



# e-SIM MMI Reference Design 1.x Training



## Table of Contents

<b>Lesson</b>	<b>Slides</b>
<b>Day 1</b>	
Welcome	<b>1</b>
Introduction to e-SIM MMI	<b>9</b>
Architecture	<b>13</b>
Main Components	<b>35</b>
<b>Day 2</b>	
Basic Architecture Exercise	
<b>Day 3</b>	
Implementation Guidelines	<b>51</b>
Customization Tools	<b>72</b>
Layout Manager	<b>77</b>
Menu and Text Generator	<b>81</b>
Multimedia Resource Manager	<b>92</b>
Customization Tools Exercise	
Synchronization of Components	<b>102</b>
<b>Day 4</b>	
Draw Method	<b>111</b>
Draw Exercises	<b>120</b>
Menu Method	<b>140</b>
Menu Exercises	<b>165</b>
<b>Day 5</b>	
Final Review Exercise	<b>178</b>
Training Summary	

© 2004 e-SIM Ltd. All rights reserved.

e-SIM Ltd.  
POB 45002  
Jerusalem 91450  
Israel



Tel: 972-2-5870770  
Fax: 972-2-5870773

Information in this manual is subject to change without notice and does not represent a commitment on the part of the vendor. The software described in this manual is furnished under a license agreement and may be used or copied only in accordance with the terms of that agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of e-SIM Ltd.

Microsoft, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other product and company names mentioned in this manual may be trademarks or registered trademarks of their respective owners.

Written and produced by e-SIM Ltd.  
Document Version 2.0




## e-SIM MMI Reference Design 1.x Training

RapidPLUS version: 8.0  
E-SIM MMI version: 1.2  
Training Book version: 2.0  
Feedback: [support@e-sim.com](mailto:support@e-sim.com)

Company Confidential. Do not copy or distribute.

1



## Course Objectives

- To explore and explain the e-SIM MMI Reference Design solution
- To teach you how to launch a project using the e-SIM MMI
- To fit the training to your specific needs

Company Confidential. Do not copy or distribute.

2

## Agenda

- Architecture of the e-SIM MMI Reference Design
- Customization tools
- Main components overview
- Implementation guidelines
- Synchronization of components
- Add new MMI module – Final project

## Syllabus: Day 1/5

Architecture	
9:00	Course Orientation
9:00-9:15	(1) Introduction to e-SIM MMI
10:00-12:30	(2) Architecture
12:30	Lunch
13:30-15:30	Architecture
15:30-17:00	(3) Main components



## Syllabus: Day 2/5

Architecture Exercises	
9:00-9:15	Summary of Day 1
9:15-11:30	Basic architecture exercise
12:30	Lunch
13:30-17:00	Basic architecture exercise

Company Confidential. Do not copy or distribute.

5



## Syllabus: Day 3/5

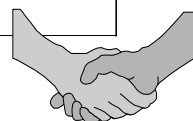
Customization Tools	
9:00-9:30	Architecture exercise summary
9:30-10:30	(4) Implementation guidelines
11:15-12:00	(5) Customization tools – Layout Manager
12:00-12:30	(5) Customization tools – Menu and Text Generator
12:30	Lunch
13:30-14:00	(5) Customization tools – Multimedia Resource Manager
14:00-15:00	Customization tools – Exercise
15:10-17:00	(6) Synchronization of components

Company Confidential. Do not copy or distribute.

6

## Syllabus: Day 5/5

Final Exercise	
9:00-9:15	Summary of Day 3
9:15-9:30	(9) Final exercise - review
9:30-12:30	Final exercise
12:30	Lunch
13:30-16:30	Final exercise
16:30-17:00	Training summery



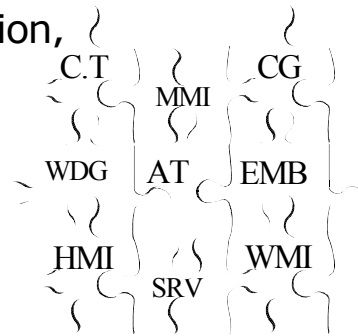
## Syllabus: Day 4/5

Advanced Methods	
9:00 - 9:15	Summary of Day 3
9:30 - 10:00	(7) Draw method
10:15 - 12:30	Draw Exercise
12:30	Lunch
13:30 - 14:00	(8) Menu method
14:10 - 17:00	Menu exercise



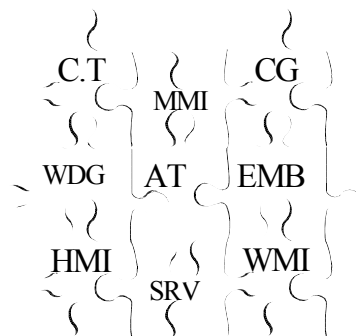
## Overview

- MMI (man-machine interface) reference design for a mobile telephone
- Provides a complete solution, including the platform, MMI application, and RapidPLUS tools that can customize the MMI application



## Overview (cont.)

- Can be easily customized to fit the needs of other platforms and projects
- The modular design also allows several developers to work on the project simultaneously





## Vocabulary

- HMI ~ MMI
- MMI modules
- Widgets
- Services
- Embedded components
- Customization tools

## Where to Get More Information


- Personal support from an e-SIM engineer
- Software Design Document
- Customization tools guides documents (3 guides)

**e-SIM MMI**  
**Reference Design 1.x**  
**Architecture**

Company Confidential. Do not copy or distribute.

13



## Component Types

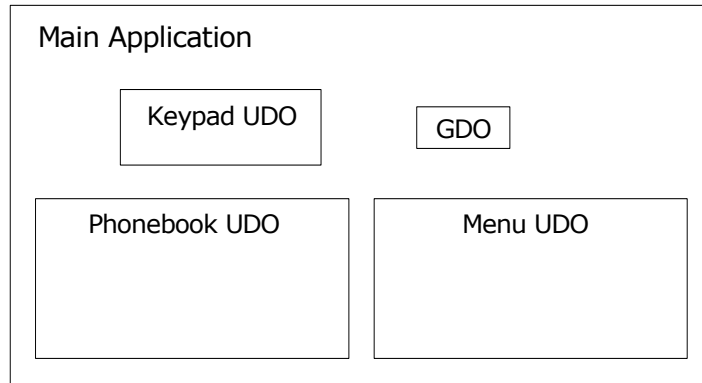
MMI	Man Machine Interface Modules (UI) referred to as HMI and WMI modules
WDG	Widgets (activated on request )
SRV	Services (Respond at any time )
EMB	Embedded (hardware/simulation)

Company Confidential. Do not copy or distribute.

14

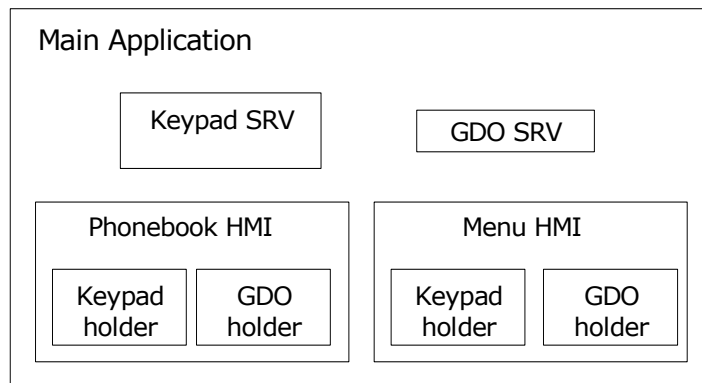
## Flat Structure

- User objects used for embedded components only



## Complex Component Structure

- User objects used to split the application logic into sub-components



## Embedded Components

Low-level system components such as keypad, display, memory, audio, battery.

They:

- Have the graphics of the product for input and output devices and for simulation panels
- Define the API to the embedded elements
- Include the minimum code that uses the object interface

**EMB**

17

Company Confidential. Do not copy or distribute.

## MMI Modules (user interface)

High-level components such as the phone book interface and call management.

They:

- Describe the main features of the product
- Handle the MMI: screens, menus, texts, and soft keys
- Are activated by the user

**MMI****EMB**

18

Company Confidential. Do not copy or distribute.

## MMI Modules (user interface) (cont.)

There are two types of MMI modules:

- WMI modules, at the lower level, are reusable MMI components that can control widgets, services, and embedded components
- HMI modules, handle a subsection of the project MMI. They can control all of the components, plus the WMI modules

**MMI**

**EMB**

## Service Components

Middle-level components such as components that control the keypad and display.

They:

- Manage the project's data and state
- Act as buffers between embedded components and MMI modules
- Can respond to requests at any time because they are always available

**MMI**

**SRV**

**EMB**

## Widget Components

Middle-level components such as the scroll bar.

They:

- Provide a single unit of user interface functionality
- Are activated on request and deactivated when no longer needed
- Have no self memory and must be set with data whenever activated
- Reusable

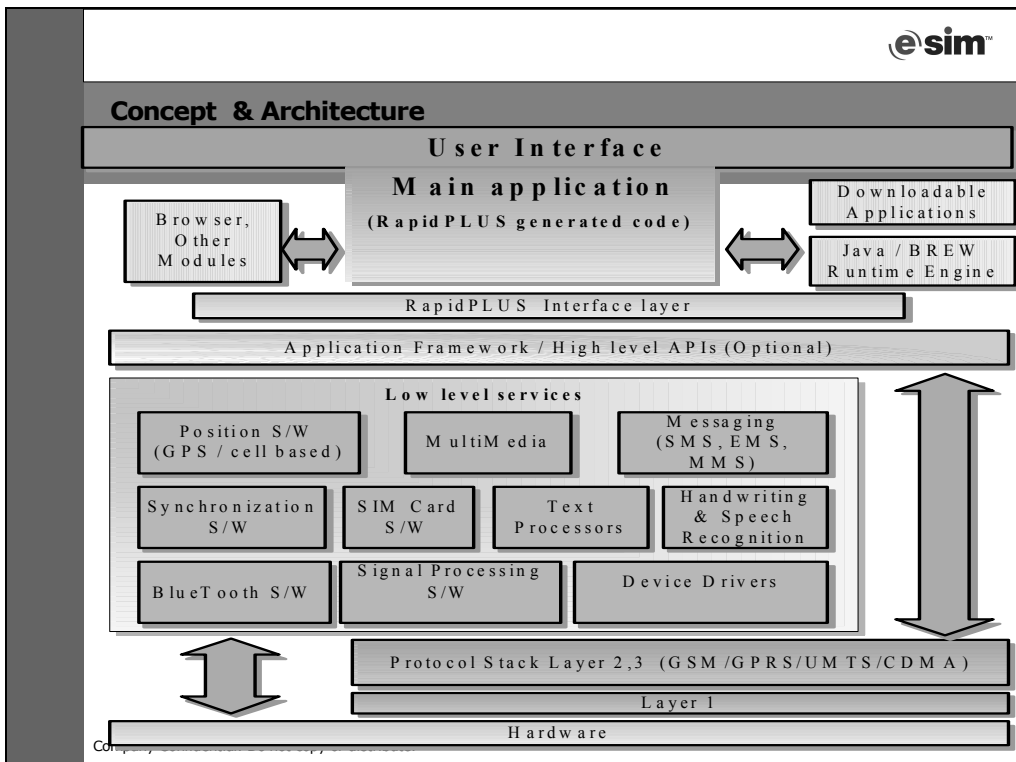
**MMI****WDG****SRV****EMB**

## Main Application

- Manages the entire project by activating the various components according to priorities.
- Is the umbrella application that contains all the other components.

**MMI****WDG****SRV****EMB**

e sim™			
<b>Component Types Summary</b>			
<b>MMI</b>	<b>WDG</b>	<b>SRV</b>	<b>EMB</b>
<b>Idle</b>	<b>Editor</b>	<b>Calls</b>	<b>Keypad</b>
<b>Call</b>	<b>Animator</b>	<b>Display</b>	<b>Battery</b>
<b>Menu</b>	<b>List</b>	<b>Bitmaps</b>	<b>Audio</b>
<b>SMS</b>	<b>Scroll bar</b>	<b>Call records</b>	<b>Memory</b>
<b>Main features</b>	<b>Parent activated</b>	<b>Always available</b>	<b>API as in embedded</b>
<b>User activated</b>	<b>Re-usable</b>	<b>Have memory</b>	<b>Have graphic objects</b>
	<b>Have no memory</b>	<b>Manage data</b>	<b>Minimum code</b>
Company Confidential. Do not copy or distribute.			23







## Device Arbitration (conflict handling)

Version 1.x:

- Main application manages resources
- One MMI module is active at any time
- HMI may start other HMI and give it the resources
- Each MMI must implement Init, Start, Stop, Pause, Resume



## Power Saving

Rapid application is event driven:

- No infinite loops or polling
- MMI is idle most of the time; activated upon external events
- The State machine works in cycles:
  - Run idle → moreToDo
  - Prevent system starvation even in simple operating system

## Power Saving (cont.)

Timer on request API:

- Timers are handles by the RapidPLUS state machine (kernel)
- Kernel API provides 2 callback functions (start/stop timer)
- The system calls a RapidPLUS function when the timer expires

## Factory Customization Concept

All resources are stored in external files:

- Bitmaps and animation bitmaps
- String literals
- Menu indexes
- Phone settings
- Layout data
- Fonts



## Factory Customization Concept (cont.)

- RapidPLUS application only needs the array entry point for each resource
- Different configurations can be prepared; no need for code generation



## Memory

- ROM
  - State machine engine; size varies between 30 to 50 KB.
  - Graphic library; size ~15 KB.
  - Application logic and objects
  - Constant objects

Total ~1 MB

## Memory (cont.)

- RAM
  - State machine queues and messages; for an empty application this represents 300 KBytes
  - Variable objects
  - (Cellular phones today has 300 KB RAM for the simplest and up to 2 MB for the advanced phones)

## Documentation

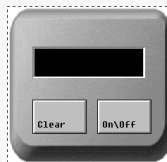
- Each of the customization tools is described in its own document
- The SDD document describes the design, implementation, and customization of the Reference Design. It also provides details about each of the components.
- Training books
- All Given in the GivenResource folder

## Architecture Exercise - Pager

- Using holders, DMA
- One HMI active at a time
- Follow the architecture slide
- PagerSys.rpd

### Main\_App.RPD

Pager.UDO



Sender.UDO

HMI\_Editor

HMI\_Ready

HMI\_TimeEd

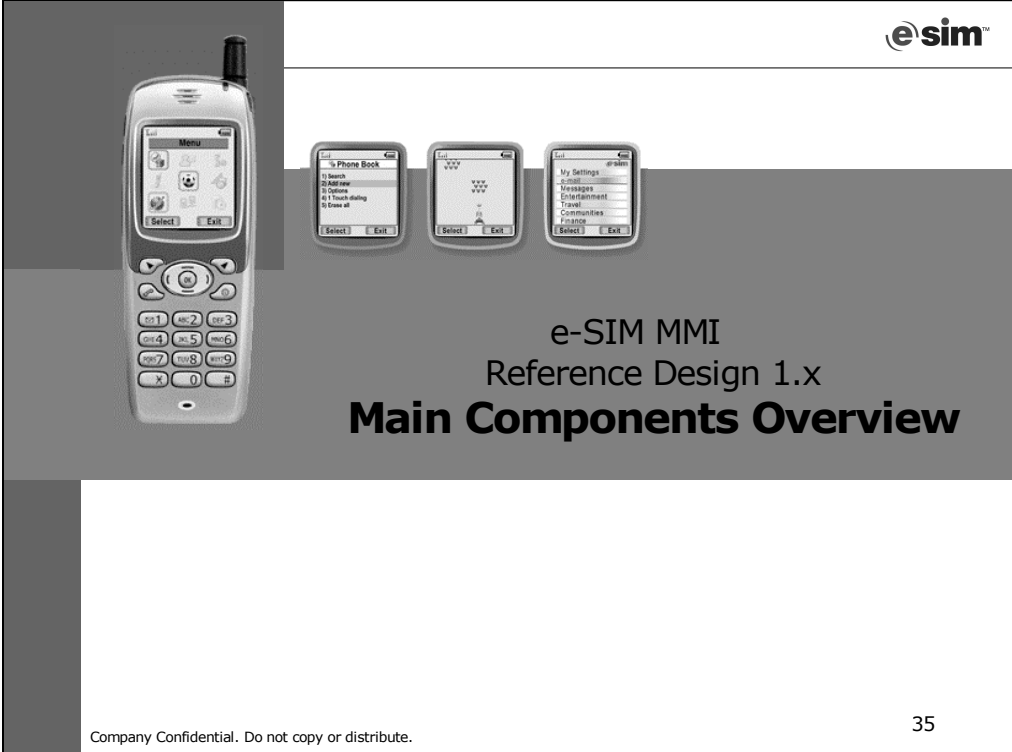
Emb\_Display

Emb\_Keypad

Srv\_Time







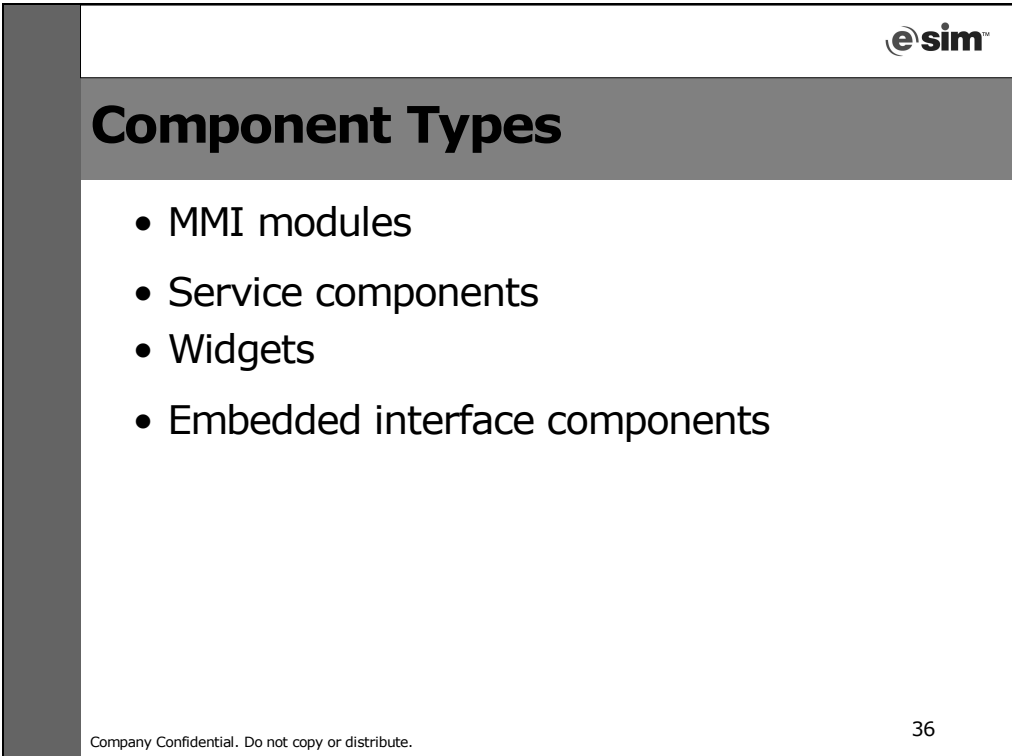
The slide features a large grey background. On the left is a silver mobile phone. To its right are three smaller screenshots of the phone's menu system: 'Phone Book', a search screen with '\*\*\*', and 'My Settings'. The 'eSIM' logo is in the top right corner. The title 'e-SIM MMI Reference Design 1.x Main Components Overview' is centered in the lower half of the slide. At the bottom left, it says 'Company Confidential. Do not copy or distribute.' and at the bottom right, the number '35'.

**eSIM**

e-SIM MMI  
Reference Design 1.x  
**Main Components Overview**

Company Confidential. Do not copy or distribute.

35



The slide has a grey header with the 'eSIM' logo on the right. Below the header, the title 'Component Types' is written in a large, bold font. Underneath the title is a bulleted list of four items: 'MMI modules', 'Service components', 'Widgets', and 'Embedded interface components'. At the bottom left, it says 'Company Confidential. Do not copy or distribute.' and at the bottom right, the number '36'.

**eSIM**

## Component Types

- MMI modules
- Service components
- Widgets
- Embedded interface components

Company Confidential. Do not copy or distribute.

36

## MMI Modules

- **HMI\_IDLE.UDO**
  - Handles all activity of the MMI when the phone is idling
  - Serves as a junction from which other MMI modules can be operated, depending on the inputs that occur
  
- **HMI\_CALL.UDO**
  - Handles phone call interaction: outgoing calls, incoming calls, multi-party calls, dialing, hold, speed dialing, and voice-mail

## MMI Modules (cont.)

- **HMI\_PBOOK.UDO**
  - Handles phone book interaction and high-level functionality
  - High-level functionality includes phone book selection, copying between phone books, selecting from the list of entries, managing speed dial keys, and memory used
  
- **HMI\_MENU.UDO**
  - Performs or initiates much of the function-ality associated with the menu items





## MMI Modules (cont.)

- **WMI\_MENU.UDO**
  - Handles menu navigation
  - Used by HMI\_MENU and by other MMI modules when menu display is needed. For example, call options and Contacts menu
- **WMI\_POPUP.UDO**
  - Handles different types of pop-up messages (timed out and non-timed out)



## Services: Modem Services

- **SRV\_CALL.UDO**
  - Manages the active calls
  - Holds data of all active calls and manages call manipulation, such as calls-on-hold and multi-party calls
  - Uses SRV\_TAPI.UDO for low-level functions
- **SRV\_TAPI.UDO**
  - Manages telephony functions, including call manipulation, network services, and certain SIM services

## Services: Modem Services (cont.)

- **SRV\_AT.UDO**
  - Known as the AT Command Dispatcher, manages all traffic to and from the modem
  - Provides basic parsing of the AT commands that are sent from the modem
  - Provides a simple synchronization mechanism to all users of the services to wait and resend their requests when the modem is busy

## Services: Display Services

- **SRV\_LAYMAN.UDO**
  - Referred to as the Layout Manager
  - Knows what to draw and where, for a given layout
    - Layouts are defined using the Layout Editor tool
    - A layout specifies the position of widgets, text, and bitmaps
    - Only one layout is active at any time



## Services: Display Services (cont.)

- **SRV\_PLOTTER.UDO**

- Provides the glue between the MMI and the layout
- When a layout is initialized, it tells the plotter about all of its primitive elements
- Application draws these elements by calling plotter functions and specifying the layout element ID as a parameter



## Services: Display Services (cont.)

- **SRV\_DISPLAY.UDO**

- Provides all the drawing functionality required by the application
- All display changes ultimately are performed through this component
- Usually used via the plotter

## Services: Display Services (cont.)

- **SRV\_BITMAPS.UDO**
  - Project bitmaps and animations are managed using the Multimedia Resource Manager
  - Graphic files can be imported into an object array in SRV\_BITMAPS.UDO
  - IDS can be imported into a constant array in RPD\_CONSTANTS.UDO

## Widgets

- **WDG\_EDIT\_BOX**
  - Graphic widget for the editor
  - Draws text on the display according to the selected orientation
  - Draws the selected cursor in the correct place in the text
- **WDG\_LIST.UDO**
  - Handles list look-and-feel and navigation



## Widgets (cont.)

- **WDG\_SCROLLBAR.UDO**
  - Provides scrolling functionality for menus and lists
- **WDG\_SOFTKEY.UDO**
  - Handles the two soft keys
  - Draws the soft key labels
  - Triggers exported events when the keys are pressed



## Embedded Interface Components



- Always generated as “interface only”
- Objects and internal logic are not generated
- **EMB\_GSRV.UDO** (general services)
  - Contains interfaces to audio, battery, backlight, microphone, clock, registry, vibrator, and ear-bud
  - Provides functions for writing to and reading from persistent (flash) memory
  - Simulation panel allows the simulation user to modify the settings of the various devices

## Embedded Interface Components (cont.)

- **EMB\_KPD.UDO** (keypad)
  - Contains the interface between the hardware and the keypad
  - Nested in SRV\_KEYPAD.UDO, which provides a more convenient interface to the MMI
  
- **EMB\_AT.UDO** (modem)
  - Responsible for communication with the modem

## Miscellaneous Component


- **TRACE.UDO**
  - This component contains assert functions



**e-SIM MMI  
Reference Design 1.x  
Implementation Guidelines**

Company Confidential. Do not copy or distribute.

51



## **Contents**

- Naming conventions for modes, objects, and logic
- Required basic modes and interfaces

Company Confidential. Do not copy or distribute.

52

## Mode Names

- Each word is capitalized
- No underscores between words

**Example:** IntializationProcess

## User Object Names

- Prefix indicates the object type:
  - EMB for embedded components
  - SRV for service components
  - WDG for widget
  - HMI or WMI for MMI modules
- Prefix followed by an underscore





## User Object Names (cont.)

- Object name in uppercase
  - Two or more words are separated by an underscore
  - When an instance of a user object is added to a parent application, its UDO name is used

**Example:** SRV\_DISPLAY



## Object Names

- Each word is capitalized
- Name followed an underscore, followed by the object type:
  - Ary for array
  - Num for number
  - Bmp for bitmap
  - Pb for pushbutton
  - Ev for event
  - Str for string
  - Img for image
  - Tmr for timer
  - Int for integer
  - Tik for timer tick

**Example:**  
Send\_Pb

## Constant Set Names (cont.)

- Constant **item** names in uppercase
- Underscores separate each word

### **Examples:**

FUNCTION\_ID-Cs (a constant set)

CALL\_WAITING (a constant set item)

## User Function Names

- Capitalization used to separate adjacent words

**Example:** drawText:onElement:

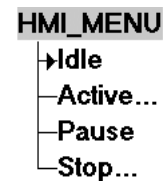
## Exported Event and Property Names

- Each word is capitalized
- Name followed an underscore, followed by the event/property type

**Example:** Selected\_Int

## Required Modes for MMI Modules

- Basic mode tree for each MMI module contains these modes:
  - **Idle:** component is awaiting activation
  - **Active:** contains entire functionality of the active component
  - **Pause:** all activity ceases. This mode is exited when the component is either resumed or stopped
  - **Stop:**



## Required Functions & Events

- Initialization User Function (init)
  - Initializes the MMI module's holders
- Start User Function (start)
  - Causes a transition from Idle mode to Active mode, using an internal event, Start\_Ev

## Required Functions & Events (cont.)

- Stop User Function (stop)
  - Causes a transition from Active mode or Pause mode to Idle mode
  - On the way from Active mode or Pause mode to Idle mode, the MMI module is responsible for closing the active layout



## Required Functions & Events (cont.)

- Stopped Event (Stopped\_Ev)
  - The result of calling the stop user function
  - Stopped\_Ev should be triggered only when there is no danger that the module would write to the display



## Required Functions & Events (cont.)

- Pause User Function (pause)
  - Initiates the Pausing process
  - Called by the main application when there is a high priority popup message that should be displayed
  - Can also be called by an HMI module for an active WMI module

## Required Functions & Events (cont.)

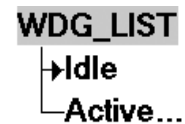
- Paused Event (Paused\_Ev)
  - The result of calling the pause user function
  - This event should be triggered only when the module is paused

## Required Functions & Events (cont.)

- Resume User Function (resume)
  - Resumes the module after it was paused
  - The module should restore the last state that was active before it was paused

## Required Modes for Widgets

- Basic mode tree for each widget component contains these modes:
  - **Idle:** component is awaiting activation
  - **Active:** contains the entire functionality of the active component



## Required Functions & Events

- Initialization User Function (init)
  - Initializes the widget's holders
  - Called by the Layout Manager, SRV\_LAYMAN.UDO
- Start User Function (start)
  - Causes a transition from Idle mode to Active mode, using an internal event, Start\_Ev

## Required Functions & Events (cont.)

- Stop User Function (stop)
  - Causes a transition from Active mode to Idle mode
  - Widget's responsibility is to clear its allocated area on the layout

## Required Functions & Events (cont.)

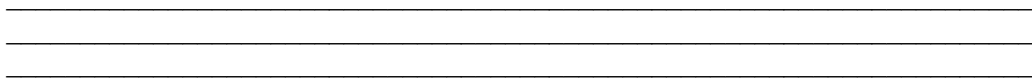
- Stopped Event
  - The result of calling the stop user function
  - Should be triggered only when there is no danger that the module would write to the display



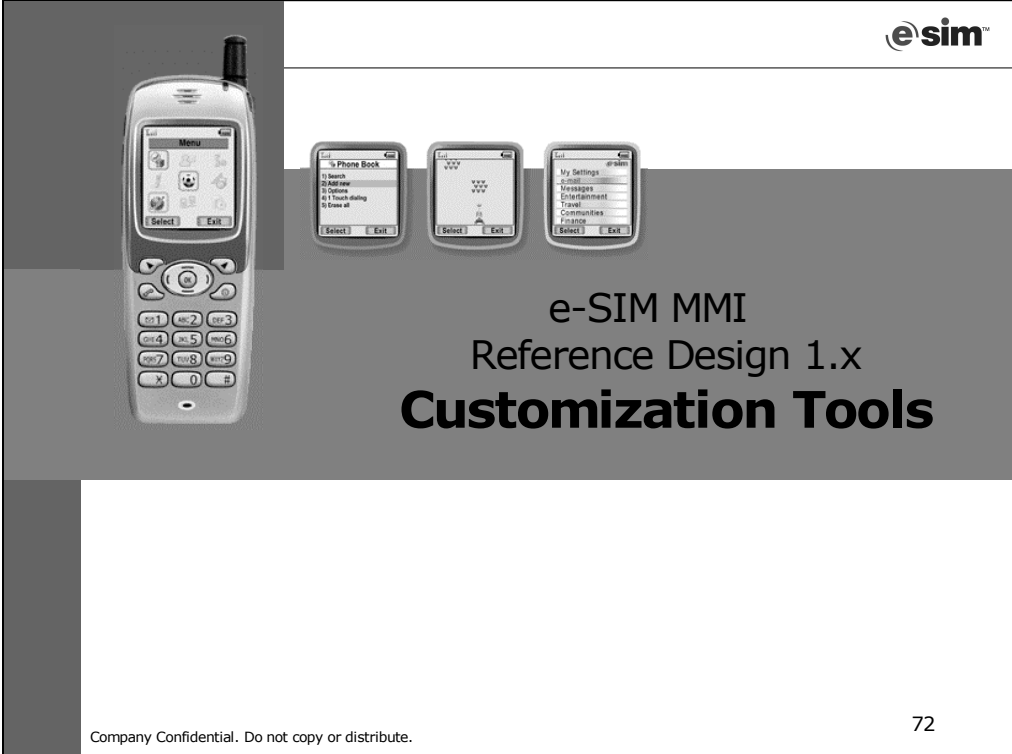


## Required Functions & Events (cont.)

- Set Rectangle User Function  
(setRectangle\_x:y:width:height:)
  - Sets the widget rectangle's position and size





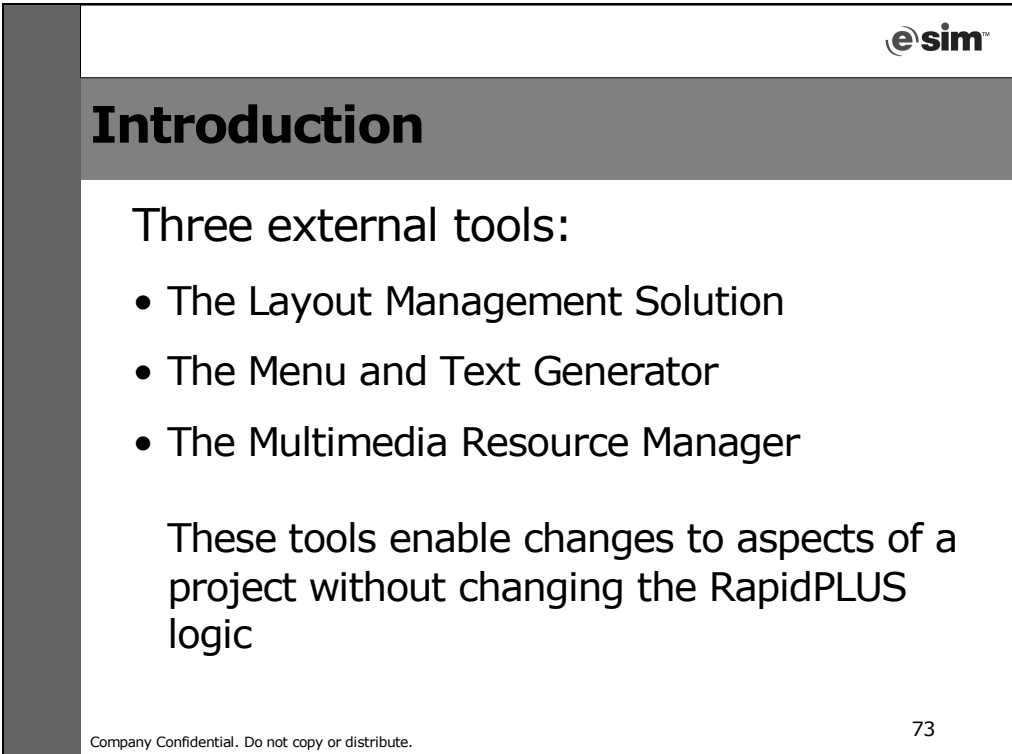


The image shows a mobile phone on the left and three small screenshots of the eSIM MMI interface on the right. The phone screen displays a 'Menu' with icons for Phone Book, Search, Options, Track dialing, Cross id, and Settings. The three screenshots show: 1) 'Phone Book' with fields for Search, Address, Track dialing, and Cross id; 2) a screen with '\*\*\*' and '\*\*\*' and a 'JA' button; 3) 'My Settings' with options for e-mail, Messages, Entertainment, Travel, Communities, and Finance.

**e-SIM MMI**  
Reference Design 1.x  
**Customization Tools**

Company Confidential. Do not copy or distribute.

72



The image shows a slide titled 'Introduction' with a list of three external tools. The eSIM logo is in the top right corner.

## Introduction

Three external tools:

- The Layout Management Solution
- The Menu and Text Generator
- The Multimedia Resource Manager

These tools enable changes to aspects of a project without changing the RapidPLUS logic

Company Confidential. Do not copy or distribute.

73

## Setting the eSim MMI

- Copy the LangAPI.DLL to c\Windows\System32
- Copy the 5 fonts from the fonts folder to controlPanel\Fonts
- Open the application from the RapidApp\_Simulation folder

## Setting the eSim MMI

- Open the TextResource.xls file
- Set the tools\security to Medium and reopen the file



## Setting the eSim MMI

- Set the path as:

3	Character Set	Unicode
4	RAR Files:	Λ
5	C and H Files:	Λ
6		
7		
8	File Name:	GraphicResourceMan.xls
9	File Location:	.\Images
10	Images Worksheet Name:	GraphicResources
11	Images Column Title:	Image ID name
12	Animations Worksheet Name:	Animations
13	Animations Column Title:	Animation ID Name
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		

Ready



## The Layout Management Solution

A two-part solution for defining and arranging the visual elements that appear on a product's screen

## Layout Management Solution

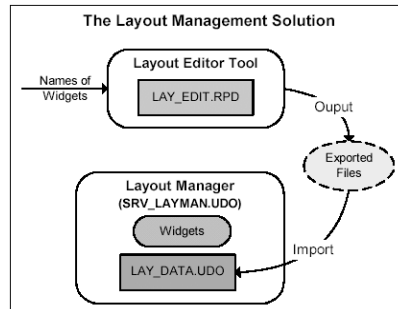
- A **layout** is a group of visual elements that have defined positions and sizes on a product's screen
- Layouts are designed in an external RapidPLUS application—the **Layout Editor** tool
- The Layout Editor generates data files that are used by the RapidPLUS project

## Layout Management Solution (cont.)

- The layouts are managed during runtime using the service component, SRV\_LAYMAN.UDO, which is referred to as the **Layout Manager**

## Layout Management Solution (cont.)

- The architecture



- Exported files, located in the Layout\_Editor\export folder:

- lo\_vt\_data.rar
- lo\_vt\_entry.rar
- lo\_vt\_enum.h

## The Menu and Text Generator

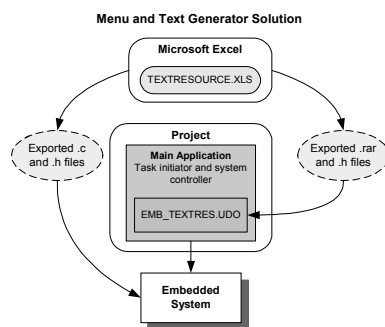
Manages multilingual menus and text resources

## Menu and Text Generator

- Based on an Excel workbook that contains all of a project's strings, translations, and menu hierarchy information
- Contains macros for generating data files that are used by the RapidPLUS project and the embedded system

## Menu and Text Generator (cont.)

- The architecture



- Exported files for the RapidPLUS project:

- CodePage.rar
- Menu\_Index.rar
- Menu\_Unicode.rar
- LoadTo\_ConstantSet.h

- Exported files for the embedded system:

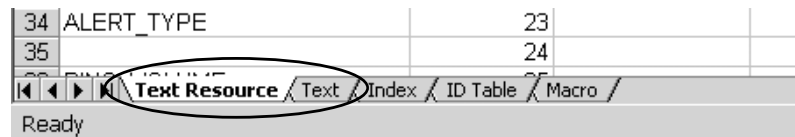
- cLng\_Chinese\_Simplified.c
- cLng\_English.c
- cLng\_French.c
- cLng\_German.c
- LoadTo\_ConstantSet.h
- xl\_IndexArray.c
- xl\_menu.h



## Menu and Text Generator (cont.)

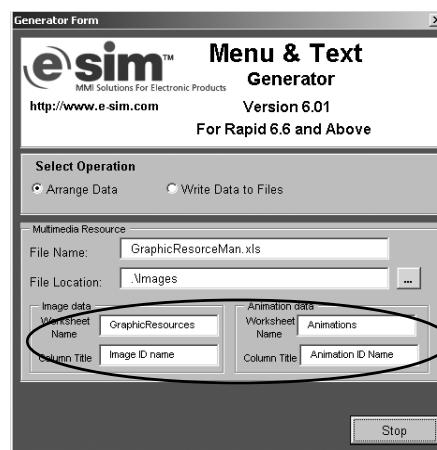
The Excel file:

- Change only the Text Resource and Text worksheets (tabs)



## Menu and Text Generator (cont.)

- Run the macro at the end



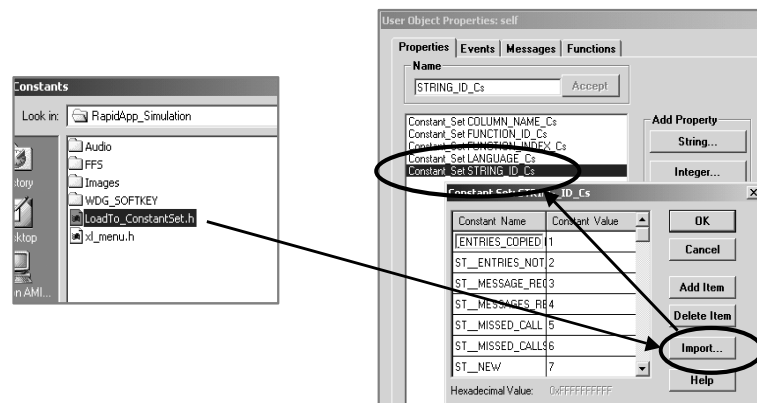
For more details, refer to the Menu and Text Generator Solution.pdf

## Menu and Text Generator (cont.)

- The output file LoadTo\_ConstantSet.h contains:
  - String IDs—for string usage
  - Function Index—for the top list item at each submenu
  - Function IDs—function names to be executed when activating a specific feature

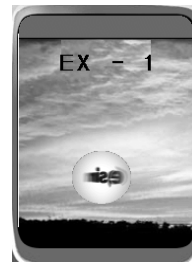
## Menu and Text Generator (cont.)

- Import constants into EMB\_TXTRES.UDO:



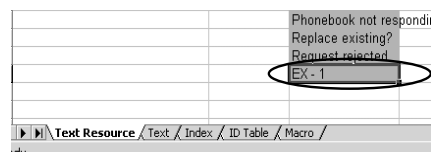
## Exercise 1: Text Generator

- Change the welcome message to EX - 1
- Change the positions of the message and animations



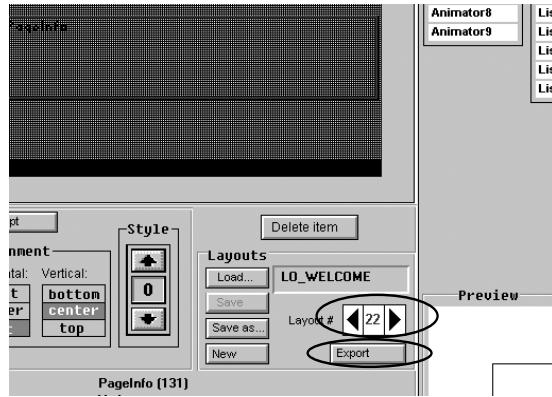
## The Solution: Step A

- Add a new text string to the Excel workbook, TextResource.xls
- Run the macro



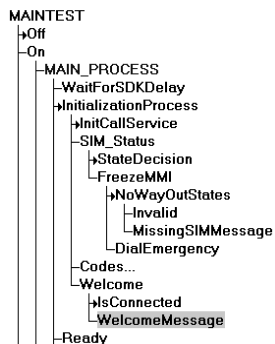
## The Solution: Step B

- Change Layout #22
- Export the layout



## The Solution: Step C

- Import constant set into EMB\_TXTRES.UDO
- Edit the code inside MAINTTEST.RPD





## The Multimedia Resource Manager

Manages bitmaps and animations

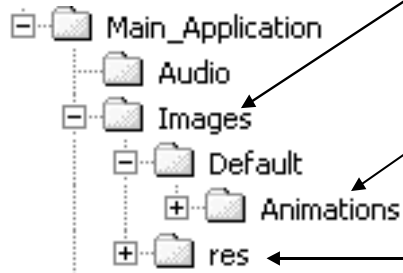


## Multimedia Resource Manager

- Based on an Excel template that contains macros for generating IDs for all of the bitmaps
- These generated IDs are used by several RapidPLUS components
- All of the project bitmaps are organized in a separate folder

## Multimedia Resource Manager (cont.)

- Folder hierarchy



All the bitmaps that are used in a project must be all together in a subfolder

Animations must be in a subfolder that holds only animation files

Used by the Excel template as the destination folder

## Multimedia Resource Manager (cont.)

- Bitmap file names

- File names must not contain blank spaces
- Each bitmap must have a unique name, even though the original bitmap files are located in different folders
- The file names will be used in the constant set in RPD\_CONSTANTS.UDO

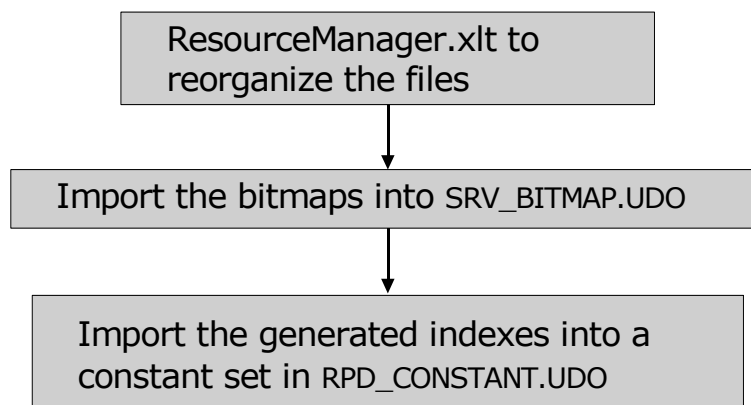


## Multimedia Resource Manager (cont.)

- **SRV\_BITMAP.UDO**
  - Contains an array of objects that holds the bitmaps, which are arranged by index number (as defined in the Excel worksheet)
  - Array elements are imported from the bitmaps in the destination folder
- **RPD\_CONSTANTS.UDO**
  - Has a constant set that contains a constant item for each bitmap. The constant set items are imported from the Excel-generated .h file



## Multimedia Resource Manager (cont.)



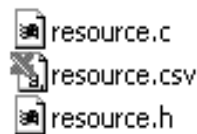
## Multimedia Resource Manager (cont.)

ResourceManager.xlt to reorganize the files

Make sure the paths are correct

	B	C	D	E
Source folder	D:\backups\2_ref_design\rapidappell\rapidapp_simulation\images			
Destination folder	D:\Courses\Installation\MMI_1_2_Phase1(1.06)\esim_mmi\RapidApp\images\res			
Animation folder	D:\backups\2_ref_design\rapidappell\rapidapp_simulation\images\default\animations			
name	ID	Source subfolder	File Name	default

Files generated to the res folder



## Multimedia Resource Manager (cont.)

Import

- Copies all of the bitmaps to the Destination folder (res)
- Imports all of the bitmaps into Excel and assigns an ID name and number to each bitmap
- Creates various files, including an .h file that can be imported into RapidPLUS
- Organizes the animation folders into a separate worksheet





## Multimedia Resource Manager (cont.)

Import the bitmaps into SRV\_BITMAP.UDO

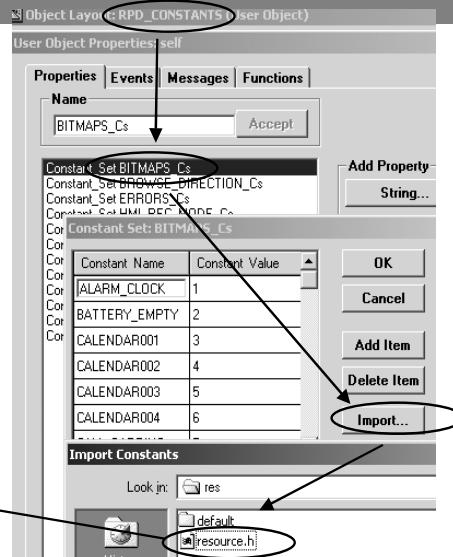
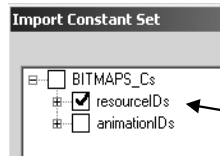
If you add a new bitmap to the Source folder:

1. Use the File|Import Bitmaps command to import all of the images from the res\default folder
2. In the "Import Bitmaps from" dialog box, select the Link to File option

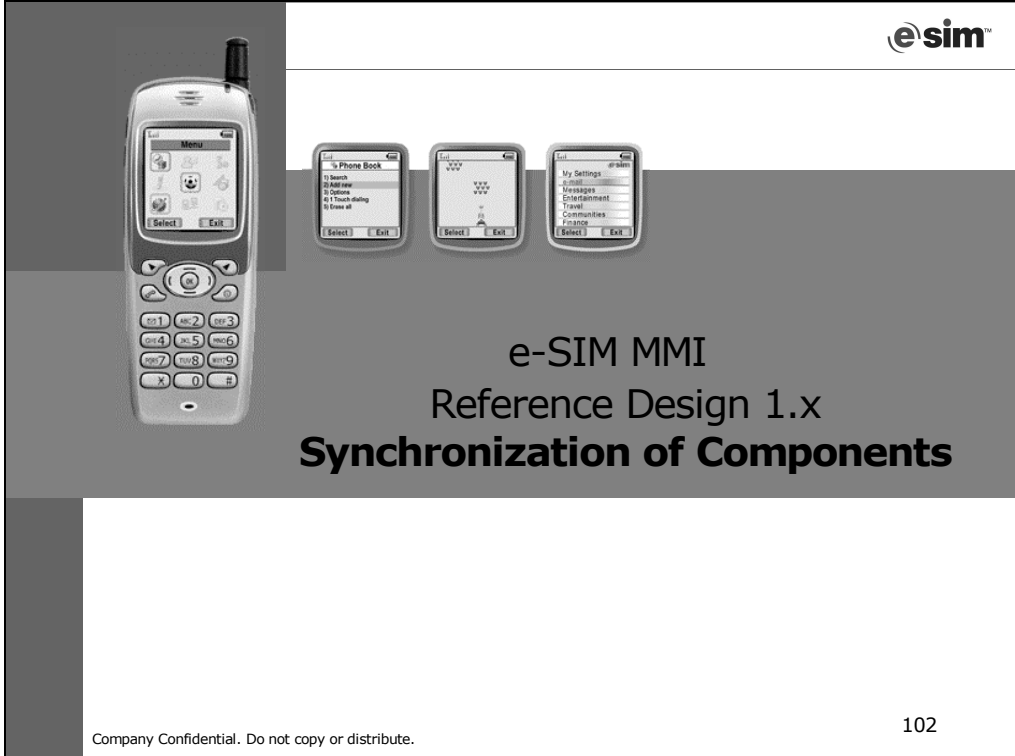


## Multimedia Resource Manager (cont.)

Import the generated indexes into a constant set in RPD\_CONSTANT.UDO



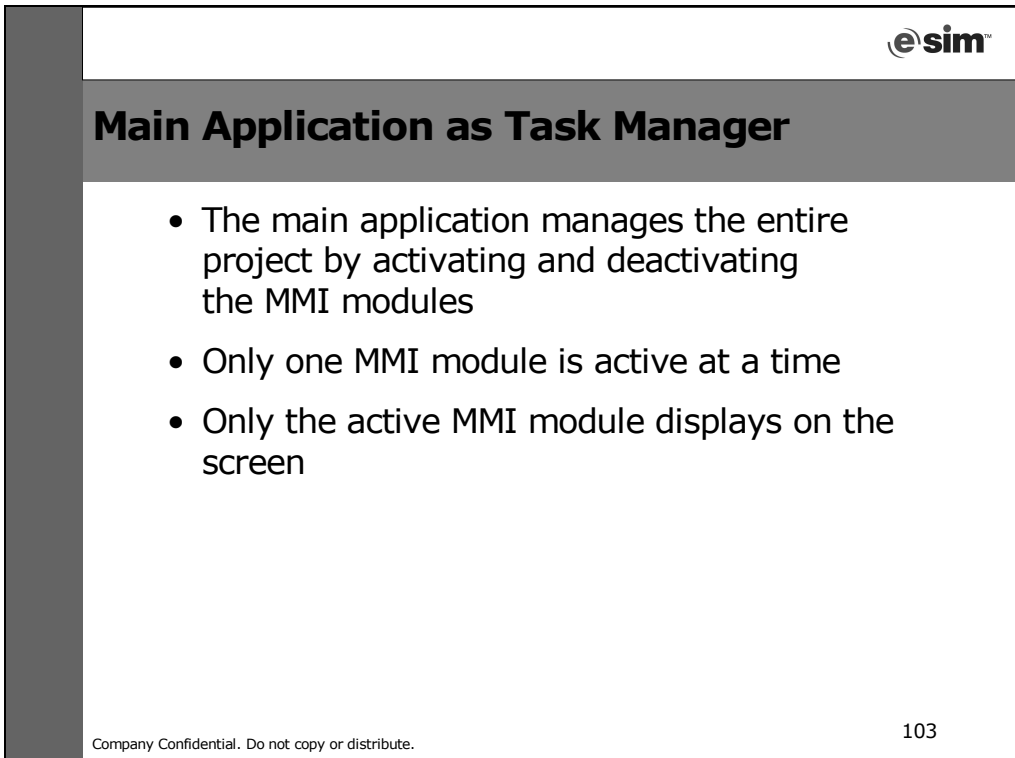




The image shows a mobile phone on the left and three small screens on the right, each displaying a different menu. The phone screen shows a 'Menu' with icons for Phone, Messages, Entertainment, Travel, Communities, and Finance. The first small screen shows a 'Phone Book' menu with options: 1) Search, 2) Address, 3) Contacts, 4) Touch dialing, 5) Close all. The second small screen shows a 'WWW' menu with options: WWW, WWW, WWW. The third small screen shows a 'My Settings' menu with options: e-SIM, Messages, Entertainment, Travel, Communities, Finance. Below the screens, the text reads: e-SIM MMI Reference Design 1.x Synchronization of Components. The eSIM logo is in the top right corner.

Company Confidential. Do not copy or distribute.

102



The image shows a slide with the eSIM logo in the top right corner. The main title is 'Main Application as Task Manager'. Below the title is a bulleted list of three points. The text at the bottom left reads 'Company Confidential. Do not copy or distribute.' and the page number '103' is at the bottom right.

**Main Application as Task Manager**

- The main application manages the entire project by activating and deactivating the MMI modules
- Only one MMI module is active at a time
- Only the active MMI module displays on the screen

Company Confidential. Do not copy or distribute.

103

## Main Application as Task Manager (cont.)

- To activate one module at a time, the “active” mode (named MainMMI) is organized into exclusive modes, one for each MMI module
- When the active MMI module’s task has been completed it terminates itself and notifies the main application (by triggering the Stopped\_Ev exported event)
- The main application receives Stopped\_Ev and returns to its Idle mode

## Working with Widgets

- Widgets reside under SRV\_LAYMAN.UDO, the Layout Manager service component
- Widgets are controlled by the Layout Manager and by MMI modules that require the widgets’ functionalities

```
SRV_LAYMAN      I
EMB_TEXTRES
LAY_DATA
SRV_DISPLAY
  SRV_BITMAPS_2
SRV_KEYPAD
EMB_KPD
Srv_Plotter
PLOTTER_DATA
WDG_ANIMATOR
ANIM_DATA
WDG_CHARPICKER
WDG_EDIT_BOX
WDG_LIST
WDG_SCROLLBAR
WDG_SOFTKEY
```



## Working with Widgets (cont.)

When an MMI modules requires a widget:

1. When the `init_layout` function is called, each widget that participates in the layout is initialized by the Layout Manager
2. Each widget is specifically initialized by the MMI module with the relevant data that is required for operation
3. Each widget is started by the Layout Manager immediately after the Layout Manager's `start` function is called by the MMI module



## Working with Widgets (cont.)

4. After a layout is started, the MMI module directly manipulates the widget according to the widget's specific interface
5. When the layout is no longer required, the MMI module calls the Layout Manager's `stop` function, which initiates each widget's Stopping process

## Pause and Resume Mechanism

- Pause and resume mechanisms allow high priority popup messages to be displayed
- When an active MMI module is interrupted for a popup message, the main application:
  1. Creates a list of MMI modules to be paused. Only modules that are active—and not paused—at the moment the process begins would be entered on the list

## Pause and Resume Mechanism (cont.)




2. Creates a list of objects to be resumed
3. Pauses relevant components
4. Displays the popup message
5. Resumes the relevant modules
6. Restores the state to where it was before the pause process began

## MMI Modules - Notes

- The MMI modules should have no graphic action in its pause mode
- When the module is resumed, it may use a deep history transition to return to the previous state
- Initializing widget data should be performed on actions of the transition entering the mode and not on entry activities, in order to be able to re-enter the mode without re-initializing the data
- Avoid using a junction mode (may cause a conflict with the deep history transition)








## e-SIM MMI Reference Design 1.x The Draw Method

Company Confidential. Do not copy or distribute.

111



## Objectives

- How to draw
- Add a new layout

Company Confidential. Do not copy or distribute.

112

## How to Draw in the MMI



## How to Draw in the MMI (cont.)

- An HMI module calls the plotter service using the function:  
Plotter drawText: onElement:
- To use the plotter you need to specify a layout
- The plotter calls the Display



## Order Of Execution: Draw

### MAINTEST.RPD/HMI:

```

Activities
ityr \--- Changed by Timur
ityr \MAINTEST drawWelcomeMessage
ityr \
ityr \GreetingMessage_Tick restart
ityr \
ityr \ Start the services intialization.
ityr InitServices_Ev trigger
ityr \
ityr CurrentLayoutID_Int ←SRV_LAYMAN initLO_Welcome_plotter: Plotter animator: Animator
ityr Plotter drawText: (TextResource.getStringByID: TextResource.STRING_ID-Cs.ST_WELCOME_) onElement: Plotter.)
ityr \--- Start animation
ityr Animator setRepetition: Animator.REPEATITION-Cs.FOREVER
ityr Animator setAnimation: Animator.ANIMATION_TYPE-Cs.LOGOANIM
ityr \--- Start layout
ityr SRV_LAYMAN start: CurrentLayoutID_Int
ode
    
```



## Order Of Execution: Draw (cont.)

### SRV\_LAYMAN.UDO:

```

Integer | initLO_Idle1_plotter: <SRV_PLOTTER_Holder:hPlotter>
\
SRV_DISPLAY setBackgroundImage: Constants.BITMAPS-Cs.LO_POPUP2_BG
WDG_SOFTKEY init: SRV_DISPLAY keypad: SRV_KEYPAD
\
<hPlotter> hold: Srv_Plotter
\
\Icons start
if self initPlotter: self.LAYOUT_ID_C_0_IDLE1 <> 0
return self.LAYOUT_ID_C_0_IDLE1
return 0
    
```

## Order Of Execution: Draw (cont.)

### SRV\_LAYMAN.UDO:

Update the plotter on which type of elements it will hold

```

Integer      initPlotter: <Integer:layID>
>           for <Integer:i> from 1 to <numElems> step 1
>>         LAY_DATA getElement: <i> ofLayout: <layID> id: tempID x: tempX y: tempY width: te
>>         if self is pictureType: tempID
>>>         Srv_Plotter addPictureElement: tempID x: tempX y: tempY width: tempWidth height: t
...         else if self is textType: tempID
>>>         Srv_Plotter addTextElement: tempID x: tempX y: tempY width: tempWidth height: t
    
```

## Order Of Execution: Draw (cont.)

### MAINTEST.RPD/HMI:

```

Activities
ityr | W— Changed by Timur
ityr | \MAINTEST drawWelcomeMessage
ityr | W
ityr | \GreetingMessage_Tick restart
ityr | W
ityr | \ Start the services initialization.
ityr | InitServices_Ev trigger
ityr | W
ityr | CurrentLayoutID_Int := SRV_LAYMAN initID_Welcome_plotter: Plotter animator: Animator
ityr | Plotter drawText: (TextResource getStringByID: TextResource.STRING_ID-Cs.ST_WELCOME_) onElement: Plotter
ityr | W— Start animation
ityr | Animator setRepetition: Animator.REPEATITION-Cs.FOREVER
ityr | Animator setAnimation: Animator.ANIMATION_TYPE-Cs.LOGOANIM
ityr | W— Start layout
ityr | SRV_LAYMAN start: CurrentLayoutID_Int
ode
    
```

## Order Of Execution: Draw (cont.)

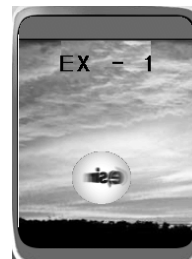
### PLOTTER.UJO:

```

Integer drawText:<String:text>onElement:<Integer:elementID>
// Draw the text for the text element specified by elementID.Return 1 if succeeded. Return
//
declare <Integer:cnt>
> <cnt> := PLOTTER_DATA.AllElements.textCount
> for <Integer:i> from 1 to <cnt> step 1
>> if PLOTTER_DATA.AllElements.textElement[ <i> ].id = <elementID>
>>> Display clearAreaAt: PLOTTER_DATA.AllElements.textElement[ <i> ].Xy: PLOTTER_
>>> Display selfStyle: PLOTTER_DATA.AllElements.textElement[ <i> ].style
>>> Display drawText: <text> tofitRect: PLOTTER_DATA.AllElements.textElement[ <i> ].X
>>> PLOTTER_DATA.AllElements.textElement[ <i> ].text := <text>
>>> self updateDisplay
>>> return 1
return 0
    
```

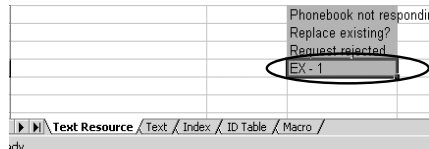
## Exercise 1: Draw Method

- Change the welcome message to EX - 1
- Change the positions of the message and animations



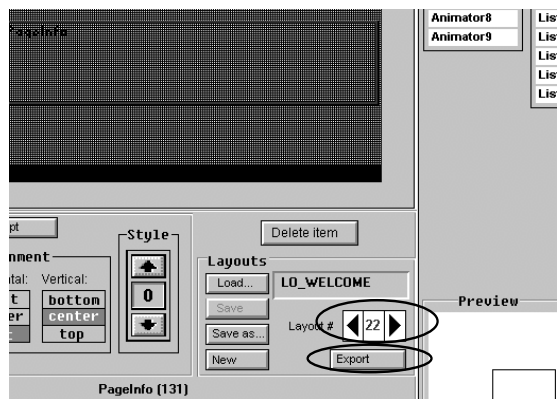
## The Solution: Step A

- Add a new text string to the Excel workbook, TextResource.xls
- Run the macro



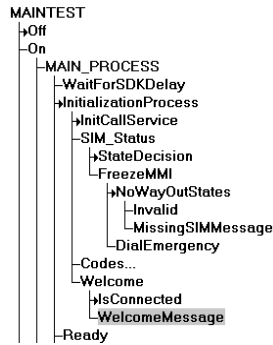
## The Solution: Step B

- Change Layout #22
- Export the layout



## The Solution: Step C

- Import constant set into EMB\_TXTRES.UDO
- Edit the code inside MAINTEST.RPD



## Draw - summary

- Example for normal page:

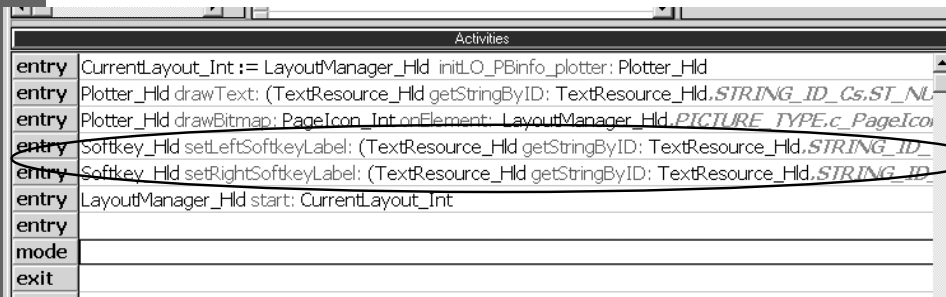
Activities	
entry	CurrentLayout_Int := LayoutManager_Hld initLO_PBInfo_plotter: Plotter_Hld
entry	Plotter_Hld drawText: (TextResource_Hld getStringByID: TextResource_Hld,STRING_ID-Cs.ST_NL
entry	Plotter_Hld drawBitmap: PageIcon_Int onElement: LayoutManager_Hld,PICTURE_TYPE.c_PageIco
entry	Softkey_Hld setLeftSoftkeyLabel: (TextResource_Hld getStringByID: TextResource_Hld,STRING_ID_
entry	Softkey_Hld setRightSoftkeyLabel: (TextResource_Hld getStringByID: TextResource_Hld,STRING_ID_
entry	LayoutManager_Hld start: CurrentLayout_Int
entry	
mode	
exit	

•Close the Layout:



Using the Soft keys:

•Use inside a layout





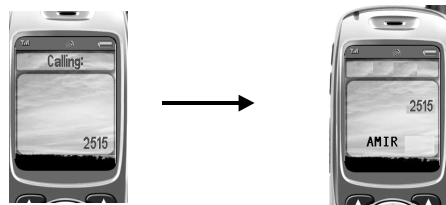
## Using the Soft keys:

- Use inside a Menu

K	L	M	N	O	
		String	String	Integer	String
el9	Level10	LeftSK	RightSK	Layout	Image ID na
		OK	Back	23	
		OK	Back	23	
		OK	Back	23	
		Select	Back	23	
		Select	Back	23	CALL_BARRI
		Select	Back	23	
		OK	Back	23	
		OK	Back	23	

## Exercise 1A: Draw Method

- Change the OutgoingCall screen Title to: your name
- Change the positions of the Title and TextBox

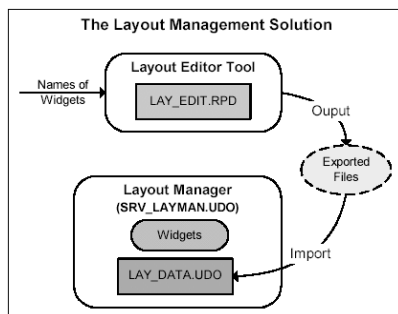


## Exercise 2: Add a New Layout

- The layout will contain a TextBox and a bitmap
- Use the layout for the welcome message screen

## Exercise 2: Add a New Layout (cont.)

- The architecture
- The exported files

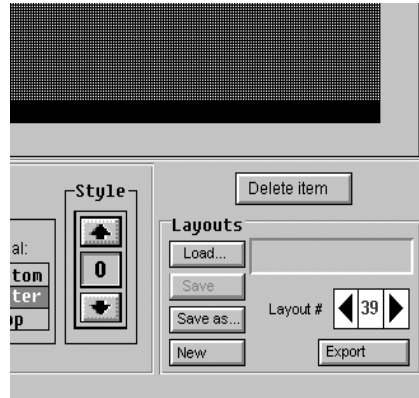


- lo\_vt\_data.rar
- lo\_vt\_entry.rar
- lo\_vt\_enum.h

(located in the Layout\_Editor\export folder)

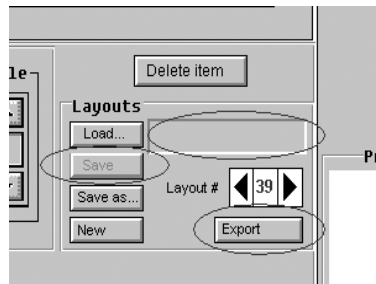
## The Solution: Step A

- Create new layout in the Layout Editor
- Place the TextBox and bitmap elements



## The Solution: Step B

- Name the layout
- Save the project, then export it

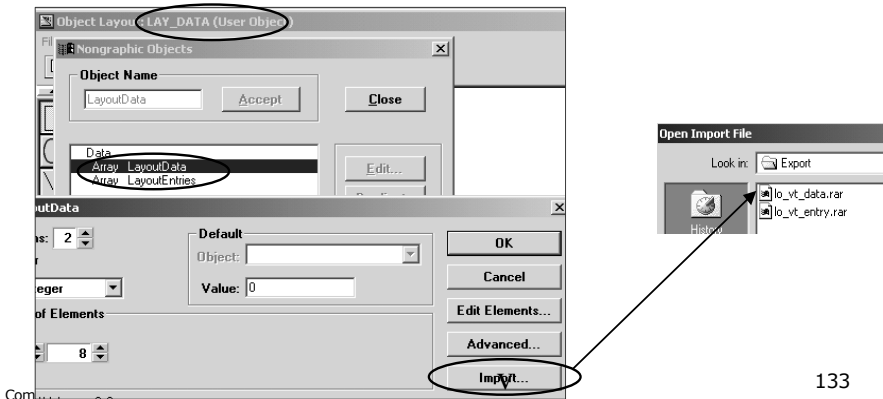


(The exported files are updated)

## The Solution: Step C

- **LAY\_DATA.UDO:**

- Update the LayoutData array by importing lo\_vt\_data.rar
- Update the LayoutEntries array by importing lo\_vt\_entry.rar

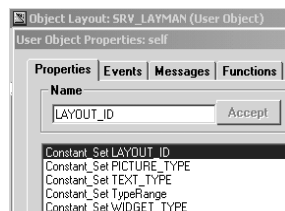


133

## The Solution: Step D


- **SRV\_LAYMAN.UDO**

- The constant set properties: LAYOUT\_ID, PICTURE\_TYPE, TEXT\_TYPE, and WIDGET\_TYPE have corresponding enumerated sets in the file lo\_vt\_enum.h.
- If you have only added a new layout, only LAYOUT\_ID must be updated



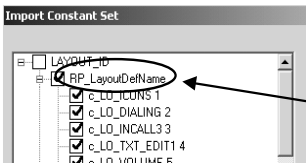
Company Confidential. Do not copy or distribute.

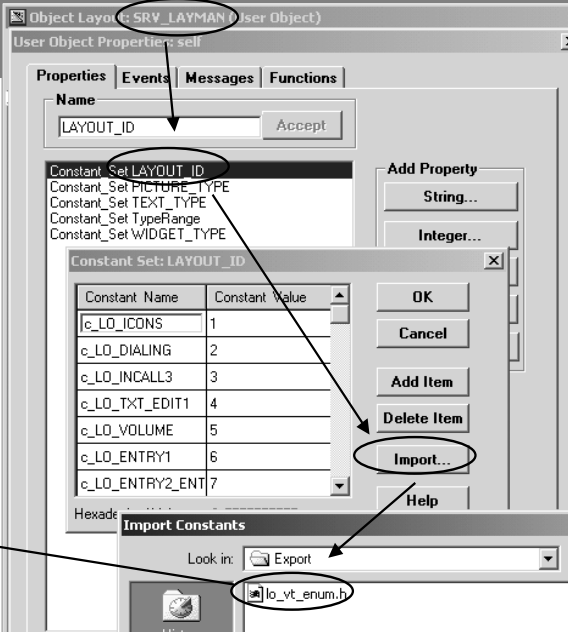
134




## Step D (cont.)

- Import lo\_vt\_enum.h
- Select RP\_LayoutDefName



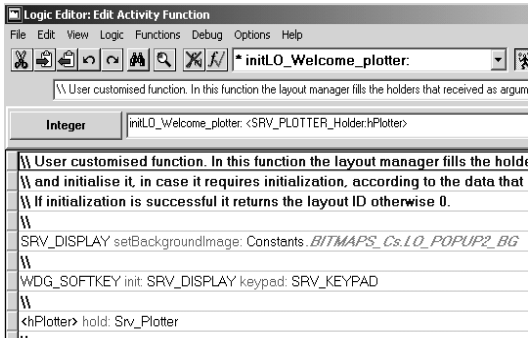


Company Confidential. Do not copy or distribute.



## The Solution: Step E

- **SRV\_LAYMAN.UDO**
  - Add an init function for the layout



- **MAINTEST.RPD**

```

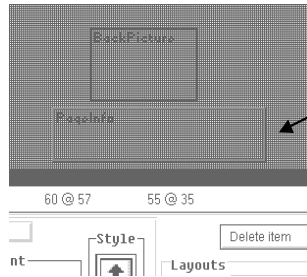
ntry CurrentLayoutID_Int := SRV_LAYMAN initLO_Welcome_plotter Plotter animator. Animator
ntry
ntry Plotter drawText: (TextResource getStringByID: TextResource.STRING_ID-Cs.ST_WELCOME_) onElement: Plotter.TEXT_TYPE
ntry
ntry Plotter drawBitmap: Constants.BITMAPS-Cs.ALARM_CLOCK onElement: Plotter.PICTURE_TYPE.c_BackPicture
ntry
    
```

Company Confidential. Do not copy or distribute.

## The Solution: Step F

- Use the drawText: function with the correct layout element
- MAINTEST.RPD:

```
entry CurrentLayoutID_Int :=SRV_LAYMAN initLO_Welcome_plotter: Plotter animator: Animator  
entry Plotter drawText: (TextResource getStringByID: TextResource.STRING_ID-Cs.ST_WELCOME_) onElement Plotter.TEXT_TYPE.c_PageInfo  
entry Plotter drawBitmap: Constants.BITMAPS-Cs.ALARM_CLOCK onElement Plotter.PICTURE_TYPE.c_BackPicture
```

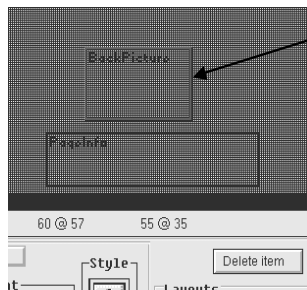


Company Confidential. Do not copy or distribute.

## The Solution: Step G

- Use the drawBitmap: function with the correct layout element
- MAINTEST.RPD:

```
entry CurrentLayoutID_Int :=SRV_LAYMAN initLO_Welcome_plotter: Plotter animator: Animator  
entry Plotter drawText: (TextResource getStringByID: TextResource.STRING_ID-Cs.ST_WELCOME_) onElement Plotter.TEXT_TYPE.c_PageInfo  
entry Plotter drawBitmap: Constants.BITMAPS-Cs.ALARM_CLOCK onElement Plotter.PICTURE_TYPE.c_BackPicture
```

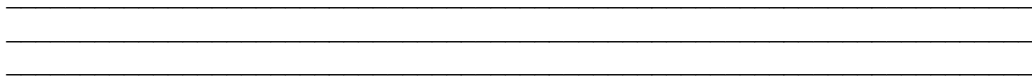


Company Confidential. Do not copy or distribute.





## Summary

You have learned to modify existing layouts and add new layouts!










e-SIM MMI  
Reference Design 1.x  
**Menu Method**

Company Confidential. Do not copy or distribute.

140



## Objectives

- Learn how to change the menu
- Learn how to add a menu leaf
- Learn how to add a new Options Menu

Company Confidential. Do not copy or distribute.

141

## Types of Menu

- The animation icons menu- Main Menu
- The Submenu
- Options menu- specific for each MMI

## MMI Modules

### **HMI\_MENU**

- Handles the main menu and the submenu
- After selecting a submenu the selected HMI will start its own menu

### **WMI\_MENU**

- Handles navigation to all menu items
- Once activated, menu navigation becomes automatic



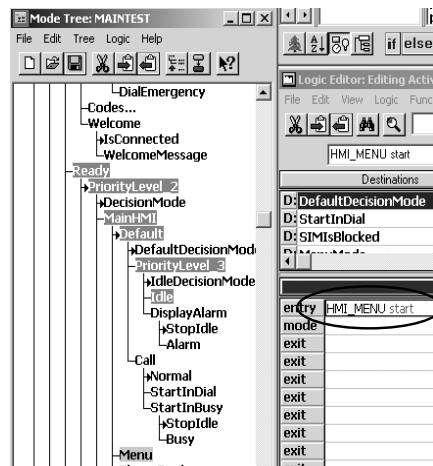
## The Menu Mechanism

# Main Menu



## The Menu Mechanism - Main Menu

- First call from MAINTTEST.RPD:



## The Menu Mechanism - Main Menu (cont.)

•HMI\_MENU:

Destinations	Event	Condition
D:Messages		& MainMenuIndex_Int = MAIN_ME
D:CallRecords		
D:Settings		

```

entry \\ Init main menu layout
entry CurrentOpenLayoutID_Int :=LayoutManager initLO_MenuMain_plotter: Plotter animator: Animator
entry \\ Set the soft keys for this menu level
entry Softkey setLeftSoftkeyLabel: (TextResource getStringByID: TextResource.STRING_ID-Cs.ST_SELECT)
entry Softkey setRightSoftkeyLabel: (TextResource getStringByID: TextResource.STRING_ID-Cs.ST_EXIT)
entry \\ Start layout
entry LayoutManager start: CurrentOpenLayoutID_Int
    
```

## The Menu Mechanism - Main Menu (cont.)

HMI\_MENU:

- A main menu item is implemented as a mode in HMI\_MENU.UDO
- Navigation of the main menu is done using WMI\_MENU\_PRIVATE
- Implementation of menu functionality is done in HMI\_MENU.UDO

```

HMI_MENU
  Idle
  Active
    MenuNavigation
      MainMenuNavigation
        Messages
        CallRecords
        Settings
        Games
        NetworkServices
        LockKeypad
        Applications
        PhoneBook
        Calendar
      SubMenuActivated
        SubMenuNavigation
        GoToMainMenu
        GoToFunction
    LeafFunctions
      HMI_SMS
      HMI_REC
      NetworkFunctions...
      ChangeBarringCode...
      LoadDefaultSettings
      GreetingMessage...
      ChangePINCode...
    
```



## The Menu Mechanism

# Submenu

Company Confidential. Do not copy or distribute.

148



## The Menu Mechanism - SubMenu

Destinations	Event	Condition	Actions
D: CallRecords	LayoutManager Stopped_ev		MenuPrivate start: TextResource.FUNCTION_INDEX-Cs.IX-INBOX
D: LockKeypad			
D: SubMenuActivated			

Activities

- entry: HMI\_MENU displayMainMenuItem: TextResource.STRING\_ID-Cs.ST\_MESSAGES
- mode: LayoutManager initAnimator\_LayoutId: CurrentOpenLayoutID\_Int animator: LayoutManager.WIDGET\_TYPE.c\_Animator1
- exit
- exit

- Transition to SubMenuActivated mode allows navigation in a submenu by start MENU\_PRIVATE (WMI\_MENU.UDO)

Company Confidential. Do not copy or distribute.

149

**e sim™**

## The Menu Mechanism - SubMenu (cont.)

•WMI\_MENU starts the new menu from the first function index item

The screenshot shows a menu grid with columns labeled Level1, Level2, and Level3. The 'Messages' cell in the Level2 column has a dropdown menu open, listing 'Inbox', 'Outbox', 'Write messages', 'Message settings', and 'Voice messages'. A callout box points from the 'Cs.IX\_INBOX' text in the Actions bar to the 'Inbox' item in the dropdown menu.

150

Company Confidential. Do not copy or distribute.

**e sim™**

## The Menu Mechanism - SubMenu (cont.)

•When WMI\_MENU navigation reaches Menu Leaf it will turn the HMI\_MENU to the function mode

Mode Tree: WMI\_MENU (User D...)

Logic Editor: Edit Activity Function

File Edit View Logic Functions Debug Options Help

getSubMenu

>WMI\_MENU.LeafFunctionSelected\_Ev trigger

<no return> getSubMenu

```

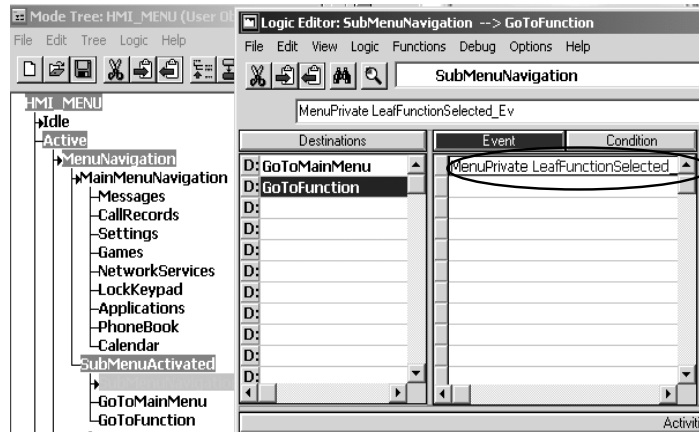
\\ Move down to the sub menu and save the information in the history array
\\
MenuHistory_Ary[ MenuHistoryLevel_Int , HISTORY_ARRAY_FIELDS_Cs.LIST_FIRST ] := List.First_Int
MenuHistory_Ary[ MenuHistoryLevel_Int , HISTORY_ARRAY_FIELDS_Cs.LIST_SELECTED ] := List.Selected_Int
MenuHistory_Ary[ MenuHistoryLevel_Int , HISTORY_ARRAY_FIELDS_Cs.FUNC_ID ] := FirstListID_Int
\\ get function ID and the next sub menu
WMI_MENU.LeafFunctionID_Int := TextResource.getItemFuncIDByListIndex: List.Selected_Int startIn: FirstListID_Int
FirstListID_Int := TextResource.getSubMenuIDByListIndex: List.Selected_Int startIn: FirstListID_Int
if FirstListID_Int = 0
>
\\ If this item has no children, the menu should call the function
> FirstListID_Int := MenuHistory_Ary[ MenuHistoryLevel_Int , HISTORY_ARRAY_FIELDS_Cs.FUNC_ID ]
> WMI_MENU.LeafFunctionSelected_Ev trigger
    
```

1

Company Confidential. Do not copy or distribute.

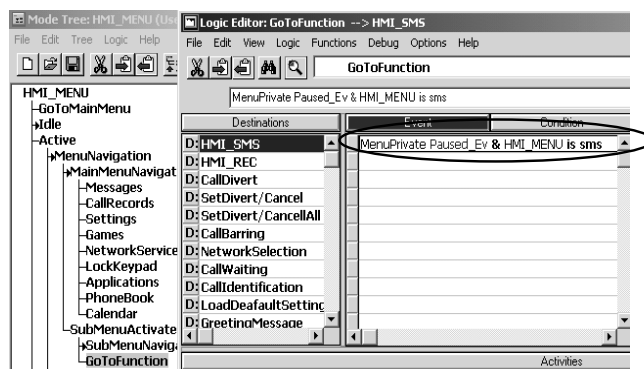


## The Menu Mechanism - SubMenu (cont.)



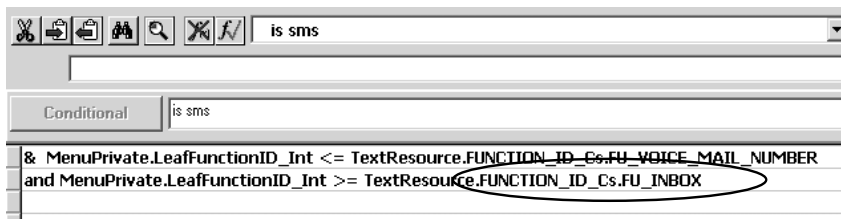
## The Menu Mechanism - SubMenu (cont.)

- Combination of WMI\_MENU\_PRIVATE's paused event and a condition on its leaf function ID property



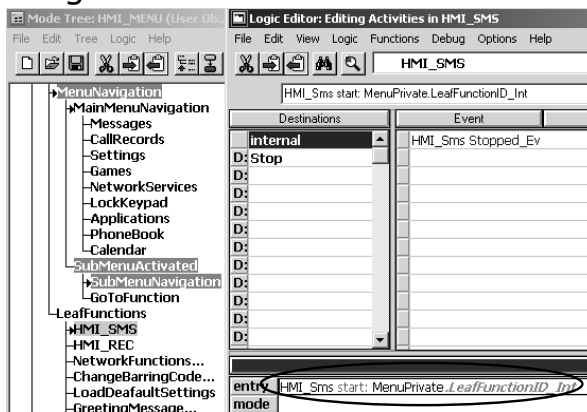
## The Menu Mechanism - SubMenu (cont.)

- Leaf function ID should equal the function ID that you assigned to the implemented function from FUNCTION\_ID-Cs (located in EMB\_TEXTRES.UDO)



## The Menu Mechanism - SubMenu (cont.)

- Starting the HMI



- From that point the HMI will take control

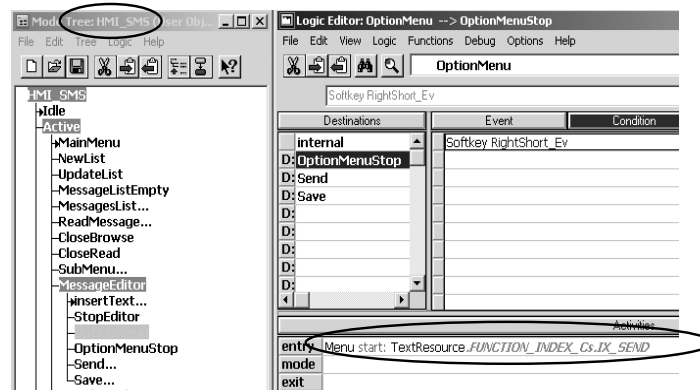


## More Mechanisms

Add Options Menu  
Change Menu Item  
Add Menu Item

## Add Options Menu

- Each HMI will start its own options menu



## Change Menu Item

- Change the string in TextResource.xls

Function ID	Level1	Level2	Level3	Level4
	Main menu			
1		Messages		
2			Inbox	
3			Outbox	
4			Write messages	
			Message settings	
5			Voice messages	Service center num
6				Listen to voice me
7				Voice mail numbe

## Add Menu Item

- Insert the new item

Function ID	Level1	Level2	Level3	Level4
	Main menu			
1		Messages		
2			Inbox	
3			Outbox	
4			Write messages	
			Message settings	
5				Service center nu
6				Listen to voice me
7				Voice mail numbe

- Don't fill the FunctionName and the Function\_ID columns



## Add Menu Item (cont.)

- Fill the following fields:  
LeftSK , RightSK, Layout,ImageID,Animation ID




L	M	N	O	P	Q	R
	String	String	Integer	String		String
<b>Level10</b>	<b>LeftSK</b>	<b>RightSK</b>	<b>Layout</b>	<b>Image ID name</b>		<b>Animation ID Name</b>
	Select	Exit		14 LO LSTICONT MESSAGES ICON		MESSAGES
	Select	Back		17 INBOX		
	Select	Back		17 OUTBOX		



## Menu Summary 1

- Example for menu managing:

Activities	
entry	<input type="text"/>
entry	Menu_Hld start: TextResource_Hld.FUNCTION_INDEX-Cs.IX_PBDEL_PH
mode	
exit	Menu_Hld stop
exit	



WMI\_MENU\_PRIVATE

## Optional

- you can use another instance of wmi\_menu, by init your HMI with the WMI\_MENU\_PRIVATE (it is a copy of the wmi\_menu


MainTest.RPD :

```

\\
HMI_PBOOK initEMB: EMB_GSRV settings: UDD_SET
HMI_PBOOK initServices: SRV_PBOOK layOutManager: SRV_LA
HMI_PBOOK initWMI: WMI_PB_BROWSE menu: WMI_MENU_e
HMI_PBINFO init_Menu: WMI_MENU_PRIVATE
HMI_PBOOK initHMI_PBinfo: HMI_PBINFO
\\
SRV_LAYMAN init: SRV_TOON
                    
```

162

Company Confidential. Do not copy or distribute.



Step 1 : New HMI

- Example for Popup

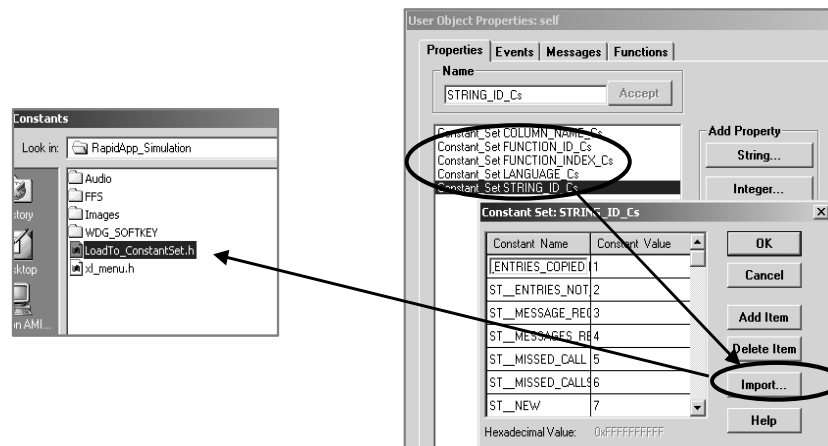
Destinations	Event	Condition	Actions
D: Options	Popup_Hld Timeout_Ev		Popup_Hld stop
D:			
Activities			
entry	Popup_Hld popMessage:(TextResource_Hld getStringByID: TextResource_Hld.STRING_ID-Cs.ST_L		
mode			
exit			

163

Company Confidential. Do not copy or distribute.

## Add Menu Item (cont.)

- Import into RapidPLUS (EMB\_TXTRES.UDO)



Company Confidential. Do not copy or distribute.

164

## Menu 1 Exercise

Details:

- Add Menu Item
- Create a new item in the Submenu
- Add the item "MySMS" to the Messages menu
- The final result will be:



Company Confidential. Do not copy or distribute.

165

## Menu 2 Exercise

Details:

- Add Options Menu
- The menu will be located on the MemoryStatus screen



Company Confidential. Do not copy or distribute.

166

## Menu 2 Exercise (cont.)

- Add a new Options menu
- Only the menu navigation will function
- Back will go to **Idle screen**



Company Confidential. Do not copy or distribute.

167



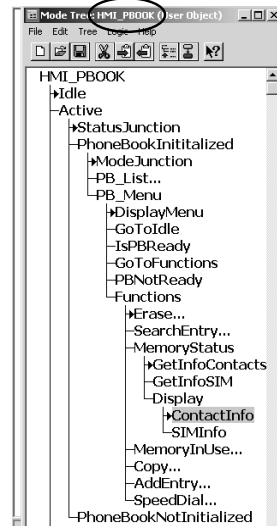
•Create new options menu

394		92	Restrict
395		Skins	
396		285	Water
397		286	Fire
398		287	Wind
399		MyOptions	
400		294	option1
401		295	option2
402		296	option3
403		TextList	
404		message received	
405		messages received	
406		missed call	
407		missed calls	
408		New	
409		(empty)	
410		Accessing Network	
411		activated	
412		active	
413		Active: Voice calls	
414		Alarm	
415		Alarm off	
416			

Company Confidential. Do not copy or distribute.



•Add new mode for the Option menu



Company Confidential. Do not copy or distribute.

- Use the Softkey

Destinations	Event	Condition
ernal	Softkey RightShort_Ev	
playMenu		

Activities
CurrentOpenLayoutID_Int := LayoutManager initLO_Menu_SB2_scrollBar: ScrollBar plot
<b>Init softkey</b>
Softkey setRightSoftkeyLabel: (TextResource getStringByID: TextResource.STRING_ID_
<b>Init title and icon</b>
Plotter drawText: (TextResource getStringByID: TextResource.STRING_ID-Cs.ST_ME

Company Confidential. Do not copy or distribute.

- Stop the curentLayout when press Options and then start the options menu (Menu start , calling the functionIndex)

Actions
HMI_PBOOK stopCurrentLayou
Menu resume

```

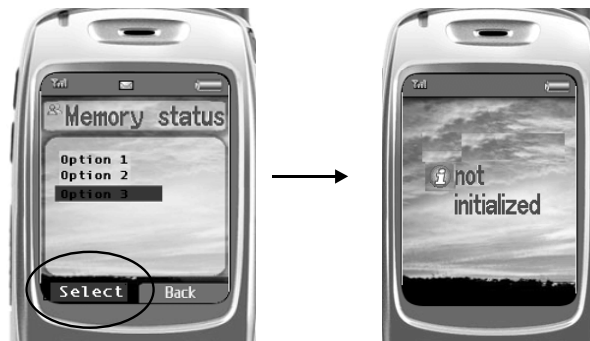
if PB_Detail.DisplayItem_Int = PB_Detail.DISPLAY_FF
Menu start: TextResource.FUNCTION_INDEX-Cs.IX
else if PB_Detail.DisplayItem_Int = PB_Detail.DISPLAY_
    
```

Company Confidential. Do not copy or distribute.



## Popup Exercise

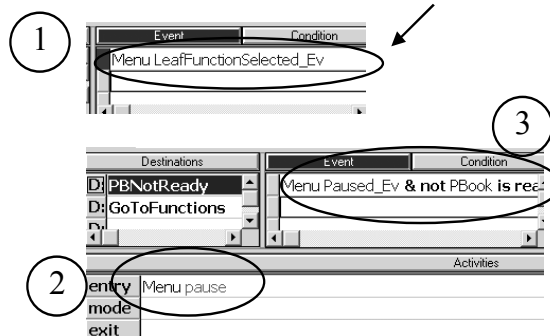
- Choosing Option3 will display 2 sec Message
- Use Popup timeout\_Evt for the timing



Company Confidential. Do not copy or distribute.

172

(On each LeftSoftKey press the menu generate):



Company Confidential. Do not copy or distribute.

173

## Pop Up Message

- Using the WMI\_Popup.UDO
- Timeout\_Ev
- Stop function , Then call menuResume function

Destinations	Event	Condition	Actions
<input type="checkbox"/> internal D: D:	Popup Timeout_Ev		Popup stop

entry Popup.popMessage: (TextResource.getStringByID: TextResource.STRING\_ID-Cs.ST\_PHONEBOOK\_NOT\_INITIAL

Popup.popMessage:'myPopup'type: Popup.TYPE-Cs.WITH\_TIMEOUT bitmap: Costants.BITMAPS-Cs.I\_SIGN leftSoftKey: 0 rightSoftKey: C

## Step 1 : Final Ex. New HMI

Define the new HMI and the screens only  
Don't display the memory data  
See FinalEx.Doc



## Step 2 : Manage the Memory

- **How to Read the memory:**

1. Create holder to SRV\_PBOOK.UDO
2. Call the usedOfIn function
3. Wait for ready\_Evt
4. Read the memory from Results.usedOf.used

Destinations	Event	Condition	Actions
D: Ac-2	SrvPB_Hld ready_Evt	PhoneEntriesCount_Int := SrvPB_Hld.Results.usedOf.used	
D:			
D:			
Activities			
entry	SrvPB_Hld usedOfIn: Costants.PHONE_LISTS-Cs.ME		
mode			
exit			

Company Confidential. Do not copy or distribute.



## Step 2 : Manage the Memory



- **How to Delete the memory:**

1. Use the SRV\_PB
2. Call deleteEntryFrom: <pb> at: <idx>
3. Wait for ready\_Evt and then call it again -Loop

Destinations	Event	Condition	Actions
D: Pbr	SrvPB_Hld ready_Evt		
D: MI			
D:			
Activities			
>	if PhoneEntriesCount_Int > 0		
mode	HMI_PBINFO deleteEntryAtPB: Costants.PHONE_LISTS-Cs.ME idx: PhoneEntriesCount_Int		
exit			
exit			

Company Confidential. Do not copy or distribute.




e-SIM MMI  
Reference Design 1.x  
**Final Exercise**

Company Confidential. Do not copy or distribute.

178



## Final Exercise

**Goal:**

- Add a new MMI including Options menu

**Details:**

- Add a new layout
- Add a new menu leaf to textResource
- Add a new options menu to textResource
- Start the new HMI from HmiMenu
- Start the options menu from the new HMI

Company Confidential. Do not copy or distribute.

179

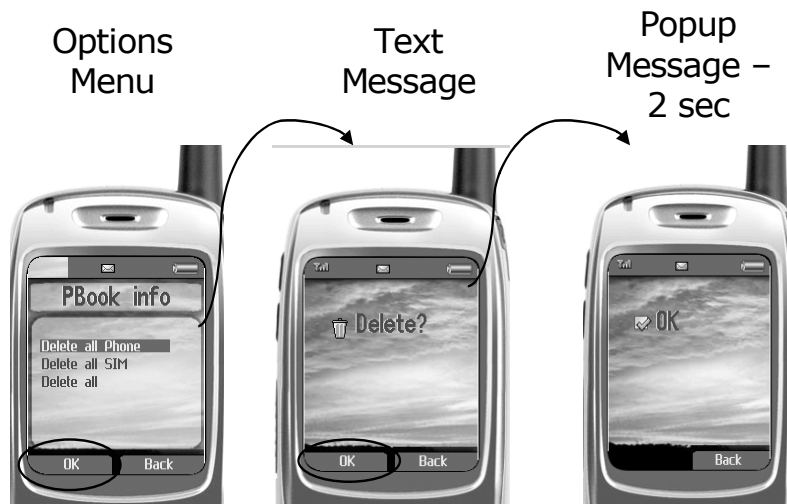
## Final Exercise – Screen Shots



Company Confidential. Do not copy or distribute.

180

## Final Exercise – Screen Shots (cont.)



Company Confidential. Do not copy or distribute.

181

## Step 1 : New HMI


Define the new HMI and the screens only  
Don't display the memory data

## Step 1 : New HMI

•Example for menu managing:

Activities	
entry	Menu_Hld resume
entry	Menu_Hld start: TextResource_Hld.FUNCTION_INDEX-Cs.IX_PBDEL_PH
mode	
exit	Menu_Hld stop
exit	

•We call the **resume** because the menu is in pause from the PBOOK.



## WMI\_MENU\_PRIVATE

### Optional

- you can use another instance of wmi\_menu
- Init the PBIInfo menu with another menu because the wmi\_Menu is still in pause from the Pbook


MainTest.RPD :

```

\\
HMI_PBOOK initEMB: EMB_GSRV settings: UDD_SET
HMI_PBOOK initServices: SRV_PBOOK layOutManager: SRV_LA
HMI_PBOOK initWMI: WMI_PB_BROWSE menu: WMI_MENU_e
HMI_PBINFO init_Menu: WMI_MENU_PRIVATE
HMI_PBOOK initHMI_PBIInfo: HMI_PBINFO
\\
SRV_LAYMAN init: SRV_ICON
            
```

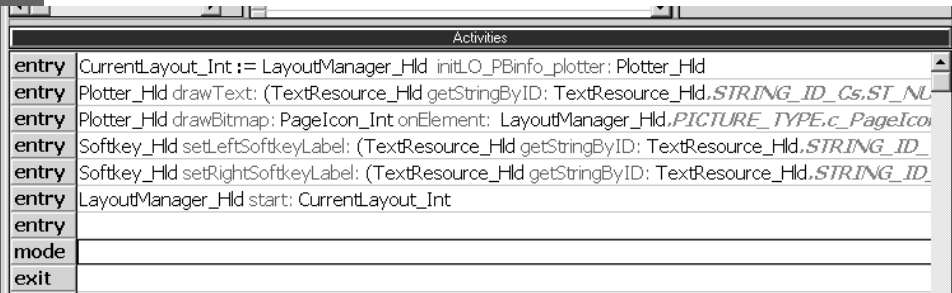
184

Company Confidential. Do not copy or distribute.



## Step 1 : New HMI

- Example for normal page:c



185

Company Confidential. Do not copy or distribute.





## Using the Soft keys:

- Use inside a layout

```

Activities
entry CurrentLayout_Int := LayoutManager_Hld initLO_PBInfo_plotter: Plotter_Hld
entry Plotter_Hld drawText: (TextResource_Hld getStringByID: TextResource_Hld.STRING_ID-Cs.ST_NU
entry Plotter_Hld drawBitmap: PageIcon_Int onElement: LayoutManager_Hld.PICTURE_TYPE.c_PageIco
entry Softkey_Hld setLeftSoftkeyLabel: (TextResource_Hld getStringByID: TextResource_Hld.STRING_ID_
entry Softkey_Hld setRightSoftkeyLabel: (TextResource_Hld getStringByID: TextResource_Hld.STRING_ID_
entry LayoutManager_Hld start: CurrentLayout_Int
entry
mode
exit
    
```



## Using the Soft keys:

- Use inside a Menu

K	L	M	N	O
LevelID	LeftSK	RightSK	Layout	Image ID na
	OK	Back		23
	OK	Back		23
	OK	Back		23
	Select	Back		23
	Select	Back	23	CALL_BARRI
	Select	Back		23
	OK	Back		23
	OK	Back		23

•Close the Layout:

Destinations	Event	Condition	Actions
internal	Softkey_Hld LeftShort_Ev		LayoutManager_Hld stop: CurrentLayout_Int
D: Options			CurrentLayout_Int := 0
D:			
D:			

Destinations	Event	Condition	Actions
internal	LayoutManager_Hld Stopped		
D: Options			
D:			
D:			

**Step 1 : New HMI**

•Example for Popup

Destinations	Event	Condition	Actions
D: Options	Popup_Hld Timeout_Ev		Popup_Hld stop
D:			
D:			

Activities	
entry	Popup_Hld popMessage:(TextResource_Hld getStringByID: TextResource_Hld.STRING_ID-Cs.ST_I
mode	
exit	



## Step 2 : Manage the Memory

- **How to Read the memory:**

1. Create holder to SRV\_PBOOK.UDO
2. Call the usedOfIn function (will generate Evt)
3. Wait for ready\_Evt
4. Read the memory from Results.usedOf.used

Destinations	Event	Condition	Actions
D: Ac	SrvPB_Hld ready_Evt	PhoneEntriesCount_Int := SrvPB_Hld.Results.usedOf.used	
D:			
D:			

Activities	
entry	SrvPB_Hld usedOfIn: Costants.PHONE_LISTS-Cs.ME
mode	
exit	

Company Confidential. Do not copy or distribute.



## Step 2 : Manage the Memory

- **How to Delete the memory:**

1. Use the SRV\_PB
2. Call deleteEntryFrom: <pb> at: <idx>
3. Wait for ready\_Evt and then call it again -Loop

Destinations	Event	Condition	Actions
D: Phor	SrvPB_Hld ready_Evt		
D: MI			
D:			

Activities	
>	if PhoneEntriesCount_Int > 0
>	HMI_PBINFO deleteEntryAtPB: Costants.PHONE_LISTS-Cs.ME idx: PhoneEntriesCount_Int
mode	
exit	
exit	

Company Confidential. Do not copy or distribute.





