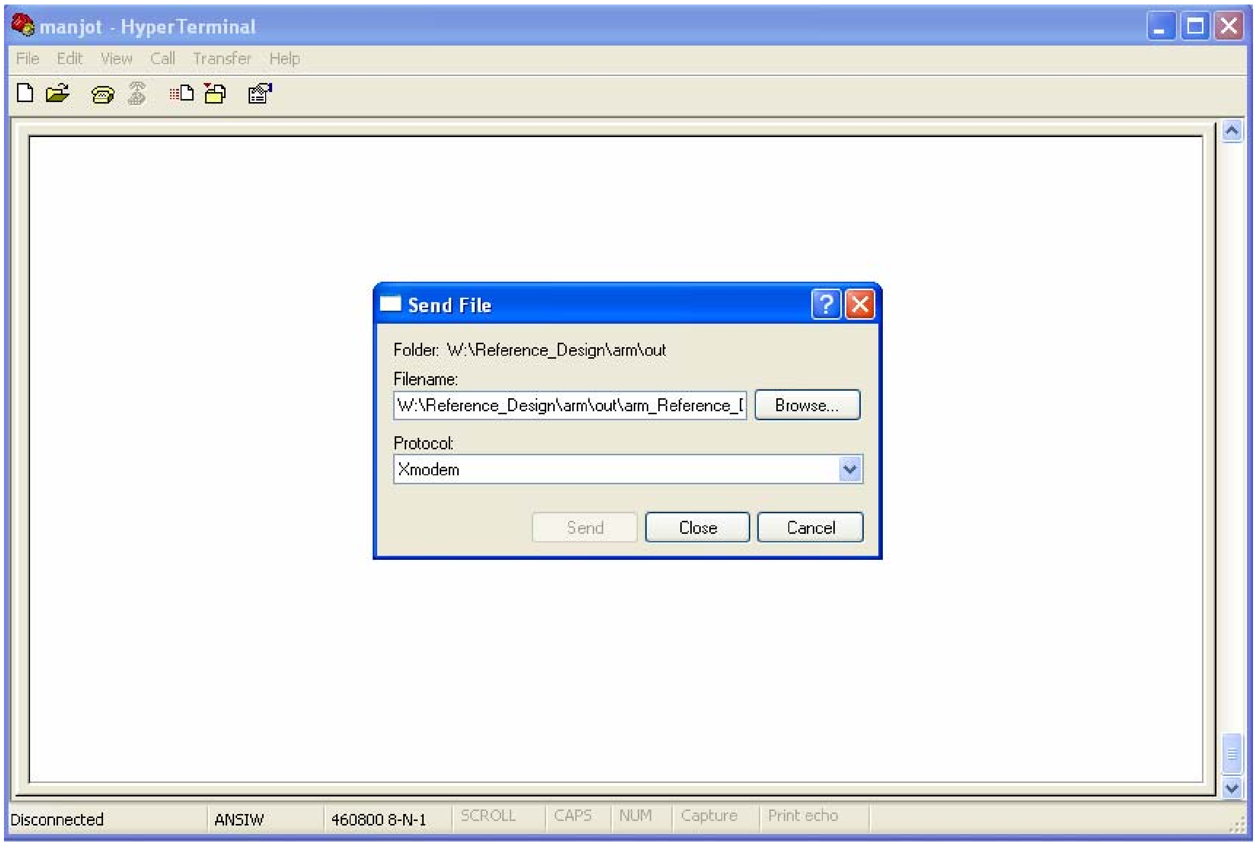
5.6.3 Download the Build on Target Platform

The steps to download the target build are:

i. Open the Hyperterminal application and use the AT+WDWL command to set the modem in download mode. Prior to the download of the application, use AT command (AT+WOPEN=6) to check if the Open AT@ application space size is large enough for the Open AT@ GTi application. The application space size can be configured with AT command (AT+WOPEN=6, <A&D size>). A "Fault 04" error will return if application space size was too small for the application after download. Erasing the application with DWLWin is the only way to recover from this error.

ii. Select the transfer protocol to 1K Xmodem.

iii. Browse to the folder Reference_Design\ads\out and select the image file and start downloading using the send button as shown in Figure 44.



Folder: W:\Reference_Design\arm\out

Filename: W:\Reference_Design\arm\out\arm_Reference_[

Protocol: Xmodem

Disconnected | ANSIW | 460800 8-N-1

Figure 44: HyperTerminal – Build Download

Note	The application can also be downloaded over the air. But there are restrictions on the application size which can be downloaded.

5.7 MMI Customization

The Open AT® GTi provides a wide range of possibilities for the customization of the MMI to suit market needs with a very short time to market. The Wavecom MMI can be customized using e-SIM and Wavecom MMI customization tools. With these tools, MMI developers can change graphics, ring tones and other multimedia content. They can also change screen sizes and layouts to provide a user friendly MMI.

The elements available for MMI customization are:

- Screen sizes and layout
- Graphics, Animations, Ring tones and other multimedia content
- Multi-lingual support to ensure a wider market for the MMI
- Feature enhancements to existing service components such as Phonebook, SMS, Call Control and Call Recordings
- Platform specific applications such as a Melody Composer, Bitmap Editor, Currency Converter, World Clock etc.
- New and exciting games can be provided and supported on the specific platform

Any changes in the MMI involve some basic steps to ensure that MMI changes get reflected on the target platform:

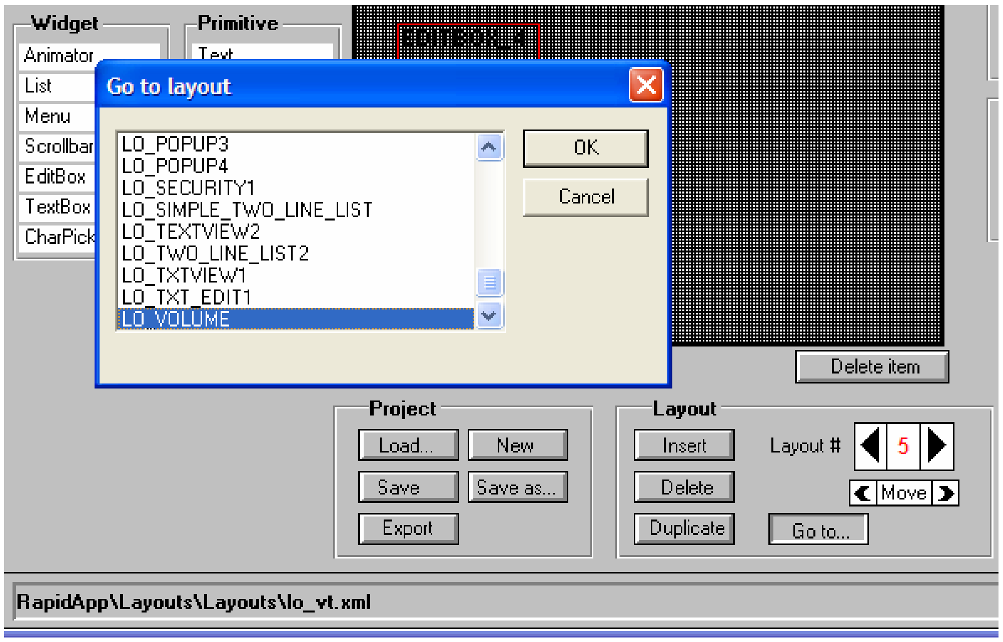
- Every time something is modified using e-SIM RapidPLUS™, code generation must be launched to update the .c and .h files
- RapidPLUS™ generates both the .c and .h files in the same folder Reference_Design\libs\ESIM\appCG. All .c files must be separated to Reference_Design\libs\ESIM\appCG\src folder and all .h files must be separated to Reference_Design\libs\ESIM\appCG\inc folder. This is done using a batch file which runs after code generation (refer to section 5.5.4)
- Once code generation is completed, the code must be compiled using the ARM compiler. If there is no error, the target image is created in the ...\ads\out folder (refer to section 5.5.2)
- The target image can be downloaded on the target platform using the Hyperterminal 1K Xmodem protocol
- The MMI can be started by running the embedded application using the AT command AT+WOPEN=1

- If the MMI settings structure is modified, it is advisable to clear the existing MMI settings using AT+WOPEN=3. If required, the existing target image can be removed using AT+WOPEN=4

5.8 Screen/Layouts - Screen Sizes, Layouts

The Open AT® GTi provides a mechanism to conveniently modify the screen sizes and layout design of the MMI using the Layout Editor tool. The Layout Editor (refer to section 2.3.1) can be used to modify layouts before the RapidPLUS™ is used for other necessary changes in the layout before generating the corresponding C code. The generated code can then be compiled using the ARM compiler.

This section explains the basic steps to perform any change in screen sizes and layouts.



RapidApp\Layouts\Layouts\lo_vt.xml

Figure 44: Screen Size/Layout Customization - Selecting Layout

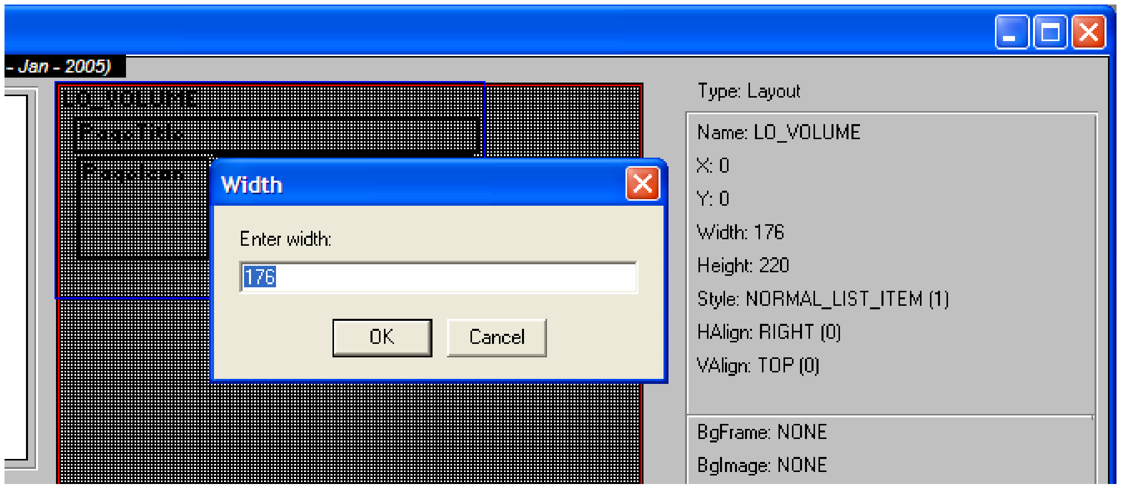
Selecting the Layout to Be Edited

i. Open the Layout Editor from the following path: Reference_Design\LayoutEditor\LAYOUT_EDITOR.RPD (refer to Figure 8).

ii. The 'Load' button can be used to load the project workspace containing information on all the existing layouts being used for the MMI. The lo_vt.xml is the project workspace file for the existing MMI. This file can be accessed from the path: \RapidApp\Layouts\Layouts in the Open AT® GTi build.

iii. Existing Layouts can be accessed using arrows keys along with 'Layout #' label. New layouts can be inserted using the 'Insert' button. Any existing layout can be deleted using the 'Delete' button. Any layout can be duplicated (and then modified) using the **Duplicate** button. The **Go to** button opens a dialog box to select the layout to be modified from the list of layout IDs (refer to Figure 46).

iv. Once the layout to edit is selected, all necessary modifications in size and location of elements in the layout can be done.



Type: Layout
Name: LO_VOLUME
X: 0
Y: 0
Width: 176
Height: 220
Style: NORMAL_LIST_ITEM (1)
HAlign: RIGHT (0)
VAlign: TOP (0)
BgFrame: NONE
BgImage: NONE

Figure 45: Screen Size/Layout Customization - Resizing Layout Elements

Moving and Re-sizing the Layout Elements

i. Any layout element can be selected and moved using the mouse pointer.

ii. The size of any layout element can be changed. Select the layout element and click the elements in the layout and then click the values of the Width or Height on the editor. Click on any of these values open a dialog box in which the new value can be entered. After the new value is entered, click **OK** on the dialog box to update this element in the Layout Editor display.

iii. Click on the current style of the element opens a dialog box with a list of all available styles in the Layout Editor. Click **OK** on the dialog box to select a new style.

iv. Similarly, other parameters related to the layout element can be updated.

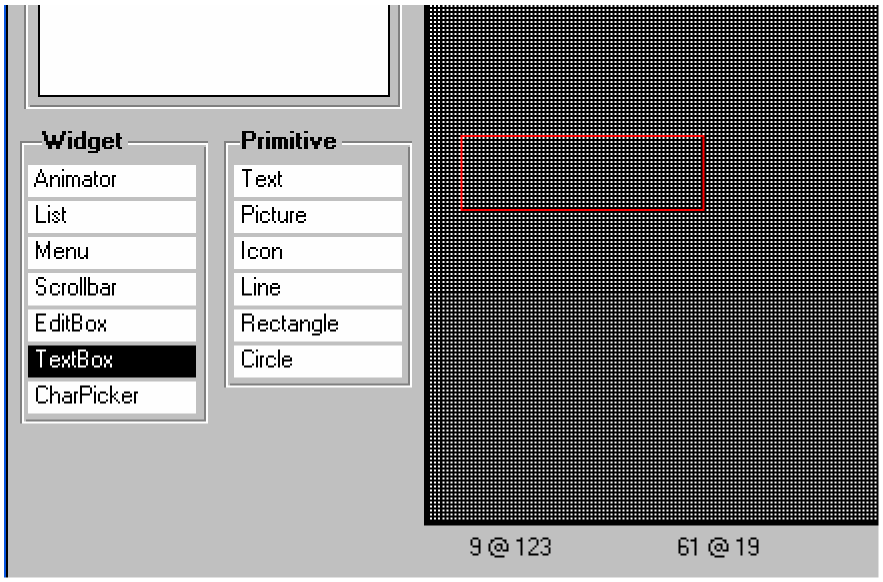


Figure 46: Screen Size/Layout Customization - Adding New Layout Element

Adding New Elements to the Existing Layout

i. New elements can be added to the existing layout from the available list of widgets and primitives (refer to Figure 47).

ii. Once added, the size, ID, and position can be changed using the steps mentioned above.

Saving and Exporting Layout Changes

i. All layout changes can be saved on the project file using the **Save** button of the Layout Editor.

ii. Saving the layout changes updates the existing project file: lo_vt.xml.

iii. Upon saving, layout changes can be exported to the .rar and .h files in the ...\RapidApp\Layouts\Export folder. The files lo_vt.c and lo_vt.h are updated after export is completed. These files are updated in the EmbdCode folder. They must be copied manually to the EmbdCode\src and EmbdCode\inc directories. These files are used by the RapidPLUS™ during code generation for the MMI.

Changes Reflected in the RapidPLUS™

Once exporting is finished using the Layout Editor, the xml and c source files, which are updated in the EmbdCode folder are used by the SRV_LAYMAN user object to generate the layout on the screen.

Generating Code and Making the Target Build

After completing all the changes needed in the RapidPLUS™, the user can follow the standard build procedure: code generation, code compilation and build download (refer to section 5.6).

5.8.1 Multimedia Resources – Graphics and Animations

The Multimedia Resource Manager (refer to section 2.3.3) is used to add and modify graphics elements and animations to the Wavecom MMI.

This tool is based on an excel template that contains macros to generate IDs for all the bitmaps. These generated IDs are used by several RapidPLUS™ components. All project bitmaps and animations are organized in a separate folder ...RapidApp\Images as shown in figure below. The Res folder is also known as the destination folder from which the application imports bitmaps. On the other hand, the Source folder is the source of bitmaps which can be added, modified or deleted by the user. Similarly, animations are kept in an Animations subfolder and animation editing is done from this folder.

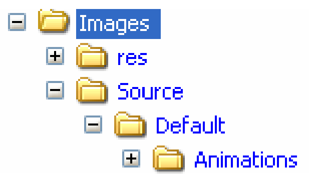


Figure 47: Multimedia Resource Manager- Folder Structure

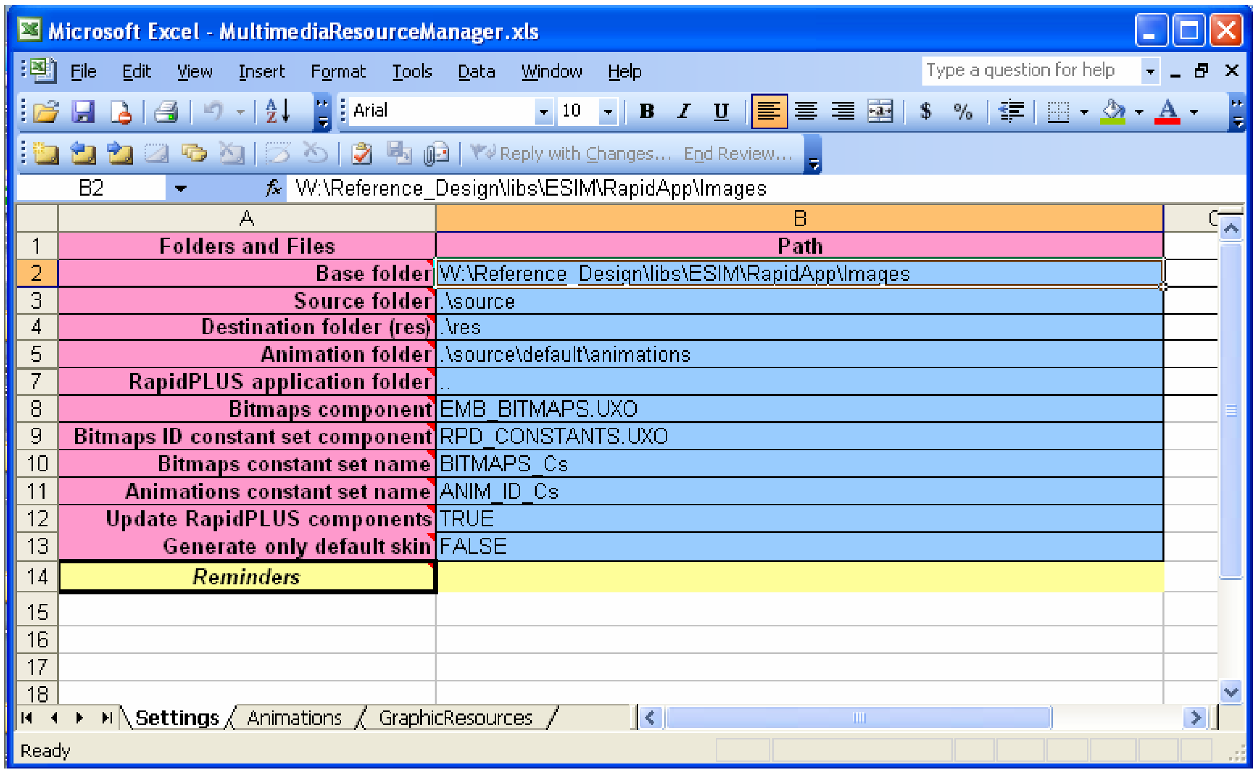
This section explains the basic steps in the customization of graphical elements (Bitmaps and Animations).

5.8.1.1 Updating an Existing Bitmap/Animation

The steps to update an existing animation or bitmap are:

i. Open the MultimediaResourceManager.xls file in the ...\RapidApp\images folder.

ii. In the settings tab of the file make sure that various components, source and destination folders are set to the default path as shown in Figure 49.



Microsoft Excel - MultimediaResourceManager.xls

B2 W:\Reference_Design\libs\ESIM\RapidApp\Images

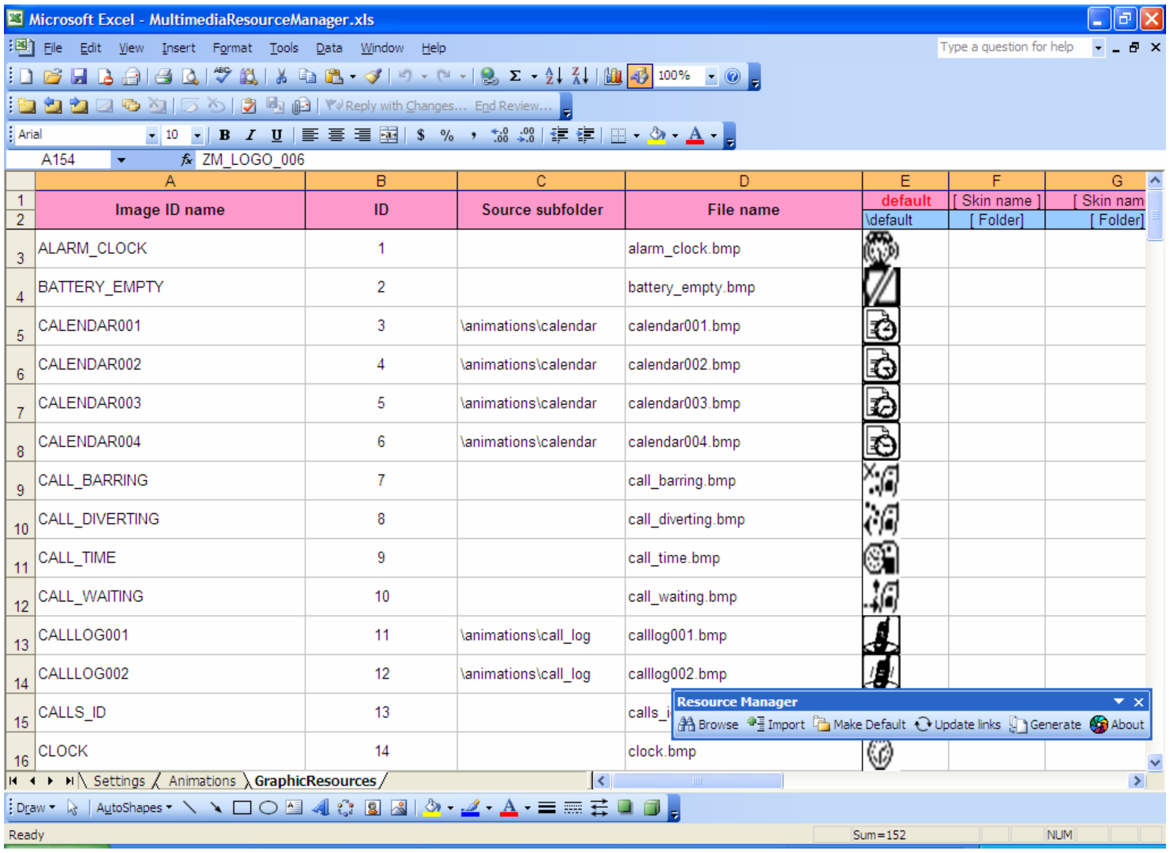
	A	B
1	Folders and Files	Path
2	Base folder	W:\Reference_Design\libs\ESIM\RapidApp\Images
3	Source folder	.\source
4	Destination folder (res)	.\res
5	Animation folder	.\source\default\animations
7	RapidPLUS application folder	..
8	Bitmaps component	EMB_BITMAPS.UXO
9	Bitmaps ID constant set component	RPD_CONSTANTS.UXO
10	Bitmaps constant set name	BITMAPS_Cs
11	Animations constant set name	ANIM_ID_Cs
12	Update RapidPLUS components	TRUE
13	Generate only default skin	FALSE
14	Reminders	

Figure 48: Multimedia Resource Manager – Settings

iii. Edit the desired image using any image editor in the source folder and save it.

iv. Select the GraphicResources tab as shown in Figure 50, and use the import facility of the Resource Manager to import the updated images from the source folder.

v. Once updated bitmaps are loaded, import them in the RapidPLUS™ application. Bitmaps are imported in the EMB_BITMAPS user object. This user object includes a list where bitmaps that must be modified can be imported.


Microsoft Excel - MultimediaResourceManager.xls
File Edit View Insert Format Tools Data Window Help
Resource Manager
Browse Import Make Default Update links Generate About
Settings Animations GraphicResources
Ready
Sum=152
NUM

A154 ZM_LOGO_006

	A	B	C	D	E	F	G
1	Image ID name	ID	Source subfolder	File name	default	[Skin name]	[Skin nam
2					\default	[Folder]	[Folder]
3	ALARM_CLOCK	1		alarm_clock.bmp			
4	BATTERY_EMPTY	2		battery_empty.bmp			
5	CALENDAR001	3	\animations\calendar	calendar001.bmp			
6	CALENDAR002	4	\animations\calendar	calendar002.bmp			
7	CALENDAR003	5	\animations\calendar	calendar003.bmp			
8	CALENDAR004	6	\animations\calendar	calendar004.bmp			
9	CALL_BARRING	7		call_barring.bmp			
10	CALL_DIVERTING	8		call_diverting.bmp			
11	CALL_TIME	9		call_time.bmp			
12	CALL_WAITING	10		call_waiting.bmp			
13	CALLLOG001	11	\animations\call_log	calllog001.bmp			
14	CALLLOG002	12	\animations\call_log	calllog002.bmp			
15	CALLS_ID	13		calls_i			
16	CLOCK	14		clock.bmp			

Figure 49: Multimedia Resource Manager- GraphicResources

Note	If the modified bitmap is part of the set of animation bitmaps (in the ...source\Default\Animations folder) changes are reflected in the animation.

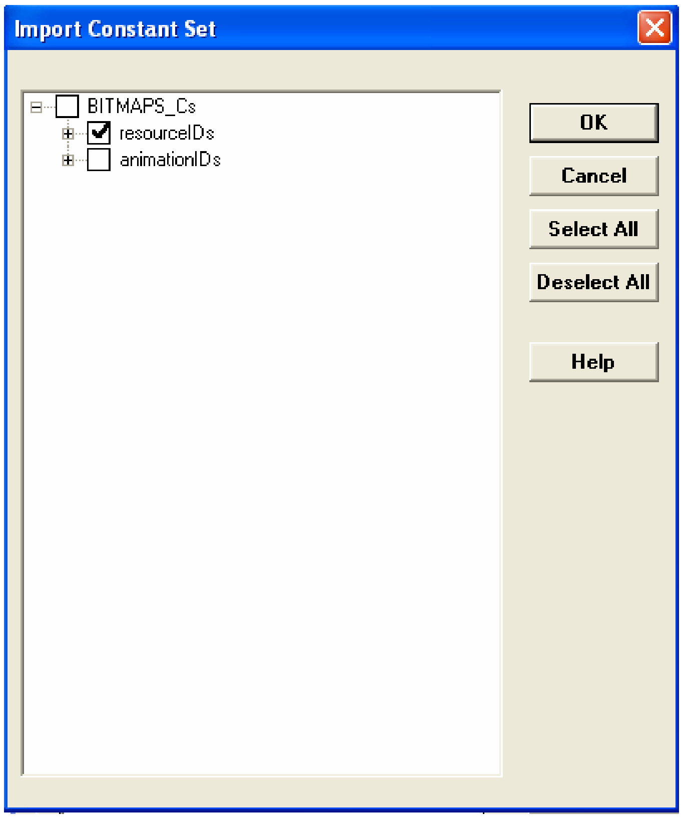


Figure 50: Multimedia Resource Manager – Updating Bitmaps in the RapidPLUS™

5.8.1.2 Adding a New Bitmap/Animation

The steps to add a new Bitmap/Animation are:

i. Open the MultimediaResourceManager.xls file and select the GraphicResources tab. The screen shown in Figure 50 is displayed.

ii. Create a new Bitmap image and copy this image in the ...\source folder. In case of animations, copy it in the ...\source\Default\Animations folder.

iii. Using the resource manager dialog control, import the images from the source folder. This also creates a new image id and copies the new image to the res folder.

iv. Update the bitmaps array in the EMB_BITMAPS user object by importing the new added bitmap.

v. Update the constant set BITMAPS_Cs and ANIM_ID_Cs with the latest bitmap and animation IDs after addition of the new bitmap as shown in Figure 51.

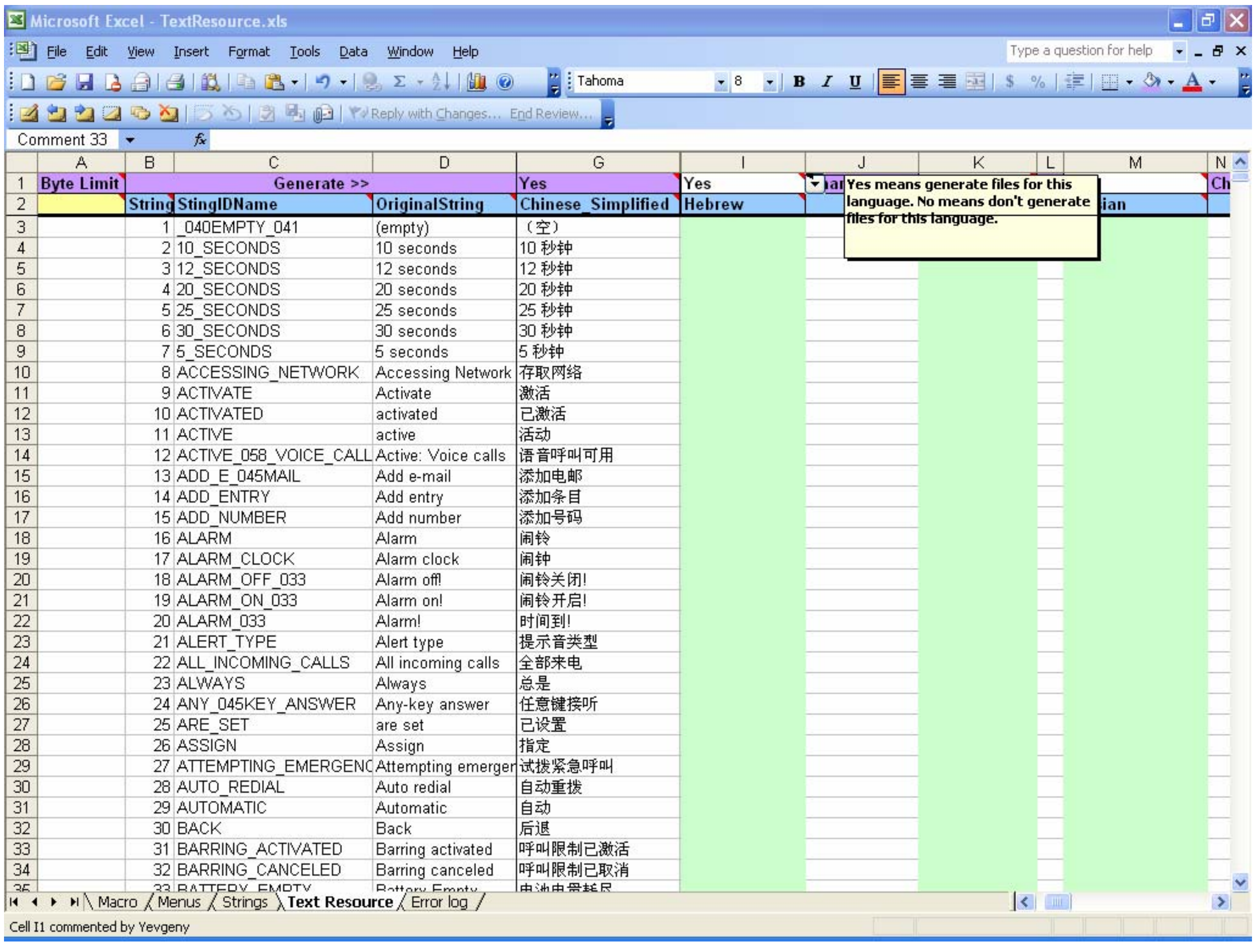
The user can now use the new Bitmap/Animation in the RapidPLUS™ application.

5.8.1.3 Deleting an Existing Bitmap/Animation

In order to remove a Bitmap/Animation from the RapidPLUS™ project, remove the bitmap from the source folder. Other steps in the deletion procedure are identical to steps for adding a new bitmap, as detailed in section 5.8.1.2 above.

5.8.2 Text Resources - Support for Different Languages

User can select different languages for the MMI. This is done by using the Textresource.xls file. In the textresources tab, the user has the option to add labels of any language and then set the generation flag for that language to Yes. The selected language is generated and updated in the EMB_TEXTRES user object. Once this user object is updated, strings in the new language can be used in the RapidPLUS™ application.


Microsoft Excel - TextResource.xls
File Edit View Insert Format Tools Data Window Help
Type a question for help
Tahoma
8
Reply with Changes... End Review...
Comment 33
Macro
Menus
Strings
Text Resource
Error log

	A	B	C	D	G	I	J	K	L	M	N
1	Byte Limit	Generate >>			Yes	Yes					Ch
2		String	StingIDName	OriginalString	Chinese_Simplified	Hebrew				ian	
3		1	_040EMPTY_041	(empty)	（空）						
4		2	10_SECONDS	10 seconds	10 秒钟						
5		3	12_SECONDS	12 seconds	12 秒钟						
6		4	20_SECONDS	20 seconds	20 秒钟						
7		5	25_SECONDS	25 seconds	25 秒钟						
8		6	30_SECONDS	30 seconds	30 秒钟						
9		7	5_SECONDS	5 seconds	5 秒钟						
10		8	ACCESSING_NETWORK	Accessing Network	存取网络						
11		9	ACTIVATE	Activate	激活						
12		10	ACTIVATED	activated	已激活						
13		11	ACTIVE	active	活动						
14		12	ACTIVE_058_VOICE_CALL	Active: Voice calls	语音呼叫可用						
15		13	ADD_E_045MAIL	Add e-mail	添加电邮						
16		14	ADD_ENTRY	Add entry	添加条目						
17		15	ADD_NUMBER	Add number	添加号码						
18		16	ALARM	Alarm	闹铃						
19		17	ALARM_CLOCK	Alarm clock	闹钟						
20		18	ALARM_OFF_033	Alarm off!	闹铃关闭!						
21		19	ALARM_ON_033	Alarm on!	闹铃开启!						
22		20	ALARM_033	Alarm!	时间到!						
23		21	ALERT_TYPE	Alert type	提示音类型						
24		22	ALL_INCOMING_CALLS	All incoming calls	全部来电						
25		23	ALWAYS	Always	总是						
26		24	ANY_045KEY_ANSWER	Any-key answer	任意键接听						
27		25	ARE_SET	are set	已设置						
28		26	ASSIGN	Assign	指定						
29		27	ATTEMPTING_EMERGENC	Attempting emerger	试拨紧急呼叫						
30		28	AUTO_REDIAL	Auto redial	自动重拨						
31		29	AUTOMATIC	Automatic	自动						
32		30	BACK	Back	后退						
33		31	BARRING_ACTIVATED	Barring activated	呼叫限制已激活						
34		32	BARRING_CANCELED	Barring canceled	呼叫限制已取消						

Yes means generate files for this language. No means don't generate files for this language.

Cell I1 commented by Yevgeny

Figure 51: TextResources – Adding a Language

To add a new language:

i. Open the file Textresource.xls and select the textresource tab.

ii. Set the generation flag column for the language to Yes and add labels under the specified language as shown in Figure 52.

iii. Once all labels are added, generate the Texresource.xls file using the Menu and Text generator.

iv. Import the new language ID into the LANGUAGE_Cs constant set in the EMB_TEXTRES user object. This ID can now be used by the RapidPLUS™ application to set the language.

5.8.3 Menus - Add/Modify/Delete Menu Items

New Menus can be added and existing menus can be modified and deleted using the textresource.xls file.

5.8.3.1 Changing an Existing Menu Item

Whenever the user needs to change the menu item for any menu in the Wavecom MMI, the textresource.xls file must be updated.

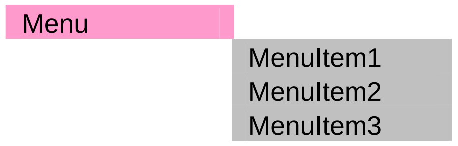


Figure 52 : Menus - Modifying a Menu Item

Proceed as follows:

i. In the ...\RapidApp folder, open the Textresource.xls file with macros enabled.

ii. Select the menu tab in the textresource.xls file.

iii. In the displayed menu tree, select the menu item to rename and change the name of this menu item as required. For e.g. as shown in Figure 53, rename the first menu item (MenuItem1) and change the name of this menu item to "ModifiedItem" (as shown in Figure 54). Save the Textresource.xls file.

iv. Select the macro tab and click the macro button. The Text and Menu generator screen is displayed as shown in Figure 11.

v. Press the **Start** button after making sure that the correct options are selected or default options as shown in Figure 11.

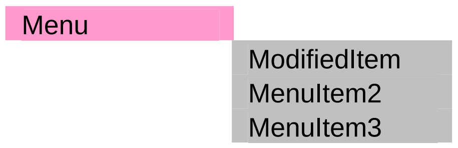


Figure 53: Menus - Modified Menu Item

vi. On completion of menu generation, the constant set STRING_ID_Cs in the EMB_TEXTRES.UXO is updated with the new string.

5.8.3.2 Adding a New Menu Item to an Existing Menu

To add a new Menu Item to an existing menu:

i. Open the TextResource.xls file and select the menu tab.

ii. From the menu tree, select the menu item above which a new menu item must be added.

iii. Right click the selected menu item and select the insert option to insert a new row on the excel sheet.

iv. Add the menu item along with the fields detailed below (refer to Figure 55 which shows an example of new Menu item added under the Sound Settings Menu):

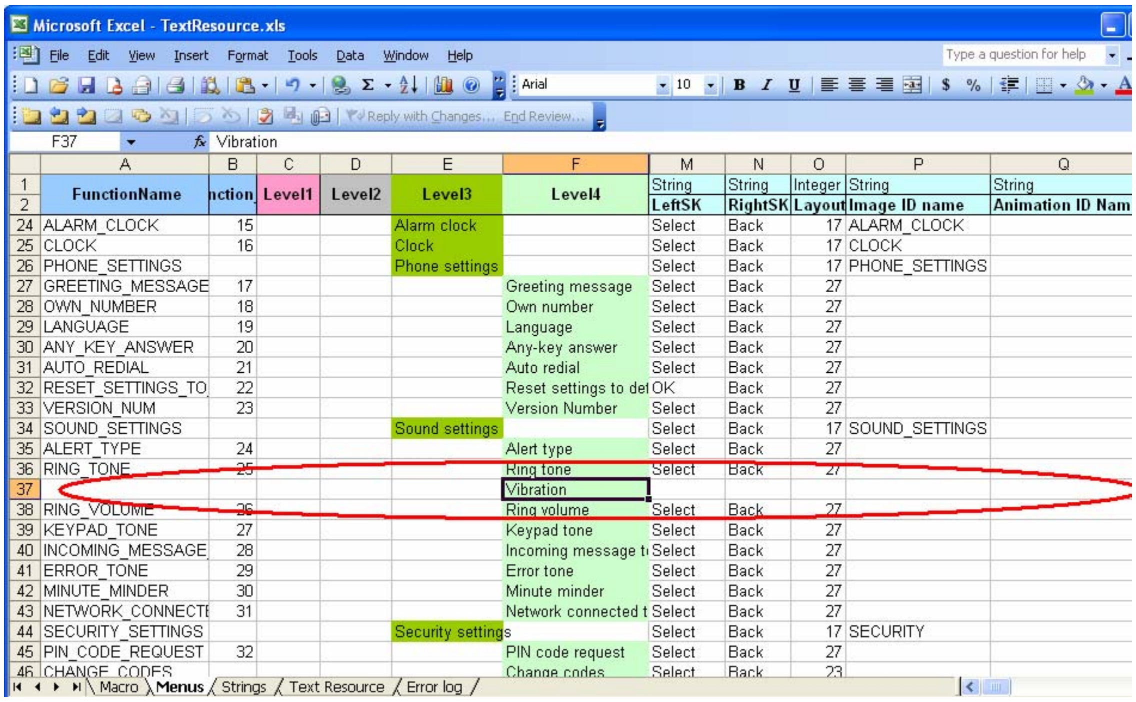
   a. LeftSK: String for the Left Soft key displayed when this menu item is selected

   b. RightSK: String for the Right Soft key displayed when this menu item is selected

   c. Layout: Numeric ID of the Layout started on selection of this menu item.

   d. Image ID: Numeric ID of the image to associate with the menu item.

   e. Animation ID: Numeric ID of the animation to associate with the menu item.


Microsoft Excel - TextResource.xls

F37 Vibration

	A	B	C	D	E	F	M	N	O	P	Q
1	FunctionName	nction	Level1	Level2	Level3	Level4	String	String	Integer	String	String
2							LeftSK	RightSK	Layout	Image ID name	Animation ID Nam
24	ALARM_CLOCK	15			Alarm clock		Select	Back	17	ALARM_CLOCK	
25	CLOCK	16			Clock		Select	Back	17	CLOCK	
26	PHONE_SETTINGS				Phone settings		Select	Back	17	PHONE_SETTINGS	
27	GREETING_MESSAGE	17				Greeting message	Select	Back	27		
28	OWN_NUMBER	18				Own number	Select	Back	27		
29	LANGUAGE	19				Language	Select	Back	27		
30	ANY_KEY_ANSWER	20				Any-key answer	Select	Back	27		
31	AUTO_REDIAL	21				Auto redial	Select	Back	27		
32	RESET_SETTINGS_TO	22				Reset settings to de	OK	Back	27		
33	VERSION_NUM	23				Version Number	Select	Back	27		
34	SOUND_SETTINGS				Sound settings		Select	Back	17	SOUND_SETTINGS	
35	ALERT_TYPE	24				Alert type	Select	Back	27		
36	RING_TONE	25				Ring tone	Select	Back	27		
37						Vibration					
38	RING_VOLUME	26				Ring volume	Select	Back	27		
39	KEYPAD_TONE	27				Keypad tone	Select	Back	27		
40	INCOMING_MESSAGE	28				Incoming message t	Select	Back	27		
41	ERROR_TONE	29				Error tone	Select	Back	27		
42	MINUTE_MINDER	30				Minute minder	Select	Back	27		
43	NETWORK_CONNECTI	31				Network connected t	Select	Back	27		
44	SECURITY_SETTINGS				Security settings		Select	Back	17	SECURITY	
45	PIN_CODE_REQUEST	32				PIN code request	Select	Back	27		
46	CHANGE_CODES					Change codes	Select	Back	23		

Figure 54: Menus – Adding a Menu Item to an Existing Menu

v. Go to the Macro tab and start the macro generation process.

vi. On completion of the Text and Menu Generation Macro, Function_INDEX_Cs and FUNCTION_ID_Cs constant sets in the EMB_TEXTRES user object are updated.

Note	The constant sets in the EMB_TEXTRES are updated from the LoadTo_ConstantSet.h file, which is updated whenever the Text and Menu generator macro is run with certain changes in menus and textresources.

5.8.3.3 Creating a New Menu (Including Menu Items)

To create a new menu:

i. Add a new menu in the Textresource.xls file.

ii. Add menu items under the menu.

The new menu is implemented in the RapidPLUS™ application using WMI_MENU user object to start the submenu.

Adding a New Menu and its Menu Items

i. Follow the steps given in section 5.8.3.2 to create a menu item.

ii. In order to add a submenu to the menu item, insert the new menu items as shown in Figure 54. i.e., menuitem1 is located column+1 and row+1 relative to the Parent menu item. Further, menu items must be added by incrementing the rows in the same column of the excel worksheet.

iii. Once menu items are added, this menu can be started by calling the start function of WMI_MENU. The parameters of the function call are the function index of the first menu item.

- E.g. WMI_MENU start: emb_textres_h.FUNCTION_INDEX_Cs.IX_SHOW_CLOCK

Menu customization is also detailed in section 2.3.2

5.8.3.4 Files Updated by the Textresource.xls Macro

The files updated by the Menu and Text Generator for the RapidPLUS™ are:

i. Datafiles updated for the RapidPLUS™ project:
   a. CodePage.rar
   b. Menu_Index.rar
   c. Menu_Unicode.rar
   d. LoadTo_ConstantSet.h

ii. Datafiles updated for the embedded system:
   a. cLng_Chinese_Simplified.c
   b. cLng_English.c
   c. LoadTo_ConstantSet.h
   d. xl_IndexArray.c
   e. xl_menu.h

5.8.4 Keypad

Before detailing the customization of Keypad, let us first understand what methods are used to integrate the Keypad and which files are involved (refer to section 5.2.1.2). The MMI application subscribes for all keypad events using an ADL API *adl_atUnSoSubscribe.* This can be seen in *DoInit* method in the file oat_appli.c.

```
adl_atUnSoSubscribe ("+CKEV:", CKEV_Handler);
```

The +CKEV are the messages that are subscribed for. These messages are handled in the CKEV_Handler method in the same file.

The CKEV_Handler calls the processKeyDown and processKeyUp methods to process key inputs. These methods are implemented in the ...\emdbcode\src\app_api.c file.

The processKeyDown method interacts with the EMB_KPD which is implemented as a RapidPLUS™ UDI object. On every key press, the code_Int property corresponding to the code of the key pressed of EMB_KPD is set and a key press event is triggered.

On getting the key press event from the EMB_KPD, SRV_KPD (implemented as a RapidPLUS™ UDO) informs the above MMI on the key press and code of the pressed key.

The MMI keypad can be customized as per the keys available on the target keypad. Keypad customization involves:

i. Adding\deleting\modifying key codes in the constant set KEY_CODES_Cs in EMB_KPD.UXO

ii. Update the switch case in the processKeyDown method in app_api.c to include a new case or modify the existing case accordingly.

iii. Add\update a corresponding event in SRV_KPD (SRV_KPD.UXO) so that the updated key can be recognized by the MMI.

5.9 Debugging Using the RapidPLUS™ and Wavecom Tools

The Open AT® Software applications are debugged using the E-SIM simulation mode, RTE (Remote Task Execution) mode, and Target Mode.

E-SIM Simulation Mode: This is the debugging process performed in the first stage of the application development in the RapidPLUS™ environment. In this mode, the application is debugged on the development platform (PC) itself. The Wireless CPU is connected to the serial port and is used as a modem to send and receive AT commands. Debugging the application at this stage is less expensive when dealing with large applications. The application developer must try to identify any application or logical errors at this stage.

Note	Although this mode is called Simulation mode, it cannot run on a standalone basis without the Wavecom Wireless CPU.

RTE Mode: Once the application is successfully built using the RTE project settings, it can be debugged in the remote mode. In remote mode, the application runs on the PC and uses its resources such as memory and clock with the serial link setup with the Wavecom CPU. The breakpoints for debugging can be set in this mode. (More information on RTE is given in section 4.4).

TARGET Mode: In target mode, the application runs solely on the target and uses its resources. Debugging in this mode is only possible using the target monitoring tools such as TMT and TE. These tools are described in section 4.2 and 4.3 respectively and are provided to assist the debugging of the application, especially, in the Target mode.

TMT is used to monitor traces and other parameters which monitor the state of the target application. Similarly, TE is used to start/stop the application and also to display the commands sent to the target and responses received from the target.

Note	Traces can only be displayed on the TMT screen provided certain flags are uncommented in the **DebugAPI.c** file in the...\EmbdCode folder. The changes that must be done are explained in section 4.2.

5.10 Wavecom Sample MMI Limitations

5.10.1 Keypad Handler

The number of keypad events passed to the E-SIM engine is limited. When the user presses too many keys, adl_atUnSoUnSubscribe is called to unsubscribe further key press event. In about 0.6 seconds, it calls again adl_atUnSoSubscribe to subscribe again. This blocking is done in oat_appli.c. The logic of the function Keypboard_TimerHandler prevents excess key press events being passed to the ESIM.

5.10.2 Battery & Charger Handler

If the user wants to use the same board for simulation and is running the MMI on hardware, the user must turn off the unsolicited battery status before switching to use. For example, when the user is using board simulation, the indication (AT+WBCM=0, 0) must be turned off using the hyper terminal application before switching the OAT GTi application on (AT+WOPEN=1). When the MMI is running on the board, the user must turn off the Wireless CPU by pressing down the power off button before turning off the application (AT+WOPEN=0). The application itself will send (AT+WBCM=0, 0) to the Wireless CPU.

5.10.3 SMS Handler

The SMS handler must be handled differently compared to other service handlers. The text must be sent with the ADL API adl_atCmdSendText rather than using the AT command AT+CMGS to send the SMS text.

5.10.4 Phonebook

The Wavecom OS supports two phonebooks (ME and SIM), but only one can be activated at a time. If a name A1 is in ME phonebook, but the SIM phonebook is the selected phonebook, if a call from A1 is received, his/her name will not be displayed.

5.10.5 Timer Handler

Timer ticks sent by the Open AT® Software are not accurate. The e-SIM system clock, running as a part of the Open AT® Software application, is based on the tick returned by the Open AT® Software. Due to inaccuracy of timer ticks, it is necessary for the MMI to obtain the accurate time periodically from the CPU via an AT command. This is implemented in the SRV_GSRV.

5.10.6 Multiple Simultaneous AT Commands

The Open AT ® GTi application sends AT commands to the Wavecom OS to perform some actions. If an external HOST also sends AT commands to the Wavecom OS, there might be some interaction problems.

In case of two commands of a same class (Phonebook, SMS), one is rejected. It is highly recommended to avoid sending AT commands at the same time from the Open AT ® GTi application and from the external HOST to avoid interaction issues.

5.10.7 Language Support

The Open AT® GTi does not support languages such as Thai, Hebrew, and Arabic. They require supplementary software development tools.

5.10.8 SIM Tool Kit

The Wavecom Open AT® GTi does not support the SIM Toolkit functionality.

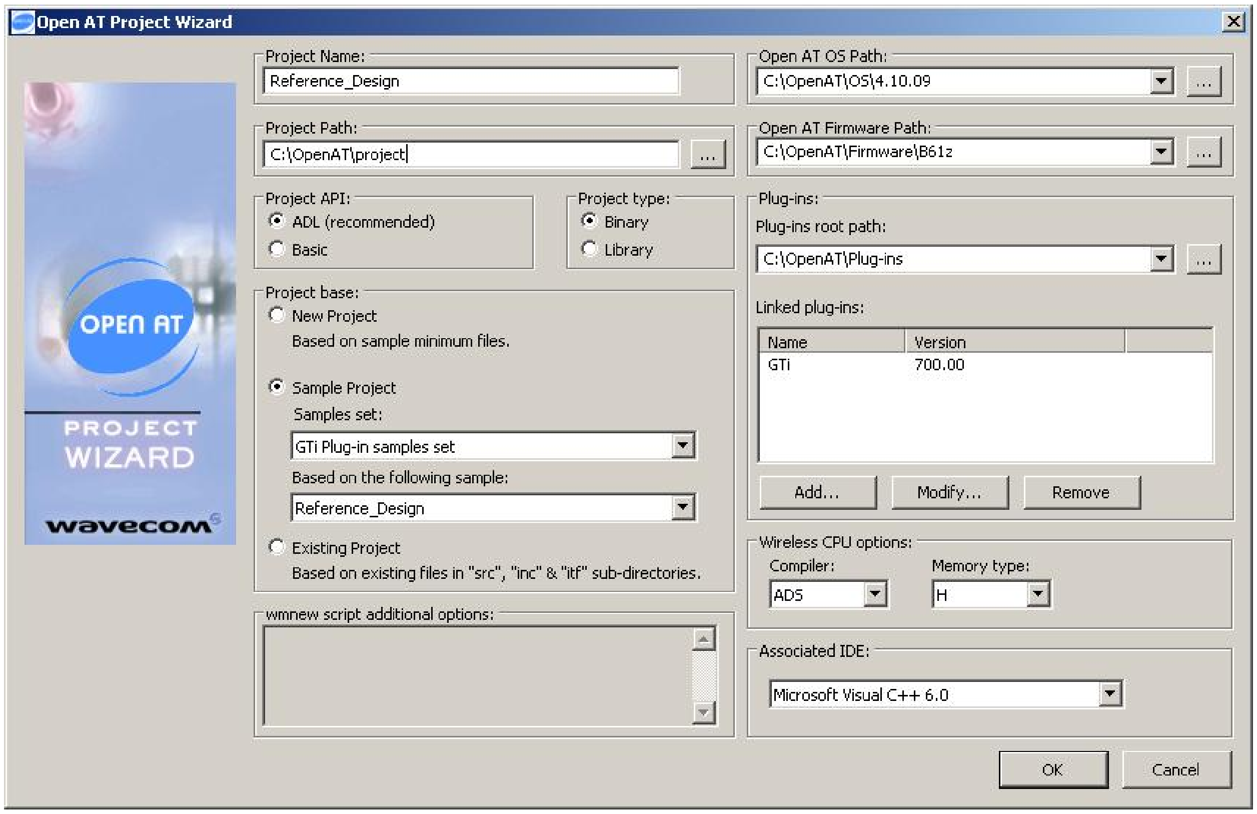
5.10.9 Simultaneous Key Press

The Wavecom Open AT® GTi does not support simultaneous key press.

5.11 Sample MMI Project

5.11.1 Reference Design Project

The Open AT® SDK came with a sample MMI project. The developers can open the project in the Open AT® project wizard.


Open AT Project Wizard

Project Name: Reference_Design

Project Path: C:\OpenAT\project

Project API: [✓ ADL (recommended)] Basic

Project type: [✓ Binary] Library

Project base:
New Project
Based on sample minimum files.
[✓ Sample Project]
Samples set: GTi Plug-in samples set
Based on the following sample: Reference_Design
Existing Project
Based on existing files in "src", "inc" & "itf" sub-directories.

wmnew script additional options:

Open AT OS Path: C:\OpenAT\OS\4.10.09

Open AT Firmware Path: C:\OpenAT\Firmware\B61z

Plug-ins:
Plug-ins root path: C:\OpenAT\Plug-ins

Linked plug-ins:

Name	Version
GTi	700.00

Wireless CPU options:
Compiler: ADS
Memory type: H

Associated IDE: Microsoft Visual C++ 6.0

Figure 55: Open AT ® Project Wizard

6 References

- TM_Tools_Manual_for_Open_AT.pdf
- TU_Tutorial_for_Open_AT.pdf
- GS_Getting_Started_with_Open_AT.pdf
- Basic_Development_Guide.pdf
- ADL_User_Guide.pdf
- AT_Commands_Interface_Guide.pdf
- Wavecom OpenAT® GTi GDS.doc
- Software Design Document 1.1.doc
- RapidPLUS™ Basics - Course Book 5.0.0.pdf
- Generating Code for Embedded Systems 8.0.pdf
- Wavecom website: www.wavecom.com
- e-SIM website: www.e-SIM.com