

UNIVERSIDADE NOVA DE LISBOA

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Electrotécnica

**REAL TIME MOBILE SYSTEM FOR SUPPORT IN FIREFIGHTING
ENVIRONMENTS**

Por:

João Reis Costa Mendes

Dissertação apresentada na Faculdade de Ciências e Tecnologia da
Universidade Nova de Lisboa para a obtenção do grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: Prof. Doutor José Manuel Fonseca

Co-Orientador: Prof. Doutor Pedro Manuel Vieira

**Lisboa
2009**

Abstract

This dissertation addresses work being performed within the context of the Fire Forest Finder system, fulfilling the requests for a multi-information application intended for a vehicle mounted mobile device.

The main objective of this dissertation is to provide a solution running on a PDA device that provides support in Fire Fighting environments. The user has access to: multi-information data of the theater of operations, an automotive navigation system (TomTom Navigator™) and a text messaging capability.

The hardware present on this system is a DLoG X7™ industrial graded PDA that possesses the right sturdiness for this task. The software developed consists of three distinct interfaces: an appropriately customized TomTom Navigator™ interface for the end-users of this system and two additional applications were created to allow a proper visualization of the available information and also have an interface with text messaging input/output capabilities;

It is important to refer that the GSM protocol, text messaging service in particular was the chosen communication mean due to the fact that this network has, at the time this dissertation was finished, the better and most reliable coverage of remote areas, such as the forests for which this system is intended to.

In terms of experimental validation a series of performance, intuitiveness and usability tests were performed and analyzed in detail with the purpose of demonstrating the validity of the ideas presented.

The thesis is completed by the depiction of the achieved results with the subsequent discussion and identification of open points as a result of the work done.

Keywords

Forest Fire Support System, PDA, GSM, TomTom, Windows CE 5.0

Resumo

No âmbito do sistema Forest Fire Finder esta dissertação tem como objectivo apresentar um sistema de processamento de multi-informação destinado a ser usado num dispositivo móvel montado num veículo.

O principal objectivo deste trabalho é providenciar uma aplicação de software a correr num PDA de modo a prestar um apoio fundamental em ambientes de combate aos incêndios. Neste sistema o utilizador tem acesso a uma panóplia de informações provenientes do teatro de operações, a um sistema de navegação automóvel (TomTom Navigator™) e à capacidade de comunicação por mensagens de texto.

O hardware presente neste sistema é o PDA industrial DLoG X7™ que possui as características de robustez apropriadas para este tipo de ambientes. Em termos do software desenvolvido o sistema é composto por três interfaces distintas: uma interface do TomTom Navigator™ personalizada para os utilizadores finais deste sistema; e duas interfaces que providenciam respectivamente, uma visualização adequada da informação do teatro de operações disponível e uma capacidade de recepção e emissão de mensagens de texto.

Como meio de comunicação do sistema com o exterior é importante salientar que o protocolo escolhido foi o GSM, com ênfase no serviço de mensagens de texto, devido ao facto de até à altura da conclusão da escrita desta dissertação, nenhum outro meio conseguir garantir uma cobertura desta qualidade ao nível de áreas remotas como é o caso das florestas no sistema desenvolvido nesta dissertação.

Em termos de validação experimental foram efectuados vários testes de performance, ergonomia e facilidade de uso, tendo em vista uma análise detalhada que procurou demonstrar a validade das ideias apresentadas.

A dissertação termina com a apresentação dos resultados conseguidos seguida de uma discussão e identificação dos vários pontos em aberto no que concerne ao trabalho efectuado.

Palavras-Chave

Sistema de Apoio a Fogos Florestais, PDA, GSM, TomTom, Windows CE 5.0

Acknowledgments

To begin my acknowledgments I would like, first and foremost, to thank my family, for all their endured sacrifices that made possible for my graduation and the comfort, encouragement and affection they have always demonstrated through all these years.

I thank Prof. Pedro Vieira for the orientation, ever-present support and patience to guide my progression even when I felt lost at times. I would like to thank him specially for making all the human and physical resources at NGNS-IS always available for my support; and also, for all the text revision, for this dissertation and the elaborated scientific paper.

To Prof. José Manuel Fonseca, for the co-orientation and the opportunity that was given to be part of this project. I thank you for all the advices, support and the careful revision of the text for this dissertation.

I would like to thank all the people at NGNS-IS that helped me on the course of this dissertation, specially Nuno Pinto, Pedro Duque and Manuela Contrim, without your help and support I would not have been able to pulled this off!

To Mr. Franz Angermeier and the DLoG company for providing essential assistance in solving specific problems that rose during the course of this dissertation.

A very special thank you goes to all my friends and colleagues, for their advice, for their comprehension, understanding, and patience to put up with me through this long process, it was worth it!

Lisboa, Setembro 2009
João Reis Costa Mendes

Table of Contents

Chapter 1.	1
1.1 Background	1
1.2 Introduction	2
1.3 Objectives and Contributions	4
1.4 General Overview of the Dissertation	5
Chapter 2.	8
2.1 HARDWARE – DLoG X7	8
2.1.1 DLoG X7 Overview	8
2.1.2 Operating System	10
2.2 SOFTWARE – TomTom	11
2.2.1 TomTom Navigator	11
2.2.2 Software Development Kit	12
2.3 SOFTWARE – Server Backoffice	13
2.3.1 SMS Protocol	14
Chapter 3.	16
3.1 Introduction	16
3.2 Windows Messages	18
3.3 MAIN Application	22
3.3.1 Main Class – Global Algorithm	22
3.3.1.1 Hardware Control	23
3.3.1.2 Message Routing between Threads and Applications	29
3.3.1.2.1 Between Applications	30
3.3.1.2.2 Between Threads	31
3.3.2 Auxiliar Classes – Algorithms	38
3.3.2.1 GSM Communication	38
3.3.2.2 Event Processing	40
3.4 SMS Application	49
3.4.1 SMS Class – Global Algorithm	49

3.4.2	Alphanumeric Keypad handling	50
3.4.3	Text message traffic handling	52
3.5	Tracking, Robustness/Ruggedness and Control Files	53
3.5.1	Global Log	55
3.5.2	GSM Log.....	56
3.6	Interfaces	57
3.6.1	FFFpdaMAIN.....	58
3.6.2	FFFpdaSMS	58
3.6.3	TomTom Navigator Customized.....	59
3.6.4	Administrator Mode	61
Chapter 4	62
4.1	Description of the Simulation Environment and Vehicle.....	63
4.2	Tests.....	66
4.2.1	Road-Book Test	66
4.2.2	System Usability	71
4.2.2.1	TomTom	72
4.2.2.2	Assorted Actions.....	74
4.2.2.3	Overall Appreciation	74
4.2.2.4	Suggestions.....	75
4.3	Conclusions	75
Chapter 5	78
5.1	General Summary	78
5.2	Conclusions	78
5.3	Future Work.....	79
REFERENCES	81
Appendix - A	83
Appendix - B	87
Appendix - C	93
Appendix - D	95
Appendix - E	133

Acronyms and Abbreviations

API	Application Programming Interface
CAB	Cabinet (file format)
CPU	Central Processing Unit
FFF	Forest Fire Finder
GIS	Geographic Information System
GPS	Global Positioning System
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communication
MCC	Mobile Control Center
OEM	Original Equipment by Manufacturer
OS	Operating System
PDA	Personal Digital Assistant
PIN	Personal Identification Number
POI	Point of Interest
RAM	Random Access Memory
ROM	Read Only Memory
SIM	Subscriber Identity Module
SMSC	Short Message Service Center
SMS	Short Message Service
SDK	Software Development Kit
TT	TomTom™
TTN	TomTom Navigator™
USB	Universal Serial Bus
WinCE	Microsoft Windows CE 5.0™

List of Figures

Figure 1.1 – FFF system overview.....	1
Figure 1.2 – Interfaces.....	4
Figure 2.1 – DLoG X7.....	8
Figure 2.2 – Win CE Architecture.....	9
Figure 2.3 – A complete coded event.....	13
Figure 3.1 – Global system architecture.....	16
Figure 3.2 – TTN interface.....	17
Figure 3.3 – Main interface.....	17
Figure 3.4 – SMS interface.....	17
Figure 3.5 – System defined messages.....	18
Figure 3.6 – Custom application defined messages.....	20
Figure 3.7 – Edit boxes scrolling hardware keys.....	24
Figure 3.8 – Application selection hardware keys.....	26
Figure 3.9 – TT events Show/Hide hardware keys.....	27
Figure 3.10 – SMS to MAIN message process.....	30
Figure 3.11 – MAIN to SMS message process.....	31
Figure 3.12 – Outgoing message processes.....	32
Figure 3.13 – Incoming message processes.....	33
Figure 3.14 – System crash message process.....	34
Figure 3.15 – Start position timer and SMS thread message process.....	35
Figure 3.16 – Global Log message process.....	36
Figure 3.17 – GSM Log message process.....	36
Figure 3.18 – Fire event detail.....	39
Figure 3.19 – Weather event detail.....	40
Figure 3.20 – Auto event detail.....	40
Figure 3.21 – Water event detail.....	40
Figure 3.22 – Human event detail.....	41
Figure 3.23 – Invalid Water event.....	44
Figure 3.24 – Track Position Update system message.....	47
Figure 3.25 – System Crash system message.....	48
Figure 3.26 – Alphanumeric keypad.....	49
Figure 3.27 – Invalid event response message.....	53
Figure 3.28 – Fire event log entry.....	55
Figure 3.29 – Invalid event log entry.....	55
Figure 3.30 – Auto-Refresh log entry.....	55
Figure 3.31 – GSM log entry types.....	56
Figure 3.32 – FFFpdaMain interface.....	57
Figure 3.33 – FFFpdaSMS interface.....	58
Figure 3.34 – TTN main interface.....	59
Figure 3.35 – TTN browse interface with events.....	59
Figure 3.36 – Customized TTN main menu.....	60
Figure 3.37 – Login interface.....	60

Figure 4.1 – Complete simulation environment.....	63
Figure 4.2 – Nokia 6020 mobile phone.....	63
Figure 4.3 – Vodafone SMS webpage.....	64
Figure 4.4 – The used vehicle mounted with the DLoG.....	64
Figure 4.5 – TT Default Screen.....	67
Figure 4.6 – TT interface options.....	67
Figure 4.7 – FFFpdaMAIN interface options.....	68
Figure 4.8 – TT Event warning message flash.....	68
Figure 4.9 – Show/Hide Events keys.....	69
Figure 4.10 – SMS interface.....	70
Figure 4.11 – SMS dialog.....	70
Figure 4.12 – Confirm the reception of multiple Events.....	71
Figure 4.13 - Trace a route to a Water event.....	71
Figure 4.14 – Go to specific Fire event location.....	72
Figure 4.15 – Show/hide Auto event.....	72
Figure 4.16 – Confirm the reception of a SMS and reading of the last two received SMS messages.....	72
Figure 4.17 – See Weather event detailed information.....	72
Figure 4.18 – Send a SMS Message to “All”	73
Figure 4.19 – Turn Off the DLoG.....	73
Figure 4.20 – Reboot the DLoG.....	73
Figure 4.21 – Overall Appreciation.....	74
Figure B.1 – DLoG X7.....	87
Figure B.2 - Simulated SMS server.....	88

List of Tables

Table 2.1 – Event/Code conversionable.....	13
Table 2.2 – SMS Protocol.....	14
Table 3.1 – TT events Show/Hide keys.....	27
Table 3.2 – Events <i>struct</i> type and array size correspondence.....	41
Table 3.3 – Event/Occurrence types.....	45
Table 3.4 – Event/Occurrence TT screen messages.....	46
Table 4.1 – Road-Book guide.....	66
Table A.1 - Event/code message types.....	83
Table C.1 – Road-Book guide.....	93

List of Algorithms

Algorithm 3.1 – Main class global.....	22
Algorithm 3.2 – Hardware control.....	23
Algorithm 3.3 – Scroll UP.....	25
Algorithm 3.4 – Application selection.....	26
Algorithm 3.5 – TT All Events Show/Hide.....	28
Algorithm 3.6 – GSM communication.....	38
Algorithm 3.7 – Event class global.....	42
Algorithm 3.8 – Incoming Event.....	43
Algorithm 3.9 – Process Event.....	44
Algorithm 3.10 – Outgoing Event.....	47
Algorithm 3.11 – SMS class global.....	49
Algorithm 3.12 – Alphanumeric keypad handling.....	50
Algorithm 3.13 – Keypad value rotating.....	51
Algorithm 3.14 – Logging action global.....	54

Chapter 1.

INTRODUCTION

1.1 Background

Technology in the last years has seen an exponential growth and has helped to provide solutions on many levels in different areas. A common objective rises: provide tools which help the human decision process, specifically in the case of this dissertation regarding a real time situation based on a multitude of available data at each moment.

Forest fire is a scourge that affects many countries and the last years have been a testimony of this situation. Once the Fire has been detected the swiftness of the reaction and response times are essential in order to minimize the damages. An optimized access to all the available information related to the theater of operations is fundamental to improve the quality and the speed of the decision process.

To respond to this problem, NGNS – Ingenious Solutions™ has developed the Forest Fire Finder™ (FFF), an innovative system of forest fire detection and location. It provides forests owners with an honest reliable system that is human independent, given its autonomous and automatic character and insures a decrease of response time to the emergency [NGNS-IS 2009] .

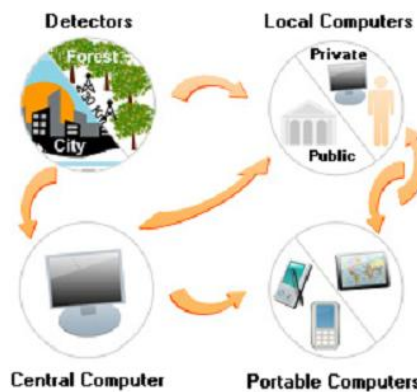


Figure 1.1 – FFF system overview

In this context rises the need for a system that has the following characteristics: capability of accurately showing the precise location of fires, other vehicles and water supplies; providing visualization of real-time periodic data from each of the referred elements; show instant detailed meteorological data and that could also be used as communication medium by means of a simple and reliable text-messaging system.

1.2 Introduction

Following the analysis of the problematic previously enunciated, the theme of this dissertation became a challenge of developing a user-friendly application which runs on industrial graded Personal Digital Assistants (PDA) adequate to the harsh environmental conditions. This system should supply the user with multiple real time data from the global fire scenario and provide communications through the Short Message Service (SMS) text messaging system.

Some solutions have been tried in the past, the most similar consisted on a hand-held mobile device approach to disaster management, based its communications in a continuous Global Communication for Mobile Communications (GSM) data-link. Due to cost and technology limitations at the time, this project was not pursued. [Fischer 2000]. The key difference to the work developed in this thesis, is the fact that instead of relying on a continuous stream of data being passed constantly to send maps, images, text, the present system offers a more efficient and cost-effective solution. By using a navigational software that provides the map visualization needed and the inherent routing tool, this method allows all the information needed to be passed exclusively by coded text messages which will be interpreted appropriately.

A Geographical Information System (GIS)-based fire management information system was also developed but focused on prediction of the fire-fronts, temperatures, wind parameters, pressure among others, besides using on a stationary operator [Lymberpoulos 1996]. Similar emergency decision support systems have also been modeled in terms of real-time update of the event, and its importance on the decision making process [Ye 2008]. In addition to these specific disaster-related solutions, interesting and inspiring concepts concerning multi-vehicle management and real-time information update have been applied in diverse scenarios as decision support systems such as: logistic and fleet management in warehouse [FMS 2009] and mining

industry [FMSy 2009]. The work developed for this dissertation combines the features of the previous mentioned examples in a unique decision support system that has the following key features: improve response time through a navigational software, provide the Firemen with a visual scenario of the theater of operations, and also update this information in real-time. Mobility on the case of this dissertation is fundamental, so the system was designed to be used in a vehicle mounted PDA device, with simple and intuitive interfaces in order to simplify the task of the Firemen. As an alternative to the common radio communications, text messaging capability is provided which can be used as an alternative broadcasting medium.

The innovation that this system proposes is to provide a mobile tool to aid the decision process, specifically, on a Fire scenario maintaining an easy and simple interaction with the Firemen. This design objective modeled the visual and interactive features of the system interfaces, providing the Firemen with an intuitive presentation of the real-time available information combined with map visualization.

The first task was to choose an adequate PDA for the problematic in hand and the decision was to have a fixed robust device on the vehicle instead of a portable handheld one. This is justified by the need for a more ergonomic firemen oriented device that could support the abuse of the physical (temperature, pressure, impact) conditions which are common inside a firefighting vehicle. In accordance to several factors, explained latter on this thesis, the DLoG X7™ was chosen.

The second concern was to decide whether to implement a Global Positioning System (GPS) navigational software, or on the other hand resort to one of the currently available on the market, that could provide the capabilities that the project required. A careful investigation on the status quo of the navigational software paradigm was performed. Taking in account that the main objective of this dissertation is the creation of an application that, above all features provides, the most quantity of useful data to the firefighter, the decision was made not to develop a navigational software. So TomTom Navigator™ 6.010 (TTN) was chosen as the software to be used. The downside of this decision is that newer versions of the TTN could not be used due to the limitations of the PDA hardware operating system (OS), Microsoft Windows CE 5.0™ (WinCE). This will be discussed in detail later in this dissertation.

Due to the fact that the TomTom™ (TT) Software Development Kit (SDK) classes are developed for C++ that was selected as the language to develop the application, using the

Microsoft Visual Studio 2005 suite. The debugging process was performed by deploying and remote debugging the several applications on the PDA X7 itself, due to the fact that the TT SDK do not allow for the use of an emulator.

Having set the base for our application, both hardware and software aspects, the next decision was to divide the application into three interfaces that can be accessed through hardware buttons: “TTN”, “Main” and “SMS” – Figure 1.2. There is an additional administrator override option which closes the running applications and brings up the standard WinCE interface and all its inherent capabilities.



Figure 1.2 – Interfaces

1.3 Objectives and Contributions

This dissertation deals with the work being performed in the context of the FFF project, but standing as an independent yet globally integrated module that seamlessly interacts with the existent platform.

The specific objectives of the project addressed in this dissertation dealt by the author are the following:

- Research on the PDA, DLoG X7™ device and all its hardware possibilities from the perspective of the end-user, in this case the firemen.
- Develop a Text Messaging protocol that carries all the communications with the backbone server, from the Global System for Mobile Communication (GSM) module signal reception, to a higher message processing level
- Implementation of the most user-friendly interface possible to allow a fast and intuitive access and interaction with the developed application.

- Implementation of a robust, error free system that after a reboot in case of an anomalous error or system failure, could be brought up-to-date automatically.
- Exhaustive study of TTN 6 and the provided SDK.

The main contribution of this dissertation is to provide within the Firefighting paradigm a useful, user-friendly tool that hopefully allows a faster and more precise response to a fire scenario on a Forest, giving emphasis on real time updated data concerning all the parts involved on the operation, human, physical and environmental.

This dissertation has resulted in a paper submitted to an international journal of the specialty, "Fire Technology".

In addition, this dissertation represents the culmination of the Electrical and Computers Engineering course of the Universidade Nova de Lisboa, This dissertation has also has the important objective of allowing the conclusion of the course as its last project. The work carried out will be evaluated with the purpose of awarding the engineering degree in the mentioned course.

1.4 General Overview of the Dissertation

This dissertation is organized in five chapters as described in the following paragraphs.

The first and present chapter ("Introduction") presents an introduction and objectives statement of the theme of this dissertation. The second chapter ("Resources Overview") introduces the system architecture on which the application runs, in terms of hardware as well as software. A brief description with a state of the art context frame accompanies each of the referred resources.

The third chapter ("System Architecture") explains all the software applications developed for this dissertation.

The fourth chapter ("Performance Analysis") is dedicated to the several tests which were carried in order to optimize and demonstrate the application performance on several parameters: response time, efficiency, robustness, security, user-friendliness.

The fifth and final chapter (“Conclusions”) of this dissertation is dedicated to summarize and state the conclusions on the work done, as well as a sub-chapter dedicated to future work possibilities of the application.

Chapter 2.

RESOURCES OVERVIEW

This chapter proceeds to present in detail the chosen tools for the realization of the intended real time mobile decision support system, both in terms of the developed software as well as the physical device on which this system is running. The first section describes the used hardware specifically the DLoG X7™ PDA. The second section depicts the GPS Navigational Software TTN and its SDK. Finally, the third and final section describes the high level SMS Protocol developed for this system.

2.1 HARDWARE – DLoG X7

This section describes the chosen PDA for this thesis, the DLoG X7™, The first section consists on an overview of the device, enlightening its features and resources and the final section is dedicated to the OS used, WinCE.

2.1.1 DLoG X7 Overview

The problematic of this thesis required a PDA that could withstand the extreme environmental(temperature, humidity, water resistance and shock) conditions that are present in forest fires environments, and that also possessed a minimum display size of 7 inches, The necessity of being vehicle mounted was also taken in account. In terms of features and software requirements, the requirements were a modem with GSM and GPS communications features and the use of a Windows™ OS.

The DLoG X7™ – Figure 2.1 – device is a multi-functional terminal which is designed for both stationary and mobile use. It is equipped with an Intel PXA270 processor and works with WinCE 5.0 operating systems, thus fulfilling one of the required requests. In terms of Random Access Memory (RAM) it has 64MB of synchronous dynamic random access memory (SDRAM) and also 64MB of Flash Read Only Memory (ROM). Regarding the communication interfaces it has two RS-232 Serial ports and two Universal Serial Bus (USB) low and full speed respectively,

which enables the transmission of the necessary data for real time debugging issues with the use of Microsoft ActiveSync 4.5™. Thanks to the robust design with aluminum housing, (IP54/IP65 Protective class) the device provides effective protection against mechanical, electrical and chemical damage and extreme ambient temperatures. It is designed without an external fan to lower maintenance requirements. The key advantage of the DLoG™ X series lies in its functionality and compact design. Various mounting brackets allow installation in the most confined spaces which met with the project demands [DLoG 2006].



Figure 2.1 – DLoG X7

The last requirement of the hardware was the necessity of having GPS/GSM capabilities which was solved by the installation of an external modem, M2501B Automotive, mounted to the side of the PDA. This choice and subsequent installation were performed by DLoG™. The modem is based on the module WISMO Quickmaster of Wavecom Q2501™ and combines a GSM voice, GPRS data transfer (class 10) and GPS tracking (16-channel GPS receiver). The M2501B is an automotive on-board computer that meets the tough requirements for use on locomotives, construction equipment and trucks. The device is intended for use in wiring with 8-36 Vdc (eg. 12 Vdc and 24 Vdc) and was designed for and against such networks as typical disturbances overvoltage, voltage drop and reverse polarity protected. It is an ideal base for the integration of a vehicle in a fleet management, contract management on-board or other backend

systems, which is the reason why it is used on several DLoG™ products [SE 2008]. The device has the following options: input for a GSM antenna, an entrance for GPS antenna, Subscriber Identity Module (SIM) card reader, a 16-pin Power / Main connector, an Auxiliar 14-pin connector and a 12-pin audio.

2.1.2 Operating System

The Original Equipment by Manufacturer (OEM) OS of the DLoG X7™ as it was previously referred is WinCE. It is a highly modular, with a relative simple kernel where optional components run as separated processes. By using separate processes for optional modules the OS becomes more reliable.

All system calls are handled in the kernel (by a software trap interface) and redirected to the correct process if needed. The main advantages of WinCE are the interoperability with the general purpose windows environment (with technologies like DCOM) and the ease to make windowing applications just like in the general purpose windows operating system. This provides a large developer base for the non real-time part of the system. The real-time part however is better designed by developers with a good understanding of real-time behavior.

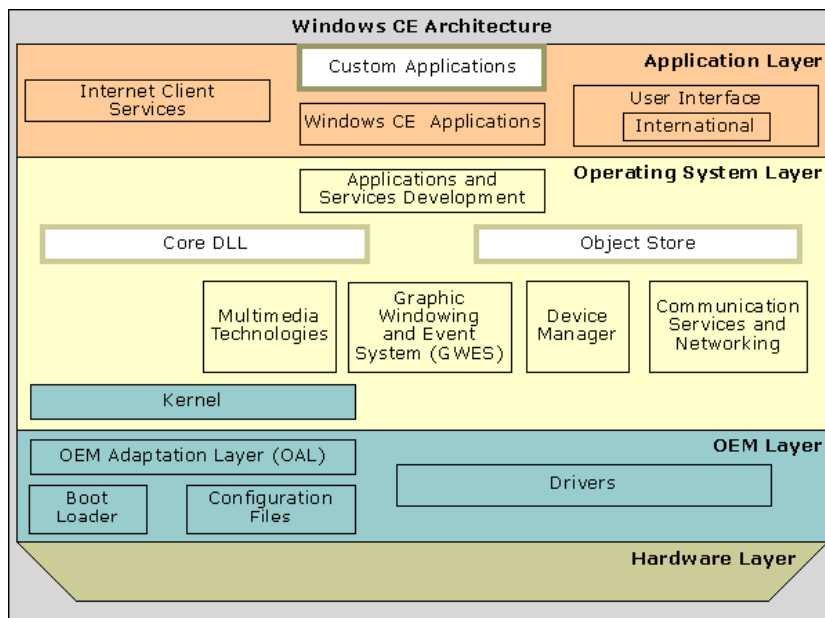


Figure 2.2 – Win CE Architecture

Although it has a modular design so that a minimal configuration can suit small systems, it is too complex and has a much larger footprint to be a good solution for deeply embedded systems.

Regarding threads and processes management, WinCE is a classical pre-emptive multitasking system with support for both processes and threads within the same process. It uses a priority based scheduler to decide which thread that is ready to run it will activate. Remark that priority number 0 is highest priority. Threads with the same priority are scheduled using a fair round-robin time slice mechanism. This time-slice quantum can be set for each thread separately.

The number of running processes in WinCE is limited to 32. The reason for this can be located in the partitioning of the virtual memory. The number of running threads within a process is only limited by the available system resources (RAM) [Dedicated Systems Experts 2004].

2.2 SOFTWARE – TomTom

This section describes the automotive navigational software TomTom Navigator™ 6 (version 6.030) that was selected for this Thesis, beginning with an overview of the application and a brief description of its most important functionalities. On the final section the Software Development Kit (SDK) is introduced given special focus to the Application Programming Interface (API) calls used on the application developed for this thesis.

In order for this version of TTN (6.030) to work with the DLoG X7™ WinCE OS there was a need to change the Cabinet (CAB) files, what was kindly performed by Franz Angermeier from DLoG™.

2.2.1 TomTom Navigator

An automotive navigation system is a satellite navigation system designed for use in vehicles. In the case of this thesis it uses the GPS feed from the M2501B Automotive external modem to acquire position data to locate the user's vehicle on a road in the unit's map database.

TomTom™ is one of the lead selling navigational software companies in the world being most known for its stand-alone units as well as the case of this dissertation its software (TTN) installation for other mobile devices. It is well recognized by its many features, specifically the speed and effectiveness of its route tracing algorithm that conjugates perfectly with the accuracy and detail of the road maps. This fact comes from a constant upgrade of the referred maps. Some

of the other functionalities consist on several route choices available at any given time (shortest, quickest, avoiding toll booths or by required arrival time), the possibility of adding custom created Points of Interest (POI) that are icon representations on the map itself which were fundamental to this Thesis (as it will be explained in detail on Chapter 3.3.2.2.). In terms of visualization it has the capability of displaying maps/routes in either 2D or 3D perspectives with a night vision oriented alternative color scheme. It also possesses a reaction algorithm in case of a missed turn or instruction which leads to an instant recalculation of the desired route from the current location.

The status bar is customizable with only the desired information which improves readability; this fact in conjunction with the simple yet clear driving instructions display gives plenty of notice and provides extra focus on the action of driving itself. With over twenty two languages as an option the universality of this software is further enhanced [TT 2008].

2.2.2 Software Development Kit

A Software Development Kit is a set of development tools that provides the capability of creating applications using aspects or even the whole existent software package, framework or operating system. In the case of this dissertation the Application Programming Interface (API) was used and all the function calls present on the developed application will be depicted on the next paragraphs. The reason why only the used functions are shown, and not a complete listing, relates to the TT SDK copyright agreement terms.

- ***AddPoi (... , char* aFilename , long aLongitude , long aLatitude , char* aName , char* aId)***

Adds a point of interest to the .OV2 file specified by the POI category argument. The point of interest will have the name and co-ordinates given by the corresponding arguments. An external identifier string can also be attached to the point of interest. When the .OV2 file does not yet exist, it is created.

- ***DeleteAllPoi (... , char* aFilename)***

Deletes all points of interest from the .OV2 file specified by the POI category argument. The POI category itself is also removed after that.

- **DeleteClosestPoi (... , char* aFilename, long aLongitude, long aLatitude)**
Deletes a point of interest from the .OV2 file specified by the POI category argument. The POI to be deleted is the closest one to the given co-ordinates among the points of interest that are within a distance of 100 meters from there.

- **FlashMessage (... , int aMilliseconds)**
Flashes a message on screen in TTN, for the specified amount of milliseconds.

- **GetCurrentPosition(...)**
Gets current GPS position (latitude, longitude, speed in km/h and direction in degrees) and GPS status (general or receiving).

- **MakeUserPoiVisible (... , char* aFilename, long aVisibility)**
Enables or disables displaying a specific user-created POI type.

- **ShowCoordinatesOnMap (... , long aLongitude, long aLatitude)**
Centers the map at the point with given coordinates.

- **StartApplication (...)**
Starts TomTom Navigator in the background.

- **StopApplication (...)**
Stops Navigator if it was running, otherwise does nothing.

- **TomTomApplicationToForeground (...)**
Brings TTN to the foreground. The difference with 'BringNavigatorToForeground()' is that this method is quicker, but does not start up TTN if it wasn't already running [TT 2007].

2.3 SOFTWARE – Server Backoffice

The following section describes a high level SMS server approach to the problematic presented on this dissertation. Due to time constrains and the fact that an eventual implementation eluded the scope of this thesis this server was not implemented at the time of this dissertation.

2.3.1 SMS Protocol

The SMS protocol present on this dissertation was designed with an important guideline: maximization of the amount of information *per* SMS. This means that each message event type has the most amount of specific information; and also in each SMS a series of complete events can be sent until the 160 characters limit is reached. Taken in account the aforementioned facts the following paragraphs describes each of the fields of a multi event SMS message and also every single event type possible.

As a design guideline it was decided that either Server or User controlling the PDA will only send completed events in each SMS message, separated with the character “#”– Figure 2.3.

222222 ; 11; xxx ;yy ; 234 ;4789 # 222223 ; 33 ; yyy # (...)

Figure 2.3 – A complete coded event

Each event type has several fields, separated by semicolons with the first two entries on each single event being: a unique global ID attached to each given event (for univocal and tracking purposes), followed by the respective event type code, as shown on Table 2.1.

SMS type	Code
Fire	11
Weather	22
Auto	33
Water	44
Human	55
Misc	66

Table 2.1 –Event/Code conversion table

The remaining fields on each message are described on Table 2.2, with detailed explanation of each field being done on APPENDIX A – SMS Protocol

SMS type	Fire	Weather	Auto	Water	Human	Misc
Code	11	22	33	44	55	66
ID	ID	ID	ID	ID	ID	ID
Name	Event or Location	Station	Number	Supply	User	X
Enable	ON/OFF	X	ON/OFF	X	X	X
GPS Coordinates	Longitude Latitude	Longitude Latitude	Longitude Latitude	Longitude Latitude	X	X
Information	X	temperature humidity pressure wind speed wind orientation pluviosity	X	X	X	X
Auto type	X	X	Jeep(0) Truck(1)	X	X	X
State	X	X	Water capacity	Water capacity/level	X	X
Receiver type	X	X	X	X	Commander(0) All Personnel(1) Received(3)	X
Text	X	X	X	X	Free	X

Table 2.2 – SMS Protocol

Chapter 3.

SYSTEM ARCHITECTURE

This chapter presents the proposed architecture for the developed real time control system that manages the PDA device actions. The first section presents a global overview of the system. The second section describes the importance and the use of Windows Messages on this system. The third and fourth sections describe the Main and the auxiliary (Event and SMS) classes in detail. The fifth section deals with all the tracking, robustness and control issues that were taken in account on this system. Finally the sixth and final section proceeds to describe all the Interfaces designed for the system.

3.1. Introduction

The main objective of this dissertation was to create a multi-interface application that could handle, route and display various types of data in real time. The data is handled transparently to the user, through the Main Class and the auxiliary classes that run on a multi-threaded system. The communication aspects are divided into two types of concerns: external and internal. The external communication relating to the server communications is handled by the SMS Class from the lower level Serial Port data processing to the higher level SMS protocol. All the internal communication between the TTN and the two interfaces, “FFFpdaMain” and “FFFpdaSMS” is performed via Windows Messages.

Although the majority of the architecture can be separated and each module described by itself, other aspects do not and are an inherent part of all the code developed throughout the applications. The most prominent cases are: windows messages handling (both system and application defined) and TTN API function calls. In the case of the Windows™ Messages and due to its vital importance and overall usage on the system there is a chapter dedicated to describing its role and outlining the message map in the system. TT API calls have already been described on Chapter 2, so at this stage it remains important to refer that the classes that call and trade information with the TTN are: the Main and Events Classes. The program flow is synthesized on Figure 3.1.

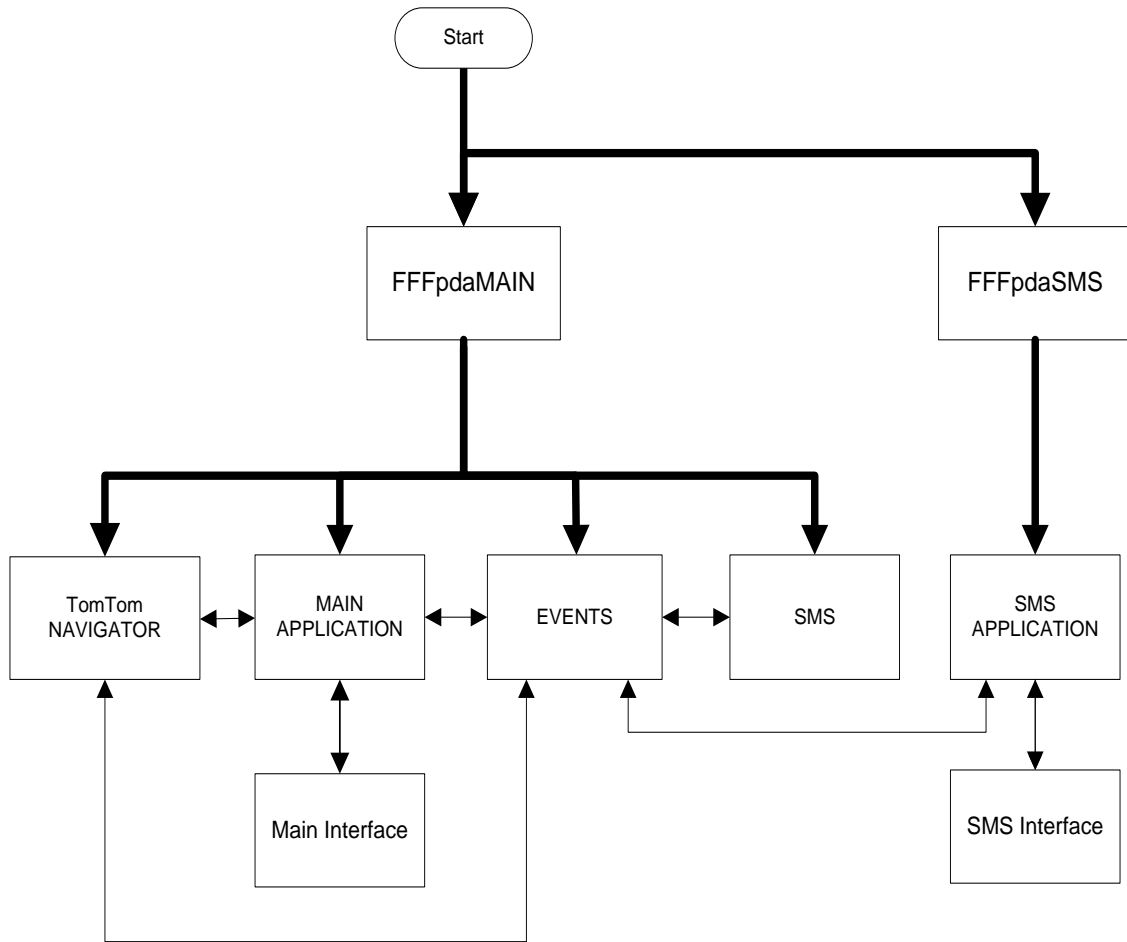


Figure 3.1 – Global system architecture

As Figure 3.1 shows, two applications are deployed on the OS boot-up: FFFpdaMAIN and FFFpdaSMS. FFFpdaMAIN is responsible for the launching of several events: TTN, the auxiliary Events and SMS Threads and its own interface. FFFpdaSMS on the other hand only launches its respective interface.

TTN – Figure 3.2 – communicates with the Main Thread in order to translate the hardware buttons requests into actions on the navigator itself. The main bulk of communication goes through the actions controlled by the Events Thread which determines the addition, updating or removal of a system related event (Fire, Vehicle, Weather Station or Water Reservoir) on the navigator itself.



Figure 3.2 – TTN Interface



Figure 3.3 – Main Interface

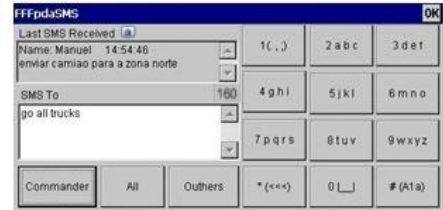


Figure 3.4 – SMS Interface

The SMS Thread is constantly checking for incoming or outgoing SMS messages. The appropriate conditioning is done for the messages that are received and passed to the Event thread and also to the messages that the Event thread requests to be sent to the server.

The Events thread as referred on the previous paragraphs is responsible for the Event Processing (incoming), Event Building (outgoing) and TTN Visual event handling. The fundamental interaction with the Main thread is also addressed for logging aspects and also establishing a connection with the other running applications. Finally, this thread also deals with the FFFpdSMS interface in terms of sending the typed Text and other specific actions that will be explained later in this chapter.

The Main class can be described as a Global System Manager in terms of its importance on all the events: interconnecting threads through message routing, hardware buttons handling, logging all the events that occurred, system robustness control and also managing its own interface – Figure 3.3.

The FFFpdSMS Application mainly handles the interface – Figure 3.4 – which consists of an alphanumeric keypad and some Text boxes for visualization purposes.

3.2. Windows Messages

“Unlike MS-DOS-based applications, Windows-based applications are event-driven. They do not make explicit function calls (such as C run-time library calls) to obtain input. Instead, they wait for the system to pass input to them. The system passes all input for an application to the various windows in the application. Each window has a function, called a window procedure that the system calls whenever it has input for the window. The window procedure processes the input and returns control to the system.

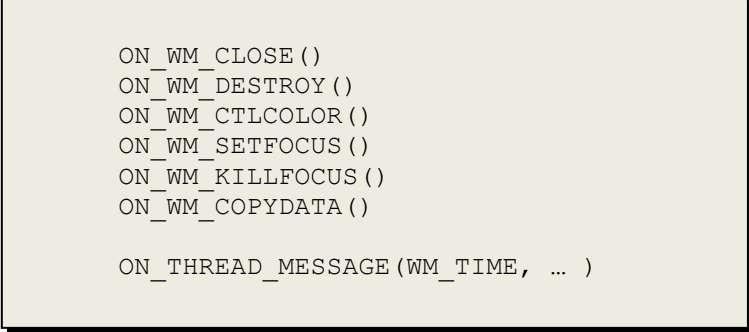
The system passes input to a window procedure in the form of messages. Messages are generated by both the system and applications. The system generates a message at each input event — for

example, when the user types, moves the mouse, or clicks a control such as a scroll bar. The system also generates messages in response to changes in the system brought about by an application, such as when an application changes the pool of system font resources or resizes one of its windows. An application can generate messages to direct its own windows to perform tasks or to communicate with windows in other applications.

The system sends a message to a window procedure with a set of four parameters: a window handle, a message identifier, and two values called message parameters. The window handle identifies the window for which the message is intended. The system uses it to determine which window procedure should receive the message. A message identifier is a named constant that identifies the purpose of a message. When a window procedure receives a message, it uses a message identifier to determine how to process the message. Message parameters specify data or the location of data used by a window procedure when processing a message. The meaning and value of the message parameters depend on the message. A message parameter can contain an integer, packed bit flags, a pointer to a structure containing additional data, and so on” [MSDN 2008a].

Message Map

“The system sends or posts a system-defined message when it communicates with an application. It uses these messages to control the operations of applications and to provide input and other information for applications to process. An application can also send or post system-defined messages. Applications generally use these messages to control the operation of control windows created by using preregistered window classes” [MSDN 2008a]. On Figure 3.5 all the system-defined messages handled on this system are listed:



```
ON_WM_CLOSE ()
ON_WM_DESTROY ()
ON_WM_CTLCOLOR ()
ON_WM_SETFOCUS ()
ON_WM_KILLFOCUS ()
ON_WM_COPYDATA ()

ON_THREAD_MESSAGE (WM_TIME, ... )
```

Figure 3.5 – System defined messages

On the case of this dissertation, custom *application-defined messages* were created for the system and its different applications as well as to communicate with other simultaneous processes. The window procedure that receives them must interpret the messages and provide appropriate processing. This type of information transmission method provides the system with a very flexible way of passing data between threads on the same application (ex: Main and Events) and also exchanging data between different applications (Main and SMS).

All systems keep a message queue in order to keep the message flow organized and controlled and with the exception of the **WM_PAINT** message, the **WM_TIMER** message, and the **WM_QUIT** message, the system always posts messages at the end of a message queue. This ensures that a window receives its input messages in the proper first in, first out (FIFO) sequence.

There are two approaches to message processing: *PostMessage()* and *SendMessage()*. ”*The **SendMessage** function sends the message to the window procedure corresponding to the given window. The function waits until the window procedure completes processing and then returns the message result. On the other hand an application can post a message without specifying a window. If the application supplies a NULL window handle when calling **PostMessage**, the message is posted to the queue associated with the current thread. Because no window handle is specified, the application must process the message in the message loop. This is one way to create a message that applies to the entire application, instead of to a specific window*” [MSDN 2008a].

All the custom *application-defined messages* that were created for this project are shown on Figure 3.6:

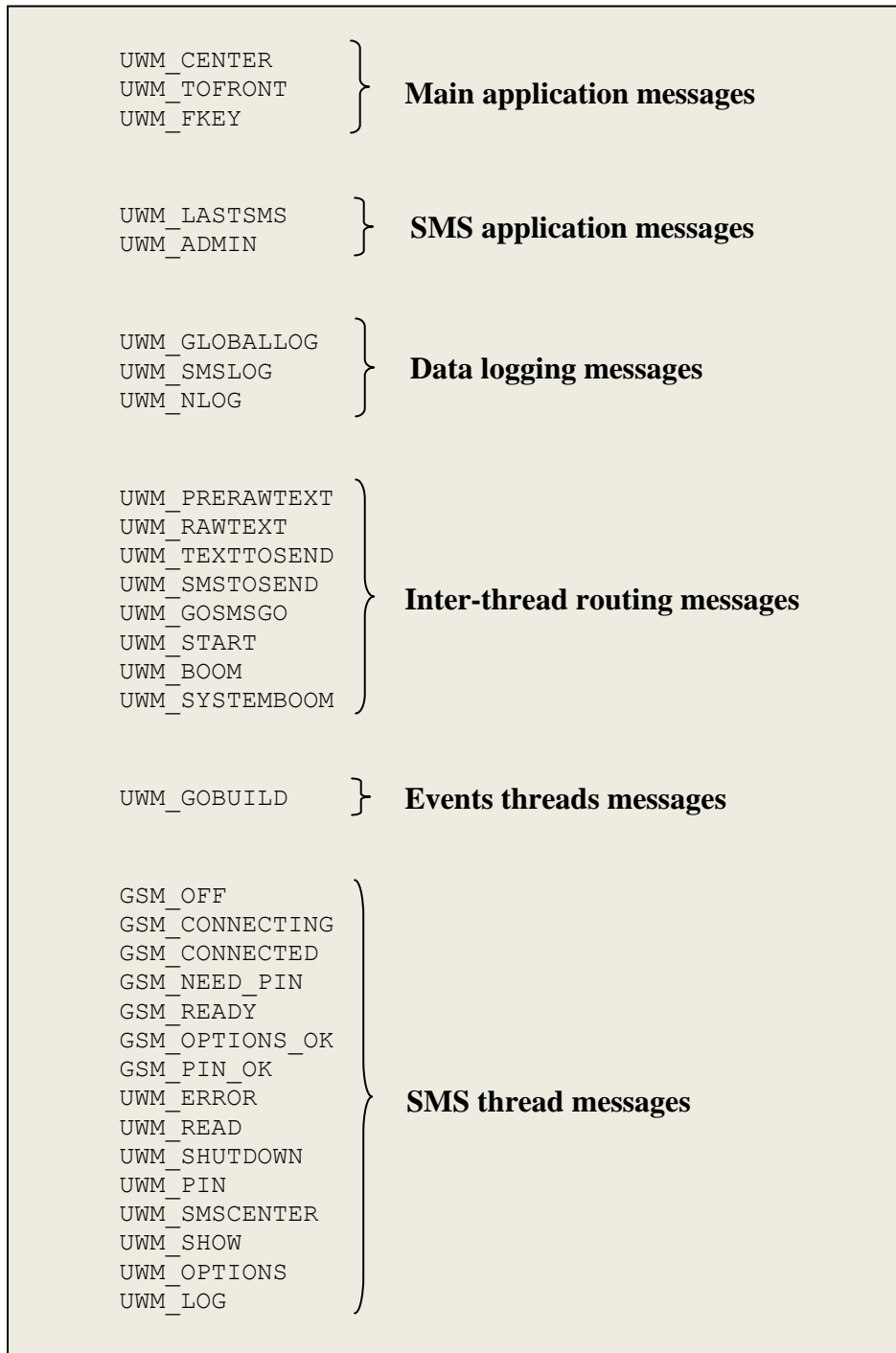


Figure 3.6 – Custom application defined messages

The particular data path between intra-thread, inter-thread messages, and also through the applications that each of these messages follows is explained in detail on Chapter 3.3.1.2.

3.3. MAIN Application

The next chapter explains and describes in detail the actions and procedures of the denominated FFFpdaMAIN application, which specifically translates on all the code developed in the FFFpdaMainDlg, FFFauxiliar and SMS classes. Although all user input is handled through a dialog window programmed on this class, there is a Chapter – 3.6, on which all the interfaces are described.

The first section will deal with the code that was written on the Main Class itself that concerns mainly with initializations and terminations processes, and hardware control handling. It also specifies the windows messages and their routing between applications and threads. The second and last section will describe all the actions performed and handled by the two auxiliary classes: Events and SMS.

3.3.1 Main Class – Global Algorithm

The sequence of the algorithm – Algorithm 3.1 – begins with the **initialization**, where the following actions take place: all hardware Hotkeys are registered; the existence of the text files for the Global and GSM logs, *globallog#.txt* and *gsmlog#.txt* is verified. Also both Events and SMS threads are started and the TTN application is launched.

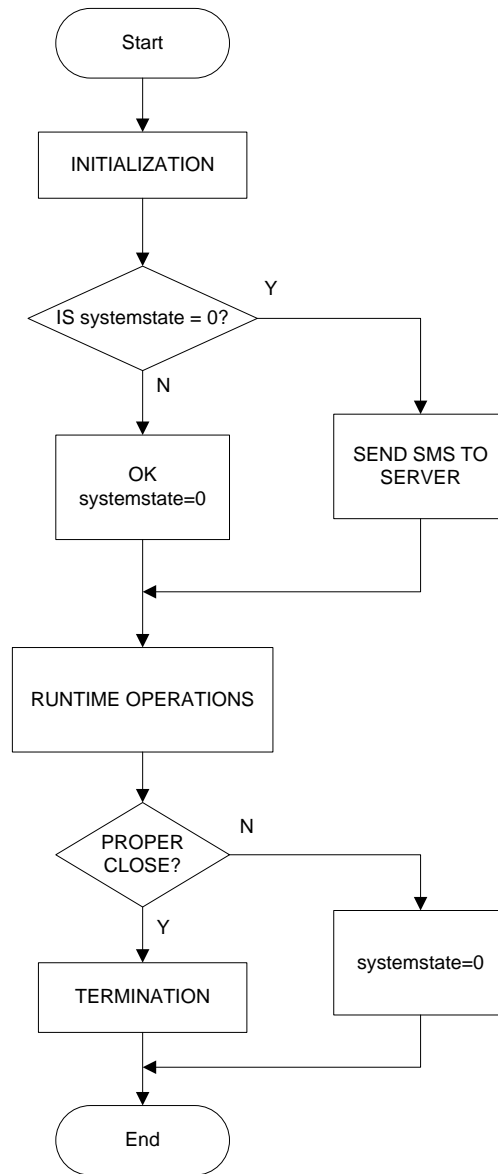
The aspects that deal with robustness, auto-recovery, and self sufficiency are described in greater detail on a latter chapter. It is important to point out that a **system crash check** is performed at this stage of the process and, if a crash did occur, a response mechanism is initiated. The process begins by sending a SMS text message to the server with the date of the first event received since the PDA was started. This will enable the server to resend all the events from that initial moment till the present, which brings the system up-to-date, and allows for a complete information recovery.

From this stage onwards it is all about **runtime operations**: hardware control, logging and windows message routing, which will be detailed on the following sub-chapters.

In accordance with the initial system crash check, a **proper closing** check action is enabled so that the system knows how to proceed on the next startup. Specifically, if the system is correctly terminated a character “1” will be written on the *system.txt*, else the “0” that was written on the **initialization** will remain and the recovery process will be called on the next startup.

The final stage of the application is the shutdown of the PDA which involves performing all the proper closing and **termination** actions that were initiated earlier on the algorithm. By order of

occurrence: unregister the hardware Hotkeys, close the Global and GSM logs text files with the adequate terminator (“end of operation”); close the Events and SMS threads and close the TTN.



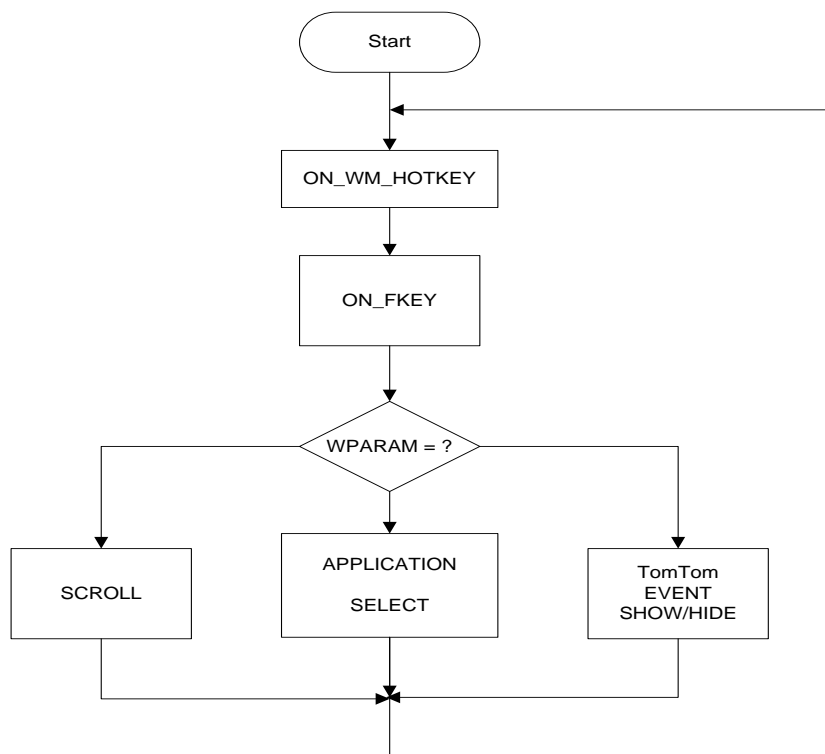
Algorithm 3.1 – Main class global

3.3.1.1 Hardware Control

All the PDA soft-touch buttons are handled using the HotKey capability that a Windows™ OS has.

This can be simply translated into the following definition: *a key or key combination that causes some function to occur in the computer, no matter what else is currently running* [ZDNET 2008a].

According to the Microsoft Development Network™ (MSDN) the process action follows this course when a key is pressed and the system looks for a match against all hot keys. When finding a match, the system posts the **WM_HOTKEY** message to the message queue of the window with which the hot key is associated. However, in the case that the hot key is not associated with a window, the message is posted to the thread associated with the hot key. This function cannot associate a hot key with a window created by another thread. [MSDN 2008b]



Algorithm 3.2 – Hardware control

In terms of the algorithm implemented as shown on Algorithm 3.2, the **WM_HOTKEY** message is caught by the **ON_FKEY** function and subsequently handled based on its Hotkey value. This value is passed through the *wparam* parameter. At this stage there are three courses of action: text scrolling, application selection, and show/hide events.

Main Application Information Scrolling Keys

One of the goals of this dissertation was to maximize the user-friendly interaction with the PDA. Taken the fact that the use of gloves is mandatory and the event information flow of data is continuous, the capability of scrolling text is of the utmost importance. So, the use of two hardware buttons on the right side of the PDA was attributed to Upwards and Downwards Scrolling actions – Figure 3.7. This gives a more analog and “in-control” feel. So the only screen touching interaction would be to choose the desired EditText on which to perform the the action.

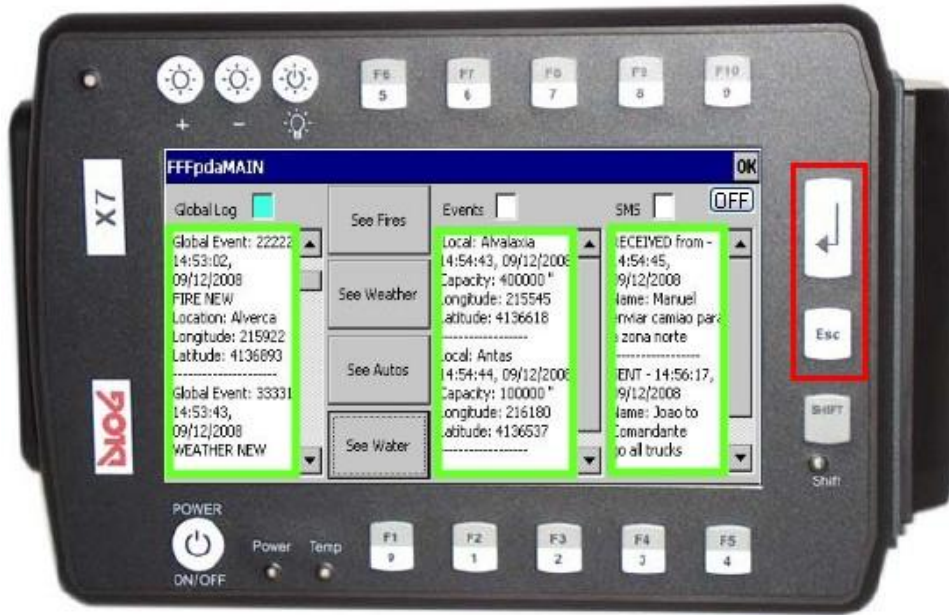
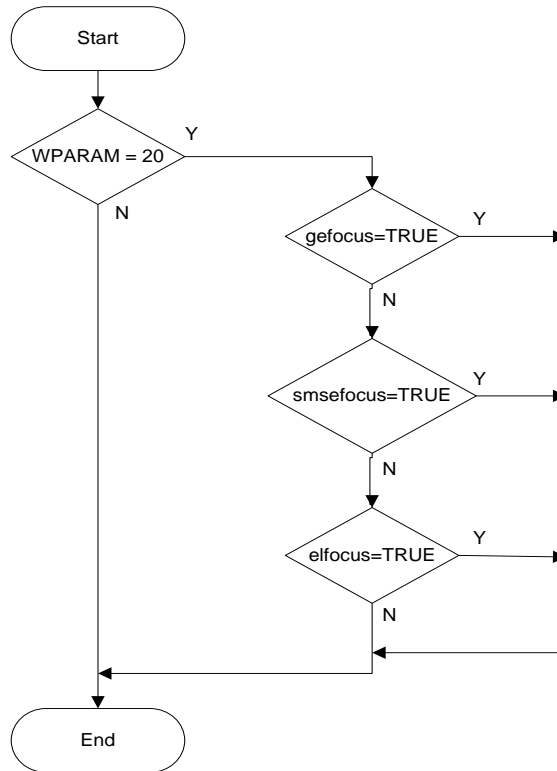


Figure 3.7 – Edit boxes scrolling hardware keys

The system knows what EditText to affect through the Hotkey filter previously explained and a series of attributed flags as it can be observed on Algorithm 3.3. The Scroll Down algorithm is similar only changing the flow of the text on the opposite direction.



Algorithm 3.3 – Scroll UP

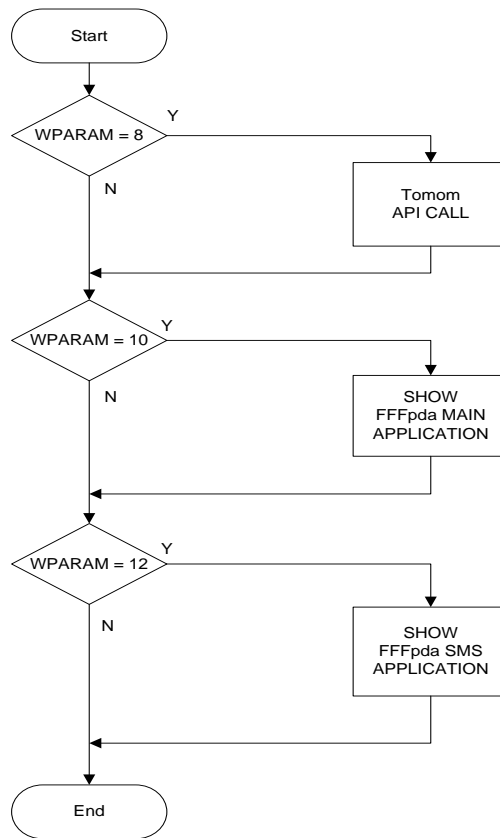
Application Selection Keys

The chosen keys were the F1, F3 and F5 as Figure 3.8 illustrates. The remaining adjacent keys, F2 and F4 were disabled for ergonomic and user friendly aspects in order to simplify and focus the firemen's range of possibilities.

The handling of these keys derives from the Hardware Control algorithm explained at the beginning of this chapter – Algorithm 3.4. The use of auxiliary flags determines what window the user is interacting with.



Figure 3.8 – Application selection hardware keys



Algorithm 3.4 – Application selection

TomTom™ Events Show/Hide Keys

The importance of intelligible interfaces is of vital importance especially on small screens and in harsh environments scenarios as is the case of this dissertation. However, sometimes visual data overload can occur. The capability to perform correct event filtering may just give the proper visualization needed by the user.

In accordance to those notions the top row of hardware buttons ranging from F6 to F10 – Figure 3.9 – was attributed to Show and Hide actions on the visual representations of the Events. These Events are: Fire, Weather, Vehicle and Water; the Event/key correspondence is detailed on Table 3.1.

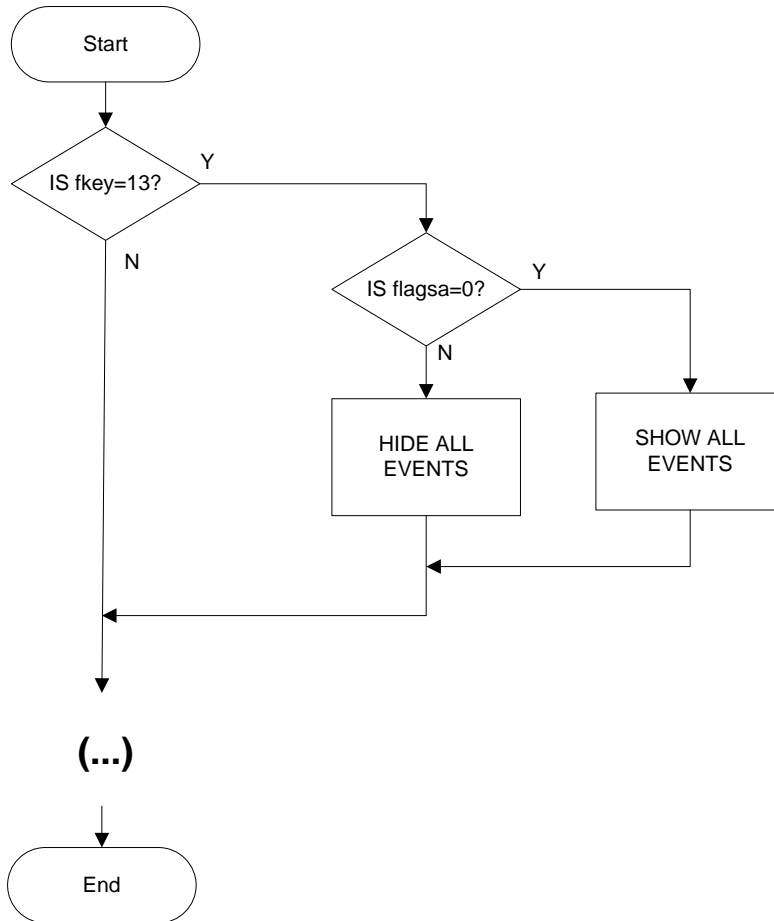


Figure 3.9 – TT events Show/Hide hardware keys

Hardware Key	Events
F6	All Events
F7	Fire
F8	Weather
F9	Vehicle
F10	Water

Table 3.1 – TT events Show/Hide keys

In a similar action with the previous algorithms all the referred events are caught by the **ON_FKEY** procedure. On this specific case the auxiliary flags allow the system to proceed in terms of the correct action required, to Show or Hide the desired type of event. This algorithm is of similar characteristics on all the keys associated to this type of actions, for illustrations purposes the *All Events* action is depicted on Algorithm 3.5.



Algorithm 3.5 – TT All Events Show/Hide

3.3.1.2 Message Routing between Threads and Applications

The messages that are exchanged between the various components have already been enunciated on Chapter 3.2. So on this chapter their objective, routes and destinations will be described in detail in order to stand out its importance and relevance through the whole system.

3.3.1.2.1 Between Applications

There are many ways to achieve inter-process communication using Visual C++. The main reason is that 32-bit applications run in a separate address space, so the address of a string in one application is not meaningful to another application in a different address space. Using the *SendMessage()* API function to pass a WM_COPYDATA message overcomes this problem.

1) Send user composed SMS text message

After the user has written the desired text on the SMS application, this text string has to pass it to the main application where it will be rerouted to the SMS thread. This is done by sending a **WM_COPYDATA** message. The first part is the telephone number of the chosen recipient (Commander, All) and the second part is text message content itself – Figure 3.10.

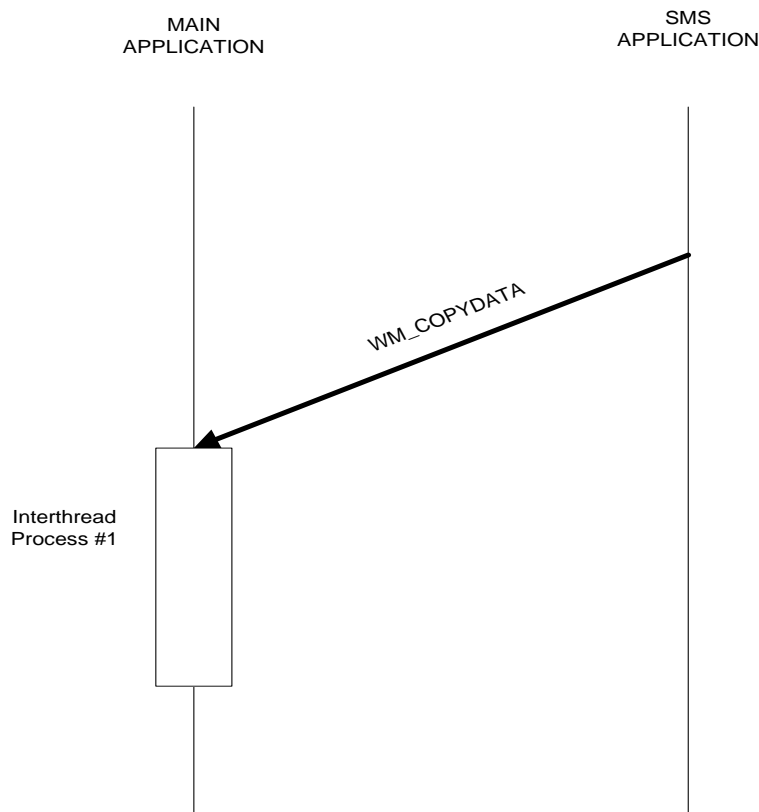


Figure 3.10 – SMS to MAIN message process

2) Display the last received SMS text message

To keep the user in-loop with human text messages being received by the PDA, the functionality of showing the last received human text message was added to the SMS application. So when this kind of message arrives, the system sends through the Main application a **WM_COPYDATA** message with the name of the sender and the received text – Figure 3.11.

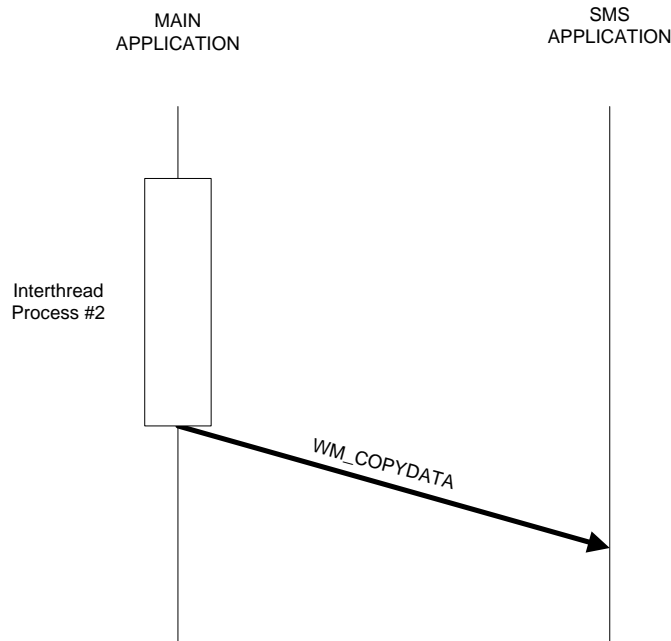


Figure 3.11 – MAIN to SMS message process

3.3.1.2.2 Between Threads

As stated before the need for a FIFO message loop is important to keep the correct chronology of the events Input/Output (I/O) flow. Also the use of the *PostMessage()* enables the system to continue its actions without runtime interruption. A limitation of Windows messages parameters is that they can only contain integers, but using a pointer to a structure, additional data can be “sent” to other threads. The inter thread messages can be divided in to two different types: General and Logs.

1) General Messages

General messages consist on the following actions: Outgoing SMS message to server, Incoming SMS message from server, System Crash Recovery SMS message and the Start Truck position Timer.

It is important to mention that the Main application will act merely as a router between threads, not producing any data modification. This task is left to the appropriate procedures on the corresponding threads.

Outgoing SMS messages to server

If a SMS text message needs to be sent through a request from the SMS application the text string is passed through a **WM_COPYDATA** message which now is processed on the Events thread. To pass the information the **UWM_TEXTTOSEND** message is sent and a specific thread process, *BuildEvent()*, will construct the message according to the defined SMS protocol. It then send it again to the Main thread through a **UWM_SMSTOSEND** message. The formatted text will finally be sent to the SMS thread via the **UWM_GOSMSGO** message which ultimately sends it to the server.

As the text to be sent is being rerouted to the SMS thread two additional messages are sent the Main thread. The first message, **UWM_GLOBALLOG**, will log the event entry. The second, **UWM_SMSLOG**, is sent if the message is of the Human type so that the system will write the text on the SMS Dialog EditBox. This last process – Figure 3.12 – will be explained in more detail on Chapter 3.6.

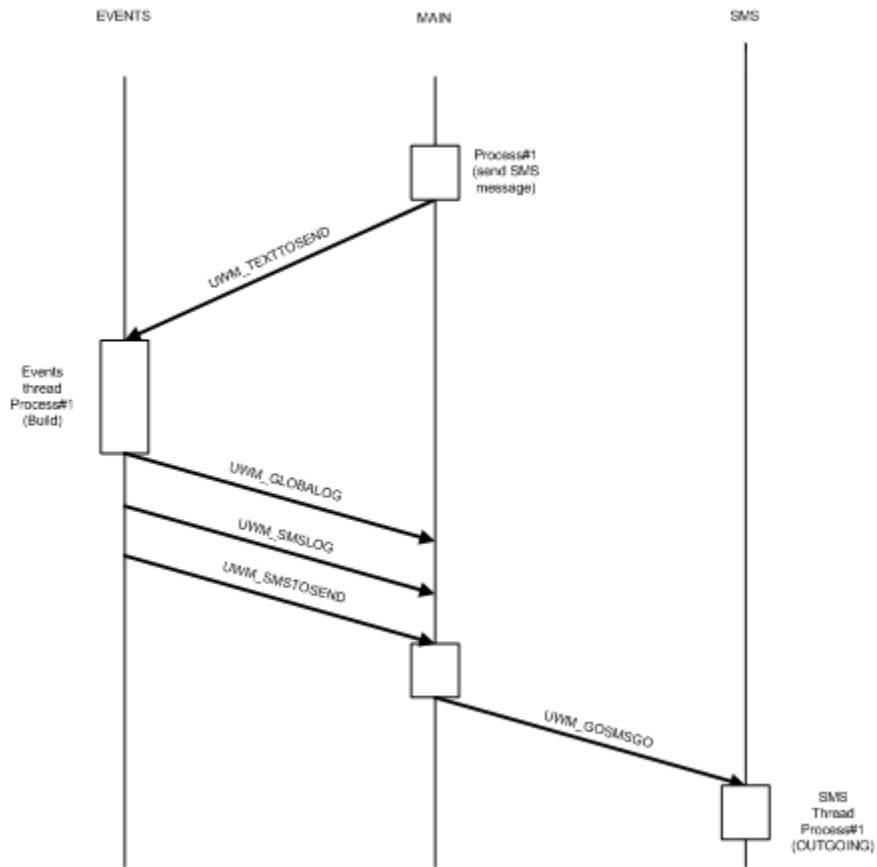


Figure 3.12 – Outgoing message processes

Incoming SMS message from server

When a SMS text message arrives, the SMS thread must pass the raw text to the Main thread through the **UWM_PRERAWTEXT** message. From this point it is rerouted to the Event thread using the **UWM_RAWTEXT** message where it will be handled by the *ProcessEvent()* function.

Similarly to the Outgoing process two messages are sent the Main thread. The **UWM_GLOBALLOG** message which will log the incoming event entry. If the message is of Human type, the message **UWM_SMSLOG** will write it on the SMS Dialog EditBox – Figure 3.13.

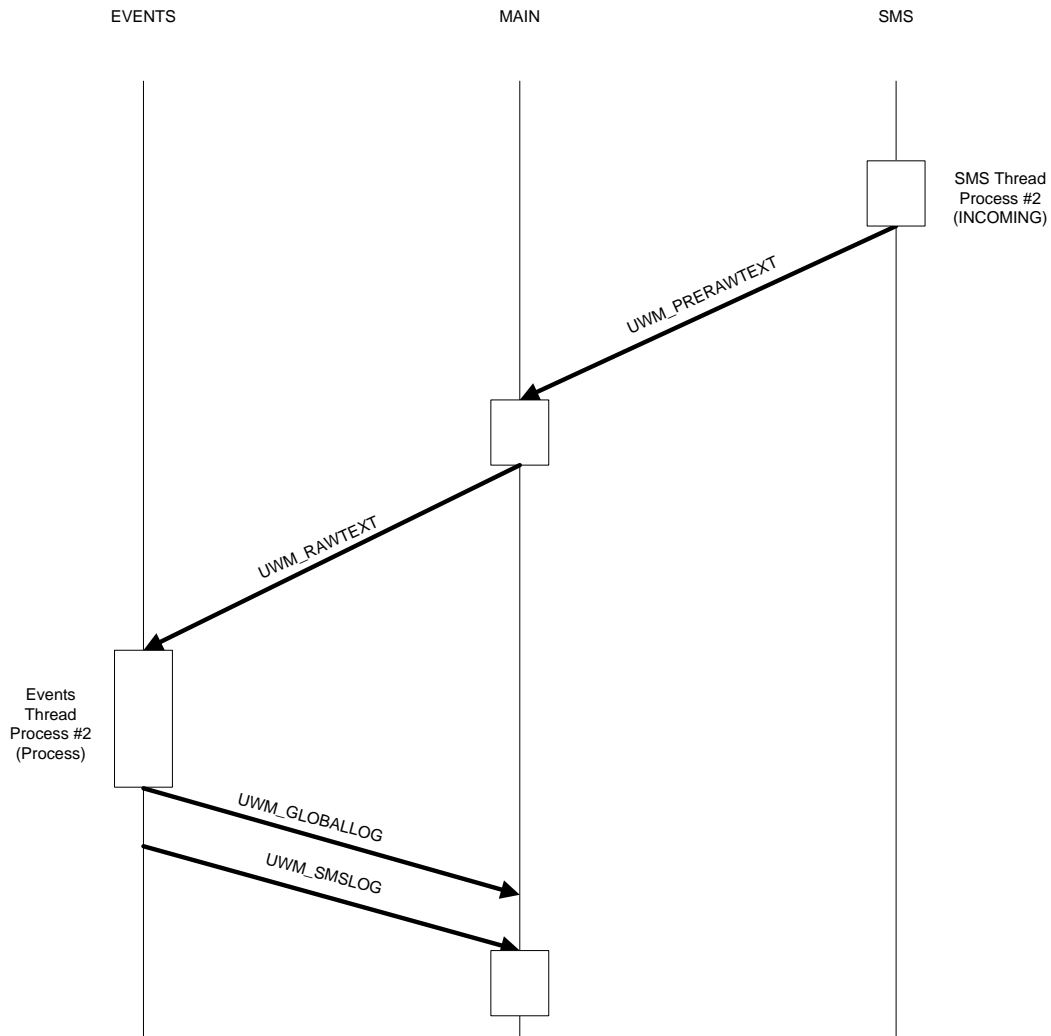


Figure 3.13 – Incoming message processes

System Crash Recovery SMS message

If the system is recovering from a crash, a special text message is sent, through the **UWM_SYSTEMBOOM** message to the Events thread. The next action consists on composing a SMS message to inform the server that all the information from a certain date onwards needs to be resent to the system. This text message is then forwarded to the Main thread through the **UWM_SMSTOSEND** message. As with the outgoing process the formatted text will be sent to the SMS thread via the **UWM_GOSMSGO** message which will relay it to the server.

Similarly to the processes previously described, a message is sent to the Main thread, **UWM_GLOBALLOG**, which will log the crash event recovery event – Figure 3.14.

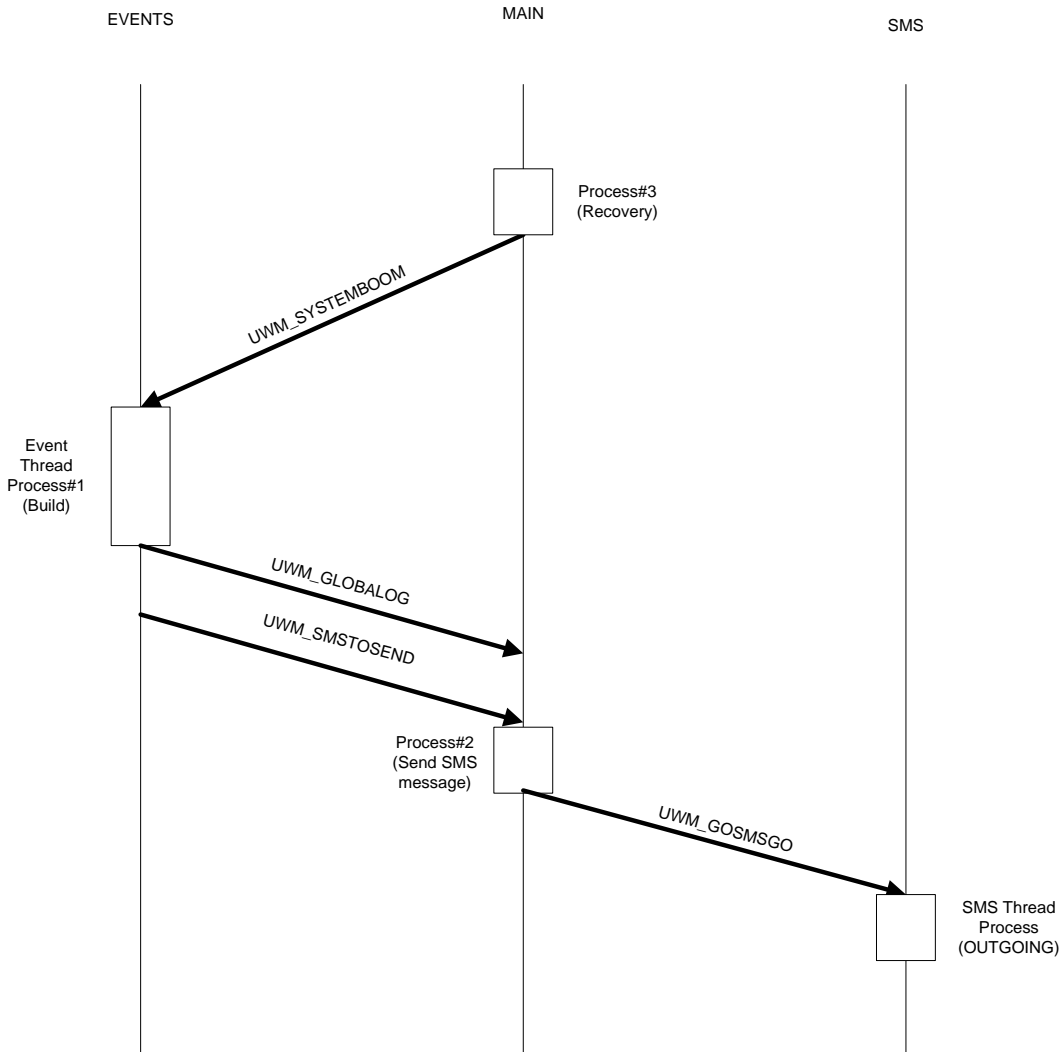


Figure 3.14 – System crash message process

Start Truck position timer and SMS thread

The vehicle where the PDA is mounted must obviously send its own GPS position to the server. However, it should only do that if its position has changed more than a certain amount of degrees every 60 seconds. Therefore, a Truck refresh position timer process must be started at system startup, which is done via the **UWM_BOOM** message to the Events thread.

The SMS thread, as it will be explained in detail on the Chapter 3.3.2.1, needs a specific windows message, **UWM_START**, so that it can initiate all the low-level mechanisms (serial

port, modem communication). This will put the modem on standby ready for any text messaging traffic that will follow – Figure 3.15.

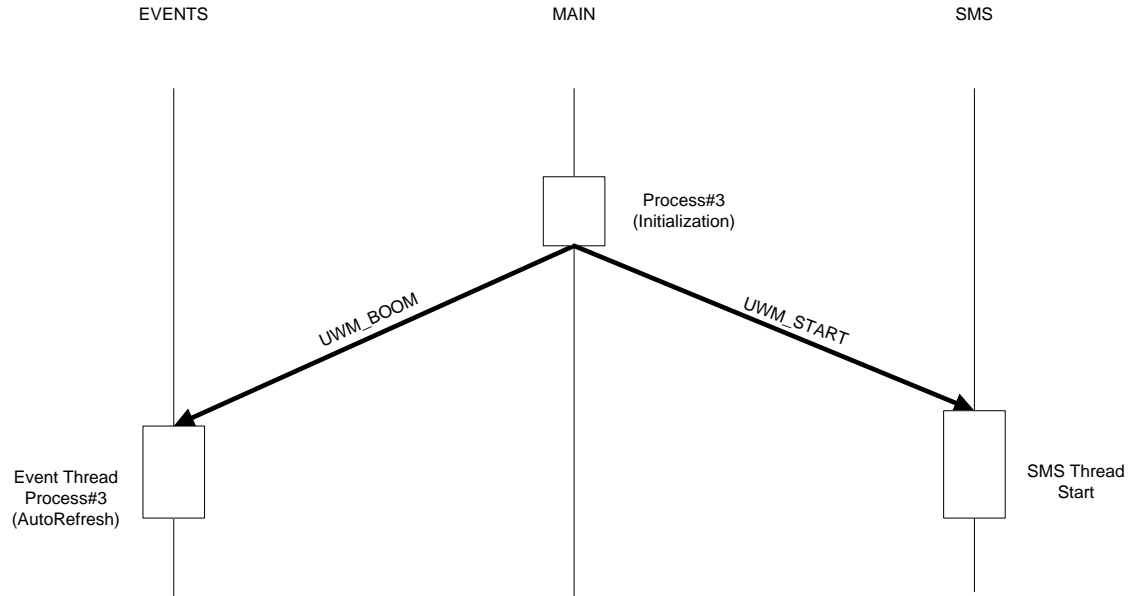


Figure 3.15 – Start position timer and SMS thread message process

2) Log Messages

This section will address the specific messages (**UWM_GLOBALLOG** and **UWM_GSMLOG**) that are sent between the Main and the Events threads, when a logging action is required. The global log entries will be shown on the Global Log EditBox. Both log entries will be written on the respective Text Files (*globallog.txt* and *gsmlog.txt*) as evidenced on the following diagrams – Figure 3.16 and Figure 3.17.

Events

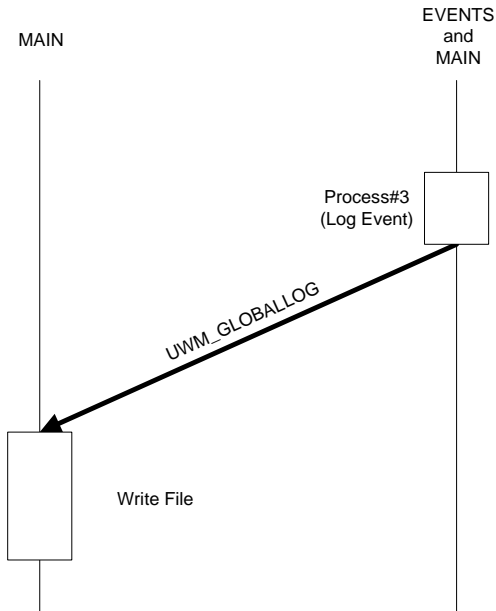


Figure 3.16 – Global Log message process

GSM

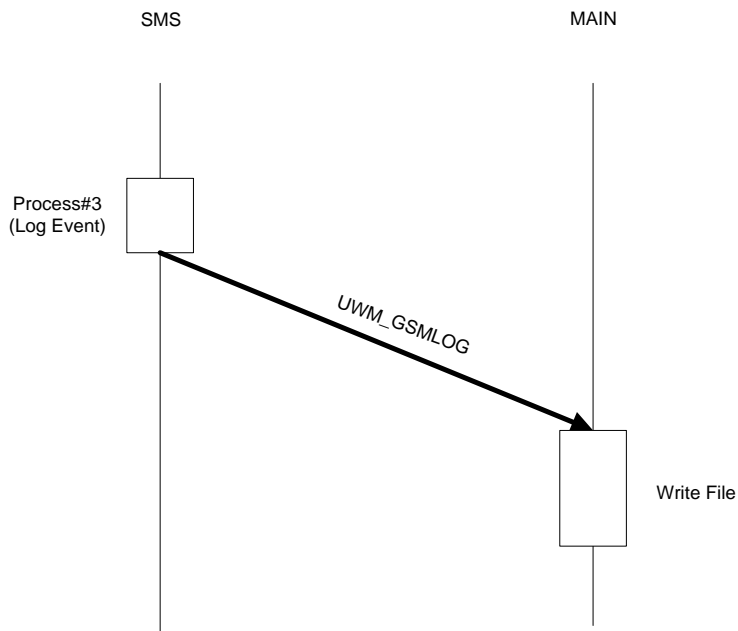


Figure 3.17 – GSM Log message process

3.3.2 Auxiliar Classes – Algorithms

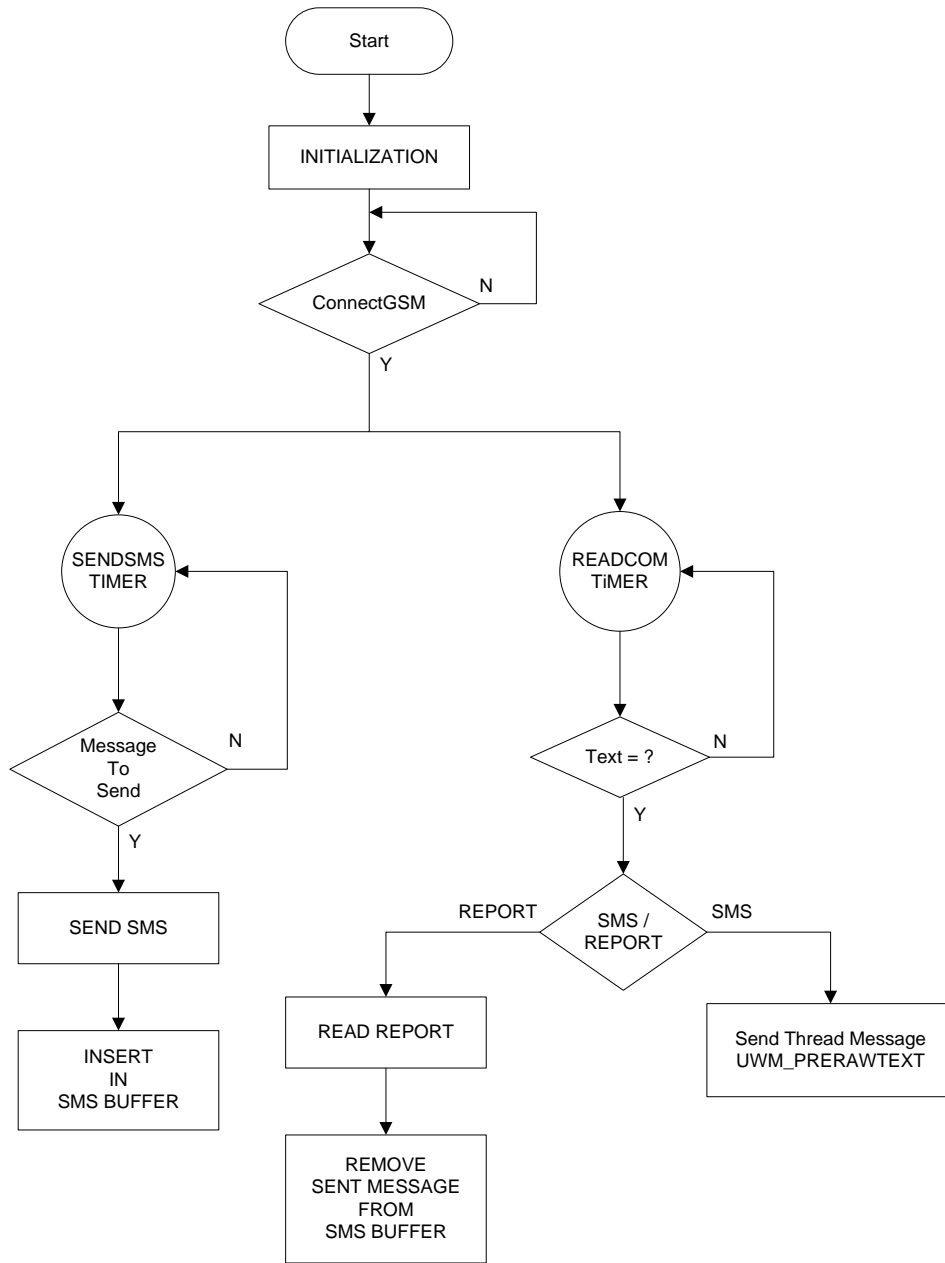
The following classes, SMS and Events, were created and adapted to provide more flexibility to the system, lighten the work load on the Main class and allow a more independent handling process. The only drawback being the information passing between classes had to be done through the described windows messages with its inherent advantages and disadvantages.

The first class that will be described is the SMS class which deals with all the GSM communication protocol. The second auxiliary class is the Events class which deals with all mechanisms that involve Event reception, processing and construction of the appropriate text messages.

3.3.2.1 GSM Communication

In order for the application to receive the Text Messages from the server a GSM Protocol had to be implemented. This task was greatly facilitated due to the fact that a protocol with a similar configuration was already implemented by NGNS Software Developer Nuno Pinto(SMS Class) which itself uses a Serial Communication Class implemented by Prof. Pedro Vieira. A certain degree of adaptation was imperative in order for these classes to work with the rest of the system, specifically:

- All the Log messages from the Modem, GSM, and Serial classes were sent to the Main Application for the appropriate handling.
- The SIM Card operation of checking for a correct Personal Identification Number (PIN) does not make much sense in the context on which the device will operate.
- On reading the serial data feed, the buffer size on the window that handles the incoming bytes was increased for a correct handling of a full raw message sent from the Server Center.



Algorithm 3.6 – GSM communication

The sequence of the algorithm begins with the **initialization**, where the SMS Center server number is retrieved from a text file, *numbers.txt*. This text file provides a more flexible and swift manner of changing its value whenever the Administrator desires.

The next step *ConnectGSM()*, initiates the modem communication and prepares the GSM modem for its functions. The algorithm then works on an internal closed loop monitored by two main

timers: **SENDSMS Timer** and **READCOM Timer**. These timers have the functions of waiting for a formatted text message to be sent or wait for a new text/report message arrival. Without going into much detail as this class was fully completed and functional, it is important to emphasize a specific point. In order to manage unsent messages that are waiting in queue, an internal thread message loop was used. The removal of sent messages from this loop is done after the reception of the correspondent report has been received.

As a final note it is important to refer that upon a text message arrival the information is passed to the Main class using the windows message **UWM_PRERAWTEXT**.

3.3.2.2 Event Processing

This sub-chapter has the objective of describing the incoming process of handling the raw SMS Text sent by the FFF Server. The outgoing process of sending all application Text Messages (human or system ones) is described on the previous classes (SMS and Serial).

One of the most important characteristics of this project is the ability to provide access to data from multiple occurrences of five types of Events (**Fire, Weather, Auto, Water** and **Human**). The necessity to create five *struct* types arose in order to provide the Firemen with specific information. This information concerns both geographical visualization, as well as specific event related data, such as: Fire location and status as a Fire event– Figure 3.18; meteorological detailed information and location from a Weather Station, as a Weather event; to Figure – 3.19; vehicle details information and location as an Auto event – Figure 3.20; water supply location and capacity availability as a Water event – Figure 3.21; and finally, human text message details and content as a Human event – Figure 3.22.

```
struct AFires
{
    EventId
    IO
    Name
    Enable
    Time
    Longitude
    Latitude
}
```

Figure 3.18 – Fire event detail

```
struct AWeathers  
  
    EventId  
    Name  
    Time  
    Temperature  
    Humidity  
    Pressure  
    WindSpeed  
    WindOrientation  
    Pluviosity  
    Longitude  
    Latitude
```

Figure 3.19 – Weather event detail

```
struct AAutos  
  
    EventId  
    IO  
    Name  
    Time  
    Enable  
    AutoType  
    State  
    Longitude  
    Latitude
```

Figure 3.20 – Auto event detail

```
struct AWaters  
  
    EventId  
    Name  
    Time  
    State  
    Longitude  
    Latitude
```

Figure 3.21 – Water event detail

```

struct AHumans
    EventId
    IO
    Name
    NameReceiver
    Time
    Text
    nText

```

Figure 3.22 – Human event detail

To maintain a limited file size and avoid program memory overflow, the dimension of the arrays of *structs* was limited depending on its specific relevance and recurrence probability. The following Table 3.2 illustrates this fact.

Struct Type	Vector Size
Fires	30
Weathers	30
Autos	50
Waters	30
Humans	50

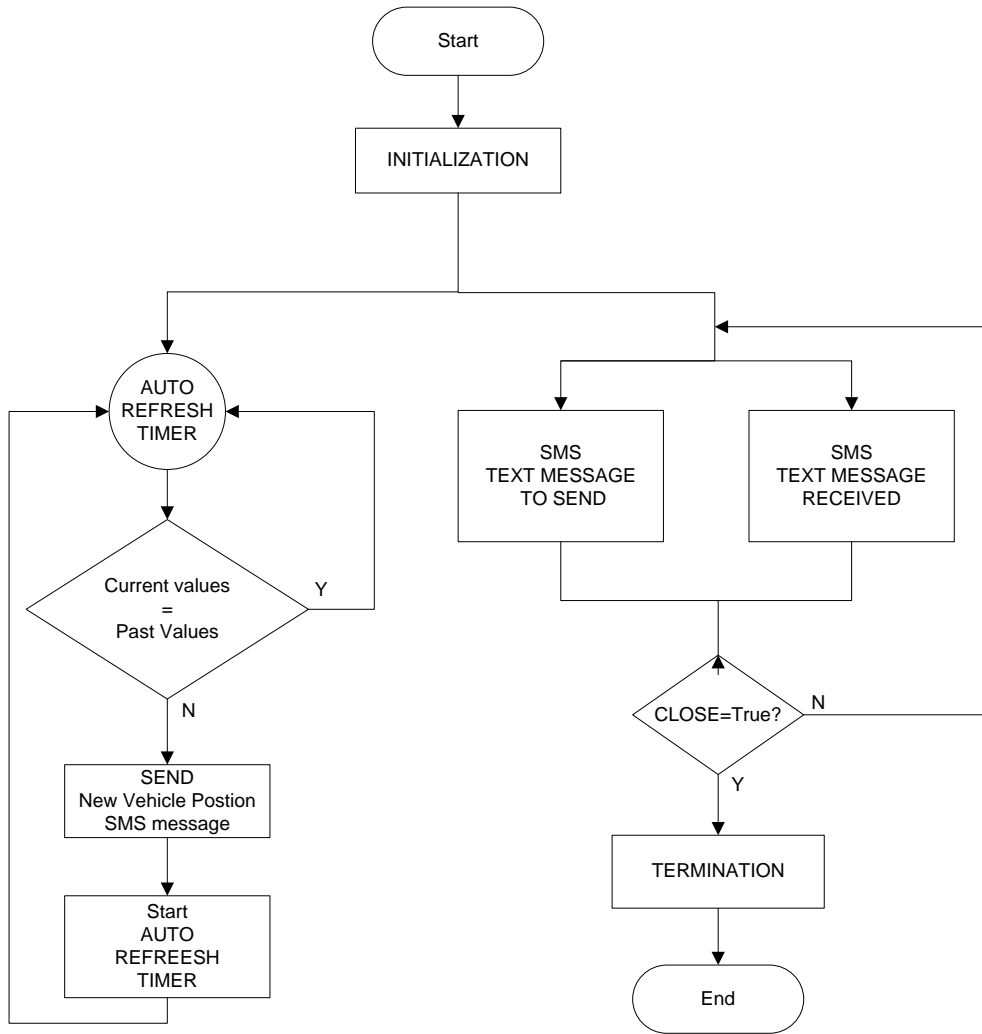
Table 3.2 – Events *struct* type and array size correspondence

Global Event Process

The sequence of the algorithm – Algorithm 3.7 – begins with the **initialization** stage, where a series of actions are done. The *structs* memory allocation is performed with values stated on Table 3.2. The systems own event number (*eventnum*) is generated. The initial system GPS coordinates are set which are related to the **Auto Position Timer** handling process. Finally the SMS Server Center number is retrieved from the *numbers.txt* text file.

The next stage can be described as parallel computing due to the fact that three processes will be running at the same time: **Truck Position Refresh**, an Incoming event task (**SMS text message to send**) and an Outgoing event task (**SMS text message received**). The Truck Position Refresh process is a very simple algorithm. It checks and keeps in memory the vehicles current GPS coordinates every

60 seconds and performs a comparison to the previous stored coordinates. If the position of the vehicle has not moved within a radius of one meter, or in angle terms more than 0.000189 degrees in Longitude and 0.000112 degrees in Latitude, the system will not take any action. However if the position has changed a new SMS text message will be sent to the server to inform of the update on the vehicle relative position.

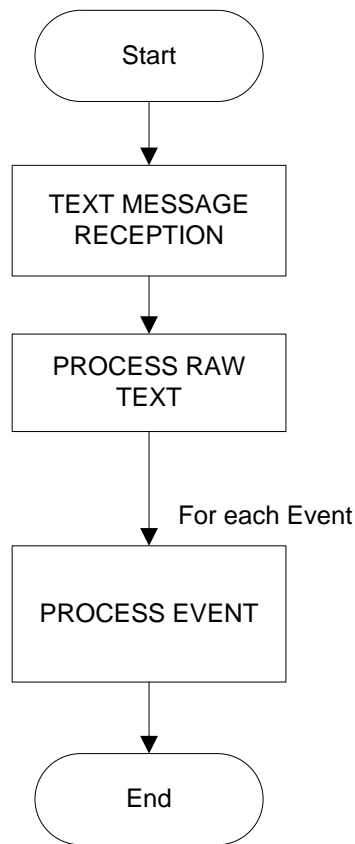


Algorithm 3.7 – Event class global

To respect the *struct* size table, before each new event is added, a dimension check is done on the respective *struct*. If the event equals the current dimension the index, of that particular event is reset to zero, and the algorithm continues to proceed as the next paragraphs will describe.

1) Incoming Event

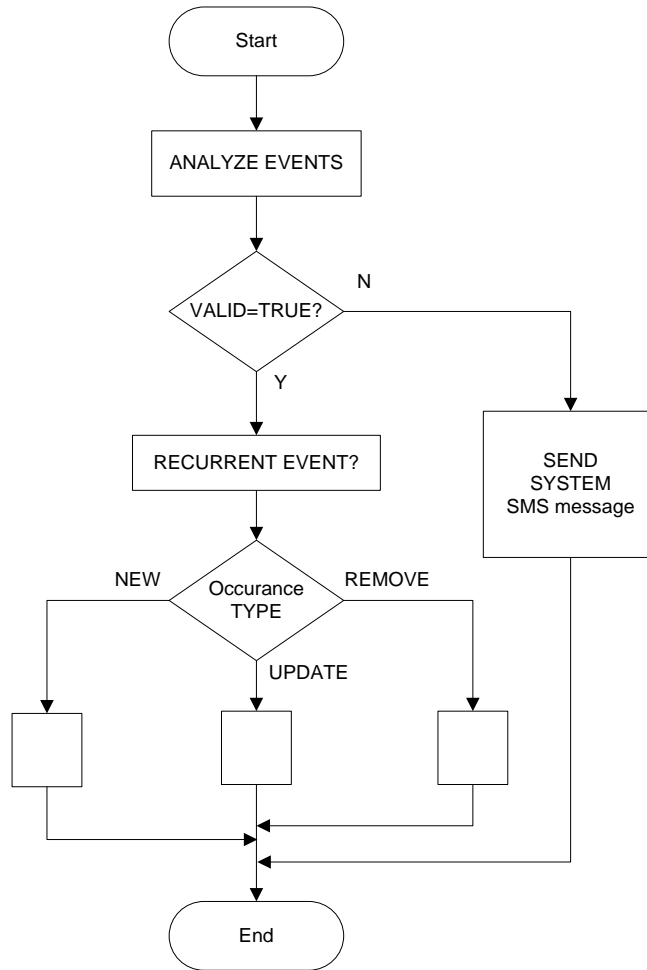
The incoming event algorithm starts with the **Text Message reception** process on the SMS thread. This sends the unformatted text to the Events thread through the windows message process previously explained. The raw text must be processed and its events parsed. This action is performed by the *ProcessRawText()* function which is basically a *for loop* that calls *ProcessEvent()* for each event – Algorithm 3.8.



Algorithm 3.8 – Incoming event

Process Event

For each event type this function is called– Algorithm 3.9. It starts by doing an event integrity-check and subsequent validity of the received event, through *AnalyseEvents()*. In case of an invalid occurrence it generates a pre-formatted SMS text message for the generic event, for example an invalid water event – Figure 3.23.



Algorithm 3.9 – Process Event

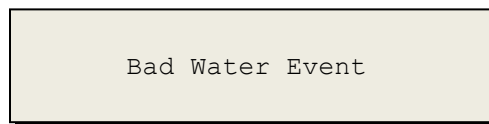


Figure 3.23 – Invalid Water event

This text message is then sent to the server, where the appropriate action will be performed. The next step consists of checking the reoccurrence of the event, so that the system knows what action to take. It can create a **NEW event**, or on the other hand **UPDATE** or **DELETE** an existent one. All the Events types and occurrence actions are shown on Table 3.3.

Event/Occurrence	NEW	UPDATE	DELETE
Fires	Y	Y	Y
Weathers	Y	Y	N
Autos	Y	Y	Y
Waters	Y	Y	N
Humans	Y	N	N

Table 3.3 – Event/Occurrence types

All of these actions are sent to the Main thread for logging purposes. Concerning the TTN a POI type is added with the type of event “Event” and the Event index “#” which translates on the following tag, in the case of a Fire Event:

Fire 1

A text message flashes on the navigator screen during five seconds with the following possible formats:

A “event” has been added !

A “event” has been updated !

A “event” has been removed!

The complete TT messages correspondence is shown on Table 3.4.

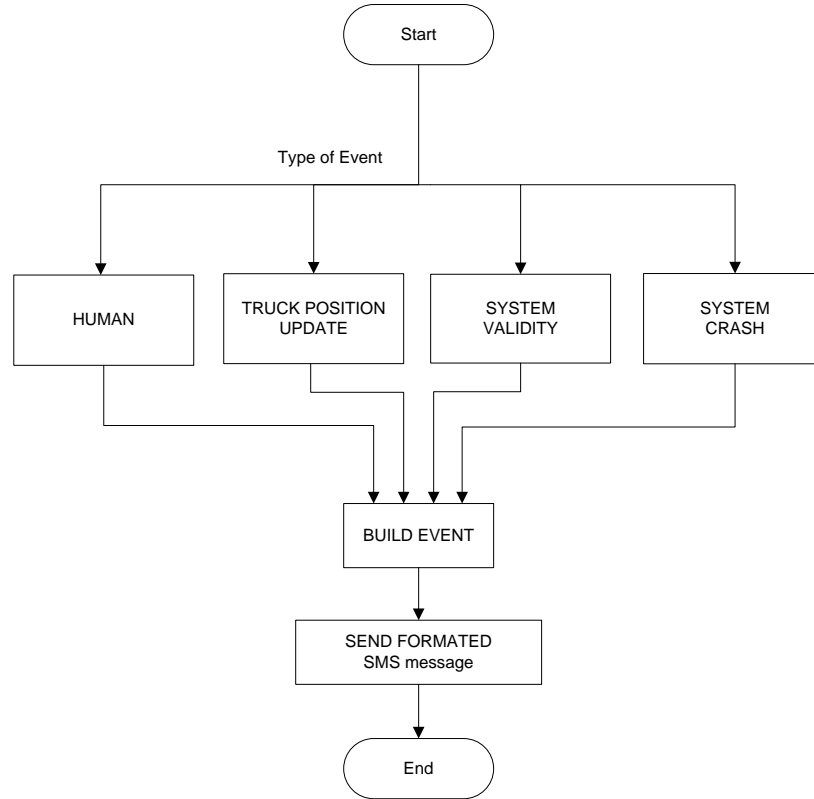
Event	Text
Fire	<p>"A FIRE has been added!"</p> <p>"A FIRE has changed position!"</p> <p>"A FIRE has been put out!"</p>
Weather	<p>"A WEATHER station has been added!"</p> <p>"A WEATHER Info Update!"</p>
Auto	<p>"A VEHICLE has been added!"</p> <p>"A VEHICLE has changed position!"</p> <p>"A VEHICLE has left the scene!"</p>
Water	<p>"A WATER source has been added!"</p> <p>"A WATER source Update!"</p>
Human	<p>"SMS received!"</p>

Table 3.4 – Event/Occurrence TT screen messages

In case the received message is of the Human type the information with the name and text of the sender is displayed on the SMS dialog EditText on the Main application and sent to the SMS application.

2) Outgoing Event

The Outgoing process is called whenever a SMS text message is to be sent to the server. The calls originate from various places on the system: normal text message, and three system warnings types (truck position refresh, invalid event and system crash) – Algorithm 3.10.



Algorithm 3.10 – Outgoing Event

Every time the user writes a text message on the SMS application the correspondent windows message process is initiated. Before being sent to the SMS thread, it needs to be formatted by the *Build Event()* function, to respect the high level developed protocol explained on Chapter 2.3.1.

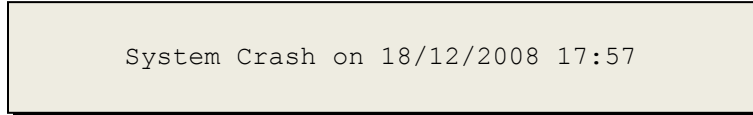
A **Truck Position Update** system message after formatting is shown on Figure 3.24:

```
22222;33;carro do joao;0;1;1;4259837;235687
```

Figure 3.24 – Truck Position Update system message

The **System Validity** invalid message has already been explained on the previous Process Event paragraph.

The last system message (**System Crash**) is one of the most important messages. This message will inform the server that it needs to resend all the events to the system, due to a system crash – Figure 3.25.



```
System Crash on 18/12/2008 17:57
```

Figure 3.25 – System Crash system message

The *BuildEvent()* function also sends the Global Log windows message to the Main application. If the message that is being sent is of the Human type it also sends the SMS Log Windows message which writes the text on the SMS EditText on the SMS Dialog box.

The last step of the algorithm consists on sending the formatted SMS text message to the server, action that is performed in the SMS class, explained on Chapter – 3.3.2.1.

3.4 SMS Application

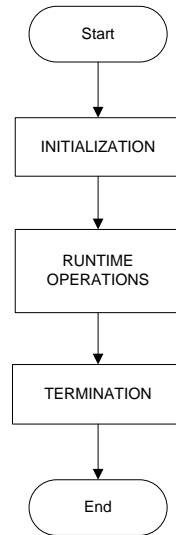
The next chapter explains and describes in detail the actions and procedures of the denominated FFFpdaSMS application.

The first sub-chapter will present the algorithm of the application itself that concerns mainly with initializations and terminations processes. The second and third sub-chapters describe the associated runtime operations: alphanumeric keypad and the text message traffic handling process.

3.4.1 SMS Class – Global Algorithm

The sequence of the algorithm starts with the **initialization**, where the auxiliary flags are reset and the keypad timers are declared. From this stage onwards it is all about **runtime operations** which are keypad handling and processing incoming and outgoing text message

The final stage of the application is when the user has decided to shutdown the PDA. This action involves performing all the proper closing and **termination** actions: kill the timers and close the application successfully – Algorithm 3.11.



Algorithm 3.11 – SMS class global

3.4.2 Alphanumeric Keypad handling

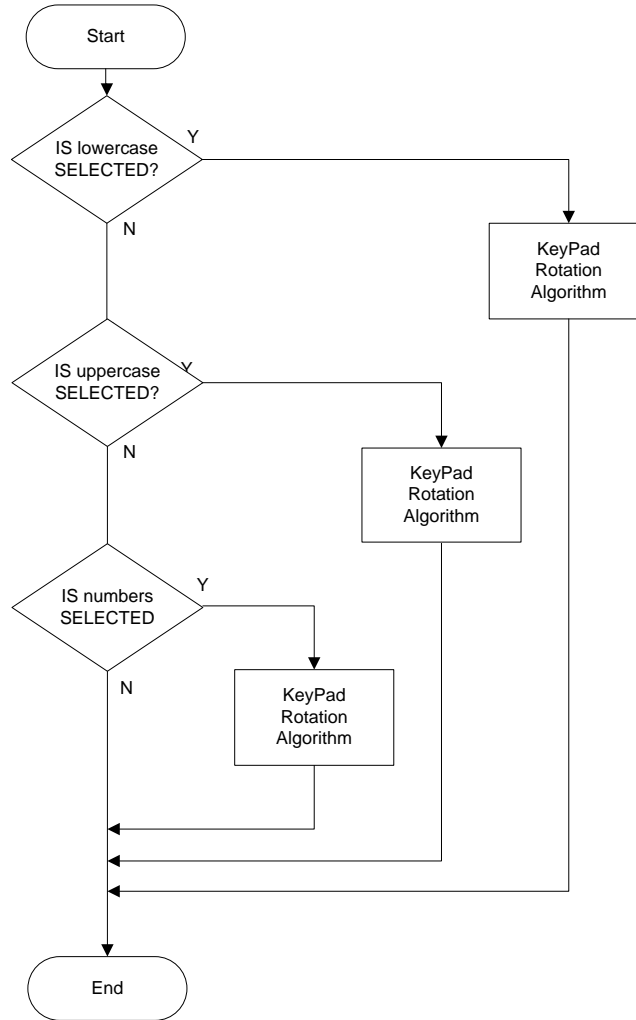
The alphanumeric virtual keypad developed for this system was developed with an important notion in mind: the creation of the most familiar, user-friendly cellular phone environment. This introduces the minimum amount of stress and thought process when the need for writing a text message arises on the worst possible scenario.

Observing the keypad itself – Figure 3.26 – the first design concern was the need of big keys/buttons. This fact in conjunction with the PDA screen size and resolution imposed a limitation on the maximum amount of buttons. The values of the keys were modeled after the most common configurations of the biggest selling mobile phone brands, once again in an attempt to ensure the maximum amount of comfort and intuitiveness to the user.



Figure 3.26 – Alphanumeric keypad

Depicted on Algorithm 3.12 is the Key value algorithm for the software Key [2 a b c] for all selection possibilities: uppercase, numbers and lowercase.



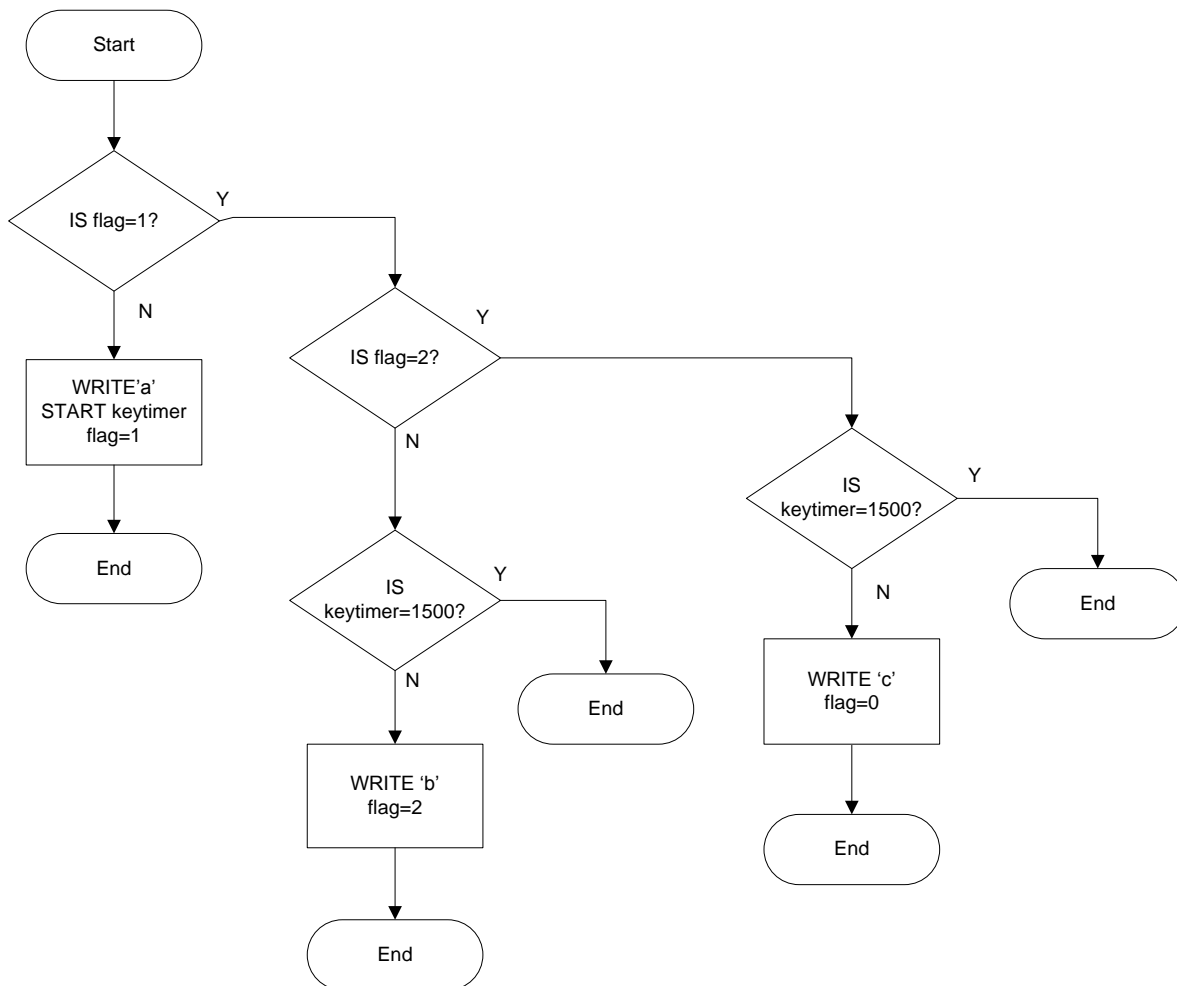
Algorithm 3.12 – Alphanumeric keypad handling

This action is identical to all the keys except the backspace/delete key [* (<<<)].

Keypad Value Rotating Algorithm

Each time a key is pressed (either the uppercase or lowercase value is selected); this state machine algorithm is initiated with the start of the key specific timer. This timer has a 1500 milisecond

cycle. If the user does not press the key until the cycle terminates, the value written on the EditText is the equivalent to first letter associated with that key. In the example of Algorithm 3.13, the letter ‘a’ is written. If the user continues to press the key, the value of letter alternates in the order that appears on the button design, ‘b’ > ‘c’ > ‘a’ until the desired character is chosen. Every time the cycle terminates the value of the auxiliary **flag** is zeroed.



Algorithm 3.13 – Keypad value rotating

3.4.3 Text message traffic handling

The main aspects of both Sending and Receiving processes have already been described on Chapter 3.3.1.2. However, it is important to refer specific details that are important on those operations.

1) Sending

The SMS To EditText dedicated to the writing of the user inputted text messages has a white background although it is as with all of the EditTextes a read-only feature. This color differentiation was implemented to increase the visibility of the written text and contrast with Last SMS Received EditText. After the user has decided to terminate the writing of the message he chooses the recipient of the message: the “Commander” or “All” the personnel present on the theater of operations.

2) Receiving

The user should keep its focus on writing the text message but also keep track of the SMS text messaging dialog. To facilitate this action the Name and text of the last received human text message is shown on the Last SMS Received EditText.

3.5 Tracking, Robustness/Ruggedness and Control Files

The necessity for reporting, tracking and auditing capabilities is a fundamental requisite of any software application. It provides the administrator or manufacturer the accountably tools on which to maintain a list of all actions performed by the system. In terms of reporting, these actions range from: logging the Fire Fighting Events sent and received by the device, logging all the GSM communication performed transparently by the system, amongst others which will be detailed on the next few chapters. The previous reporting characteristic by itself provides enough data in terms of knowing what and when an action is being done.

All these system capabilities allow the administrator a much easier task of storing the data of each action taken place during a certain Fire scenario call. It also provides useful information that the manufacturer can interpret in terms of remote maintenance and debugging. Ultimately it provides an adequate backtracking capability to ascertain responsibility in case of a system malfunction.

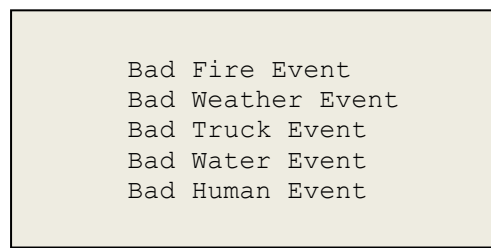
Concerning Robustness, the system was developed to ensure that only the actions desired by the user should be performed. The applications TTN, Main and SMS, appear on a Full Screen size. The only way to change that is by entering the administrator mode that will be explained on Chapter 3.6.4.

In terms of TTN application control some issues arose that made the task extremely difficult to handle due to the very poor customization capabilities allowed by the TT API SDK's. However, the menu actions on the primary TTN “main menu” were customized to the system specifications. Once

the user enters the TTN sub-menus some of the additional actions that become available can lead to misuse by an inexperienced person.

Although all SMS received messages come from the server and theoretically correctly formatted, an event failsafe was implemented upon reception. Specifically the *AnalyzeEvent()* function where the text message is parsed and the first field informs the type of event that is present. With this knowledge the system can check if the number of fields present is correct. If not, a system message is sent to the server with one of the following possibilities – Figure 3.27.

Concerning the logging and control aspects an algorithm was used for both the Global Log and the GSM Log – Algorithm – 3.14. On the first event Global log occurrence the specific time and date is written on *dateofevent.txt* file for system recovery concerns which have already been explained on Chapter 3.3.1. After this initial action, every time an entry is about to be recorded on the text file named *globallog#.txt* it has to pass through a several dimension checks. This action prevents a text file dimension overflow. Due to the limited amount of hard disk space the file was limited to ten Megabytes.



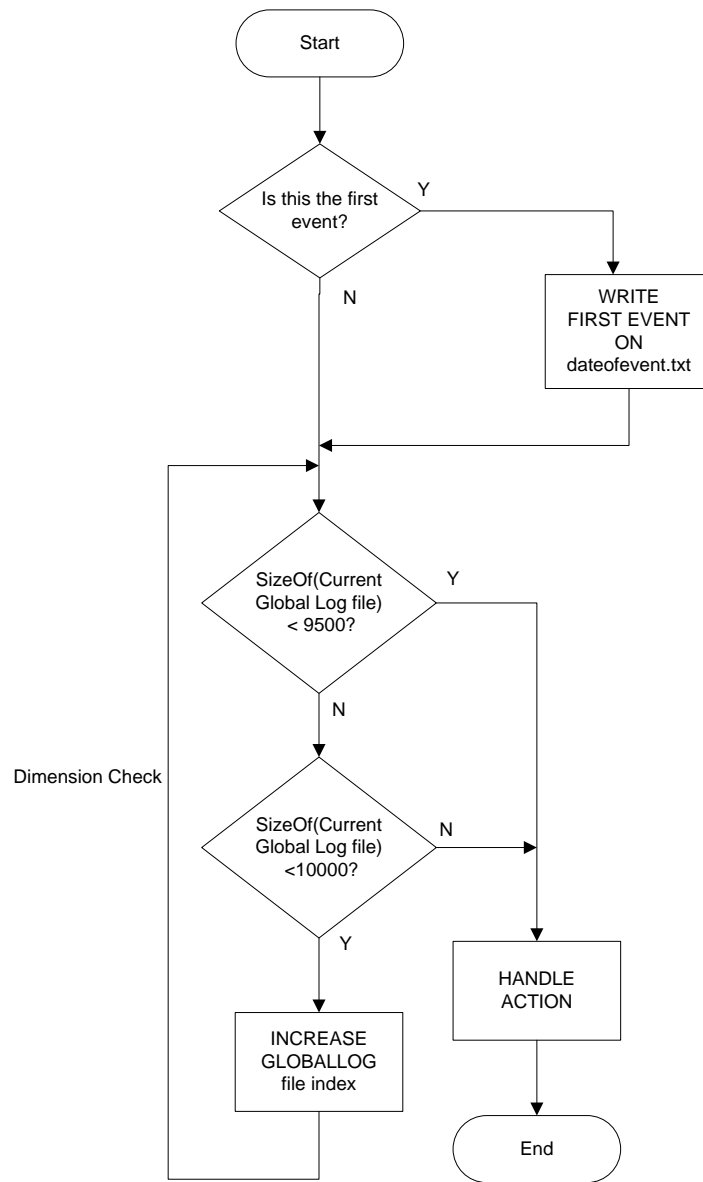
```
Bad Fire Event
Bad Weather Event
Bad Truck Event
Bad Water Event
Bad Human Event
```

Figure 3.27 – Invalid event response messages

Another design decision was to keep the file creation from growing progressively larger by limiting the file creation to ten text files, and in case the system needs to write additional information it overwrites on the first Log File, *globallog1.txt* or *gsmlog1.txt* accordingly.

The GSM Log process is analogous to the Global Log algorithm. It differs on the first event date writing text file and on the standard name of the file that keeps the logs, in this case, *gsmlog#.txt*.

To keep the index order of the text files, the current index number is kept and updated on two text files dedicated to this task: *gl.txt* and *gsmlt.txt*



Algorithm 3.14 – Logging action global

3.5.1 Global Log

This Log records entries for all the occurrences that the system produces or responds such as: Events, bad system Events, auto refresh Event. All the events received and sent (a NEW Fire on the example shown below) are constructed following the common format: an initial tag “Global Event: xxxx” is written. The next line logs the time/ date and finally the Event and Occurrence type. The next set of lines depicts the content of the Event vector entry associated the particular event to be logged, and finally the terminator – Figure 3.28.

```
Global Event: 22221
13:15:26, 09/12/2008
FIRE NEW
Location: Chamusca
Longitude: 216478
Latitude: 4136890
-----
```

Figure 3.28 – Fire event log entry

The invalid events that are received which have been detected by the system, assume the following format: a tag is added “SYSTEM >> Invalid xxxx Event -“, terminated with the current date/time – Figure 3.29.

```
SYSTEM >> Invalid HUMAN event - 13:15:26, 09/12/2008
```

Figure 3.29 – Invalid event log entry

The last log type is dedicated to the vehicle position refresh algorithm. Each time the position is updated it records the entry on the log with the simple format tag: “SYSTEM >> AUTO REFRESH “ with the date/time of the event – Figure 3.30.

```
SYSTEM >> AUTO REFRESH - 13:15:26, 09/12/2008
```

Figure 3.30 – Auto-Refresh log entry

3.5.2 GSM Log

All the logs from the SMS and SERIAL classes are passed to the main application and then processed accordingly. The same file size protection algorithm is used in this case, the changes being the name of the control file which in this case is *gsmlt.txt*, and the name of text files where the logs are

written is of the form: *gsmlog(value).txt*. Where *value* is the number of the current file being written. To illustrate this algorithm three different examples are shown next, each regarding in of the different types of logging entries: Serial, Modem and SMS – Figure 3.31.

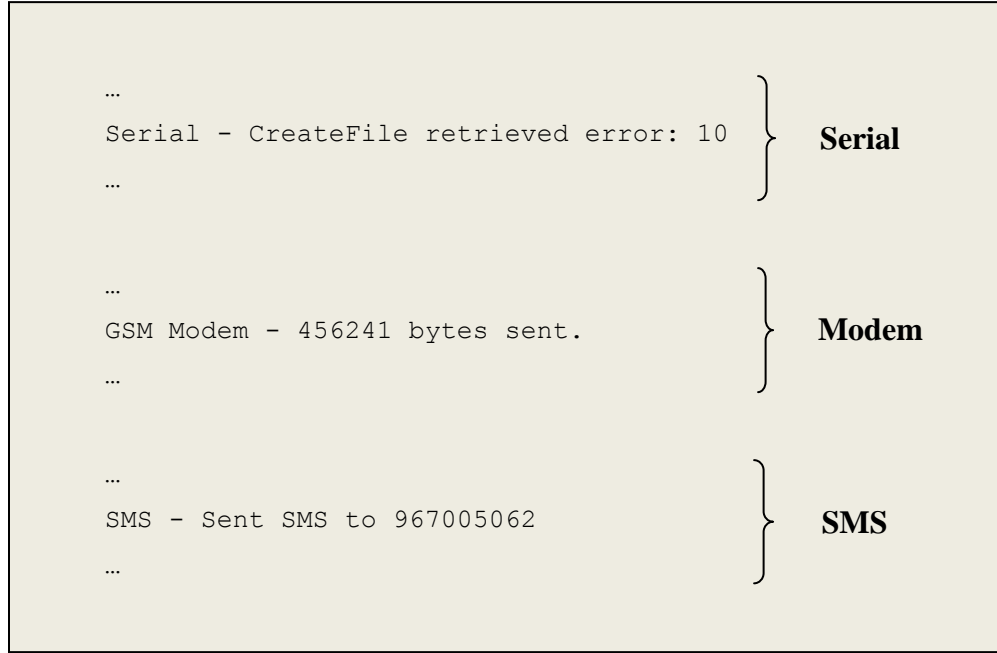


Figure 3.31 – GSM log entry types

3.6 Interfaces

The several interfaces that the user has at its disposal on this system were designed with some notions that have already been referred on this dissertation: ergonomics, intuitiveness and user-friendliness. These objectives were fulfilled to the extent of what the hardware allowed, the exception being the TTN Interface which, due to software customizations limitations could only be altered to some extent.

On the following three sub-chapters an overview of the interfaces will be pursued, detailing the presence of all the aspects referred on the first paragraph. However, some aspects are recurrent and are worth mentioning: all EditBoxes are read-only to avoid any voluntarily or involuntarily misuse by the user; all the system was designed to be the most automated and intuitive as possible, so the touch-screen interactivity with the user is as direct as possible.

Taking in consideration the fact that the Firemen will be using gloves, the buttons are as big as the screen size/resolution allowed. The exception is made for the administrator login.

3.6.1 FFFpdaMAIN

This interface – Figure 3.32 – consists of four major buttons: “**See Fires**”, “**See Weather**”, “**See Autos**” and “**See Water**”, each of them showing the full detailed information associated to each vector entry of the following events respectively: Fires, Weather, Auto and Water.

To visualize the information there are three EditBoxes: **Global Log** (where each system occurrence is logged), **Events** (showing the details of the given Event chosen) and **SMS** (where all the human text messages traded with server are displayed to create the notion of a dialog).

The user also has the capability of pressing on any given EditBox enabling the Text Scrolling actions explained on Chapter 3.3.1.1. The three small boxes next to the EditBox name have the task of giving the user the visual information of what edit box is selected. Finally an “**OFF**” button is also present to insure a proper close sequence for the system.



Figure 3.32 – FFFpdaMain interface

3.6.2 FFFpdaSMS

This interface – Figure 3.33 – consists of three main buttons: “**Commander**”, “**All**” and “**Others**”. The “**Commander**” button informs the system that it should send the written text to the

Chief of Firemen. The “**All**” button will inform the system that it should send the written text to all parties on the theater of operations. Finally, the “**Others**” button at the time of the conclusion of this dissertation had not been assigned to any number/person. The remaining buttons belong to the alphanumeric keypad that has already been described on Chapter – 3.4.2.

To visualize the SMS text messages there are two EditBoxes: **SMS To**, where the user will write the text and **Last Received SMS**, where the last received text message will be shown. In order for the user to keep track on the amount of characters that have been written, a counter has been added. The actual limit for SMS text messages is one hundred and sixty characters.

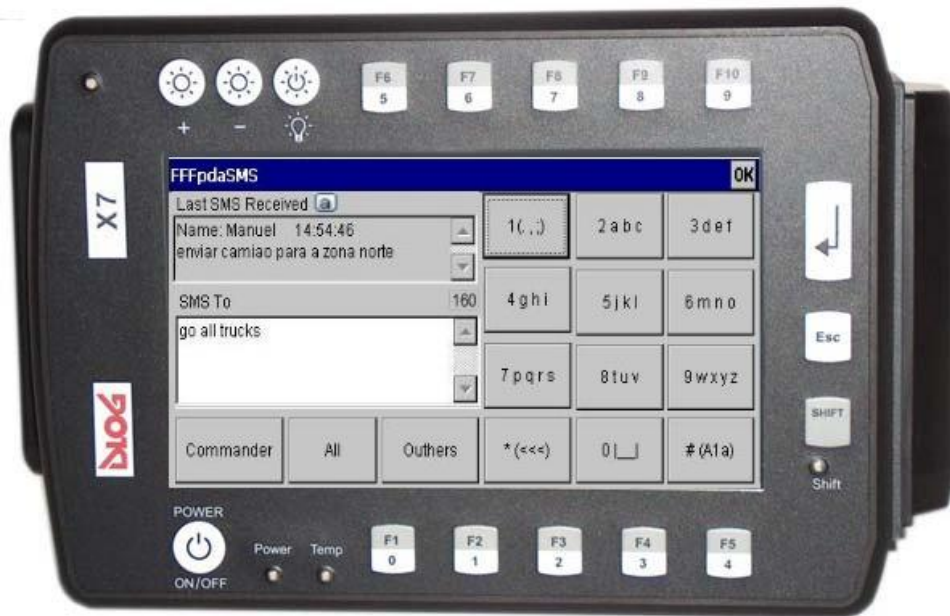


Figure 3.33 – FFFpdaSMS interface

3.6.3 TomTom Navigator Customized

When interacting with the TTN the user has three options: the default interface – Figure 3.34 – shows where the vehicle is at the moment. The main browse interface – Figure 3.35 – where the user can go directly to the Fire scenario and observe the events that have been received. In case the user decides to press the middle of the screen on the TTN interface the following TTN customized main menu appears – Figure 3.36. On this last menu the user can choose one of the actions: **Go To Event** where he will chose what event will the TTN trace a route to; **Browse Map** where he can go directly to the desired POI event and finally can enable/disable the 3D display option.



Figure 3.34 – TTN main interface



Figure 3.35 – TTN browse interface with events



Figure 3.36 – Customized TTN main menu

3.6.4 Administrator Mode

The last interface – Figure 3.37 – on this system is a very simple Dialog window that pops up after the user has pressed the small “a” button on the SMS application. If the user introduces the correct combination of User/Password names all system applications plus TTN will be closed and the windows desktop will appear. The User/password check is done by comparing the entries on the text file *admin.txt* with the data inputted on the correspondent EditBoxes.

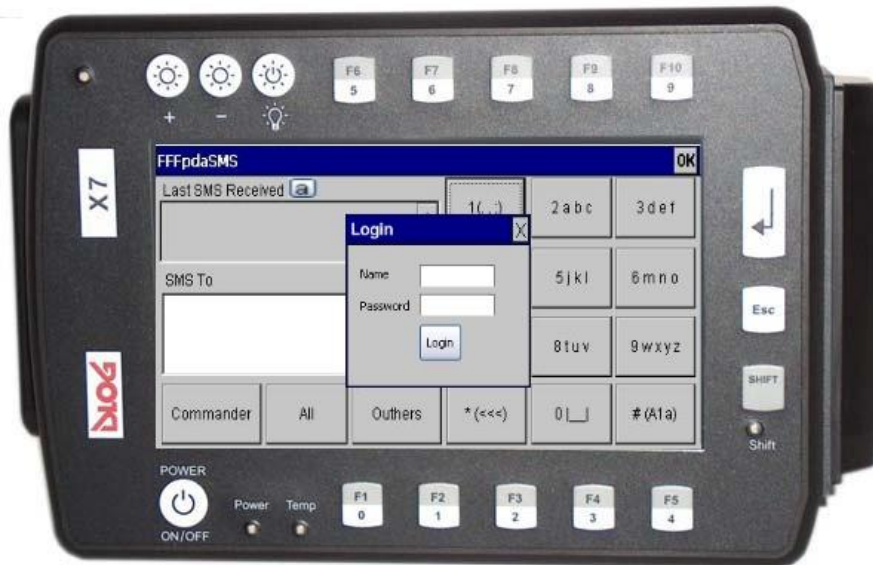


Figure 3.37 – Login interface

Chapter 4.

PERFORMANCE ANALYSIS

After presenting the system architecture proposed on this dissertation, the following chapter describes the different tests that were performed, in order to validate the intended objectives, as well as the subsequent analysis.

The chapter is divided in three sections. The first section discusses the simulation environment and the vehicle that was used when performing the aforementioned tests. The second section addresses the tests that were performed and it is divided in two sub-sections: Road-Book and System Usability. Each sub-section provides an explanation of the mentioned tests followed by an analysis of the results that were achieved. On the last section conclusions are devised from the achieved results.

Before the performance of this system is described on the following chapters it is important to describe the conditions, limitations and impediments that led to the lack of tests of the designed system on a real world Forest Fire scenario.

The first and most crucial flaw was originated by a TTN version 6 software bug that does not allow for an external application to introduce negative coordinates as parameters [In the Hand 2007a], which is a big setback in the particular case of this dissertation location (Portugal based) and even more at a global scale. This is not directly linked to the TT SDKs but it is inherent with the TTN itself. The problem was detected at the very early stages of development when using the version 6.010 of the SDKs, but with a release of a new TT SDK version eminent, that supposedly would fix this and other bugs, it was decided to proceed with the development of the remainder of the system.

When the new version of the SDK became available another drawback occurred after a required update of the TTN application. The updated version could not be deployed on the PDA even after many technical discussions through email with Franz Angermeier, Operating System Developer from DLoG™. Towards the end of this dissertation there was an opportunity of having a software training session in Germany at the DLoG™ headquarters where the problem was finally discovered: the amount of RAM memory (64 Megabytes) of NGNSs version of the DLoG X7™ was insufficient for the TTN

newest version to be able to start. This was proven when the program launched without problems on DLoG™s newer prototype version of the X7 which has 128 Megabytes of memory.

Although this might have indicated that a solution was found, two major impediments prevented it: the new DLoG X7™ had an expected release date, according to information given by Thorsten Kraus from, the DLoG™s Head of Business Development, around the second semester of 2009; the other contrariety was the fact that TT ceased to sell the version of the maps needed by the updated version of TTN a few months ago. A more detailed description to the TTNs problematic and other possible approaches to address and resolve this problem is properly depicted on Chapter 5.2 and Chapter 5.3.

Despite these drawbacks, in order to ensure that this dissertation could be validated in terms of a PDA in-vehicle mounted Road test, a short trip was taken although all the events were launched on Barcelona, thus overcoming the negative coordinate issue, which will be described in detail on Chapter 4.2.1.

4.1 Description of the Simulation Environment and Vehicle

This system, in its commercial version, will interact with the existent server for the FFF, but due to time constraints and the fact that the modification of the current protocol implicated software development in an area of expertise that was outside the theme of this dissertation, the use of a simulated server environment was chosen as the most cost efficient solution in terms of financial and easiness of use.

So the simulation environment – Figure 4.1 – was set up so it could react and process the servers' text messaging traffic. The Incoming aspect of the server was handled by a regular mobile phone, specifically a Nokia 6020 – Figure 4.2. The telephone phone number was written on the system text file (*numbers.txt*) as described on Chapter 3.3.2.1. For the Outgoing text messaging tasks it was decided that a more cost-efficient and user-friendlier approach was needed. Although a few of the initial tests were done by sending the messages from a variety of mobile telephones the chosen option was to use the Vodafone site SMS sending page – Figure 4.3 – to handle all the outgoing text message traffic.

Regarding the Road-Book test the DLoG X7™ was mounted on a NGNS vehicle, a Nissan Navarra™, – Figure 4.4 – where two operators were assigned for execution: a driver and a co-driver that operated the DLoG X7™ PDA.

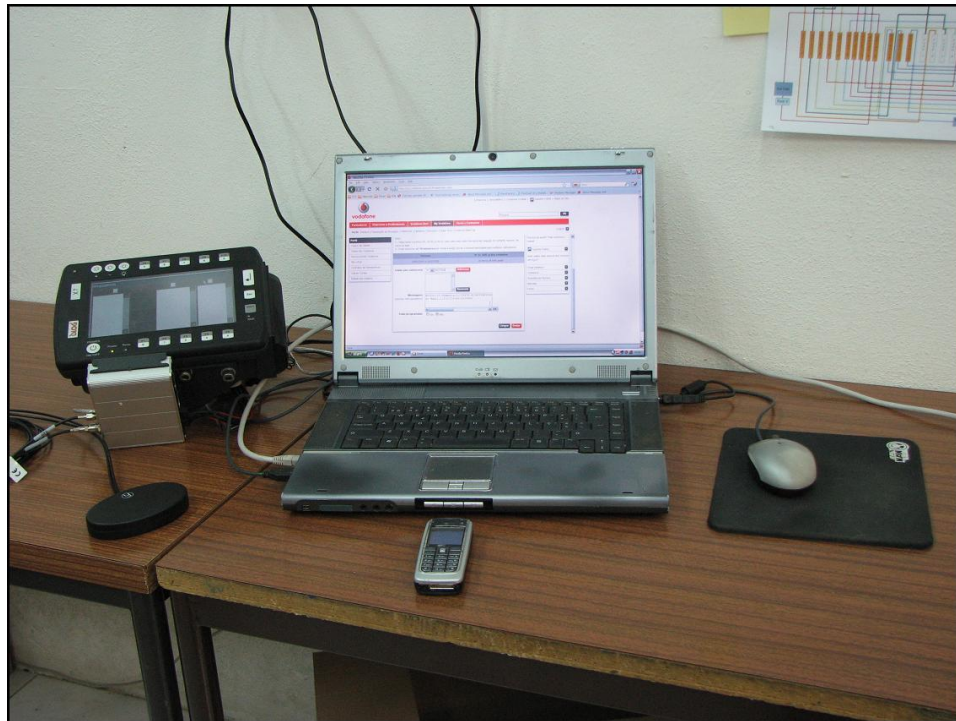


Figure 4.1– Complete simulation environment



Figure 4.2 – Nokia 6020 mobile phone

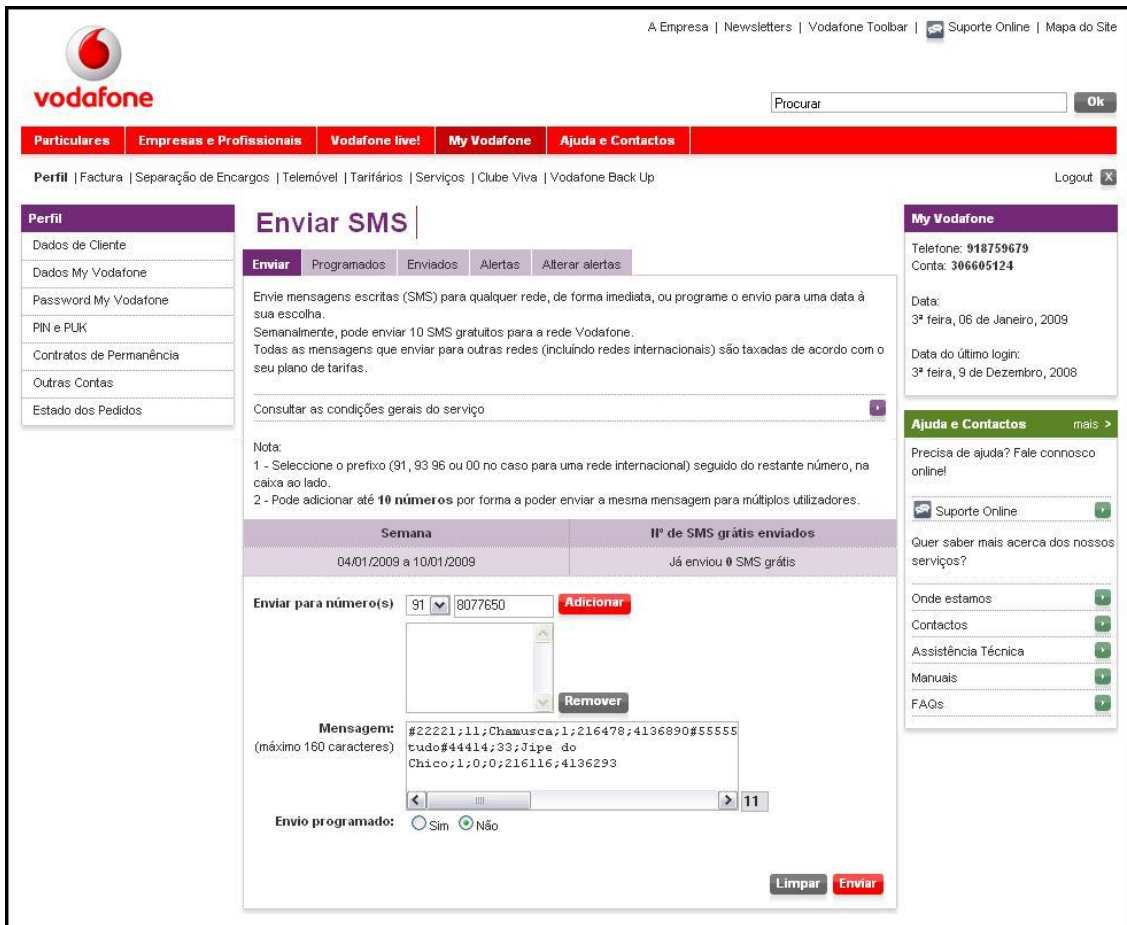


Figure 4.3 – Vodafone SMS webpage

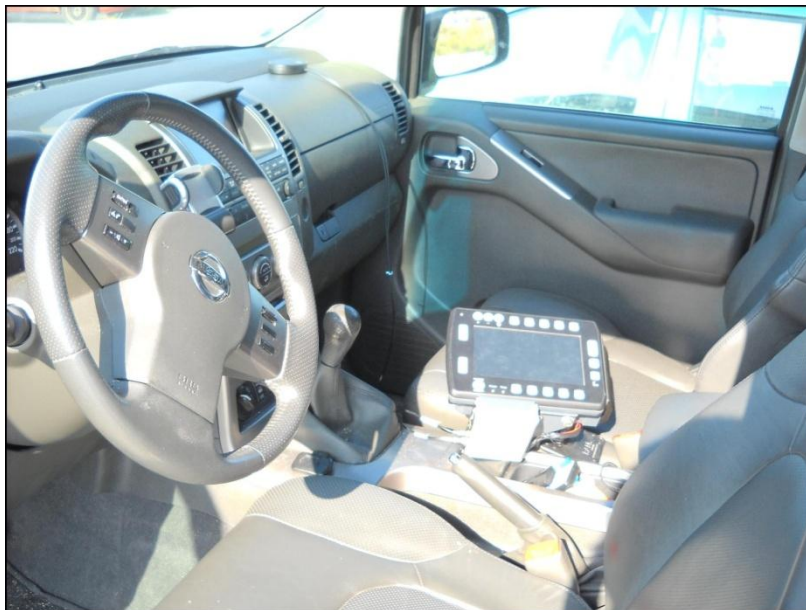


Figure 4.4 – The used vehicle mounted with the DL0G

4.2 Tests

The first issue that most system performance tests address is time. Due to an amount of special characteristics of this dissertation, hardware as well as software-wise, the time factor *per se* could not be an appropriate standard on which the quality of this system could be measured, with a few particular exceptions. The referred exceptions are the immediate response that the system provides on user input requests, both in terms of software buttons and hardware keys and the whole system reactions to the arrival or sending of a text message. Notwithstanding the importance of the previous aspects, the performance of the system ultimately relies on the level of the hardware on which the OS runs. Although being an industrial graded PDA that for previously mentioned reasons does not have the state of-the-art components, the systems global reaction time satisfied the desired objectives.

In order to validate the performance and usability of the developed system two types of tests were performed. A Road-Book was designed to test the system in a real world situation, in terms of reaction and response time, robustness and GPS signal conditions change. The second test performed concerned the usability and ergonomics of the software application from the point of view of the firefighter. In order to have some degree of feedback of the system, a small questionnaire was devised in order to gauge the performance of the system.

4.2.1 Road-Book Test

This test was performed to demonstrate the use of the developed system mounted in a vehicle, taking part of a response team to an initial Fire detection occurrence. The Road-Book was written to work primarily as a guide for the SMS virtual server human operator to send the events on the correct timing. This necessity rose due to the previously mentioned fact that the NGNS server was not fully integrated with the developed system at the time.

In order to showcase the features and robustness of the system, this Road-Book was designed to have all types of Events (Fire, Auto, Weather and Water) and all possible actions on those Events (New, Update and Removal). It also simulates a system crash and its correspondent automatic system recovery process.

The developed Road-book is illustrated on Table 4.1, and on APPENDIX C – Road-Book Test.

Test Time	Server Action	User Action	Observations
0:00		Power ON PDA	
0:01		Send Auto-Position Refresh(AUTO)	
0:02	Send FIRE Event(Fire#1)		NEW Event
0:03		Trace Route to Fire#1	
0:04	Send AUTO Event Send AUTO Event		NEW Event NEW Event
0:05		System Crash: Send SMS to Server (AUTO)	
0:06	Re-send previous events		
0:08	Send WATER Event Send HUMAN Event		NEW Event NEW Event
0:09	Send WEATHER Event Send HUMAN Event		NEW Event NEW Event
0:10		Send SMS message to Commander	
0:12	Send FIRE Event(Fire#2) Send AUTO Event		NEW Event UPDATE Event
0:13	Send FIRE Event Send WEATHER Event		REMOVE Event UPDATE Event
0:14		Trace Route to Fire#2	
0:15		Close/Power OFF PDA	

Table 4.1 – Road-Book guide

The test begins by powering ON the PDA which will result on all the interfaces being initiated, showing the TT default screen – Figure 4.5, and an automatic Auto position SMS is sent to the server, informing the server of the PDA vehicle position.



Figure 4.5 – TT Default Screen

The first Fire event is received and the user now has several possibilities respectively on the TT and FFFpdaMAIN interfaces. The TT interface allows the user to trace a route to the Fire event (“Go to Event”), or center the map with a certain zoom on the event itself (“Browse”) – Figure 4.6. The FFFpdaMAIN interface allows the user to visualize specific Fire event information using the “See Fires” button – Figure 4.7.



Figure 4.6 – TT interface options

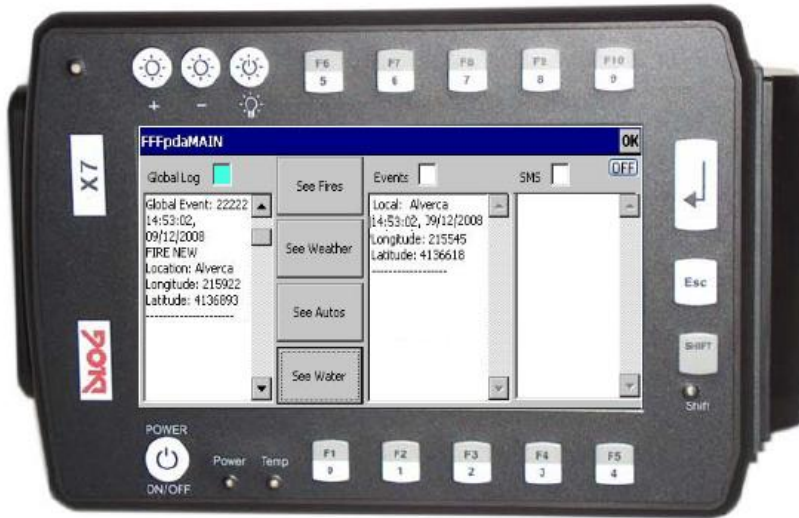


Figure 4.7 – FFFpdaMAIN interface options

Following the scripted actions the user traces a route to Fire#1 and the driver initiates the trip to the desired location. The server then sends two Water events, which are received and a visual warning is given to the operator on the top right corner of TT interface, as shown on Figure 4.8



Figure 4.8 – TT Event warning message flash

To simulate a system crash derived from a vehicle malfunction or operator misuse the recovery process is tested by unplugging the power abruptly. After the operator powers the DLoG™, the system

automatically sends a message to the server noting the system crash. All events from the initial startup moment are resent to the system. The operator resumes normal action, by tracing the route to the Fire once again. Several types of events are then sent, to illustrate all the possible events and specific information that is available. The operator has the option of visualizing only certain Events by using the appropriated hardware keys (“F6”...”F10”) as shown on Figure 4.9.



Figure 4.9 – Show/Hide events keys

In terms of the SMS capabilities the FFFpdaSMS interface allows the user to send a Text message to three possible recipients, “**Commander**”, “**All**” and “**Others**”, by means of a generic alphanumeric keypad – Figure 4.10. The operator also has the possibility to follow the text messaging dialog that is being performed through the system. There are two options available to the operator: the last text message that was received is always written in the “**Last SMS Received**” display on the FFFpdaSMS interface; the second option is available on the FFFpdaMAIN interface specifically on the “**SMS**” display, where all SMS text dialog is being constantly logged at all time on the SMS text display – Figure 4.11.

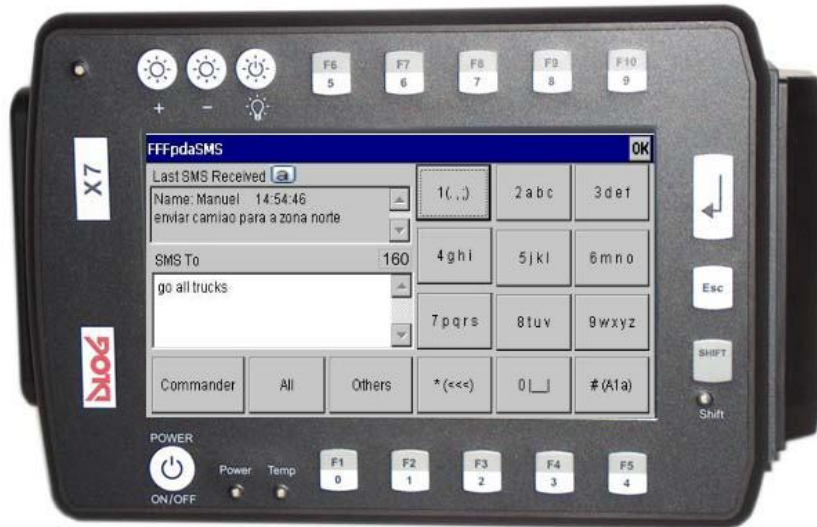


Figure 4.10 – SMS interface

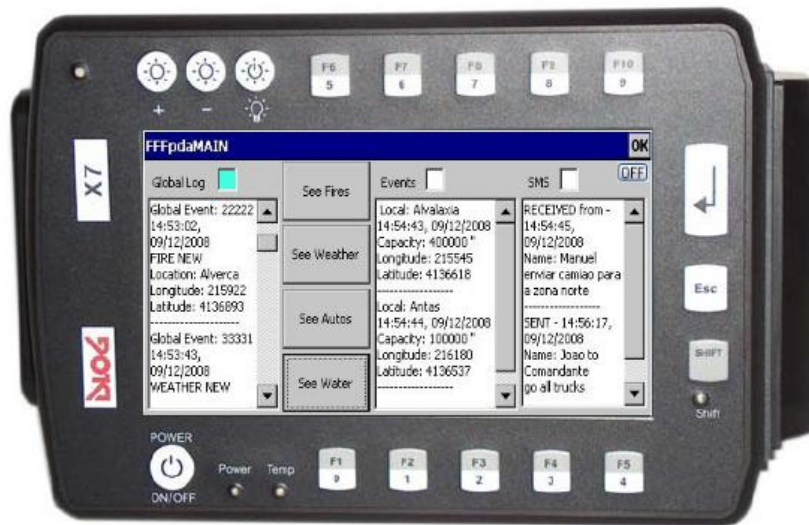


Figure 4.11 – SMS dialog

4.2.2 System Usability

This section concerns the results achieved by performing a questionnaire – APPENDIX B - User Test – to a sample of non-expert people, upon a pre-requisite reading of the systems user manual – APPENDIX D - User and Administration Manual. The evaluation system that was devised quantified the degree of difficulty to complete the task; from **Easy** (value 1) to **Hard** (value 5). The universe consisted of five testers.

The sub-chapter is divided into five sections. The first section tested the capabilities of the TT interface. The second section evaluated the performance of the custom Interfaces (FFFpdaMAIN and FFFpdaSMS). On the third section, assorted PDA actions were proposed. The fourth section concerns overall appreciation level of the complete system. Finally, the fifth section addresses some suggestions that were kindly given in order to improve the system even further.

4.2.2.1 TomTom

The following figures illustrate graphically the relative percentages for each of the requested actions. The first action “**Confirm the reception of multiple events**” was considered Easy by 80% of the testers, and 20% responded that the action was moderately Easy. The second action “**Trace a route to Water event**” showed that 80% of the universe considered the action Easy and 20% responded that it was of medium difficulty. The third action “**Go to specific Fire event location**” divided the universe in terms of distribution of the responses: 40% considered it Easy, 40% moderately easy and 20% of medium difficulty. The last action “**Show/Hide Auto event**” divided the universe between Easy, 80% and 20% moderately Easy.

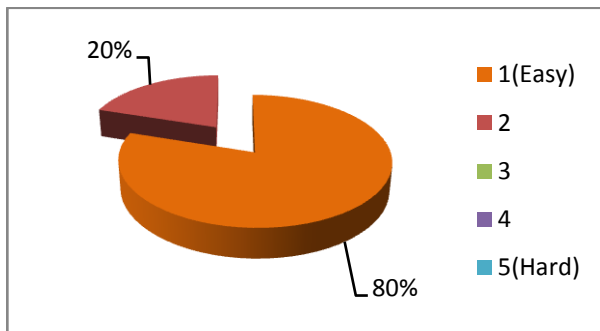


Figure 4.12 – Confirm the reception of multiple events

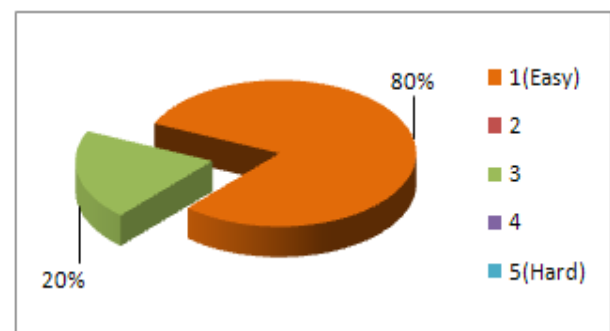


Figure 4.13 – Trace a route to a Water event

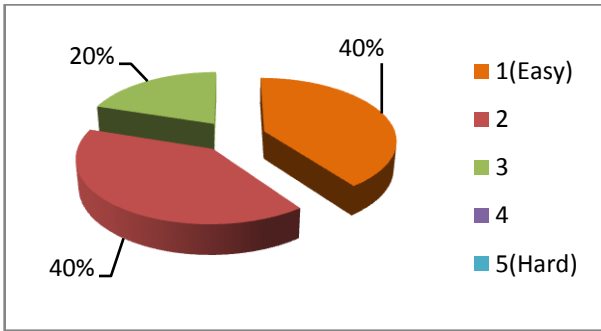


Figure 4.14 – Go to specific Fire event location

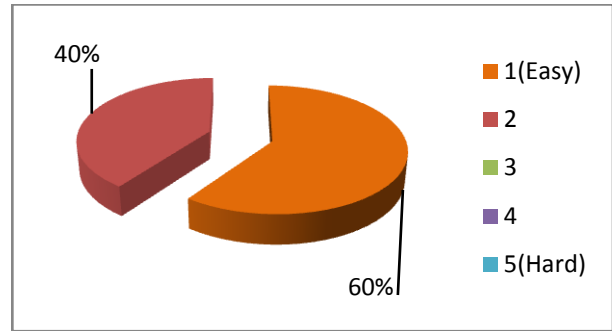


Figure 4.15 – Show/hide Auto

This section regards the analysis of the interfaces that were created for this system: FFFpdaMAIN and FFFpdaSMS. The first action “**Confirm the reception of a SMS and reading of the last two received SMS messages**” was considered Easy by all the universe of the testers. The second action “**See Weather event detailed information**” showed that 80% of the universe considered the action Easy and 20% responded that it was medium difficulty. The last action “**Send a SMS Message to “All”**” obtained the consensus of the universe on classifying the task as Easy.

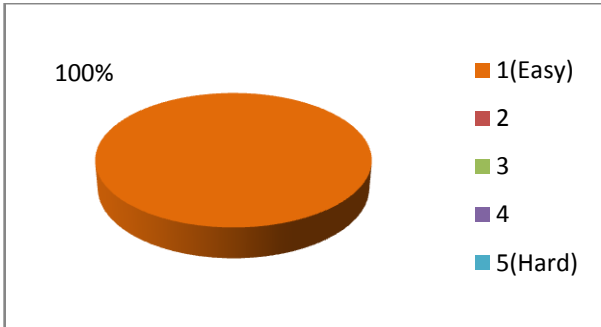


Figure 4.16 – Confirm the reception of a SMS and reading of the last two received SMS messages

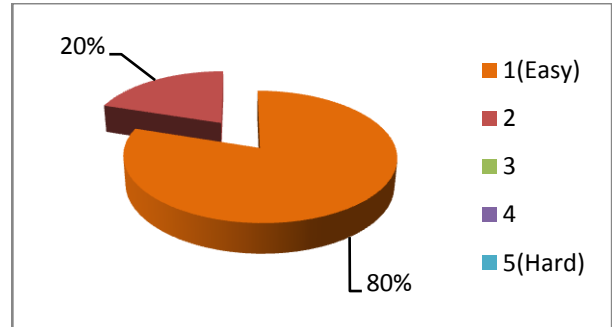


Figure 4.17 – See Weather event detailed information

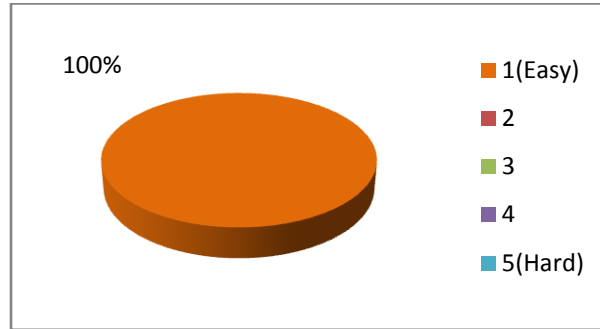


Figure 4.18 – Send a SMS Message to “All”

4.2.2.2 Assorted Actions

This section addresses assorted actions performed on the PDA. The first action “**Turn Off the DLoG**” showed that 80% of the universe considered the action Easy and 20% responded that it was moderately Easy. The other action “**Reboot the DLoG**” showed that 80% of the universe considered the action Easy and 20% responded that it was medium difficulty.

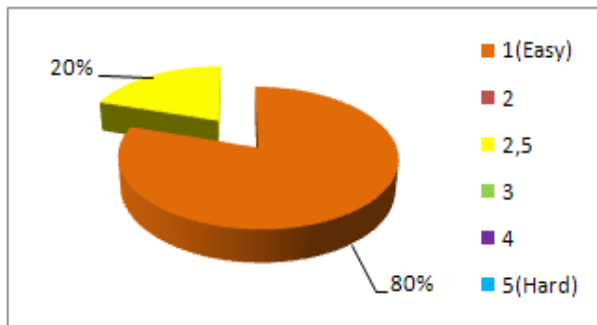


Figure 4.19 – Turn Off the DLoG

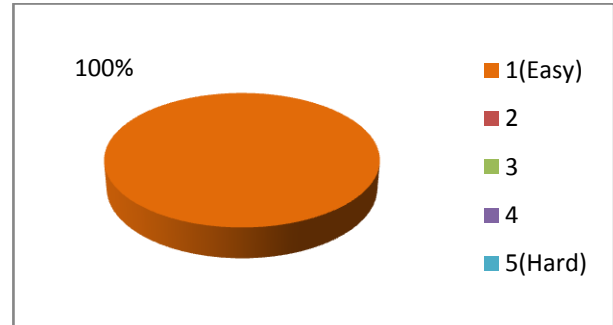


Figure 4.20 – Reboot the DLoG

4.2.2.3 Overall Appreciation

The last section questioned the testers on a final consideration of the overall system. The results where that 60% considered it Excellent, 20% Very Good and the other 20% considered it Good.

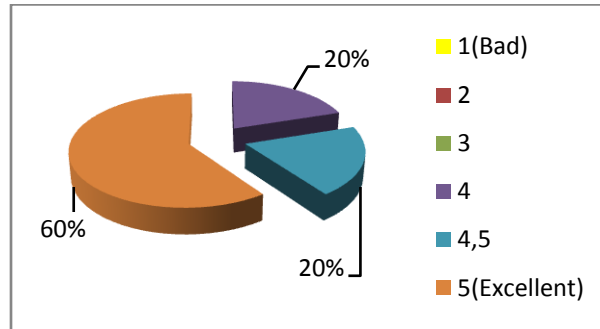


Figure 4.21 – Overall Appreciation

4.2.2.4 Suggestions

The final part of the questionnaire requested the opinion of the testers on several fields such as: Ergonomics, Ease of use, Aesthetics, User-Friendliness and Intuitiveness. As this took the analysis to a subjective level, the results could not be quantified but certain suggestions are important to mention. The TT interface was the most criticized. Primarily the lack of intuitiveness of the menus, and the morose process to perform the route tracing and browse actions. Some screen glitches especially when zooming in and out were also criticized. These facts were caused by the limited access provided by the SDKs to the libraries of the TTN, making full customization impossible. The user manual was considered well constructed and easy to comprehend, aiding positively to make the experience intuitive and user-friendly.

4.3 Conclusions

The system behaved as expected in both tests, apart from a slight delay due to the Vodafone SMS server and the GPS signal acquisition process, both processes external to the system.

On the first test, Road-Book, the system proved to be effective and promptly responded to all the solicitations of the operator. The TT rerouting capabilities also responded well to course alterations. This test served the purpose of testing the performance of the system on a real situation with all the additional factors and considerations that need to be in account. The robustness of the system was also tested and proved its reliability.

The second part of the tests, System Usability was conducted on the controlled environment of the laboratory, due to the logistic impossibility of taking all the testers on individual trips. The results were positive and demonstrate the value of the system in terms of meeting the initial proposed

objectives: an user-friendly, intuitive system that provides the necessary information in various formats, according to the needs of the end-user. From the analysis of the tests, the TT interface did not have the intuitiveness and ease of use that the remainder of the system undoubtedly has. As previously explained, the proper customization to meet those standards was not possible due to limitations of the SDKs.

Chapter 5.

CONCLUSIONS

On this last chapter, in its first section, a global overview is given on the premises and aim of this project. The last two sections resume the main contributes of this dissertation, as well as offering some guidelines on further developments.

5.1 General Summary

This dissertation focus on the development of a user-friendly application running on industrial graded PDAs adequate to harsh environments. The focus of the application is to acquire multiple real time data from the global fire scenario and provide communication through the SMS text messaging system. The system was described to its full extent on Chapter 3, employing the resources depicted on Chapter 2. The task was successful in the case of the hardware and the developed software. Finally, on Chapter 4 results were discussed and considerations were produced.

5.2 Conclusions

All the objectives proposed on Chapter 1.3 were accomplished. The development of an user-friendly system that will help the Fire Fighting Services was achieved. The end-user has a tool that provides a multitude of available and easily filtered information: related to the fire, meteorological data and also from the deployed resources for the theater of operations, such as Fire vehicles and Water reservoirs.

The exploration of the DLoG X7™ was accomplished successfully to the level that was described on Chapter 3. Several keypad assignments were customized so that some TTN reactions to external hardware input could be bypassed. It is important to refer also that, in order for the deployment of the TTN on the DLoG™ X7 PDA WinCE OS, some adaption was performed to the TTN cab files.

Regarding the creation of the text messaging high level protocol it was successfully achieved at all levels, in particular the efficiency of the codification for each event. The inherent but unstated

objective was to maximize the amount of information for each 160 character SMS message. This action decreased the overall financial cost and increased the useful information per message ratio.

The visual representation was developed successfully with all the interfaces. This ensured that ergonomic, simple and direct actions were kept to the bare essential. The effectiveness of this action was fundamental in detriment of a multiplicity of overproduced features. The Fireman's attention should be kept to the essential and easy to access information.

Concerning the system crash occurrence and respective automated system recovery the initial idea of having all the information extracted from the Global Log text files was quickly abandoned. Memory size constraints were taken in account, combined with the rewriting aspects of the used algorithm, meant that an accurate and complete event recovery was untenable. So, the decision was made, as shown on Chapter 3.3.1, to rely on NGNSs server database and basically resend the appropriate events to the PDA.

The choice of TTN as the automotive navigation system was based primarily on the quality of the TTN application itself, the detailed maps and route tracing capabilities and its extensive array of features, described on Chapter 2.2. As the knowledge of the SDKs increased throughout the development stages, certain TTN flaws prevented the degree of customization needed. Some even prevented the use of the program itself, due to the previously mentioned difficulties on Chapter 4. After a thorough research, it was concluded that TTs is not oriented to software development by third parties, as the closed API proves, despite lacking TTs official confirmation. This leads to a much reduced number of third party application projects and developers. As Franz Angermeier from DLoG™ informed us and posterior research confirmed, TT decided to terminate the release of their SDKs and also their SDKs Customer Service. This decision limits the range of posterior development of the conceived system using TT software.

However, despite the pessimistic scenario, the system developed for this dissertation was conceived on its majority by a moduled object oriented approach so the interaction with TTN is limited to the necessary. This makes adaptation to any other automotive navigation software simple and most recommended.

5.3 Future Work

The work that was developed for this dissertation covered the practical aspects on which this system was implemented. As proved on the previous sub-chapter some of those aspects limit and

prevent the system from performing at its best. This chapter suggests and describes various approaches and solutions for the future releases of the FFFpda, as well as other working environments where this type of application can be applied.

The SDKs limited access to TTN libraries diminished the intended level of software control and restricted the usage of TTN to only a small part of its capabilities. These factors in conjunction with the end of the SDKs releases stagnates the progression of the applications.

During the training days at DLoG™, some of the software developers pointed out the existence of an automotive navigation system produced by Elektrobit™ named Street Director. This software consists of a multi-platform, multi-map (Navteck™ and TeleAtlas™ compliant features) and also possess a C++ language based SDK which enables full application customization [ELEKTROBIT 2008]. The software is even more appropriated due to inter-company agreements and projects between DLoG™ and Elektrobit™, which reduces the hardware/software learning curve significantly.

The next version of this system will be developed on the new DLoG X7™ which, besides the mentioned RAM increase, will have General Packet Radio Service (GPRS). Its high data rate transference introduces a series of new possibilities such as image transmission and Remote Desktop capabilities. Psion Teklogix™ Mobile Control Center (MCC) provides customers with a complete set of integrated tools to control their mobile devices remotely and cost-effectively. The MCC allows customers to centrally monitor their devices (PDAs in this case) and their status. It also generates a series of actions such as: assets and activity reports, installation and software update, and remote control of devices. It also enables customers to troubleshoot and fix problems, configure and provision the systems network and security properties easily, and secure the devices by locking them down in case of need. Additionally, MCC enables customer to perform mass cloning, and centrally back up their PDAs. This application will allow the administrator to manage the PDA more effectively, hassle-free; reducing the support costs, while maximizing uptime and productivity [PSION 2008].

Although this dissertation addresses a vehicle mounted PDA, the concept is easily translated in many mobile devices. A hand-held industrial PDA, as Psion Teklogix™ Ikôn™ could be given to the chief of the Firemen with added Cellular Phone features as well as an on-board camera. With this the Chief of Firemen could give, for example a terrain update of the fire in photographic terms to all the parties involved on the theater of operations.

REFERENCES

[NGNS-IS 2009] "Introduction", available on <http://www.ngns-is.com/>, Date of access: 12 August 2009

[Fischer 2000] Fischer, P., Fusco, L., Brugnoli, G., "Integration of Portable Information Technology and GSM Based Communication Devices for Disaster Management Operations", available on <http://adsabs.harvard.edu/full/2000ESASP.458..177F>, Date of Access: 30 June 2009

[FMS 2009] "Fleet Management Solutions, Product Overview", available on <http://www.fmsgps.com/frontend/overview.aspx>, Date of Access: 20 June 2009

[Lymberopoulos 1996] Lymberopoulos, N., Papadopoulos, C., Stefanakis, E., Pantalos, N., Lockwood, F.A., "GIS-Based Forest Fire Management Information System", available on <http://www.dblab.ntua.gr/pubs/uploads/TR-1996-2.pdf>, Date of Access: 25 June 2009

[FMSy 2009] "Fleet Management System", Wenco available on <http://www.wencomine.com/Real-Time.html>, Date of Access: 22 June 2009

[Ye 2008] Ye, X., Wang, Y., Li, H., Dai, Z., "Emergency Decision Support System Based on the General Decision Process", available on <http://www2.computer.org/portal/web/csdl/doi/10.1109/WIAT.2008.173>, Date of Access: 23 June 2009

[Dedicated Systems Experts 2004] "RTOS Evaluation Project: Windows CE 5.0", available on <http://www.dedicated-systems.com>, Date of access: 15 February 2009

[DLoG 2006] "*Manual DLoG X 7, DLoG X 10, DLoG X 12*" available on <http://www.dlog.com>, Date of access: 30 October 2007

- [SE 2008] "M2501B automotive combines GSM and GPS capabilities" available on <http://newsletter.spezial.com/artikel.php?AID=52&TID=498>, Date of access: 16 February 2009
- [TT 2007] "*Supported API calls*" available with through the documentation of TomTom Software Development Kit, version 6.0 (for Pocket PC), 11 of October 2007
- [TT 2008] "NAVIGATOR 6 - Software & Maps of Western Europe on DVD - Functionalities" available on <http://www.tomtom.com/products/features.php?ID=260&Category=2&Lid=1>, Date of access: 15 November 2008
- [MSDN 2008a] "Messages and Message Queues", available on [http://msdn.microsoft.com/en-us/library/ms644927\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms644927(VS.85).aspx), Date of access: 30 September 2008
- [MSDN 2008b] "Register HotKeyFunction", available on [http://msdn.microsoft.com/en-us/library/ms646309\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms646309(VS.85).aspx), Date of access: 2 October 2008
- [ZDNET 2008a] "Definition for: Hotkey", available on <http://dictionary.zdnet.com/definition/Hotkey.html>, Date of access: 2 October 2008
- [In the Hand 2007a] "TomTom Negative Coordinate Bug", available on <http://32feet.net/blogs/announcements/archive/2007/03/26/tomtom-in-the-hand-6-1-released.aspx>, Date of access: 25 July 2008
- [In the Hand 2007 b] "TomTom SDK release", available on <http://inthehand.com/forums/p/1391/4206.aspx#4206>, Date of access: 25 July 2008
- [ELEKTROBIT 2008] "EB Street Director", available on <http://www.elektrobit.com/index.php?925>, Date of access: 2 December 2008

Appendix - A.

SMS Protocol

SMS messages FORMAT:

UNIQUE GLOBAL ID (6 characters) from SERVER

Code	Message type
11	Fire
22	Weather
33	Auto
44	Water
55	Human
66	Misc

Table A.1 – Code/Event message types

Each message event type has the most amount of specific information; and also in each SMS a series of complete events can be sent until the 160 characters limit is reached. Both Server and User only send completed events in each SMS message, separated with the character ‘#’

#(EVENT)#(EVENT)#(...)

Event Types

#FIRE (Fire location and status)

Communication: Server generated, user inputted

TYPE(11)

NAME (type of event:“Fire” or location: “Chamusca”)

ENABLE(1 - ”ON”, 0 – “OFF”)

COORDINATES (two sets of 6/7 digit numbers):

LONGITUDE

LATITUDE

Example:

3333333;11;Fire;1;123454;9234545

#WEATHER (Meteorological information on the fire location)**Communication: Server generated****TYPE(22)****NAME(station name:“Station 1”)****INFO 6 type of data (type of weather info:“temperature”, “humidity”, “pressure”, “wind speed”, “wind orientation”, “pluviosity”)****COORDINATES (two sets of 6/7 digit numbers):****LONGITUDE****LATITUDE**

Example: 555555;22;Station 1;25;30;34;300;N;200;123456;9234547

#AUTO (vehicles location and detailed information)**Communication: Server generated, User inputted (automatic periodic refresh)****TYPE(33)****NAME (of the vehicle) number****ENABLE(1 - ”ON”, 0 – “OFF”)****AUTOTYPE: Jeep(0), Tank(1), etc(3), ...****STATE: water capacity...****COORDINATES (two sets of 6/7 digit numbers):****LONGITUDE****LATITUDE**

Example: 444444;33;Car of Manuel;1;2;0;123454;9234545

#WATER (information on static water supplies)**Communication: Server generated****TYPE(44)****NAME (of the water supply)****STATE (capacity/level of water)****COORDINATES (two sets of 6/7 digit numbers):****LONGITUDE****LATITUDE**

Example: 666666;44;Dam 2;450000;123454;9234545

#HUMAN (free text)

Communication: Server generated , User sent

TYPE(55)

NAME (of the user)

RECEIVERTYPE: Commander(0), All(1), in case of reception on the PDA(3)

TEXT(free text)

Example: 654456;55;John;0;send truck to north region

Appendix - B.

User Test

1. Set Up

1.1 Test Environment

The environment for this test consists on the *DLoG X7™* PDA with the GSM/GPS Module attached and the respective antenna – Figure B.1.



Figure B.1 – DLoG X7

The SMS server is simulated by a laptop and a cellular phone to handle the incoming/outgoing the SMS message traffic – Figure B.2.

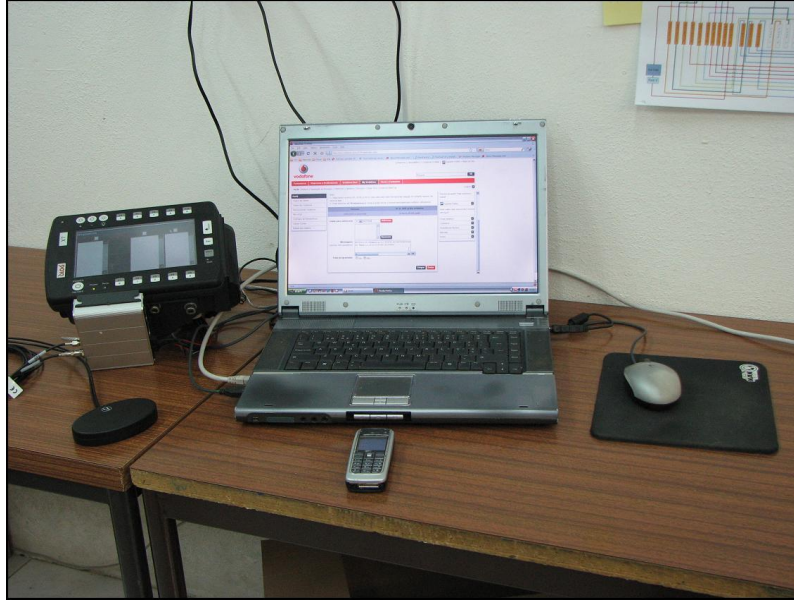


Figure B.2 – Simulated SMS server

1.2. SMS communications

The events received on the PDA by the tester during the simulation will be deployed remotely from the laptop by an operator; and the reception of the SMS messages sent by the tester will be done on the pictured cellular phone, and that fact will be acknowledged by the operator.

2. Guide

Evaluation 1..to..5 (1=easy and 5= hard)

2.1 TomTom

2.1.1 Confirm the reception of multiple events on the TomTom™ interface

(Suggestion: see sub-chapter 2.1 TomTom interface, Page 7; and chapter 3 section 3.4)

Grade:

2.1.2 Trace a route to Water 2 event

(Suggestion: see sub-chapter 2.1 TomTom interface, Page 7)

Grade:

2.1.3 Go to specific Fire 1 event location

(Suggestion: see sub-chapter 2.1 TomTom interface, Page 8)

Grade:

2.1.4 Show/hide Auto events

(Suggestion: see sub-chapter 2.1 TomTom interface, Page 7/8 and also Page 6/7 specifically "Event Zoom In/Out")

Grade:

2.2 Interfaces

2.2.1 Confirm the reception of a SMS and read the last two received SMS.

(Suggestion: see sub-chapter 2.1 FFFpdaMAIN interface, Page 13; and chapter 3 section 3.4))

Grade:

2.2.2 See Weather 2 event detailed information

(Suggestion: see sub-chapter 2.1 FFFpdaMAIN interface, Page 11)

Grade:

2.2.3 Send a SMS message to "All" with the following text: "Fireman down"

(Suggestion: see chapter 3 "How To", section 3.3)

Grade:

2.3 Assorted Actions

2.3.1 Turn Off the DLoG X7 PDA

(Suggestion: see chapter 3 “How To”, section 3.2)

Grade:

2.3.2 Reboot the DLoG X7 PDA

(Suggestion: see chapter 3 “How To”, section 3.1)

Grade:

2.4 Overall Appreciation

Evaluation 1..to..5 (1=bad and 5= Excellent)

Grade:

3. Suggestions/Comments

Please feel free to add any suggestions or improvements on the following fields:

Ergonomy

Ease of use

Aesthetics

User-friendliness

Intuitiveness

Appendix - C.

Road-Book Test

This Road-Book was elaborated to provide accountability of the features of the developed FFF PDA system. It simulates all the actions during regular operation: the expected and possible (routes, events), as well as some of the unexpected (power outage, incorrect shutdown, system crash).

Test Time	Server Action	User Action	Observations	Check
0:00		Power ON PDA		
0:01		Send Auto-Position Refresh(AUTO)		
0:02	Send FIRE Event(Fire#1)		NEW Event	
0:03		Trace Route to Fire#1		
0:04	Send AUTO Event Send AUTO Event		NEW Event NEW Event	
0:05		System Crash: Send SMS to server(AUTO)		
0:06	Re-send previous events			
0:08	Send WATER Event Send HUMAN Event		NEW Event NEW Event	
0:09	Send WEATHER Event Send HUMAN Event		NEW Event NEW Event	
0:10		Send SMS message to Commander		
0:12	Send FIRE Event(Fire#2) Send AUTO Event		NEW Event UPDATE Event	
0:13	Send FIRE Event Send WEATHER Event		REMOVE Event UPDATE Event	
0:14		Trace Route to Fire#2		
0:15		Close/Power OFF PDA		

Table C.1 – Road-Book guide

Appendix - D.

User and Administration Manual

This manual contains important information regarding the system operation. Please read it thoroughly before starting to use it. It is an essential step to use all its capabilities and avoid incorrect use that may lead to malfunctions. Please note that this manual follows its own page numbering.

User and Administration Manual

Forest Fire Finder



PDA

4. Getting Started

4.1 Package Contents

The package contains *DLoG X7™*, the GSM/GPS modem and respective antenna, Automobile mounting bracket and a USB 1.0 interface cable.

4.2 Technical Requirements

For software upgrades and specific administrator modifications the PC which connects to the *DLoG X7™* PDA must be equipped (minimum requirements) with:

- Windows XP™
- Microsoft ActiveSync 4.5™
- 1GB of RAM
- One USB 1.0 Port

2. Overview

The *Forest Fire Finder* PDA (FFF PDA) application is divided in three separate interfaces: **TomTom™**, **FFFpdaMAIN** and **FFFpdaSMS**:

- **TomTom™** Interface – Events visualization, filtered Events visualization, Route calculation
- **FFFpdaMAIN** Interface – Visualization of the acquired information (iEvent dedicated); Visualization of the exchanged SMS messages and Global log entries. Mandatory PDA Shutdown.
- **FFFpdaSMS** Interface – Alphanumeric keyboard for text message composition, two/three recipients pre-dialed. Visualization of the last received SMS human text

Figure 1 – Interface description

In all the interfaces there are two types of interaction possible: software buttons (screen touch enabled) or physical hardware buttons as showed on the **Figure 2**.

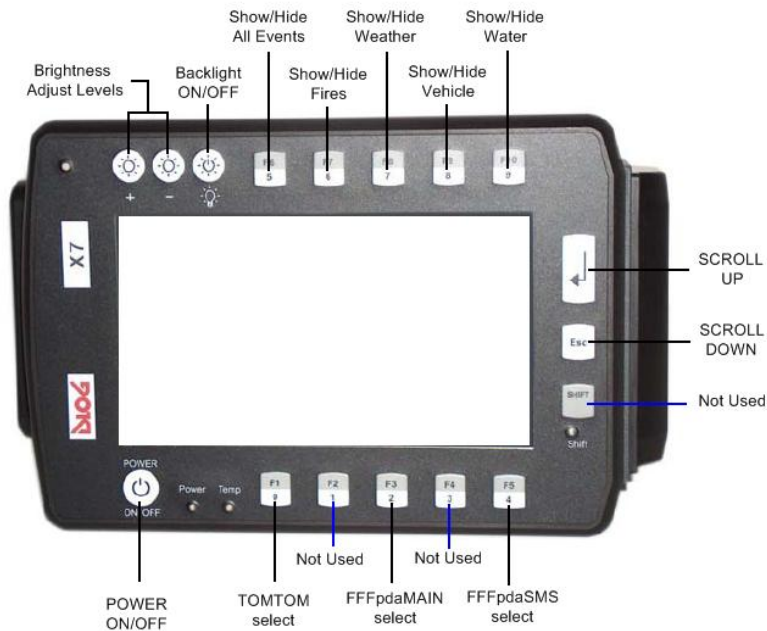


Figure 2 – DLoG X7 Key Layout

Common to all the interfaces present on the system are the three inferior buttons available which enable the desired interface (**TomTom™**, **FFFpdaMAIN** and **FFFpdaSMS**) to come to the foreground.

At any given time the user may switch interfaces (**FFFpdaMain**, **FFFpdaSMS** or **TomTom™**) simply by pressing the appropriate hardware button.

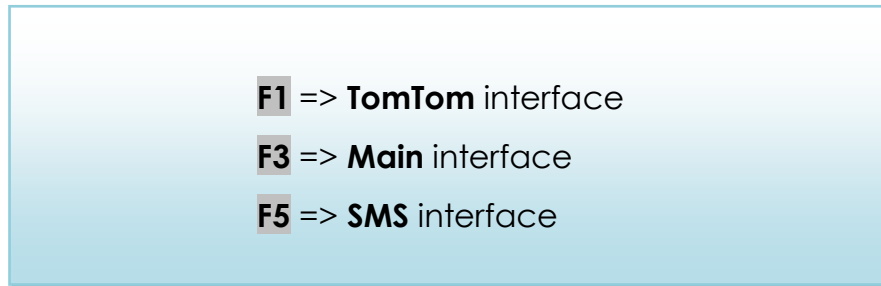


Figure 3 – Interface Selection Keys

Note: The **F2** and **F4** hardware buttons are disabled on all interfaces.

2.1 TomTom™ Interface



Figure 4 – TomTom Default Screen

This is the default **TomTom™** interface where the blue arrow represents our vehicle at any given time. If a route has been traced, a continuous green highlighted segment will appear indicating the selected path.

If a route has been selected, multiple types of information will appear on the inferior portion of the screen as shown on **Figure 3**. This information includes the remaining distance and estimated time to the desired destination and also the GPS signal strength on a series of vertical bars.

Software Buttons

The interface has two types of buttons:

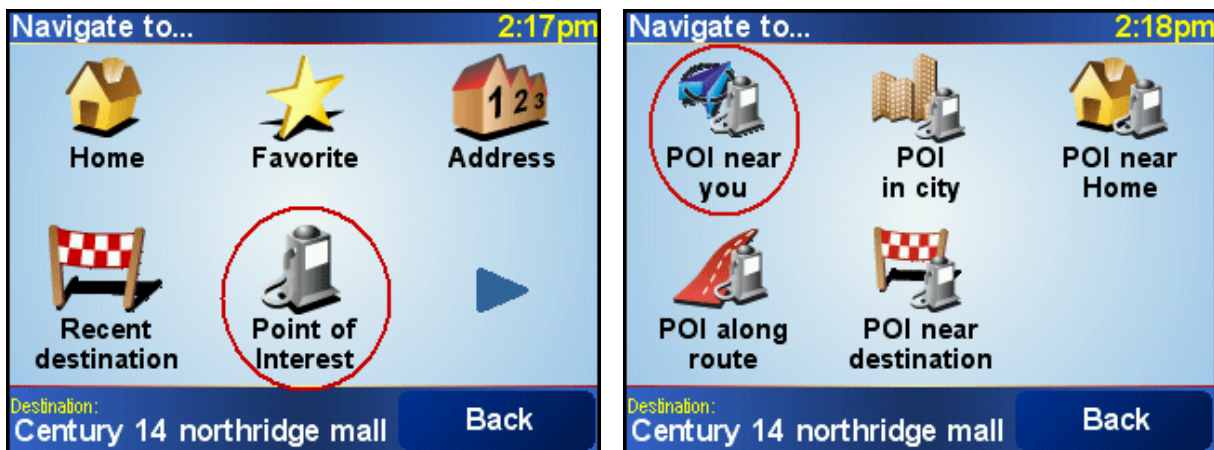
Access to the main menu by pressing the TomTom™ default screen and Map Zoom control in the specific screen area

Menu Access



Figure 5 – TomTom Default Menu

By pressing the TomTom™ default screen the following options became available **“Go to Event”** which will trace a route to the event. This opens another menu where the option **“Point of Interest”** must be selected followed by **“POI near you”** on the next menu:



The user will then choose a type "Point of Interest" (Fire, Water, Weather and Truck).



On the following sub-menu the specific event (**Event#1, Event#2, ...**) is selected.

"Browse" centers the map/user on the specified event location. The TomTom™ Menu actions required to browse for a specific event are exactly the same as the previously explained for the **"Go to Event"** option.

"Turn on 3D Display" alternates the default TomTom™ interface between the 2D and 3D type of display.

Event Zoom In/Out

If the **"Browse"** option has been chosen, the selected event area is shown, allowing the user to view in more detail the surrounding terrain or on the other hand, have a bigger perspective of the geographical area surrounding it. The user has at its disposal a vertical "zoom" bar on the right side of the screen, controlled by a dragging motion up and down, while maintaining the zoom bar indicator.



Figure 6 – TomTom Map Zooming Bar

Hardware Buttons

In terms of hardware buttons this interface provides a visible/hide action on the several types of events using the upper buttons of the *DLoG X7™*.

As previously referred, all the interfaces (**TomTom™**, **FFFpdaMAIN** and **FFpdaSMS**) present on the system are brought to the foreground by the three inferior buttons of the *DLoG X7™* PDA.

Event ICON (Point Of Interest) Enable/Disable action

The user has at its disposal on the vertical hardware buttons numbered respectively F5 to F9 the option to make visible or hide the events.

- F6** => Show/Hide **All** Events
- F7** => Show/Hide **Fire** Events
- F8** => Show/Hide **Weather** Events
- F9** => Show/Hide **Vehicle** Events
- F10** => Show/Hide **Water** Events



Figure 7 – Show/Hide Events Keys

Note: These hardware buttons will only function on the **TomTom™** interface.

Reception of events

When an event arrives, the system will flash a message on the top right corner of the screen with one of the following texts:

Event	Text
Fire	<p>"A FIRE has been added!"</p> <p>"A FIRE has changed position!"</p> <p>"A FIRE has been put out!"</p>
Weather	<p>"A WEATHER station has been added!"</p> <p>"A WEATHER Info Update!"</p>
Auto	<p>"A VEHICLE has been added!"</p> <p>"A VEHICLE has changed position!"</p> <p>"A VEHICLE has left the scene!"</p>
Water	<p>"A WATER source has been added!"</p> <p>"A WATER source Update!"</p>
Human	<p>"SMS received!"</p>

Table 1 – Event/Occurrence TomTom screen messages

2.2. FFFpdaMAIN Interface



Figure 8 – FFFpdaMain Interface

This interface allows the user to access and filter all the Events logged in the system. For that there are three information areas available: The Global Log Entries, Events and SMS. The information is accessed by a set of software/hardware buttons.

The logged entries on the various displays are shown by chronological order, oldest-first.

Available Displays

“Global Log Entries”, allows the visualization of all the information that the application handles using the several log entries being performed in real time, such as: received and sent events, visualizations requested, bad events received and finally the sent and received text messages.

“Events”, when the user presses an event Type button, all the present events of the chosen type and its detailed information are shown on this display.

“SMS”, this display keeps updating the SMS text message dialog in real time in order to provide the user with an easier remembrance of all the conversations kept till the present.

Software Buttons

In terms of software buttons this interface has four different buttons that will allow the detailed visualization of Events Type information and the **OFF** (Shutdown) buttons.

Selection of Events Detailed Information

When the user presses any of the four different buttons (**“See Fires”**, **“See Weather”**, **“See Autos”** and **“See Water”**), all the detailed information on the several entries of that specific Event Type will be shown on the **“Events”** display being shown in chronological order, oldest event first, divided by this separator **“-----”**.



Figure 9 – FFFpdaMain Displays

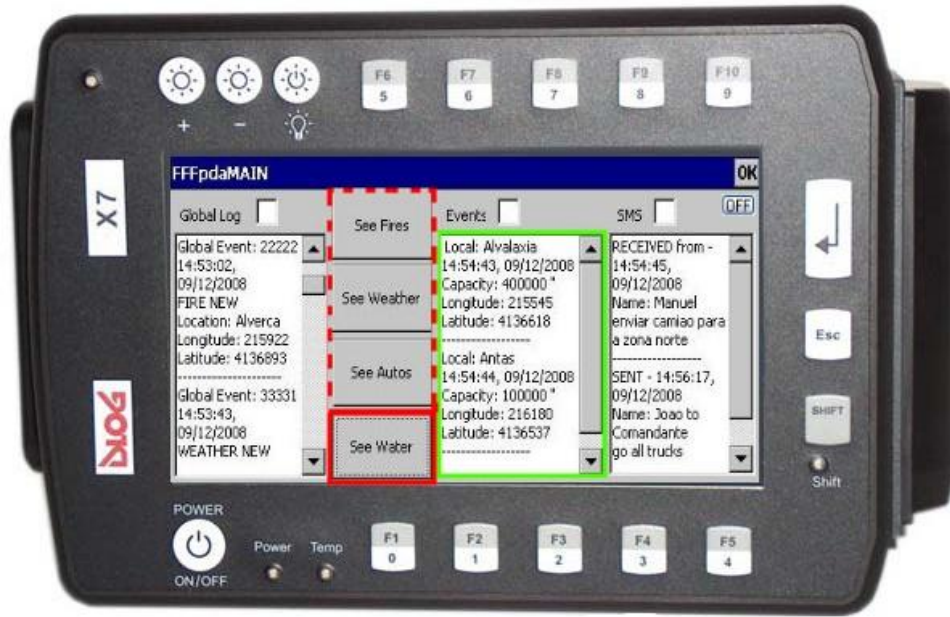


Figure 10 – FFFpdaMain Event Selection Buttons

OFF Button

To terminate the FFF PDA correctly before turning off the vehicle engine, the only action required is to press the **“OFF”** button.

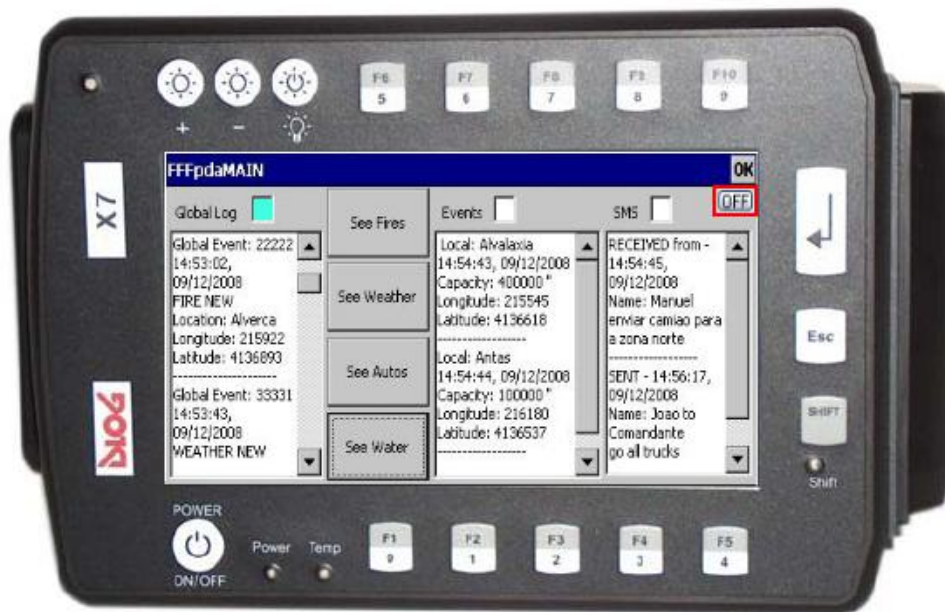


Figure 11 – System Off Button

Hardware Buttons

In terms of hardware buttons this interface provides a scrolling action on the text present on the various displays using the side buttons of the *DLoG X7™*.

Scroll Text Up/Down

To scroll the text on the displays the user must first select the desired display by pressing anywhere on its corresponding area. This action will make appropriate square turn into fluorescent blue indicating that the display is associated to the scrolling hardware keys



Figure 12 – FFFpdaMain Text Scroll Up/Down Keys

Then the user may scroll the displayed text **UP** by using the **“ENTER”** key, and scroll **DOWN** respectively by using the **“Esc”** key.

2.3 FFFpdaSMS Interface

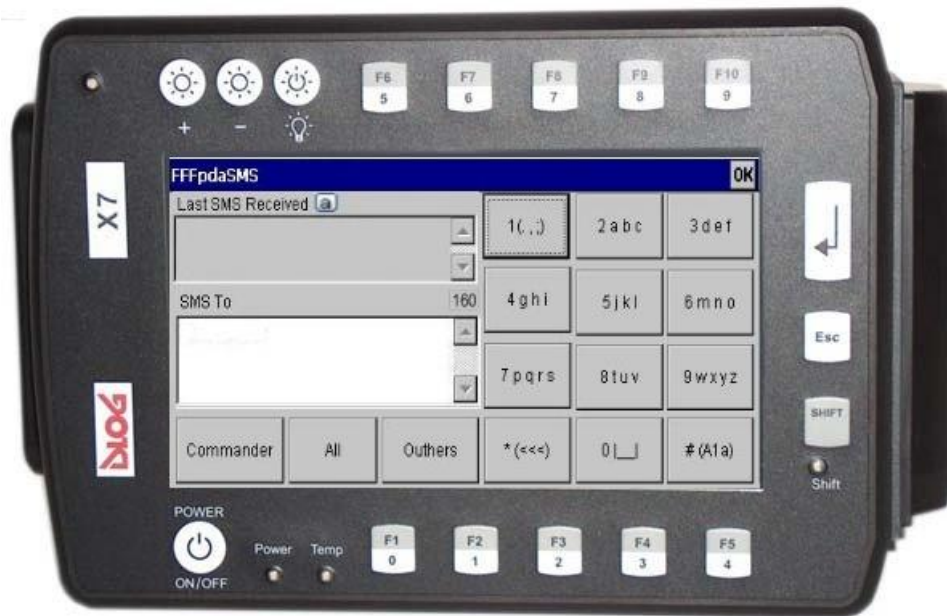


Figure 13 – FFFpdaSMS Interface

This interface, as the name suggests, has a similar action to a cellular phone text message application with some adequate adjustments for the specificity of this system. The interface has a pair of displays, an alphanumeric keypad to input the text and destination receiver choice buttons.

Available Displays

“Last SMS Received”, in order for the user to have a more immediate recollection of the last SMS text message received, this information is presented on this display with the following format: the name of the user who sent it, the time and the content of the text.

“SMS To”, is the display where the user writes his text. In order for the user to keep track on the amount of characters that have been written, a counter has been added. The actual limit for SMS messages is 160 characters.

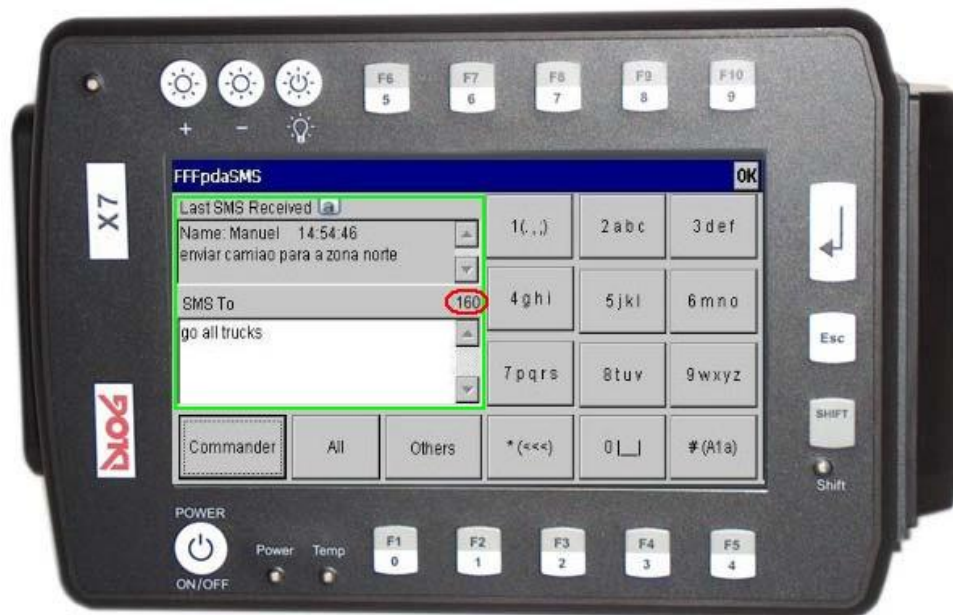


Figure 14 – FFFpdaSMS Displays

Software Buttons

In terms of software buttons this interface has two different sets of buttons: the SMS Destination Buttons and the Alphanumeric Keypad.

SMS Destination Choices

There are three different destination possibilities:

- **“Commander”** which will only send the message to the Commander of the present operation.
- **“All”** which sends the message to all personnel at the scene of the at the theatre of operations.
- **“Others”** which is reserved for any alteration or outside personnel that is not on the list of the two previously referred recipient types.

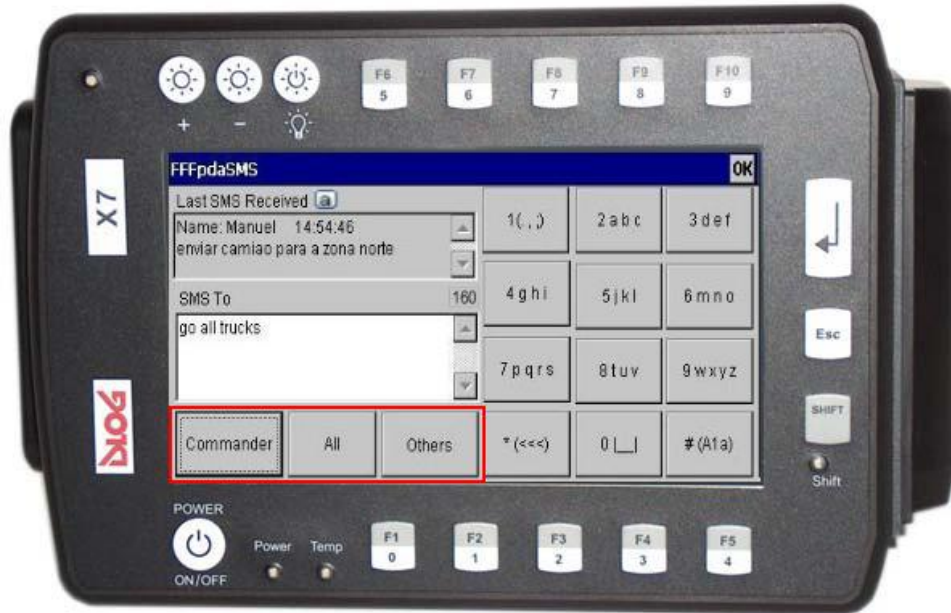


Figure 15 – FFFpdaSMS SMS Destination Buttons

Alphanumeric Keypad

The keypad available in the system is similar to the one found in most mobile phones.



Figure 16 – Alphanumeric keypad

The time interval between the letter/number cycles on each key press was defined to two seconds.

The following table resumes the button/action correspondence.

1(.,:)	=> digit 1 / dot, comma, semicolon
2 a b c	=> digit 2 / letter a, b or c
3 d e f	=> digit 2 / letter d, e or f
4 g h i	=> digit 2 / letter g, h or i
5 j k l	=> digit 2 / letter j, k or l
6 m n o	=> digit 2 / letter a, b or c
7 p q r s	=> digit 7 / letter p, q, r or s
8 t u v	=> digit 8 / letter t, u or v
9 w x y z	=> digit 9 / letter w, x, y or z
*(<<<)	=> eliminate character(s)
0 _ 	=> digit 0 / insert a space character
#(aA1)	=> Rotate between small, capitals and numbers

Figure 17 – Specific key values

3. “How to”

3.1 How do I turn ON the PDA?

Please press the **“POWER ON/OFF”** hardware button for 2 seconds and system will boot automatically.

3.2 How do I turn OFF the PDA?

To terminate the *FFF PDA* application correctly before turning off the vehicle engine, the only action required is to press the **“OFF”** button on the **FFFpdaMAIN** interface, and after a few seconds a message will appear informing that all shutdown procedures have been completed, disabling all further actions until the user presses the **“POWER ON/OFF”** hardware button.



Figure 18 – Shutdown Sequence

3.3 How do I send a SMS?

To send a SMS text message, the user must first bring the **FFFpdaSMS** interface to the foreground by pressing **“F5”** (See **Figure 1**) which selects the appropriate interface, then compose the text using the alphanumeric keypad and choose a destination from the three software buttons: **“Commander”**, **“All”** and **“Others”**.

NOTES: To delete text the user must press the **“(←←←)”** keypad button the appropriated amount of times.

To change the capitulation of the letters or activate the NUMLOCK the user must press the **“(αA1)”** keypad button.

3.4 Receiving SMS events on **TomTom** interface?

When an event arrives, the system will flash a message on the top right corner of the screen with one of the following texts:

Event	Text
Fire	<p>“A FIRE has been added!”</p> <p>“A FIRE has changed position!”</p> <p>“A FIRE has been put out!”</p>
Weather	<p>“A WEATHER station has been added!”</p> <p>“A WEATHER Info Update!”</p>
Auto	<p>“A VEHICLE has been added!”</p> <p>“A VEHICLE has changed position!”</p> <p>“A VEHICLE has left the scene!”</p>
Water	<p>“A WATER source has been added!”</p> <p>“A WATER source Update!”</p>
Human	<p>“SMS received!”</p>

Table 1 – Event/Occurrence TomTom screen messages

3.5 Receiving a SMS on **FFFpdaMain** and **FFFpdaSMS** interface?

When a human SMS text message arrives it will be added to the SMS Dialog kept on the **FFFpdaMAIN** interface, specifically on the **SMS** display.

The last SMS text message received is always shown on the **FFFpdaSMS** interface **“Last SMS Received”** display.

4. Administration

This section is dedicated to the systems administrator access and the description of its areas of action, concerning system maintenance, operation data retrieval, specific text files alteration and menu customization.

4.1 Accessing the Administrator Login

To access to the administrator section the system needs to be terminated correctly, so that the qualified user can have access to the Windows Operative System. The administrator then has the possibility of retrieving, modifying specific files that in some cases are being used by the system during the applications runtime. In the case of the TomTom™ menu customization a system reboot is needed in order for the alterations to take effect.

To access the section the user needs to press the button **"a"** which pops up a login window.

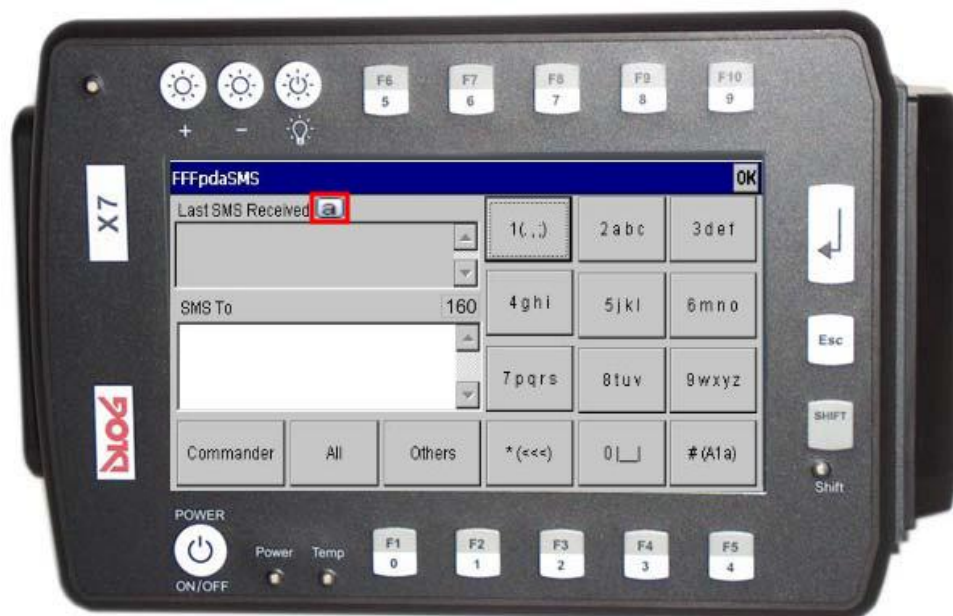


Figure 19 – FFFpdaSMS Administrator Access Button

On this window, the default name is **admin** and the password is **gogo**.

The login information is changeable and will be explained the procedure on sub-chapter – 5.1.

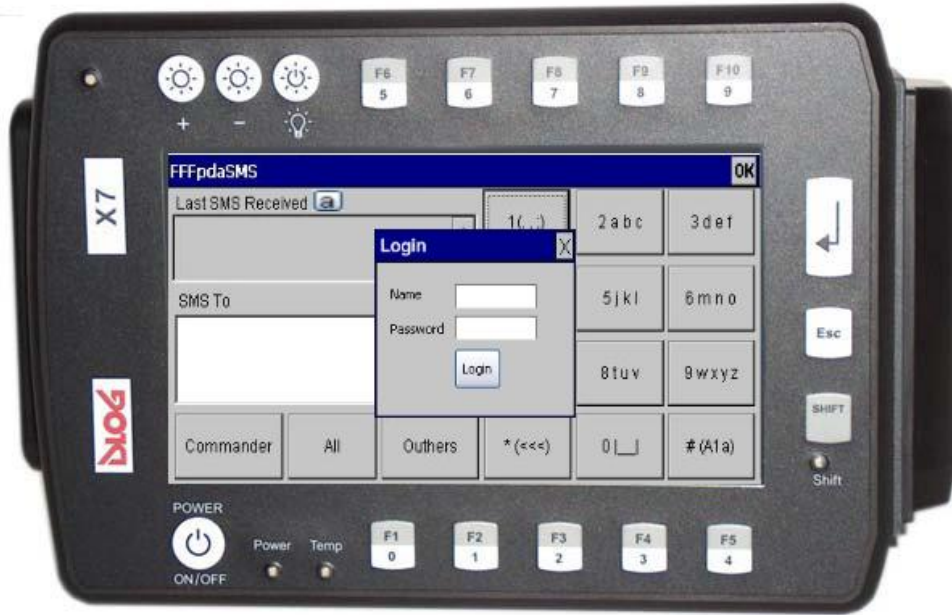


Figure 20 – Administrator Login Interface

When the login action has been concluded, the user should see the Windows CE 5.0™ desktop screen, and should retrieve the Start menu bar by touching the inferior part of the LCD screen.

4.2 Customizing the TomTom Menu

TomTom™ allows a certain degree of further customization to the TomTom Navigator™ menus in terms of available options, color scheme amongst other details. To accomplish this action a special file must be produced according to a specific set of rules. The section in “**Chapter 5 – TomTom Customization Menu**” explains all the procedures in detail.

4.3 Text Files

All the text files are stored and handled on the same directory of the PDA storage card. Please go this location "**Windows Explorer/Storage Card/Testes/**".

On this directory there are several text files, the following system files should not be changed: "**system.txt**", "**recup.txt**" and "**dateofevent.txt**".

However the remaining text files can and should be modified if required: "**glf.txt**" and "**gsmlt.txt**" are internal system indexes; "**tempt.txt**" is where the username and password of the Administrator is kept; "**globallog#.txt**" and "**gsmlog#.txt**" are the files where the entry logs are stored and finally "**numbers.txt**" is where the SMS destination number is saved.

- "**glf.txt**" and "**gsmlt.txt**"

Although the global log file index auto resets itself, in case of an abnormal system malfunction any value between 0 and 9 is a valid option.

- "**tempt.txt**"

When the need to change login user **Name** and **Password** arises you will need to open the file with any text editor and change:

- the **first** line with the new user **name**,
- the **second** line with the respective **password**

- "**globallog#.txt**" and "**gsmlog#.txt**"

All the information stored during the operation is stored respectively:

- "**globallog#.txt**", concerning the events and user actions entries.
- "**gsmlog#.txt**", regarding all modem related events.

- "**numbers.txt**"

When the administrator wants to change the SMS destination number, the "numbers.txt" file should be opened with any text editor and change the first line with new number.

5. TomTom Customization Menu

The following document is property and intricate part of the TomTom™ Software Development Kit, more specifically the documentation, "configuration2.html".

1. Introduction

Several default configuration components of TomTom NAVIGATOR can be adjusted, allowing users to modify the appearance of TomTom NAVIGATOR.

By creating certain files to be placed in certain directories, the appearance of the menu can be modified or the list of colour schemes can be extended with user-defined colour schemes.

This section explains how to define a customized menu and how to add a colour scheme.

2. How to customize the menu

2.1 Example customized menu

Copy the following text into a file called 'TomTom.mnu' or use the corresponding file from the \TomTomPPC-SDK\examples\additional_configuration\ folder.

```
MENUBLOCK|BLOCK_MAIN|BTM_GPS_POSITION|
```

```
MENUPAGE|TASK_PAGE1|Custom menu 1|
MENUITEM|TASK_MENU_PLAN_ATOB|
MENUITEM|TASK_NAVIGATE_TO||"Go Somewhere"|
MENUITEM|TASK_MENU_FIND_ALTERNATIVES|"none.bmp"|"Alternatives..."|
MENUITEM|TASK_SDK1|
MENUITEM|TASK_PAGE5|TASK_MENU_PREFERENCES|TASK_MENU_PREFERENCES|
MENUITEM|TASK_PAGE2|
MENUPAGE|TASK_PAGE2|Custom menu 2|
MENUITEM|TASK_ADD_FAVORITE|
MENUITEM|TASK_SHOW_MAP|
MENUITEM|TASK_MENU_ROUTE_INSTRUCTIONS|
MENUITEM|TASK_DIAL_POI|
MENUITEM|TASK_ITINERARY|
MENUITEM|TASK_PAGE3|
MENUPAGE|TASK_PAGE3|Custom menu 3|
MENUITEM|TASK_SHOW_WEATHER|
MENUITEM|TASK_MENU_DOWNLOAD|
MENUITEM|TASK_MENU_TRAFFIC|
MENUITEM|TASK_TUTORIAL|
MENUITEM|TASK_CONFIRM_DELETE_ROUTE|
MENUITEM|TASK_PAGE4|
MENUPAGE|TASK_PAGE4|Custom menu 4|
MENUITEM|TASK_EXIT_APP|
MENUITEM|TASK_PAGE1|
```

```
MENUBLOCK|BLOCK_PREF|BTM_DONE|
MENUPAGE|TASK_PAGE5|Custom preferences|
MENUITEM|TASK_SET_COLOR_SCHEMES|
MENUITEM|TASK_SET_CLOCK_TYPE|
MENUITEM|TASK_SWITCH_BLUETOOTH|
MENUITEM|TASK_SET_VOICE|
MENUITEM|TASK_CONFIRM_DELETE_ROUTE|
MENUITEM|TASK_PAGE1|
```

Put the file in the directory \TomTom\SdkRegistry\. Also, make sure at least one of the previously defined capability files is available in \TomTom\SdkRegistry\. The command defined in the capability file will now show as the 5th menu item of the first main menu as defined by "MENUITEM|TASK_SDK1|". If you do not have a capability file available and you still want to complete the example, just remove that line from the 'TomTom.mnu' file.

Now restart TomTom NAVIGATOR and browse through the main menu. The menu will now show your custom menu.

2.2 Defining the menu

To define a menu, a text file should be created with the following name:

```
TomTom.mnu
```

In this file the menu items to be shown on each page of the menu are defined.

The format of the TomTom.mnu file is pipe-separated ASCII, where each line consists of at least a level and one of the predefined names defined for that level.

Three levels can be distinguished:

- **MENUBLOCK:** The definition of a menu block. The main menu block, map cursor menu block and preferences menu block can be defined.
- **MENUPAGE:** The definition of a menu page, i.e. page1, page2, etc.
- **MENUITEM:** The definition of a menu item, i.e. "Navigate to", "Clear Route, etc.

To define a menu, you first have to define the menu block:

```
MENUBLOCK|BLOCK_MAIN|BTM_GPS_POSITION|
```

where

- BLOCK_MAIN is the name of the main menu block
- BTM_GPS_POSITION is the GPS-information, to be shown on the bottom of the main menu

Every line after this will be interpreted as a page or an item of the main menu.

Then a page is defined, by:

```
MENUPAGE|TASK_PAGE1|Custom menu 1|
```

where

- TASK_PAGE1 is the internal name of the menu page
- My Menu 1 is the name of the menu page, to be show on the top of the menu page

Note: The name of the menu page does not require doubles quotes.

Now, the menu items are defined for each page. The first item of the first page of the main menu is the predefined menu item "Advanced Planning":

```
MENUITEM|TASK_MENU_PLAN_ATOB|
```

The second item of the menu says "Go Somewhere" and selecting it brings you to the default "Navigate To"-functionality:

```
MENUITEM|TASK_NAVIGATE_TO||"Go somewhere"|
```

The description "Navigate to..." has been replaced with "Go somewhere", the icon is the default TomTom NAVIGATOR icon.

The next menu item in the example is the predefined "Find Alternative". In this case the icon for the menu option has been replaced by a user-defined icon and the description has been replaced by "Alternatives...":

```
MENUITEM|TASK_MENU_FIND_ALTERNATIVES|"none.bmp"|"Alternatives..."|
```

To use your own bitmaps, place your .BMP file in the /TomTom/SdkRegistry/ directory. If the bitmap isn't available, the default icon will be displayed.

To browse through the menu, one of the 6 menu items of each page should be reserved for the next-screen item:

```
MENUITEM|TASK_PAGE2|
```

This shows the default arrow for going to the next page and selecting it from the menu will bring you to the next page.

Finally, an icon can be greyed out by adding 1 as a final argument:

```
MENUITEM|TASK_MENU_PHONE||"My Phone"|1|
```

The menu option "My Phone" will show on the menu, but it is greyed out and can not be accessed.

Please note that the pages are interpreted in the order in which they are defined and not by the number of the TASK_PAGE, so make sure the pages are defined in following order.

For an overview of the predefined menu items, see:
Section **2.5 Overview Menu Items**.

A special item TASK_SDKn is defined:

```
MENUITEM|TASK_SDK1|
```

This item represents the command to be sent to an external application, as defined in a capability file.

If you have several commands to send or several applications to start, you should define a different TASK_SDKn for each command. If you have three commands defined in capability files, these items should be defined in the menu: TASK_SDK1, TASK_SDK2, TASK_SDK3. Please note that adding an icon or a description to the SDK_TASK line, the icon and description specified in the capability file will be overwritten.

The external commands will be interpreted in the same order as they were on the extended menu.

After you have created the file and defined the menu, put it in the following directory:

```
\TomTom\SdkRegistry\
```

When TomTom NAVIGATOR is started, this SdkRegistry directory is searched for the file 'TomTom.mnu'. If available and correct, the customized menu will show instead of the default menu.

2.3 Summarize TomTom menu

A customized menu is defined by:

```

MENUBLOCK|«blockname»|«btm_option»|
MENUPAGE|TASK_PAGE1|«title»|
...
...
...
MENUPAGE|TASK_PAGE«n»|«title»|

```

where

- «blockname» is one of the [Blocks]
- «btm_option» is one of the [BTM_option]
- «title» is the title of the menu page, to be shown at the top of the screen
- TASK_PAGE1 through TASK_PAGE«n» are the main menu pages, defined by:

```

MENUITEM|«taskname»|«icon»|«description»|«accessibility»|
MENUITEM|«taskname»|«icon»|«description»|«accessibility»|
MENUITEM|«taskname»|«icon»|«description»|«accessibility»|
MENUITEM|«taskname»|«icon»|«description»|«accessibility»|
MENUITEM|«taskname»|«icon»|«description»|«accessibility»|
MENUITEM|TASK_PAGE«n+1»

```

where

- «taskname» is one of the predefined Menu Items
- «icon» is the name of the bitmap for the menu item (optional). If left out, the default TomTom NAVIGATOR icon will show
- «description» is the description of the menu item (optional). If left out, the default TomTom NAVIGATOR description will show
- «accessibility» is 0 or empty for 'allow', any other integer for greyed out (optional). If left out, the default accessibility will show
- TASK_PAGE«n+1» is the reference to the next page. For the last menu page, define TASK_PAGE1 here in order to go to the first screen again

2.4. Custom Preferences

In your custom menu, you can also specify a set of preferences tailored to your needs. To do that, you have to do two things:

- specify the entry point for the preferences:

```
MENUITEM|TASK_PAGE5|TASK_MENU_PREFERENCES|TASK_MENU_PREFERENCES|
```

- define the preferences block

```
MENUBLOCK|BLOCK_PREF|BTM_DONE|  
  
MENUITEM|«taskname»|«icon»|«description»|«accessibility»|  
MENUITEM|«taskname»|«icon»|«description»|«accessibility»|  
MENUITEM|«taskname»|«icon»|«description»|«accessibility»|  
MENUITEM|«taskname»|«icon»|«description»|«accessibility»|  
MENUITEM|«taskname»|«icon»|«description»|«accessibility»|  
MENUITEM|TASK_PAGE«n+1»
```

The parameters are the same as for the main block.

The preferences block can have more than one page. You define navigation between pages in the similar fashion as for the main block: the last menuitem could be the next preferences page, the first preferences page or the first menu page.

2.5. Overview menu items

For more information on TomTom NAVIGATOR menu items, please refer to the TomTom NAVIGATOR manual.

Blocks

There are three blocks that can currently be defined in a menu file:

- BLOCK_MAIN , Main Menu Block
- BLOCK_PREF, Preferences Menu Block
- BLOCK_CURSOR, Cursor Menu Block

BTM_options

At block-level, the following items can be defined to be shown at the bottom of the screen. Each menu-page of that block will show the BTM_options defined for that block.

- BTM_GPS_POSITION , GPS Information
- BTM_DONE , Done Button
- TRAFFIC_INFO , Traffic Information

Menu Items

The following menu items can be used in the customized menu file for the main menu and the preferences menu.

Special Tasks

- TASK_EMPTY, no task and this option will be shown as a blanc spot in the 6-option menu
- TASK_NONE, no task and no placeholder for it in the menu
- TASK_PAGEn, (n is in the range 1-20) a number identifying the custom menu page
- TASK_SDKn, (n is in the range 1-50) a command to be sent to a client application, as defined in a capability file

Navigation

- TASK_AVOID_ROUTE_LINE, Choose part of the route to avoid
- TASK_CONFIRM_DELETE_ROUTE , Clear Route with asking the user confirmation first
- TASK_DELETE_ROUTE, Clear Route immediately
- TASK_ITINERARY , Itinerary planning
- TASK_NAVIGATE_TO , Navigate to
- TASK_MENU_FIND_ALTERNATIVES , Go to "Find alternative" menu
- TASK_MENU_PLAN_ATOB , Advanced planning
- TASK_MENU_RECALC_BLOCK, Go to "Avoid roadblock" menu
- TASK_MENU_ROUTE_INSTRUCTIONS, Show the menu with options for viewung route
- TASK_PLAN_VIA, Calculate a route that goes via specified address
- TASK_SET_TOLL, Set the preferences for toll roads
- TASK_SET_PLANTYPE, Set the preferences for route planning
- TASK_SET_CRADLE, Set cradle preferences
- TASK_SHOW_ROUTE_DEMO, Demonstrate planned route
- TASK_SHOW_ROUTE_INSTRUCTIONS, Show the list with route instructions
- TASK_STEP_ROUTE_INSTRUCTIONS, Browse through the route instructions as images
- TASK_SWITCH_2D3D, Switch between 2D and 3D navigation view

Downloads

- TASK_MENU_DOWNLOAD , Download extras
- TASK_MENU_TRAFFIC , TomTom Traffic
- TASK_SHOW_WEATHER , TomTom Weather

Manage Maps

- TASK_DELETE_MAP, Delete a map
- TASK_SET_MAP, Choose a current map
- TASK_SET_HIDEMAP, Specifies whether the map should be replaced with a simpler view under certain conditions
- TASK_SHOW_MAP , Browse map

Favourites

- TASK_ADD_FAVORITE , Add favourite
- TASK_MAINTAIN_FAVORITES, Maintain the list of favorites

Preferences

- TASK_CHANGE_HOME_LOCATION, Specify new home location
- TASK_LEFTHANDED, Switch between lefthanded and righthanded UI
- TASK_MENU_PREFERENCES , Change Preferences
- TASK_RESET_SETTINGS, Restore original application settings
- TASK_ROTATE_DISPLAY, rotate display by 180 degrees
- TASK_SET_BRIGHTNESS, Adjust the brightness level for the device
- TASK_SET_BACKLIGHT, Adjust the backlight settings for the device
- TASK_SET_CLOCK_TYPE, Set clock type
- TASK_SET_COLOR_SCHEMES, Choose day and night color schemes
- TASK_SET_COMPASS, Specify how to show the compass in the navigation view
- TASK_SET_DAY_COLOR_SCHEME, Choose day color scheme
- TASK_SET_DIST_UNITS, Choose distance units
- TASK_SET_KEYBOARD_SIZE, Set the size for the onscreen keyboard
- TASK_SET_LANGUAGE, Choose application language
- TASK_SET_NIGHT_COLOR_SCHEME, Choose night color scheme
- TASK_SET_PROMPT, Specify the additional settings for voice prompts
- TASK_SET_STATUS, Specify the information to show at the bottom of navigation view
- TASK_SET_VOICE, Choose the voice for reading instructions
- TASK_SET_VOLUME, Adjust the voice volume
- TASK_SHOW_GPS_STATUS , Configure GPS
- TASK_SHOW_STATUS, Show route status
- TASK_SWITCH_BLUETOOTH, Enable/disable bluetooth connectivity
- TASK_SWITCH_NIGHTVIEW, Switch to night view mode
- TASK_SWITCH_SOUND, Enable/disable sound
- TASK_SWITCH_TIPS, Enable/disable tips

POI

- TASK_CONFIGURE_POI, Choose which types of POI to show
- TASK_MAINTAIN_POI, Go to POI maintenance menu
- TASK_SWITCH_POI, shows/hides POI

Traffic

- TASK_MENU_TRAFFIC, go to traffic menu
- TASK_TRAFFIC_ENABLE, enable traffic support
- TASK_TRAFFIC_EXPLAIN, help for traffic
- TASK_TRAFFIC_REPLAN, replan current route
- TASK_TRAFFIC_SETTINGS, specify traffic settings
- TASK_TRAFFIC_UPDATE, update traffic information
- TASK_TRAFFIC_VIEW, go to traffic incidents view

Miscellaneous tasks

- TASK_ABOUT, show NAVIGATOR information screen
- TASK_EXIT_APP, exit the application
- TASK_SET_NAME_DISPLAY, specify naming preferences for map/navigation views
- TASK_BUDDIES, go to buddies menu

Downloads

- TASK_DOWNLOAD_MAP, Download map
- TASK_DOWNLOAD_POI, Download POI
- TASK_DOWNLOAD_VOICE, Download voice
- TASK_DOWNLOAD_SCHEME, Download color scheme
- TASK_DOWNLOAD_VERSIONNUMBER, Upgrade the NAVIGATOR application
- TASK_MANAGE_MAPS, Go to "Manage maps" menu
- TASK_MENU_DOWNLOAD, Go to "Downloads" menu
- TASK_SERVICE_LOGIN, Perform login to the TomTom Plus services
- TASK_SHOW_WEATHER, Get the current weather information

Help

- TASK_HELP_GENERAL, General help
- TASK_HELP_ITINERARY, Help for itinerary
- TASK_HELP_MAINMENU, Help for the main menu
- TASK_HELP_MAPBROWSER, Help for the map view
- TASK_HELP_PLANNING, Help for route planning
- TASK_HELP_TRAFFIC, Help for traffic
- TASK_HELP_ZOOMING, Help for zooming
- TASK_TUTORIAL, Guided tour

Phone

- TASK_DIAL_POI, Dial the POI
- TASK_MENU_PHONE, Mobile phone
- TASK_PAIR_WITH_PHONE, Connect to the phone via Bluetooth

Multimedia

- TASK_MULTIMEDIA, Go to multimedia menu
- TASK_JUKEBOX, Go to jukebox menu
- TASK_IPOD, Go to IPod control
- TASK_IMAGE_BROWSER, Go to image library
- TASK_DOC_BROWSER, Go to document reader

The following menu items can be used in the customized menu file for the map cursor menu.

Cursor menu

- TASK_PLAN_TO_MAPLOC, Navigate to map cursor location.
- TASK_NEARBY_POICAT, Find POI near map cursor location.
- TASK_CENTER_ON_MAP, Center map on map cursor location.
- TASK_DO_ADD_FAVOURITE, Add map cursor location as favourite.
- TASK_CURSOR_INTO_POI, Add map cursor location as POI.
- TASK_TRAVEL_VIA, Travel via map cursor location.
-

3. How to add a colour scheme

TomTom NAVIGATOR comes with several map colour schemes, which can be selected by selecting 'Set day colour scheme' or 'Set night colour scheme' from the Preferences menu. It is now also possible to create your own map colour scheme and add it to the list of schemes.

3.1. Example colour scheme

Copy the following code into a file called 'mycolours.clr' or use the corresponding file from the \TomTomPPC-SDK\examples\additional_configuration\ folder.

```
;
; Belgica
;
204,255,204 ; background
255,204,153 ; built-up area (city)
168,168,255 ; water solid
204,255,153 ; park
```

153,204,153 ; woodland

255,255,153 ; beach/dune/sand

204,153,255 ; industrial zone

204,204,204 ; harbor/marina

255,204,204 ; moor/heath

153,255,204 ; marsh

255,204,255 ; pedestrian zone

153,153,204 ; airport

204,204,204 ; runway

192,192,192 ; route background ('below' the road)

136,000,000 ; route foreground (on top of the road)

255,102,153 ; blocked road (dotted on top of any road)

255,000,153 ; traffic jam (dotted on top of any road)

000,136,000 ; highlighted / selected (line on top of any road)

255,153,255 ; unreachable / not allowed to drive (dotted on top of any road)

255,000,000 ; motorway solid

255,000,000 ; motorway edge

255,255,000 ; motorway inside

136,000,000 ; motorway text

153,051,000 ; major/international road solid

136,000,000 ; major/international road edge

255,255,000 ; international road inside

255,255,000 ; major road inside

051,051,000 ; major road text
255,255,000 ; secondary road small
255,255,000 ; secondary road solid
000,000,000 ; secondary road edge
255,255,000 ; secondary road inside
153,153,000 ; secondary road text
255,255,000 ; connecting road small
255,255,102 ; connecting road solid
000,000,000 ; connecting road edge
255,255,102 ; connecting road inside
153,153,000 ; connecting road text
255,255,255 ; (major)local road small
255,255,255 ; major local road solid
255,255,255 ; local road solid
000,000,000 ; local/destination road edge
255,255,255 ; local/destination road inside
085,085,085 ; local road text
255,255,255 ; destination road solid
085,085,085 ; destination road text
255,255,255 ; ferry background
000,000,136 ; ferry dots
000,000,136 ; ferry text
170,170,170 ; railroad dashed A
000,000,000 ; railroad dashed B

```
153,153,255 ; water edge outlines
000,000,000 ; borders (dot-dash)
000,000,000 ; arrows (road direction)
255,255,255 ; arrows border (road direction)
000,128,000 ; Route indication arrow color, normal
000,255,000 ; Route indication arrow color, highlighted
000,128,000 ; Tollroad edge color
133,227,135 ; Tollroad inside color
000,000,092 ; NavView: status/safeview panel background (default:dark
blue)
080,080,190 ; NavView: status/safeview panel borders (default:blue)
255,255,255 ; NavView: status/safeview panel text (default:white)
206,207,255 ; NavView: status/safeview panel dim text (default:lightblue)
000,000,092 ; NavView: next highway background
255,255,255 ; NavView: next highway border
255,255,255 ; NavView: next highway text / iRouteDirectionArrowColor
000,128,000 ; NavView: next highway EXIT background
255,255,255 ; NavView: next highway EXIT border
255,255,255 ; NavView: next highway EXIT text
095,100,215 ; NavView: routeplanning progressbar color
192,092,092 // Distant route color
```

Put the file 'mycolours.clr' in a directory called 'SCHEMES' from the root of the SD card. First-time SDK users will probably have to create this directory. Within TomTom NAVIGATOR browse to Preferences => 'Set day colour scheme' or 'Set night colour scheme' Tap on the arrows to cycle through the colour schemes. A plug-in scheme will be available and selecting it will show you the new scheme 'mycolours'.

3.2. Creating a colour scheme

Choose the name of your colour scheme and create a file with that name and the extension CLR. The name will be displayed on the scheme selection page, so choose a sensible name.

In the .CLR file define the colours of the different components of a map; what colour do you want a park to have, how do you want a motorway to look like, a city, etc. The colours are defined by their RGB values: the red, green, and blue intensity values that are used to define a colour. The comment behind each set of values, which is preceded by either '/' or ";", describe which component of the map that line defines.

Please note that the colour-definition on a line in the colour file is always assigned to the same component; the colour for woodland is always the 5th colour defined in the file, a runway always the 13th, etc. So, commenting out some line at the beginning of the file, should give you some funny results, as each component gets the previous component's colour-definition.

Refer to the comments of the example 'mycolours.clr' to see which component of a map each line represents.

3.3. Adding a colour scheme

To add a colour scheme, put a valid CLR file in the following directory:

```
\SD Card\SCHEMES\
```

When selecting a colour scheme from the Preferences Menu, a plug-in scheme will be available. Selecting it will show the newly created colour scheme 'mycolours'. created colour scheme 'mycolours'.

6. Technical Support

If you encounter any problem and/or want to make a suggestion regarding the Forest Fire Finder PDA, please contact our services by e-mail or letter.

Pedro Vieira

Madan Parque de Ciência

Campus da Caparica Faculdade de Ciências e Tecnologia da

Universidade Nova de Lisboa, Edifício IV

2829-516 Caparica PORTUGAL

Appendix - E.

Submitted Paper

Please note that this paper follows its own page numbering.

Real Time Mobile System For Support in Firefighting Environments

João C.Mendes, Department of Electrical Engineering, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Lisbon, Portugal

Nuno Pinto, NGNS Ingenious Solutions Ltd. Lisbon, Portugal

José M.Fonseca. Department of Electrical Engineering, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Lisbon, Portugal

Pedro M.Vieira¹, Department of Physics, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Lisbon, Portugal

Abstract. The main objective of this work is to provide a solution running on a rugged Personal Digital Assistant (PDA) device that provides decision support in Fire Fighting environments. Using this system the user has access to: multi-information data of the theater of operations, an automotive navigation system and text messaging capability. The hardware that supports this system is a DLoG X7™ industrial graded PDA that possesses the right sturdiness for this task. Two interfaces were developed, one allows a proper visualization of the available information and the other provides text messaging input/output capabilities. An additional TomTom™ interface was customized for the specific needs of the end-users of this system. It is important to refer that the text messaging service of GSM protocol was the chosen communication way, due to its reliable coverage in remote areas, such as the forests. Results show that in terms of performance validation in the field, the system responded to the challenges proposed. After a series of usability tests it was shown that in terms of overall appreciation 80% of the users considered it Excellent or Very Good. Specifically concerning the developed interfaces 93% or more considered them very Easy to interact, thus validating the intended objectives.

Keywords: *Decision Support System, Forest Fire, PDA, GSM, GPS*

¹ Correspondence should be addressed to: Pedro Vieira , E-mail: pmv@fct.unl.pt

1. Introduction

Forest fire is a scourge that affects many countries and the last years have been a testimony of this situation. Once the Fire has been detected the swiftness of the reaction and response times are essential in order to minimize the damages. An optimized access to all the available information related to the theater of operations is fundamental to improve the quality and the speed of the decision process.

Some solutions have been tried in the past: a hand-held device approach to disaster management [1] that due to cost and technology limitations of the time was not pursued. A GIS-based fire management information system was also developed but focused on a stationary operator [2]. Similar emergency decision support systems have also been modeled [3]. In addition to these specific disaster-related solutions, interesting approaches have been applied in diverse scenarios as decision support systems: logistic and fleet management in warehouse [4] and mining industry [5].

The work developed for this paper combines the features of the previous mentioned examples in a unique decision support system that has the following key features: improve response time through a navigational software, provide the Firemen with a visual scenario of the theater of operations, and also update this information in real-time. Mobility on the case of this paper is fundamental, so the system was designed to be used in a vehicle mounted PDA device, with simple and intuitive interfaces in order to simplify the task of the Firemen. As an alternative to the common radio communications, text messaging capability is provided which can be used as an alternative broadcasting medium.

The innovation that this system proposes is to provide a mobile tool to aid the decision process on a Fire scenario yet maintaining an easy and simple interaction with the Firemen. This design objective modeled the visual and interactive features of the system interfaces, providing the Firemen with an intuitive presentation of the real-time available information, combined with map visualization.

1.1. Objectives and Contributions

This paper addresses the work being performed in the context of the Forest Fire Finder™ project developed by NGNS-IS™, but standing as an independent yet globally integrated module that seamlessly interacts with the existent platform.

Several objectives were devised: in-depth research on the industrial PDA, DLoG X7™ and its versatile and rugged design that withstands harsh environments; development of a Text Messaging protocol that carries all the communications with the backbone server, from the GSM module signal reception, to a higher message processing level; implementation of the most user-friendly, simple interface possible to allow a fast, intuitive access and interaction with the developed application.

The main contribution of this system is to provide within the Firefighting paradigm a useful, user-friendly tool that hopefully allows a faster and more precise response to a fire scenario on a Forest, giving emphasis on real time updated data concerning all the parts involved on the operation: human, physical and environmental.

2. Resources Overview

This section presents the chosen tools for the realization of the decision support system, in terms of the physical device on which this system is running, as well as the used software platform. The first section describes the used hardware, the DLoG X7™ PDA [6]. The second section presents the automotive navigational software TomTom Navigator™ (TTN) and its Software Development Classes (SDK).

2.1. Hardware - DLoG X7™

Due to the extreme conditions that the PDA will be working, the decision was made of choosing a fixed robust device mounted on the vehicle instead of a portable handheld one. This is justified by the need for a more ergonomic Firemen oriented device that could support the abuse of the physical aggressions which are common inside a firefighting vehicle, specifically under stress.

The DLoG X7™ PDA fulfilled the requirements in terms of the need of a PDA that could withstand the extreme environmental (temperature, humidity, water resistance and shock) conditions that are present in forest fires environments. The hardware also possessed the specific features desired for the system: a minimum display size of 7 inches and in terms of features and software requirements, a modem with GSM/GPS communications features and the use of a Microsoft Windows™ OS, Win CE 5.0.

2.2. Software – TTN

TomTom™ is one of the leading navigational software companies in the world, being well known for its stand-alone units as well as the case of this paper, for its software TomTom Navigator™ (TTN) installation for other mobile devices. It is recognized by many features, specifically the speed and effectiveness of its route tracing algorithm that conjugates perfectly with the accuracy and detail that their road maps usually have.

Since TomTom™ SDK classes are developed for C++ that was the adopted language to develop the application. In the case of this system the Application Programming Interface (API) was used. The Integrated Development Environment selected was the Microsoft Visual Studio 2005™ Suite.

3. Interfaces

The several interfaces that the user has at its disposal on this system were developed, first and foremost, with the intent of aiding and simplifying the task of the Firemen in response to a fire. A particular emphasis was given to the notions that have already been referred on this paper: to be ergonomic, intuitive and user-friendly. These objectives were fulfilled to the extent of what the hardware allowed.

3.1. FFFpdaMAIN

The main goal of this interface (Figure 1) is to provide immediate visualization of the most recent information. There are three text displays: “Global Log” where each system occurrence is logged; “Events” that displays the details of the existent Event entries and “SMS” where all the human text messages traded with the server are displayed to create the notion of a dialog. Four major buttons filter the information on the “Events” display: “See Fires“, “See Weather“, “See Autos“ and “See Water“, each of them showing the full detailed information associated to each entry of the following events respectively: Fire, Weather, Auto and Water.



Fig 1 FFFpdaMain interface

The “Global Log” display indicates that a new Fire event has been received as well as new Meteorological station has been added to the database. The “Events” display shows the latest updated information concerning the two existing Water supplies. The “SMS” display is showing two text messages traded between the system and external users.

The user also has the capability of pressing on any given text display enabling the hardware button controlled Text Scrolling actions. The three small boxes next to the display name have the task of giving the user the visual information of which is selected. This provides better control of the text flow, avoiding a screen drag action which could be hard to perform while driving on difficult terrains. Finally an “OFF” button is also present to insure a proper close sequence for the system.

3.2. FFFpdaSMS

This interface (Figure 2) consists of three main buttons: “Commander”, “All“ and “Others“. The “Commander“ button action sends the written text message to the person responsible for the coordination of theater of operations. The “All“ button action sends the written text to all parties on the

theater of operations. Finally, the “Others” button action can be customized to any desired additional number. The remaining buttons belong to the alphanumeric keypad.

To visualize the SMS text messages there are two text displays: “SMS To”, where the user will write the text and “Last Received SMS”, where the last received text message will be shown. In order for the user to keep track on the amount of characters that have been written, a counter has been added. The actual limit for SMS text messages is 160 characters.

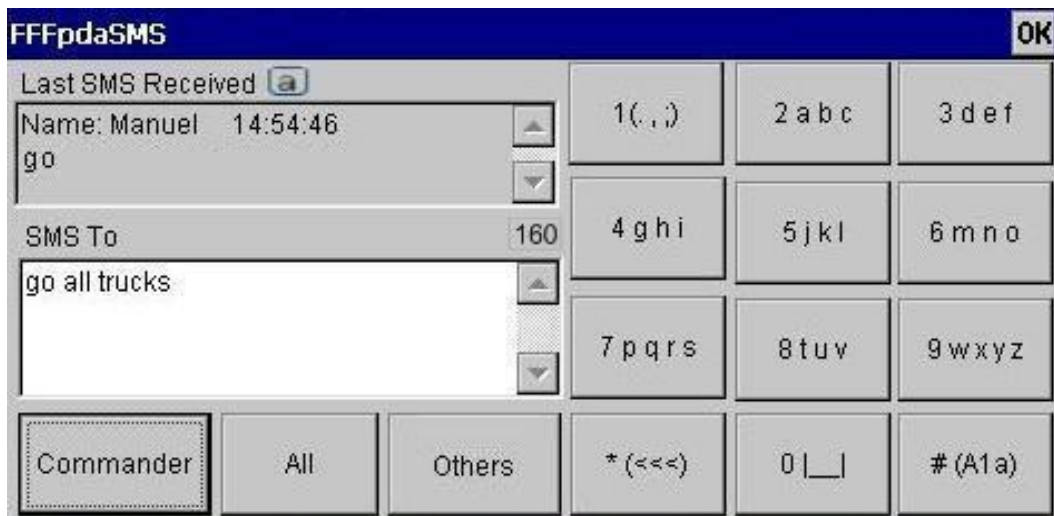


Fig 2 FFFpdaSMS interface

3.3. TT Interface Customized

When interacting with the TTN the user has two options: the default TT interface that shows where the vehicle is at the moment; and the main browse interface (Figure 3) that allows the user to go directly to the Fire scenario and observe the events that have been received.



Fig 3 TTN browse interface with all events enabled

Note: This image is adapted from the TomTom Navigator™, “Browse Map” interface

If the user wants to filter the events shown onscreen, five PDA hardware buttons have been programmed to Show/Hide the different events, one for each possible event and a global filter which hides all the events from the screen. In case the user decides to press the middle of the screen on the TT interface a TTN customized main menu appears. On this menu the user can choose one of the actions: “Go To Event” where he will chose what event (Fire, Water, Weather, Truck) will the TTN trace a route to; “Browse Map” where he can zoom the map directly to the desired event and finally the option to enable/disable the “3D display”.

4. System Architecture

This section presents the proposed architecture for the real time control system that manages the PDA device actions. The first section presents a global overview of the system. The second and last section describes the SMS protocol developed purposely for the system.

4.1. Global Overview

The main objective of this paper was to create a multi-interface application that could handle, route and display various types of data in real time. The data is handled transparently to the user, through the Main Class and the auxiliary classes that run on a multi-threaded system. The communication aspects are divided into two types of concerns: external and internal. The external communication, relating to the server, is handled by the SMS Class from the lower level Serial Port data processing to the higher level SMS protocol. All the internal communication between the TTN and the two interfaces, “Main” and “SMS”, is performed via Windows Messages.

Although the majority of the architecture can be separated and each module described by itself, other aspects do not and are an inherent part of all the code developed throughout the applications. The most prominent cases are: windows messages handling (both system and application defined) and TTN API function calls.

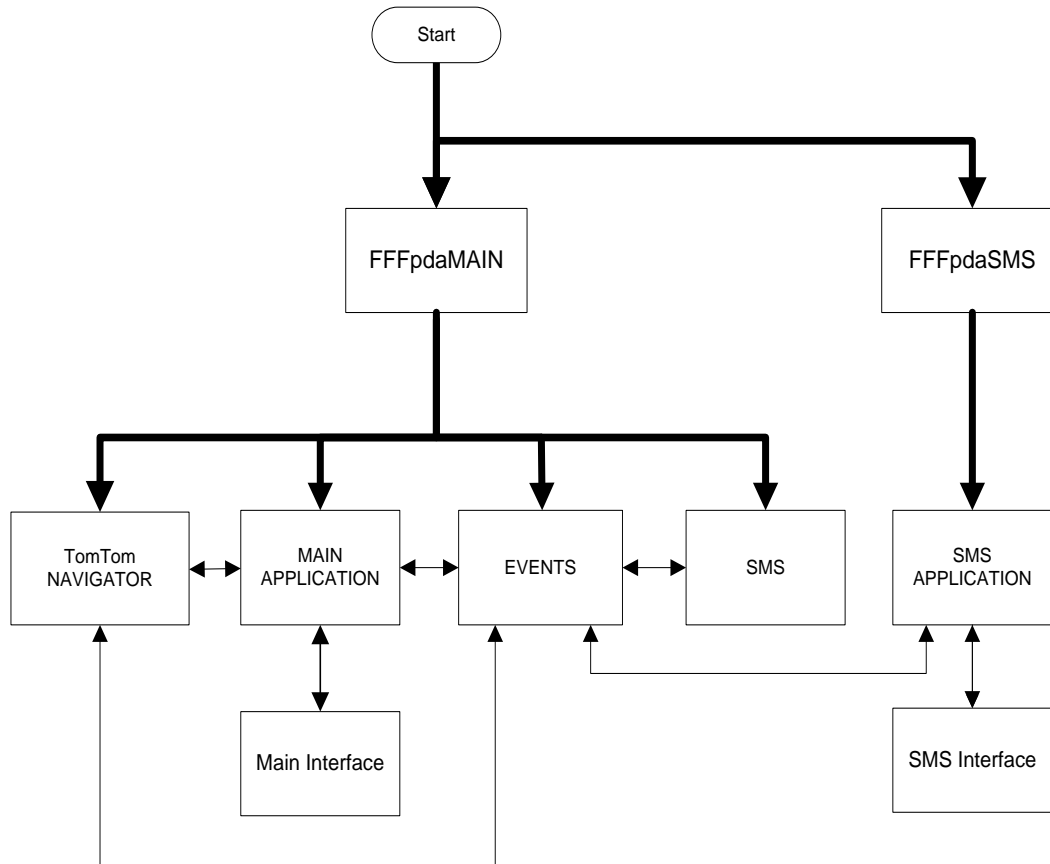


Fig 4 Global system architecture

As Figure 4 describes, two applications are deployed on the OS boot-up: FFFpdaMAIN and FFFpdaSMS. FFFpdaMAIN is responsible for the launching of several events: TTN, the auxiliary Events and SMS Threads and its own interface. FFFpdaSMS on the other hand only launches its respective interface.

TTN communicates with the Main Thread in order to translate the hardware buttons requests into actions on the navigator itself. The SMS Thread is constantly checking for incoming or outgoing SMS messages. The appropriate conditioning is done for the messages that are received and passed to the Events Thread and also to the messages that the Event Thread requests to be sent to the server. The fundamental interaction with the Main thread is also addressed for logging aspects and also establishing a connection with the other running applications.

The Main class can be described as a Global System Manager in terms of its importance on all the events that occur: interconnecting threads through message routing, hardware buttons handling, logging all the events that occurred, system robustness control and also managing its own interface (Figure 1).

The FFFpdaSMS Application mainly handles the interface (Figure 2) which consists of an alphanumeric keypad and some Text boxes for visualization purposes.

The necessity for reporting, tracking and auditing capabilities is a fundamental requisite of any software application. It provides the administrator or manufacturer the accountably tools on which to maintain a list of all actions performed by the system. In terms of reporting, these actions range from: logging the Fire Fighting Events sent and received by the device, as well as logging all the GSM communication performed transparently by the system. The previous reporting characteristic by itself provides enough data in terms of knowing what and when an action is being done.

Concerning robustness, the system was developed to ensure that only the intended actions desired by user should be performed. The applications TTN interface, FFFpdaMain and FFFpdaSMS, appear on a Full Screen size. The only way to change that is by entering the administrator mode. Although all SMS received messages come from the server correctly formatted, an event failsafe was implemented upon reception. In case of an invalid event a system message is sent to the server as an example of an invalid Fire event: “Bad Fire Event”, and so forth with all other events.

4.2. SMS Protocol

The SMS protocol was designed with an important guideline: maximization of the amount of information *per* SMS. This means that each message event type has the most amount of specific information; and also in each SMS a series of complete events can be sent until the 160 characters limit is reached. Taken in account the aforementioned facts, the following paragraphs describe each of the fields of a multi event SMS message and also every single event type possible. As a design guideline it was decided that either Server or User controlling the PDA will only send completed events in each SMS message, separated with the character “#”.

A Fire event already coded is shown as an example (Figure 5).

```
#Fire
Type (11)
Name (“Paris”)
Active ( 1 - ”yes”, 0 – “no”)
Coordinates (Two Sets Of 6/7 Digit Numbers): Longitude, Latitude

Ex: 3333333;11;Paris;1; 236370;4887280
```

Fig 5 Example of Fire event in coded SMS protocol

Each event type has several fields, separated by semicolons with the first two entries on each single event being: a unique global ID attached to each given event (for univocal and tracking purposes), followed by the respective SMS type code. The remaining fields on each message are described on Table 1.

Table 1

Summary of SMS protocol

Message Fields	SMS type				
	Fire	Weather	Auto	Water	Human
Code	11	22	33	44	55
ID(*)	ID(*)	ID(*)	ID(*)	ID(*)	ID(*)
Name	Event or Location	Station	Number	Supply	User
Enable	ON/OFF		ON/OFF		
GPS Coordinates	Longitude	Longitude	Longitude	Longitude	
	Latitude	Latitude	Latitude	Latitude	
Information	Temperature				
	Humidity				
	Pressure				
	Wind Speed				
	Wind Orientation				
	Pluviosity				
Auto type			Jeep(0)		
			Truck(1)		
State			Water Level	Water Level	
Receiver type			Commander(0)		
			All Personnel(1)		
			Received(3)		
Text			Free		

Note: (*) This message field value is generated from an algorithm, for each event being sent from the system. The backoffice server generates the value following the same algorithm for each event sent to the system.

5. Performance Analysis

This system, in its commercial version will interact with the existent server for the FFF. However the system was only tested in simulated Forest Fire.

In order to validate the performance and usability of the developed system two types of tests were performed. A Road-Book was designed to test the system in a real world situation, in terms of reaction and response time, robustness and GPS signal conditions change. The second performed test concerned the usability and ergonomics of the software application from the point of view of the firefighter. In order to have some degree of feedback from the users of the system, a small questionnaire was devised in order to gauge the performance of the system.

5.1. Road-Book

This test was performed to demonstrate the use of the system mounted in a vehicle, taking part of a response team to an initial Fire detection occurrence. In order to showcase the features and robustness of the system, this Road-Book (Table 2) was designed to have all types of Events (Fire, Auto, Weather and Water) and all possible actions on those Events (New, Update and Removal). It also simulates a system crash and the automatic system recovery process that ensues.

On this test the system proved to be effective and responded to all the solicitations of the operator promptly. The TTN software rerouting capabilities responded well to course alterations. This test served the purpose of validating the performance of the system on a real situation with all the additional factors and considerations that need to be taken into account. The robustness of the system was also tested and proved to be reliable.

5.2. System Usability

This section concerns the results achieved by performing a questionnaire to a sample of layman people, upon a pre-requisite reading of the systems user manual. The evaluation system that was devised quantified the degree of difficulty to complete the task from Easy (value 1) to Hard (value 5). The universe consisted of five testers.

This section is divided into four parts. The first part tested the capabilities of the TT interface. The second part evaluated the performance of the custom Interfaces (FFFpdaMAIN and FFFpdaSMS). On the third part, assorted PDA actions were proposed to be performed. Finally, the fourth and last part concerns the overall appreciation level of the complete system.

Table 2

Road-Book guide

Test Time	Server Action	User Action	Observations
0:00		Power ON PDA	
0:01		Send Auto-Position Refresh(AUTO)	
0:02	Send FIRE Event(Fire#1)		NEW Event
0:03		Trace Route to Fire#1	
0:04	Send AUTO Event Send AUTO Event		NEW Event NEW Event
0:05		System Crash: Send SMS to Server (AUTO)	
0:06	Re-send previous events		
0:08	Send WATER Event Send HUMAN Event		NEW Event NEW Event
0:09	Send WEATHER Event Send HUMAN Event		NEW Event NEW Event
0:10		Send SMS message to Commander	
0:12	Send FIRE Event(Fire#2) Send AUTO Event		NEW Event UPDATE Event
0:13	Send FIRE Event Send WEATHER Event		REMOVE Event UPDATE Event
0:14		Trace Route to Fire#2	
0:15		Close/Power OFF PDA	

5.2.1. *TT Interface*. The devised questions for this group, concerned features related to actions performed on the TT interface, such as: reception of multiple events, route tracing, map browsing, and event visualization filtering.

Table 3

TT Interface related questions

Question	1(Easy)	2	3	4	5(Hard)
Confirm the reception of multiple events	80%	20%	0%	0%	0%
Trace a route to Water event	80%	0%	20%	0%	0%
Go to specific Fire event location	40%	40%	20%	0%	0%
Show/Hide Truck event	60%	40%	0%	0%	0%

5.2.2. *Custom Interfaces*. This section regards the analysis of the features present on the interfaces that were created for this system, FFFpdaMAIN and FFFpdaSMS. These features concern the SMS text message sending and reception operations, as well as the visualization of specific event information.

Table 4

Custom Interfaces related questions

Question	1(Easy)	2	3	4	5(Hard)
Confirm the reception of a SMS and reading of the last two received SMS messages	100%	0%	0%	0%	0%
See Weather event detailed information	80%	20%	0%	0%	0%
Send a SMS Message to “All”	100%	0%	0%	0%	0%

5.2.3. *Assorted Actions*. This section concerns assorted actions, specifically turning off and rebooting the PDA.

Table 5

Assorted Actions related questions

Question	1(Easy)	2	2,5	3	4	5(Hard)
Turn Off the PDA	80%	0%	20%	0%	0%	0%
Reboot the PDA	100%	0%	0%	0%	0%	0%

5.2.4. *Overall Appreciation.* The last section questioned the testers on a final appreciation of the overall system.

Table 6

Overall Appreciation related questions

Question	1(Bad)	2	3	4	4,5	5(Excellent)
Overall Appreciation	0%	0%	0%	20%	20%	60%

The second part of the tests, System Usability, was conducted on the controlled environment of the laboratory, due to the logistic impossibility of taking all the testers on individual trips. The results were positive and demonstrate the value of the system in terms of meeting the initial proposed objectives: an user-friendly, intuitive system that provides the necessary information in various formats, according to the needs of the end-user. From the analysis of the tests the TT interface did not have the intuitiveness and ease of use that the remainder of the system undoubtedly has. As previously explained, the proper customization to meet those standards was not possible due to limitations of the SDK.

6. Conclusions

The development of an user-friendly system that will help the Fire Fighting Services was effectively achieved. A system is available for the personnel in the field that provides a multitude of available and easily filtered information, specifically: fire related, water reservoir and meteorological station updates and from the deployed vehicle resources on the theater of operations.

In terms of a future work perspective the costs of each device take a more relevant role, so the use of a less expensive and easier to license, automotive navigation system becomes an important condition to take in consideration. An alternative software is Street™, by the Elektrobite™ Company. The key feature which differentiates it from the navigational software used is that the licensing fees *per* equipment are more cost effective, and it is also able to use either Navteq™ or Tele-Atlas™ maps which increases the versatility of the product.

The systems visual representation proved to be adequate and intuitive, as the results of the performance tests proved. The effectiveness of this action was fundamental in determent of a multiplicity of overproduced features, as the Fireman's attention should be kept to the essential and easy to access information. The text messaging high level protocol performed as intended, particularly in terms of the efficiency of the codification for each event. This decreased the overall financial cost and increased the useful information per message ratio.

Although this paper addresses a vehicle mounted PDA, the concept is easily translated in many mobile devices. A hand held industrial PDA, as Psion Teklogix™ Ikôn™ could be given to the chief of the Firemen with added Cellular Phone features as well as an on-board camera. With this the manager could give a terrain update of the fire in photographical terms to all the parties involved on the theater of operations.

7. References

1. Fischer, P., Fusco, L., & Brugnioni, G (2000), Integration of Portable Information Technology and GSM Based Communication Devices for Disaster Management Operations, available on <http://adsabs.harvard.edu/full/2000ESASP.458..177F>
2. Lymberopoulos N., Papadopoulos C., Stefanakis E., Pantalos N., Lockwood F.A, GIS -Based Forest Fire Management Information System
3. Xin Ye, Yanzhang Wang, Hui Li, Zailin Dai, Emergency Decision Support System Based on the General Decision Process, available on <http://www2.computer.org/portal/web/csdl/doi/10.1109/WIIAT.2008.173>
4. Fleet Management Solutions, Product Overview, available on <http://www.fmsgps.com/frontend/overview.aspx>
5. Fleet Management System, Wenco available on <http://www.wencomine.com/Real-Time.html>
6. "DL0G X7™ technical data" available on <http://www.dlog.com/en/products/7-industrial-computer/dlog-x-7/x-7-technical-data.html>