

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**INTERNETWORKING: THE INTEROPERABILITY OF
COMMERCIAL MOBILE COMPUTERS WITH THE USMC
DIGITAL AUTOMATED COMMUNICATIONS TERMINAL (DACT)**

by

James C. Cummiskey

September 1996

Thesis Advisor:
Second Reader:

Don Brutzman
Lou Stevens

Approved for public release; distribution is unlimited.

19970311 006

DTIC QUALITY INSPECTED 1

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 1996		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE INTERNETWORKING: THE INTEROPERABILITY OF COMMERCIAL MOBILE COMPUTERS WITH THE USMC DIGITAL AUTOMATED COMMUNICATIONS TERMINAL (DACT)			5. FUNDING NUMBERS	
6. AUTHOR(S) Cummiskey, James, C.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
The United States Marine Corps has begun to develop a system called the Digital Automated Communications Terminal (DACT). The DACT system is based around a subnotebook-sized, tactical input/output battlefield situational awareness system and communications terminal. The problem is that DACT's excessive weight, size, cost and complexity might ultimately prevent its successful integration into the rapidly evolving Marine Corps style of maneuver warfare. This thesis provides a study of palmtop-sized mobile computing platforms to include the Hewlett-Packard family of palmtops, as well as the emerging Microsoft Pegasus mobile operating system. Furthermore, various messaging standards, protocols and commercial digital transmission channels are analyzed for their suitability to DACT requirements. Finally, a system prototype called the "Rapid Electronic Delivery of Messages over Asynchronous Networks" (REDMAN) is implemented to disseminate field orders under combat conditions. REDMAN speeds the flow of accurate information to all levels of command within a Marine infantry battalion using a commercial palmtop platform. This commercial palmtop is 3-7 times lighter and 20-30 times less expensive than DACT. Wireless networked palmtop computing will completely change the scope of Marine warfighting. This thesis provides a proof of concept system that demonstrates such fundamental change is feasible today.				
14. SUBJECT TERMS Digital Automated Communication Terminal (DACT); Mobile Computing; Palmtop; Commercial Communication Infrastructure; Joint Variable Message Format			15. NUMBER OF PAGES 333	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited

**INTERNETWORKING: THE INTEROPERABILITY OF
COMMERCIAL MOBILE COMPUTERS WITH THE USMC
DIGITAL AUTOMATED COMMUNICATIONS TERMINAL (DACT)**

James C. Cummiskey
Major, United States Marine Corps
B.S., Computer Science, California State University, Dominguez Hills, 1987

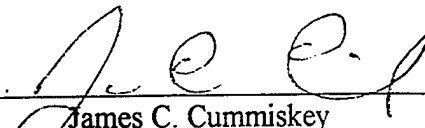
Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

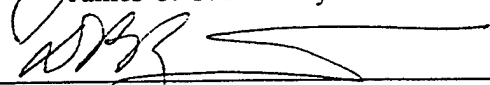
from the

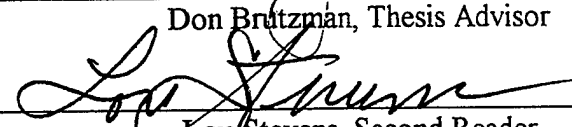
**NAVAL POSTGRADUATE SCHOOL
September 1996**

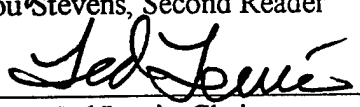
Author: _____


James C. Cummiskey

Approved by: _____


Don Brutzman, Thesis Advisor


Lou Stevens, Second Reader


Ted Lewis, Chair
Department of Computer Science

ABSTRACT

The United States Marine Corps has begun to develop a system called the Digital Automated Communications Terminal (DACT). The DACT system is based around a subnotebook-sized, tactical input/output battlefield situational awareness system and communications terminal. The problem is that DACT's excessive weight, size, cost and complexity might ultimately prevent its successful integration into the rapidly evolving Marine Corps style of maneuver warfare.

This thesis provides a study of palmtop-sized mobile computing platforms to include the Hewlett-Packard family of palmtops, as well as the emerging Microsoft Pegasus mobile operating system. Furthermore, various messaging standards, protocols and commercial digital transmission channels are analyzed for their suitability to DACT requirements. Finally, a system prototype called the "Rapid Electronic Delivery of Messages over Asynchronous Networks" (REDMAN) is implemented to disseminate field orders under combat conditions. REDMAN speeds the flow of accurate information to all levels of command within a Marine infantry battalion using a commercial palmtop platform. This commercial palmtop is 3-7 times lighter and 20-30 times less expensive than DACT. Wireless networked palmtop computing will completely change the scope of Marine warfighting. This thesis provides a proof of concept system that demonstrates such fundamental change is feasible today.

TABLE OF CONTENTS

I. INTRODUCTION	1
A. MOTIVATION.....	1
B. OBJECTIVES.....	1
C. ORGANIZATION OF THE STUDY.....	2
II. BACKGROUND AND RELATED WORK	5
A. INTRODUCTION	5
B. BACKGROUND.....	7
1. Mobile Computing Applications in the U.S. Military	8
2. Why Mobile Computers in the Military?	9
3. Existing Mobile Computing Military Programs.....	11
4. Reorientation of the Information Flow in Mobile Systems	13
5. Downsizing Today's Military Mobile Computing Plans	15
C. RELATED WORK	15
1. Ongoing Research into the Challenges of Mobile Computing.....	15
2. Related NPS Internetworking and Mobile Computing Research	17
D. SUMMARY.....	17
III. THE DIGITAL AUTOMATED COMMUNICATION TERMINAL (DACT)	19
A. INTRODUCTION	19
1. Shortcomings of the DACT System.....	19
B. CONCEPT OF EMPLOYMENT.....	22
1. Overview	22
2. Method of Employment.....	24
3. Mission Effectiveness Criteria.....	26
C. DACT SPECIFICATIONS.....	28
1. Introduction	28
2. Detailed Specifications.....	29
D. SUMMARY.....	30
IV. COMMUNICATION PROTOCOLS AND STANDARDS IN DISTRIBUTED MILITARY NETWORKS.....	31
A. INTRODUCTION	31
B. JOINT VARIABLE MESSAGE FORMAT (VMF) MESSAGES.....	31
1. Overview of Joint VMF.....	31

2. The Joint VMF Field Orders Message Format	33
a. Transmitting Combat Orders with Joint VMF	33
b. Bit Orientation of Joint VMF	34
c. Bit Error Rates and Joint VMF	35
d. Lack of Flexibility in Joint VMF	35
e. Packet Sizes of Combat Order Messages Joint VMF	37
f. Other Joint VMF Design Flaws	38
3. Joint VMF Alternatives	39
a. The Risks of Imposing Artificial Constraints on Message Content	39
b. Traditional Industry Standard Data Structures and HTML	39
C. FORWARD ERROR CORRECTION (FEC)	41
1. Overview	41
2. Reed Solomon Algorithm	42
D. MOBILE INTERNET PROTOCOL (MOBILE IP)	43
E. IEEE 802.11 WIRELESS LAN AND INTER-ACCESS POINT PROTOCOLS	43
1. Overview	43
2. The Quest for Interoperability	44
F. MIL-STD-188-220A WIRELESS COMMUNICATION PROTOCOL	45
1. Introduction	45
2. Flow Control	45
3. Open Systems Interconnection (OSI) Model	46
4. Forward Error Correction (FEC)	46
5. Communications Security (COMSEC)	47
6. Intranet Topology Updates	48
7. Net Access Control	48
8. MIL-STD-188-220A Summary	49
G. SUMMARY	50
V. COMMERCIAL CONNECTIVITY TECHNOLOGIES FOR MOBILE COMPUTING	51
A. INTRODUCTION	51
B. MILITARY APPLICATION OF COMMERCIAL COMMUNICATION SYSTEMS	51
C. WIRED COMMUNICATION CHANNELS	53
1. RS-232	53
2. Universal Serial Bus	53
3. IEEE 1394 FireWire	53
4. Ethernet Wired LANs	54
D. WIRELESS ETHERNET LANS	54
1. Overview	54
2. AIRLAN Wireless Bridges	56
F. INFRARED	57
1. Overview	57
2. Ongoing Research	57
3. Commercial Infrared Connectivity	58

G. PACKET RADIO WIDE AREA NETWORKS (WAN)	58
1. Advanced Radio Data Information Service (ARDIS)	58
2. RAM Mobile Data	59
3. Metricom and the Ricochet Wireless Network Service	59
H. PAGING AND NARROW-BAND PCS	60
1. Data Critical Corporation	61
2. Motorola Advisor Pager	63
3. Motorola Tango Pager	64
I. CELLULAR TELEPHONE SERVICES	65
1. History	65
2. Advanced Mobile Phone System (AMPS)	65
3. Multiple Access Techniques	66
4. Packet vs. Circuit Switched Cellular	68
5. Cellular Digital Packet Data (CDPD)	68
6. Personal Communication Services (PCS)	70
7. Global System for Mobile (GSM) Communications	70
8. Personal HandyPhone System (PHS)	70
9. Using a Mobile Computer with Circuit Switched Analog Cellular	71
J. SATELLITE COMMUNICATIONS (SATCOM)	73
1. International Maritime Satellite Organization (INMARSAT)	73
2. Qualcomm's OmniTRACs	73
3. American Mobile Satellite Corporation (AMSC) SKYCELL	73
4. Low Earth Orbit Satellites (LEOS)	74
a. The Iridium Project	74
b. Globalstar	75
c. Orbcomm	75
d. Teledesic	76
e. ICO Global Communications	76
5. Sky Station—Stratospheric Geostationary Dirigibles	77
K. SUMMARY	77
VI. THE HEWLETT PACKARD FAMILY OF PALMTOP COMPUTERS	79
A. INTRODUCTION	79
1. The HP95LX	79
2. The HP100LX and HP200LX	80
3. The OmniGo 100 and 700	81
4. Strengths of the HP 200LX Palmtop	83
B. THE ANATOMY OF THE 200LX	84
1. Hardware Features	84
2. Software Features	87
3. Connectivity with Desktop Platforms (Cochran 1996)	90
C. SHORTCOMINGS OF THE HP200LX	90
D. HP PALMTOP PROGRAMMING ENVIRONMENTS	91
1. HP200LX Built-In Programming Tools	91
2. General Programming Languages for the HP200LX	91
3. The HP Program Applications Library (PAL)	92

4. HP200-LX Database-Centric Programming.....	92
E. CRITICAL SOFTWARE FOR THE HP200LX	92
1. General Telecommunications Programs	93
2. Internet Direct Connection Software	93
3. HTML Viewing Software: HV	94
4. World Wide Web (WWW) Browsers.....	94
5. Internet Software Suites	94
a. Minnesota Internet User's Essential Tool (MINUET)	95
b. Net-Tamer.....	95
F. EXTENDING THE 200LX WITH ADD-ON HARDWARE	95
1. RAM and Clock Doubling Extensions.....	95
2. Parallel Port Adapters	95
3. Flash Memory Cards.....	96
4. External Floppy Drives and Hard Drives.....	96
5. Geographic Positioning System (GPS) Sensors.....	96
6. Wireless Packet Radio Modems.....	97
a. Megahertz Allpoints card	97
b. Motorola 100D Personal Messenger Card.....	98
c. EnBlock R/F PalmStation Plus	99
7. Pagers.....	100
8. Ethernet LAN Adapters	101
9. PC-Card Modems	101
10. Cryptographic Cards	102
11. PCMCIA Tactical Communications Interface Module (P-TCIM) Modem.....	102
G. MILITARY APPLICATIONS FOR THE HP200LX	103
1. The HP Palmtop in Action	103
2. Land Navigation and Target Engagement with the Grid Calculator.....	105
H. SUMMARY.....	108
VII. THE PEGASUS MACHINE AND OPERATING SYSTEM.....	109
A. INTRODUCTION	109
1. Non-Disclosure Agreement Specifications.....	109
2. The Pegasus OS: "32 bit Mobility with a Familiar User Interface".....	109
B. PEGASUS HARDWARE.....	111
1. Hardware Overview	111
2. Power Management	111
3. Memory	112
4. Pegasus Input Devices.....	113
5. Pegasus Output Devices	113
6. Pegasus Serial Ports	114
7. Pegasus Infrared Communications	114
8. Pegasus Hardware Expansion Slot.....	115
9. Hardware Preview Program.....	115
C. PEGASUS OPERATING SYSTEM.....	115
1. Software Overview.....	115
2. Operating System Components	116
a. Kernel	116

b. Windows and Event Managers	116
c. Graphical Device Interface (GDI)	117
d. File system, Registry and Database.....	117
3. Characteristics of PEG/OS	117
4. Object Store	118
a. System Registry API.....	118
b. File System API.....	118
c. Database System API.....	118
d. Remote Procedure Call (RPC) API	119
5. Memory Management	119
6. Peg/OS Messages.....	120
7. Future Directions for Peg/OS	121
 D. MILITARY USE OF PEGASUS INTERNAL APPLICATIONS	122
1. Word Processing, Spreadsheet and Personal Information Management	122
2. Pegasus Mail (PMail).....	123
3. Pegasus Internet Explorer (PIE)	123
4. Pegasus Help.....	124
 E. PROGRAMMING FOR THE PEG/OS	124
1. Introduction.....	124
2. Pegasus SDK and DDK Utilities	125
3. Pegasus Programming Fundamentals	125
 F. SUMMARY	126
 VIII. THE RAPID ELECTRONIC DELIVERY OF MESSAGES OVER ASYNCHRONOUS NETWORKS (REDMAN)	129
A. INTRODUCTION	129
B. WHY REDMAN?	132
C. REDMAN COMBAT ORDER PROCESSING SUB-SYSTEM	133
1. Defaults/Options/Baseline T/O Module	134
2. Incoming Combat Order Viewing Module.....	134
3. Outgoing Combat Order Generation Module.....	135
4. Combat Order Communication Module.....	135
D. COMMUNICATION CHANNELS USED IN REDMAN/REDWEB.....	135
1. REDMAN File Transfer Connectivity	135
2. REDWEB HTML Connectivity	138
E. THE USMC OPERATIONS ORDER FORMAT	139
F. REDMAN DATA MODEL	140
1. Introduction	140
2. ORDERS.DB Table	142
3. TO-<n>.DB Table.....	144
4. TO-BASE.DB Table	145
5. MISSIONS.DB Table.....	146
6. MSN-AREA.DB Table.....	146
7. CHECKLST.DB Table.....	146
8. ELEMENTS.DB Table	147

9. OUT-ORD.DB Table	147
10. TO-EMPTY.DB Table	148
G. CRITICAL RAD DESIGN DECISIONS OF THE REDMAN APPLICATION	148
H. SUMMARY	149
IX. REDMAN IMPLEMENTATION USING RAPID APPLICATION DEVELOPMENT (RAD)	
TOOLS	151
A. INTRODUCTION	151
1. REDMAN, REDWEB and Web Server Interactivity	151
B. DELPHI - THE 16 AND 32-BIT RAD DEVELOPMENT ENVIRONMENT	152
1. What is Delphi?	152
2. The Borland Database Engine (BDE)	153
C. WEBHUB HTML APPLICATION FRAMEWORK	155
1. Introduction	155
2. Benefits of the Webhub Framework	156
a. Saving State	156
b. Process Control	156
c. Integration with Delphi Debugging Environment	156
d. Easy Site Maintenance and Flexibility	157
e. Scalability and High Performance	157
3. WebHub Connectivity and Data Flows	157
4. WebHub Uniform Resource Locator (URL) Syntax	158
D. COMMERCIAL DELPHI COMPONENTS USED WITHIN REDMAN	159
1. Maelstrom TDBOutline Component	159
2. Xceed Zip Compression Library	160
3. TcsNotebook Tabbed Notebook Component	162
4. Infopower	163
5. WebHub & Tpack Components	164
E. REDMAN ORDER PROCESSING SYSTEM FUNCTIONAL OVERVIEW	165
1. Introduction	165
2. REDMAN Combat Order Processing	166
F. WEBSITE WEB SERVER	177
G. REDWEB IMPLEMENTATION	178
1. Introduction	178
2. RedWeb HTML Order Generation	181
3. Format of the WebApp INI file	190
H. REDWEB HTML PROCESSING SYSTEM FUNCTIONAL HIGHLIGHTS	192
1. Converting a Limited Level Recursive Task Organization to an Unordered HTML list	192
I. SUMMARY	194
X. CONCLUSIONS AND FUTURE WORK	197

A. CONCLUSIONS.....	197
1. Introduction.....	197
2. Where to Go from Here.....	198
3. Summary.....	199
B. RECOMMENDATIONS FOR FUTURE WORK.....	199
1. Software Enhancements.....	200
a. REDMAN Order Processing Software.....	200
b. RedWeb HTML Processing Software.....	201
c. Palmtop Grid Calculation Software.....	202
2. Additional Research Areas.....	202
APPENDIX A. MESSAGES PLANNED FOR IMPLEMENTATION IN THE DACT.....	205
APPENDIX B. AN EXECUTIVE OVERVIEW OF THE DACT.....	209
APPENDIX C. SAMPLE USMC INFANTRY BATTALION COMBAT ORDER.....	213
APPENDIX D. SOURCE CODE FOR THE GRIDCALC APPLICATION.....	219
APPENDIX E. SOURCE CODE FOR THE REDMAN APPLICATION.....	237
APPENDIX F. REDWEB HTML AND SOURCE FOR THE SAMPLE ORDER.....	273
APPENDIX G. SOURCE CODE AND MISCELLANEOUS INI AND HTML FILES FOR THE REDWEB APPLICATION.....	285
REFERENCES.....	299
INITIAL DISTRIBUTION LIST.....	307

LIST OF FIGURES

3.1. DACT Employment	25
3.2. Early DACT Prototype	29
5.1. PortLAN Systems	55
5.2. Metricom Street Lamp PoleTop Radio	60
5.3. Data Critical Critical Link Connectivity	62
5.4. Motorola Advisor Pager	63
5.5. SkyTel Tango 2-Way Pager	64
6.1. HP200LX Palmtop	80
6.2. HP Omnigo 100	81
6.3. HP Omnigo 700	83
6.4. Trimble Mobile GPS Locator 110	97
6.5. Megahertz AllPoints Card	98
6.6. Motorola Personal Messenger 100D	99
6.7. EnBloc R/F PalmStationPlus	100
6.8. Data Critical Critical Link Connectivity	100
6.9. EtherLink III V.34 Modem / Ethernet LAN Card	101
6.10. Allied Signal Fortezza Plus/LP Crypto Card	102
6.11. HP Palmtops in Kuwait	103
6.12. GridCalc 2.0 Opening Screen	105
6.13. GridCalc 2.0 Range/Azimuth Menu Selection Screen	106
6.14. GridCalc 2.0 Range/Azimuth Data Entry SubForm Screen	107
6.15. GridCalc 2.0 Range/Azimuth Calculation Results Screen	108
8.1. DACT/Palmtop Combat Information Flow	130
8.2. REDMAN Combat Order Processing System Modules	134
8.3. REDMAN File Transfer Connectivity	137
9.1. REDMAN, REDWEB, Web Server Interaction	152
9.2. Borland Database Engine Connectivity	154
9.3. WebHub Connectivity and Data Flows	158
9.4. TDBOutline Recursive Delete Code Modules	160
9.5. Xceed Outgoing Order Compression Code Module	161
9.6. TDBOutline Tabbed Notebook Component Screen	162
9.7. MDI Child Form Pointer Handling Code Modules	164
9.8. REDMAN Order Processing System Interactivity	165
9.9. REDMAN Task Organization Screen	166
9.10. REDMAN Nested Task Organization Screen	167
9.11. REDMAN Defaults/Options T/O Screen	168
9.12. REDMAN Execution Commander's Intent Screen	169
9.13. REDMAN Outgoing Order Generation Menu Screen	170
9.14. REDMAN Outgoing Order General Info Screen	171
9.15. REDMAN Outgoing Order T/O Screen	172
9.16. REDMAN Outgoing Order Mission Screen	173
9.17. REDMAN Order Checklist Items Screen	174
9.18. REDMAN Order Elements Screen	174
9.19. REDMAN Order Elements Screen	175
9.20. REDMAN Transmit/Receive Screen	176
9.21. WebSite WWW Server Properties Interface Screen	177
9.22. RedWeb Connectivity and Data Flows	178
9.23. 16-bit WebHub Connected Status Screen	179
9.24. RedWeb Application Page Test Screen	180
9.25. Netscape Web Browser with RedWeb HomePage Screen	181

9.26. RedWeb Order Selection Screen	182
9.27. RedWeb Unit Selection Screen	183
9.28. RedWeb Unit Selection Scrolling Screen	184
9.29. RedWeb Invalid Password Screen.....	185
9.30. RedWeb HTML Operation Order Screen	186
9.31. RedWeb HTML Task Organization Screen.....	187
9.32. RedWeb HTML Tasks Screen.....	188
9.33. RedWeb HTML Command and Signal Screen.....	189

LIST OF TABLES

4.1. Extract from K05.15 Field Orders Message Format	34
4.2. File Sizes for Typical Field Orders Message Packets.....	37
5.1. Cellular Technology Comparison	68
6.1. HP200LX Serial Port Pin Outs to RS-232	86
7.1. Peg/OS Thread Priority Levels	117
8.1. REDMAN Communication Channel Theoretical Throughputs	138
8.2. USMC 5-Paragraph Order Format.....	140
8.3. REDMAN Data Model and Table Composition (Part 1/2)	141
8.3. REDMAN Data Model and Table Composition (Part 2/2)	142
8.4. REDMAN Fields in Data Model.....	143
8.5. REDMAN USMC 5 Paragraph Order Fields	144
8.6. T/O Table Data Schema	145
8.7. Missions Table Data Schema.....	146
8.8. Mission Area Table Data Schema.....	146
8.9. Checklist Item Table Data Schema.....	147
8.10. Order Elements Table Data Schema	147
10.1. Uses of Mobile Computers in Combat and Garrison.....	198

ACKNOWLEDGMENTS

The author gratefully acknowledges the financial and technical support of the Marine Corps Systems Command (PM-CIS) and the Communications Systems Division of the Marine Corps Tactical Systems Support Activity. Acknowledgment is also due the Microsoft Corporation for their support of the author as an independent software developer of their emerging Pegasus operating system. Readers should note that at the time of this writing, the contents of Chapter VII are currently protected under a Non-Disclosure Agreement (NDA) (see the pertinent chapter for details).

This research is dedicated to Tawnya, whose love, patience and understanding helped me stay mobile during this arduous project. Semper Mobilis!

I. INTRODUCTION

A. MOTIVATION

The future of mobile computing is inextricably intertwined with the future of our military forces. As a component part of the Department of Defense's digitization of the battlefield initiative, the United States Marine Corps has begun to develop a system called the Digital Automated Communications Terminal (DACT). The DACT system is based around a subnotebook-sized, tactical input/output battlefield situational awareness system and communications terminal. However the DACT program has some shortcomings that might ultimately prevent its successful integration into the rapidly evolving Marine Corps style of maneuver warfare. Foremost among these are the expense and weight of the DACT due to the misrepresentation of the DACT as a primary battlefield sensor. Mobile computers should not be considered as sensors for input to the Global Command and Control System (GCCS) network, but rather as a means of helping each Marine accomplish his individual mission.

This thesis investigates mobile computing and wireless networking in three ways. First, it provides a study of selected mobile computing hardware platforms in both the commercial and military domains. Secondly, this research analyzes various messaging standards, protocols and digital transmission channels and their applicability to military communication requirements. Finally, a system prototype is developed to support the dissemination of combat messages via one or more of these selected transmission channels. This research demonstrates that both current and future-generation palmtop-sized commercial computers can be affordably integrated into the emerging DACT system. The use of hypertext transfer protocol servers in conjunction with back-end databases will be the foundation of military data communication on tomorrow's battlefields. Wireless networked palmtop computing will completely change the scope of Marine warfighting.

B. OBJECTIVES

The primary objective of this study is to demonstrate a viable end-to-end solution to mobile battlefield computing within the framework of the existing Marine Corps DACT program. This research is conducted on a conceptual foundation of cost, ease of use, and suitability to task. The dissemination of combat orders often dictates a fundamentally

different approach than the one proposed by the current vision of DACT. For example, subordinate commanders and key personnel are often called forward to "huddle up" in a confined area when combat orders are disseminated. At the battalion level, these confined areas can sometimes offer shelter from the elements and artificial lighting (such as that provided in a tented Command Post). Hence, the analysis presented here will not simply confine itself to long-haul transmission mediums such as combat net radio. Rather, this thesis examines a wide-range of diverse commercial technologies and their application to military mobile applications.

This thesis demonstrates that both current and future-generation palmtop-sized mobile computers--working in conjunction with both military and commercial communication infrastructures--can be successfully integrated into the emerging DACT system. Mobile networked computing can provide dramatic new warfighting capabilities for the U.S. Marine Corps. Integrating low-cost high-performance internetworked palmtop computers to the fireteam level is now technically and economically feasible.

C. ORGANIZATION OF THE STUDY

Chapter II offers the reader a brief introduction to background work in mobile computing and wireless networking, including its applicability to the unique requirements of the U.S. military.

Chapter III discusses specific strengths and shortcomings of the DACT program and presents the emerging concept of employment for this critical system.

Chapter IV examines some of the more significant communication and data standards and protocols and analyzes their applicability towards the vision of integrating and internetworking highly mobile computers on the battlefield. The Joint Variable Message Format (VMF) standard, in particular, is analyzed in detail resulting in the conclusion that there are several significant problems with its architecture that can be resolved with established international commercial standards.

Chapter V compares and contrasts existing commercial communication technologies applicable to mobile computing, as well as discusses emerging communication channels that represent the future of military mobile computing applications.

Chapter VI focuses on the Hewlett-Packard family of palmtop computers and examines specific MS-DOS and HP-specific developmental tools used in building mobile computing software for military applications.

Chapter VII examines an as-yet-unannounced 32-bit operating system (OS) under development by the Microsoft Corporation for the emerging hand-held Pegasus device. The Pegasus OS is based on a subset of the Win32 API and consists of a Microsoft Windows operating system optimized for mobile palmtop platforms, a shell, and a suite of personal information manager (PIM) applications. The entire system is designed to run on a 100 MIPS pocket-sized mobile computer weighing under one pound.

Chapter VIII presents the proof-of-concept application called the Rapid Electronic Delivery of Messages over Asynchronous Networks (REDMAN)—a highly automated mobile computing suite of applications for transmitting digital messages among military units on the joint battlefield. Specifically, the chapter focuses on the system design regarding the transmission of combat order message packets from the U.S. Marine Corps infantry battalion level to the fireteam level via a wide range of digital communication channels.

Chapter IX details the implementation of a working prototype of selected GUI-based REDMAN applications based on the system design of Chapter VII. These applications include numerous ease-of-use enhancements to allow each level of command to rapidly assimilate the higher level commander's directive; select a course of action; and format and transmit the new order to his subordinate commanders.

Finally, Chapter X summarizes the conclusions of this thesis research, and offers a road map for future work for REDMAN and other important research areas in the dynamic field of military mobile computing.

II. BACKGROUND AND RELATED WORK

A. INTRODUCTION

The world is poised at the beginning of yet another sweeping computer revolution. Most observers would agree that the computing scene of the 1970's was dominated by stand-alone mainframe computers working in conjunction with time-sharing terminals. These "dumb terminals" provided users with remote access to centralized databases and applications running on the mainframe. During the decade of the 1980's, users became empowered with the flexibility of the personal computer (PC). PCs worked very well in conjunction with decentralized applications and databases residing on these powerful standalone work stations. The first half of the 1990's saw the advent of multi-tiered client/server technology and the emergence of high performance networked PCs. Now we find ourselves in the latter half of this decade clearly dominated by global internetworking via the Internet; the phenomenon of the World Wide Web; and the appearance of network-mobile Java byte-code applications.

Where do we go from here? Since 1895, when Marconi first introduced us to radio, the promise of truly mobile, unwired communications has been the driving force for continuous research in radio wave propagation (Duff, 1996). Miniaturization has been the theme of the last several decades as computers embrace the principles of "smaller, faster and cheaper." Notebook-sized computers are already powerful enough to become the full-featured desktop computer of yesterday. Palmtop-sized and smaller devices are quickly assuming the role of today's notebook computers. Hence, the first decade of the new millennium will undoubtedly involve ubiquitous, widely distributed mobile computers. As our world makes the transition to the promise of Nicholas Negroponte's "digitally communicating cufflinks" (Negroponte, 1995), mobile computing is destined to become foremost in the minds of our brightest engineers. As Alan Kay (Kay, 1996) has said, "the computer is about to leap off the desk and merge with the telephone to become part of our clothing." This is not so preposterous if we examine the research being conducted at Carnegie Mellon's Engineering Design Research Center where wearable computers are being routinely fabricated and used (Finger, 1996).

Clearly today's mobile computers are far from perfect. They have numerous shortcomings including hard-to-read screens, diminutive keyboards, limited processing

power and relatively short battery life. However, the utility of carrying a device more powerful than the revolutionary original IBM PC in one's coat pocket more than makes up for these deficiencies in the eyes of many users. User satisfaction is improving constantly as pen-based data entry is becoming the standard means of providing input to our mobile devices. Powerful palmtop-sized computers with intuitive graphic user interfaces (GUIs) are already on the market which accommodate even more sophisticated, easy-to-use applications. With the advent of RISC-based processors such as the StrongArm SA-110 (a 32-bit RISC chip operating up to 200mhz with internal MPEG-2 video circuitry and internal V.34 soft modem--all operating on a single pair of AA batteries), the promise of robust voice recognition becomes feasible (Bajarin, 1996). Similar advances are being made in screen technologies as aggressive research is being conducted into alternatives to the traditional means of displaying two-dimensional visual output to the user. Meanwhile, technological innovations in wireless networking and data communication exponentially increase the promise of mobile computing. Electronic mail promises to be the "killer app" of mobile computing for many busy professionals. The world is about to discover the allure of wireless distributed computers providing "any time—anywhere" access to an increasingly mobile work force.

Or, is it? After all, bandwidth to these devices is still severely limited. As the pundit Bob Metcalfe has stated, "Wireless mobile computers will eventually be as common as today's pipeless mobile bathrooms. Port-a-potties are found on planes and boats, at construction sites, rock concerts and other places where running pipes is very inconvenient. But bathrooms are still predominantly plumbed. For more or less the same reasons, computers will stay wired." (Frezza, 1995)

Notwithstanding Metcalfe's amusing analogy, mobile computing appears to be here to stay. After all, mankind always seems to be able to conquer what it sets out to do. Mobile computing will happen because we want it to. Users simply do not want to be locked to a single location anymore. Cutting the tether to one's desk and becoming "unwired" reflects the desire of the human spirit to retain its natural mobility. At the same time, wireless computing is not a panacea. Wireless will almost certainly never completely replace wire owing to the physical limitations of the bandwidth of the electromagnetic spectrum. In a wired world, bandwidth is essentially unlimited--if one needs more bandwidth, one simply adds another fiber or cable. Wireless communication channels, however, can be used in many places where wire cannot be physically or economically applied.

Thus mobile computing is essentially about convenience. Computing will become completely transparent to mobile users as they move around from one environment to another. Correspondingly, one of the challenges of mobile computing is making the environment aware of the user to allow each application to adjust itself to the needs of that particular person. These factors include the user's location, the devices they are using and communicating with, the bandwidth available, and whether or not the user is in motion. Rapidly changing connectivity and latency characteristics of a given communication path will become the norm. Whenever and wherever we move around, the underlying system must always know who we are, where we are, and what services we need. In traditional models of computing, users have always been aware of their environment--now, the networked environment must become aware of us.

B. BACKGROUND

Mobile computing has been defined as "the system support and application support for self-powered, hand-held, mobile computers with wireless connections." (Imielinski, 1995) Leonard Kleinrock prefers the term "nomadicity" and defines it as "providing a rich set of computing and communication capabilities and services to nomads as they move from place to place in a transparent, integrated and convenient form." (Kleinrock, 1996) Kleinrock claims, however, that it is not just about being untethered or unplugged—rather it is about the development of a robust infrastructure that is available to the nomad irrespective of his location. Complete connectivity of these devices using a massively distributed architecture while always exploiting frequency reuse is the ideal infrastructure for maximizing the utility of mobile computers. Nomadic computing builds on the premise that most users are nomads. Any infrastructure that permits nomadic computing must be intelligent enough to know where a user is, the user's identity, and the kinds of services that the user requires. (Kirvan, 1996)

Mobile computing can be particularly confusing to researchers. Many scientists erroneously conclude that mobile computing is simply another term for distributed computing due to many similarities between the two fields. However, this is not the case. Mobile computing is distinctly different from distributed computing. These differences include:

- Limited power availability, memory and CPU processing power
- Widely changing bandwidth

- Different platforms
- Requirement for ad hoc services
- Requirement for handoffs between cells due to the movement of users
- Variance in location and number of users

Hence the environment in which mobile computing users must operate is essentially unpredictable. The ideal mobile computing framework provides transmission locations, signal strengths and costs to the application layer in the system's network stack. Hence, common network architectures may need to be revised to provide this independence between the applications and the network layer in order to fully realize the promise of mobile computing. Furthermore, systems engineers must consider that quality of service constraints may dictate different paths in an application program. Since wireless channels typically have limited bandwidth and high bit-error rates (BERs), applications need to constantly adapt themselves to fluctuating conditions. For example, an optimized World Wide Web (WWW) browser may sense that a user is currently using a low-speed, error-prone communication link, and automatically not display inline graphic images to increase performance and user satisfaction. Naturally, we also need to focus on the interoperation between wireless and wired networks, and build systems which allow a graceful navigation of this extraordinarily unpredictable environment.

1. Mobile Computing Applications in the U.S. Military

Who is more mobile than a member of the U.S. military forces? The potential of mobile computing offers unparalleled advances in the military's ability to prosecute wars. As our American military embraces the tenets of our acclaimed "Revolution in Military Affairs (RMA)" (Kelly, 1993), the promise of mobile computing offers a road map towards success on the battlefields of tomorrow. Mobile computing in military terms is about the flow of real-time information to maintain synchronization and a common picture of the battle, and to stay within the decision cycle of the enemy to achieve battlespace dominance.

However, traditional methodologies of acquiring and developing automated information systems (AISs) in the military have not always been very successful. The military relies on a system development methodology which just doesn't seem to work in today's rapidly changing technological landscape. If history can be the foreteller of the future, military mobile computing systems will most likely be fielded late; have a simply

terrible user interface with lots of undiscovered flaws in the software; and be quickly rendered obsolete and superseded by another emerging technology. As the CEO of Silicon Graphics said, "The equipment used in today's active U.S. military is reminiscent of a technology museum." (McCracken, 1996) The military's mobile computing programs are laudable, but seem destined for only marginal success due to the inherent flaws of our systems acquisition process. The military should refocus its efforts on rapid application development (RAD) and non-traditional models of system development. We need to lose our preoccupation with MIL-SPEC and focus on the "realm of the possible" emphasizing the utility of commercial de facto standards and protocols. In particular, the use of civilian telecommunications infrastructure for military operations must become the rule, rather than the exception.

At bottom, mobile computers add significant value to a highly mobile armed forces. Due to the aforementioned advances in consumer mobile computers, we now have the economic and technological capability to fundamentally improve virtually every facet of the leadership and management skills necessary for success as a military leader. However, mobile computers will only achieve their potential when the device itself is truly mobile (and we can provide our users with a flexible and robust communication infrastructure to transmit their message packets). Therefore, highly portable battery-powered computers, communicating wireless whenever necessary, are the obvious solution for an ever-mobile Marine Corps. These "palmtop" mobile computers are cheap, powerful and relatively rugged; more importantly, they're here and ready for use today.

2. Why Mobile Computers in the Military?

In order to use mobile computing technology sensibly, it must be exploited by people who are not only trained in its application, but completely familiar with its underlying purpose. In the Marine Corps' case, this purpose is to move information around better and faster than the bad guy. As author Tom Peters has said, "the basis of competitive advantage today is how well and rapidly an organization deploys its knowledge assets." Moreover, "warfare is no longer primarily a function of who puts the most. . .[equipment]. . . on the battlefield, but of who has the best information about the battlefield." (Peters, 1994) The Marine Corps seems to have become so immersed in Clausewitz's "fog of war" theorem, that we sometimes forget Sun Tzu's dictum about the importance of real-time information flow in choosing the correct course of action:

Success depends on foreknowledge. What is called foreknowledge cannot be elicited from spirits, nor from gods, nor by analogy with past events, nor from calculations. It must be obtained from men who know the enemy situation. (Sun Tzu, From Griffith, 1978)

Marine officers want to believe Clausewitz when he tells us that we are destined to be inundated with conflicting and erroneous information on the battlefield. Hence, it appears easier to emphasize strong characters and force of will in our leaders, rather than rely on communicating information from widely-dispersed forces. However, Clausewitz formed this theorem on a much different battlefield than the one we occupy today. Certainly, there will be always be friction and fog in modern warfare. However, real-time information flow is not only possible with today's micro-electronic revolution, it is the key to success. Keeping one's situational awareness in a fluid combat environment is an ideal problem for resolution by today's technology. An ultra-mobile, rugged microcomputer seems an ideal solution for the U.S. Marine Corps.

The Marine Corps needs a mobile computing solution today—both for combat and for peacetime. We need to acquire it with the understanding that it will be made obsolete in just a few short years by an even better one. The pace of technological advancement in microcomputers is an anathema to traditional military acquisition methodologies. Hence, we need to take a serious look at adapting off-the-shelf commercial microcomputer solutions.

Recent government legislation regarding Commercial-Off-The-Shelf (COTS) procurement is particularly applicable to the acquisition of microcomputer technology. We need to completely reorient ourselves towards adapting commercial solutions to military applications. We simply can't research, design, and develop solutions to problems, and then begin to acquire the gear through the normal acquisition channels fast enough to keep pace with the technology. We have to find a way to make our laborious acquisition process work for us. As the Director of the General Services Agency (GSA) put it, "the procurement regulations are fundamentally steeped in mistrust. Faith and trust needs to be in the federal vocabulary, as well as waste, fraud, and abuse." (Johnson, 1994)

3. Existing Mobile Computing Military Programs

Both the Marine Corps and the U.S. Army have long recognized the utility of mobile computers with the Department of Defense's (DoD) emerging vision of applying information technologies to acquire, exchange, and employ timely digital information throughout the battlespace. The Marine Corps' Digital Automated Communications Terminal (DACT) program and the Army's Appliqué system give us a hint into the future of this revolutionary technology. However, both programs seem to be missing much of the point of mobile computing: the mobility just isn't there. Unfortunately, these programs are designing their systems around hardware that is far too heavy and bulky to be of continuous practical use to the soldier or Marine employing it. A five or six-pound, battery-consuming device just isn't convenient enough to be carried and operated everywhere a Marine needs to go.

Where did these well-intentioned development programs go astray? At the base of the problem is the broad scope of the functionality that is being over-engineered into these emerging systems. The wide range of features demanded by military planners is far too encompassing to be accommodated in a truly portable solution with today's technology. However, a truly portable solution is exactly what is required. The Mitre Corporation is the nation's foremost Federal Funded Research Development Center which is providing the technical oversight for the development of the Appliqué system. According to Mitre, the Army's original vision was to design a system based around a small, hand-held palmtop-sized computer which might be carried everywhere on the battlefield. This computer would have had only a small subset of the currently mandated feature set, yet would still retain remarkable power and flexibility (to include some limited communication capability). (McCarthy, 1995)

Unfortunately, this clear vision has been obscured by military planners piling feature upon feature into the design until the hardware required to run all this software has become far too bulky to be really useful. After all, if the device doesn't help get the Marine or soldier on top of the hill in battle, why will he carry it? If a mobile computer is too cumbersome, the infantryman might very well be compelled to throw it in the mud when the bullets start to fly.

Essentially, we're designing too much complexity and functionality into the mobile systems currently being developed. Computers are useful things, but they first must be

used to be useful. Our individual Marines must become convinced that mobile computers are contributing more to their mission than what they're costing them in weight, training time and maintenance. The mandatory commitment to the proper level of training with any technology we decide to adopt is paramount. Sadly, we continue to buy all sorts of equipment in the Department of Defense without the proper motivation towards both the initial and sustained training necessary to maximize our return on investment. As a result, extraordinarily expensive gear sits around our warehouses and armories--all of it destined for the Defense Reutilization Marketing Office (DRMO) auction block at a small fraction of what the military paid for it. We simply can't afford to do this any longer. We must first convince our leaders of the extraordinary value that the potential of this mobile computer technology provides us in helping us prepare for combat.

A significant problem exists which continues to confound the program managers and project officers involved in developing high-technology systems. The acquisition cycles of even widely available Commercial-Off-The-Shelf (COTS) hardware are still far too long—regardless of the significant effort towards reform. Every piece of hardware in support of a major program generally must be specified, bid, awarded, tested, and finally approved for purchase. By the time the software which runs on this hardware is built, the hardware originally is often two or three generations behind the state of the art. Moreover, the software which was painstakingly designed to run on a specific platform will often not work properly on the newer hardware. In a computer industry which practically reinvents itself every eighteen months, a serious paradigm shift is needed in fielding small, mobile computers and its companion software in a rapid manner. Essentially, if we can't design, buy, build, and field a prototype within six months, we shouldn't do it.

Much of the problem with the DACT and the Appliqué systems has been the focus on the use of mobile computers as communication devices, Geographic Positioning Systems (GPS), and primary sensors and input devices for the massive Global Command and Control (GCCS) effort currently consuming the military research community. From the ancient times of war, commanders have been frustrated with their inability to impose their will on the forces under their command. However, as LtGen Paul Van Riper tells us, "a commander doesn't need reams of data and every last picture of the battlefield in order to make effective decisions in spite of uncertainty." (Van Riper, 1995) The answer is not to focus on using mobile computers as sensors for input to the GCCS network, but rather as a means of helping each Marine accomplish his individual mission. Clearly, the GCCS

is a critical component of our emerging military command and control infrastructure. However, the limitations of today's technology seem to suggest a different focus when we consider the mobility requirements of our foot-mobile Marines and soldiers: the individuals themselves.

4. Reorientation of the Information Flow in Mobile Systems: Down, Not Up!

The direction of information flow in all these emerging mobile computing systems seems oriented primarily in the wrong direction. The devices and software that are being fielded are too focused on reporting information up the chain. As a result, the individual Marine suffers because the current state of technology does not allow us to build a truly portable mobile computer that can also function continuously as a transmitter, an intelligence report, a positioning system, etc. With the current limited technology at our disposal, we need to concentrate instead on moving useful information down the chain to help that Marine get on top of the hill. All information flow up the chain of command needs to be passive, unless the Marine using the mobile system specifically wants to submit a combat report. The reason for this reversal is based on simple physics: it takes far less battery power to receive combat information than it takes to transmit it. Conserving energy in our mobile computers is the key to their utility on the battlefield.

In short, we don't need to establish a mobile infrastructure which will allow commanders to see and filter every nuance of the fight from each Marine's perspective. Primarily, we need to design a portable tool to help the infantrymen (and the Marines and sailors that support him) do their jobs. These tools are available today in the form of commercial palmtop-sized computers. However, many of our current system acquisition programs are focusing on designing automated systems which burden our Marines and soldiers with heavy, complex equipment whose primary function seems to turn him into some sort of remotely-controlled sensor for a senior commander far outside his immediate chain of command. It seems as if some of the military's technical advisors don't understand the dynamics of what the combat troops really need, while the infantry leadership doesn't know how to express what they really want.

At the 1995 Mobile Computing and Networking Conference in Berkeley, California, members of the Army Research Office and the Advanced Research Project Agency (ARPA) proudly announced that they are designing their Digital Battlefield mobile computing infrastructure to allow "the White House to communicate with every foxhole

on the battlefield.” However, from the perspective of the soldier in a foxhole, the last guy anybody would want to talk to from a battlefield is someone in the White House. The existing global mobility vision is grand, but seems to discount the human factors in integrating technology into our way of warfighting. After all, we still have to convince the Corporals, the Gunnies, and the Lieutenant Colonels that this technology is useful. That isn’t easy. By rethinking the flow of information, reducing the scope of the functionality of mobile computers on the digital battlefield, and providing an incremental, logical methodology for introducing the Marines to this kind of technology, we can reap huge rewards almost immediately. Remember, people don’t do things simply because their leader has told them to do so. They only do them when there is an unconscious consensus that it is the right thing to do. Hence, building consensus is an important component to any paradigm shift in thinking about mobile computers on the battlefield.

To be clear, the vision of the future proposed by the Marine Corps in their DACT program and by the Army in their Force XXI campaign is essentially a good one. Specifically, the Army hopes to apply portable, mobile, digital, and multimedia communications technology to provide real-time combat information to the tactical and strategic forces from departure to destination throughout any conflict. This is an extremely hard problem. Imagine the difficulties associated with this environment even in a relatively small Marine Expeditionary Brigade (MEB) sized conflict. We might have thousands of Marines, widely deployed over hundreds of miles, in widely varying types of terrain where line of sight, network connectivity, and bandwidth are all very limited. These problems in respect to mobile computing can be reduced tremendously by refocusing our efforts towards first providing useful Personal Information Management (PIM) oriented applications both on the battlefield and in garrison. The intent for both GCCS and the Army’s Appliqué program is commendable--however, it ignores the fact that the great majority of Marines don’t possess trust and confidence in high-technology solutions to solve all their problems. The technology community can help build this all-important consensus by providing the individual officer and NCO useful tools to do their jobs immediately. The global mobile vision of the future is technically attainable on the battlefields of tomorrow, however we’re not quite there today. We can reap immediate and tangible benefits on today’s battlefields if we simply provide our Marines with a truly portable computer and teach them how to use it.

Finally, the military leadership needs to revisit the idea that mobile computers are only to be used in tactical environments. The reality is that today’s Marines spend more

than 95% of their time in garrison. Even the infantry and other combat arms communities don't spend nearly as much time as desired in the field in peacetime--primarily because we can't afford it. The realities of the modern military make it simply impossible to train as hard as we would like. After all, if a Marine leader is doing things right, he is spending a lot of time planning, thinking, and organizing for the next week's training cycle. Good training is expensive—both in dollars and in preparation time. Automating the mundane, time-consuming administrative requirements of garrison life will allow us to focus on what is important when it really counts. The ubiquitous use of palmtop computers in garrison can greatly increase the efficiency of a company commander tirelessly preparing his Marines to go to war.

5. Downsizing Today's Military Mobile Computing Plans

So, much like we're downsizing the military, we need to start downsizing today's overly ambitious military mobile computing plans. Programs like DACT and Appliqué can continue unabated for tomorrow's fight; however, palmtop computers can also make a critical contribution to this vision of the future immediately. Existing palmtops are small enough to go everywhere and routinely provide weeks of battery life which can be easily replenished with off-the-shelf batteries. These miniature full-featured computers can hold many megabytes of useful information and are fully capable of wired, infrared, and wireless communication with minimal external hardware.

Furthermore, palmtops are much more affordable than their larger, heavier, more powerful cousins. As discussed in Chapter III, the current concept of employment of the Marine Corps DACT for infantry units limits it to battlefield applications terminating at the platoon leader level of command. Although patrol leaders and other lower level groups may occasionally be issued the DACT, limited resources make the acquisition of adequate numbers of these computers prohibitively costly. By providing a cost-effective means of integrating palmtop computers into the DACT concept of employment, all levels of leadership can reap the benefits of digitization both on the battlefield and in garrison.

C. RELATED WORK

1. Ongoing Research into the Challenges of Mobile Computing

Intermittent connectivity though widely varying communication conditions such as fading, noise and interference, power constraints, limited bandwidth, and channel use

optimization problems abound within the domain of mobile computing. The challenge is to maintain a high level of interoperability and adaptability among various networking environments (ranging from high-bandwidth wireless LANs to low-bandwidth cellular and infrared communication links). The development of a massively distributed "pico-cell" architecture of wireless base stations can provide islands of connectivity with low bandwidth location signaling (Imielinski, 1995).

Due to the limited power capacity of mobile computers, reduction of energy use in our handheld computers is also a key issue. Some solutions include slowing down the CPU based upon heuristics of past utilization; implementing flash memory storage mechanisms to optimize disk access; and minimizing the acknowledgment of packets via the adaptation of multicast vice unicast transmission of data. System architectures should be designed so that the CPUs will always "sleep" whenever possible. Similar to the model provided by cordless phones, energy can be conserved by only listening to one's IP address for a small fraction of the time (assuming the latency can be tolerated). A variety of techniques are currently being explored by researcher and developers.

Furthermore, battery technologies are not keeping pace with either the dramatic increases in processing power, nor the similar reductions in the cost of memory, storage, and CPU MIPS. In contrast to CPU and memory speeds (which increase by approximately 50% a year), battery capacity and power is only expected to increase by 20% annually over the next 10 years (Imielinski, 1995). Integrated solar recharging strategies, in particular, offers numerous advantages for minimizing power consumption. In many cases, we can adopt a television broadcast publishing paradigm--users can save energy by only receiving the channels they need at a pre-determined time. Hence, there is no need to transmit a client request (since transmitting requires much more power than receiving). The IEEE 802.11 Wireless Ethernet LAN standard (discussed in Chapter V) includes low-energy feature specifications which are applicable to this problem.

Hoarding of data or "predictive caching" is also a critical research area which will pay large dividends in increasing battery life. The work by Kistler and Satyanarayanan with their CODA project is illustrative of some of the inspired techniques in developing robust hoarding systems which scale well for mobile computing applications (Kistler and Satyanarayanan, 1992). After all, when there is a cache miss on a wired system, the system may lose a few microseconds. However, when a cache miss occurs on a wireless system, the user may be out of work for an extended period. As the bandwidth shrinks,

we need to move more of the server to the client. Thus the initiative towards "thin-client" and the "network computer" led by such companies as Sun may not be in concert with the harsh realities of a wireless world.

2. Related NPS Internetworking and Mobile Computing Research

This section lists related studies of work by members of the Naval Postgraduate School (NPS) including the NPS Mobile Computing and Communications Research Group (NPSMOB) and the NPS Information Infrastructure Research Group (IIRG).

- **Internetworking: Technical Strategy for Implementing the Next Generation Internet Protocol (IPV6) in the Marine Corps Tactical Data Network** (Nierle, 1996) - Masters thesis offering a strategy for migrating the joint tactical internet protocol towards Internet Protocol version 6 (IPv6) via a gradual transition from the IPv4 architecture. The author outlines how Ipv4 provides universal interoperability with other networking technologies. However, he claims that the requirements of the Tactical Data Network (TDN) dictate the incorporation of numerous Ipv6 enhancements such as secure mobility. Furthermore, the author proposes a tactical IP addressing plan for TDN to ensure universal interoperability and hardware-independent evolution of tactical applications and networking technology.

- **Wireless Applications for Marine Air Ground Task Forces** (Duff, 1996) - Masters thesis proposing the integration of tactical digital cellular telephones into the Marine Corps style of warfare. This research addresses the inherent advantages that make Broadband Code Division Multiple Access (B-CDMA) technologies promising for third generation digital cellular networks. Furthermore, the study recommends changes to our current Marine Expeditionary Unit (MEU) structure to better use the new wireless technologies introduced in his research.

D. SUMMARY

Mobile networked computing is rapidly becoming the next major computing field. The global wireless infrastructure that is currently being built at an unprecedented pace which will dramatically increase the utility of mobile computers both in the commercial sector and the military.

Palmtops can assist in the integration of information from the six command, control, communications, computers and intelligence (C⁴I) functional areas: maneuver, fire support, intelligence, air operations, combat service support, and C⁴I warfare. Furthermore, they can serve as the primary platform for numerous applications in garrison environments. Truly mobile computers will play a large role in achieving battlespace dominance for tomorrow's fight.

III. THE DIGITAL AUTOMATED COMMUNICATION TERMINAL (DACT)

A. INTRODUCTION

The United States Marine Corps is currently in the middle of the systems acquisition process for a device known as the Digital Automated Communications Terminal (DACT). The DACT is a subnotebook-sized, tactical input/output battlefield situational awareness system and communications terminal. The device is designed to seamlessly integrate with the legacy Tactical Combat Operations (TCO) command and control system. This TCO system provides integrated intelligence, logistics, and maneuver control for USMC force commanders. At the heart of the TCO system is the capability to establish and maintain a robust network of linked systems, allowing immediate distribution of orders, guidance, maneuver graphics and other information critical to combat and deployment operations worldwide. (MCTSSA, 1995)

Although the DACT is considered to be a Digital Messaging Transfer Device (DMTD), it is not a practical mobile computer as defined in Chapter II. Rather, it is a subnotebook-sized computer designed to execute its programs on a Win32-API-based operating system (Windows 95 or NT). As a result of various interim constraints, all current developmental and operational test software being written for the DACT is focused on the Windows 3.11 platform (or its pen-based equivalent). Consequently, all Windows software written in conjunction with this thesis for the DACT is designed for the 16-bit Win 3.X platform whenever practicable.

1. Shortcomings of the DACT System

There is a fundamental problem with both the 16- and 32-bit Windows operating system (OS) for mobile environments. Windows was designed for a full-powered personal computer as it places extraordinary resource demands on the hardware. Both variants of the operating system are large monolithic systems with a large memory footprint as well as high battery power and CPU requirements. Numerous other operating systems optimized for mobile computers offer a much more promising solution (for example, the Geoworks GEOS operating system discussed in Chapter VI, or the unannounced Pegasus Operating System reviewed in Chapter VII). However, these optimized mobile computing OSs have problems of their own. Foremost among these is the absence of abundant quality

commercial software, development environments and tools. Hence, the strength of the WinTel platform is primarily one of market share.

The DACT system currently envisioned by military planners has numerous additional shortcomings. A significant problem with the DACT is that the system as designed is too heavy and bulky to meet the needs of foot-mobile Marines in combat situations. The seven-pound threshold weight of the current DACT is at least five times what it needs to be in order to be effectively employed by our already over-burdened combat troops (MCTSSA, 1995). Although the objective weight of the eventually fielded DACT is three pounds, this is still too heavy to be practical for continuous use. Additionally, due to the Marine Corps' overly rigid specifications and insistence on compliance with MILSPEC and emerging joint communication and messaging standards, the cost of each DACT is expected to be in excess of \$15,000. This figure represents just the hardware costs. However, software and training costs have always been the most expensive aspects of developing any automated information system. Therefore, the software required for the DACT promises to be extraordinarily expensive if the Marine Corps continues to develop its application software in conformance with traditional models of systems acquisition (for example, ADA language compliance). As a result of these excessive costs, the Marine Corps simply cannot afford to purchase the system in the quantities necessary to achieve the rapid flow of combat information from the battalion to the fireteam level. Hence, the current procurement numbers within the \$60M DACT program budget only allow limited fielding down to the platoon level of command.

The DACT is theoretically being competitively procured as a "Non-Developmental Item (NDI)" during the test and evaluation phase of its acquisition (NAWC, 1995). Although the Marine Corps intends that the DACT will be built upon commercial off-the-shelf (COTS) and government off-the-shelf (GOTS) hardware and software, this is not what is happening in practice. The overly demanding technical requirements of the DACT requires the Marine Corps to develop virtually all of the DACT software from scratch in conjunction with its industry consultants. Moreover, the extensive communication and ruggedization requirements of the DACT preclude buying affordable off-the-shelf solutions—they simply don't exist in the commercial sector. Hence, the government is forcing vendors to put enormous sums of money into fabricating custom hardware oriented specifically towards the DACT specifications. This high cost is naturally being passed on to the government in the form of extremely high prices for each DACT.

It is likely no surprise that the delays and bureaucracy typically found in military system acquisition are impeding the successful integration of the DACT into the USMC way of warfighting. The DACT system planners are being forced to embrace an archaic procurement system designed for the acquisition of strategic bombers and tanks. Our current systems acquisition process was designed for equipment with a 30 year life-cycle. However, the DACT does not fit this model. USMC system planners have failed to recognize that the DACT is essentially a disposable technology which warrants an extremely compressed systems acquisition process. Failing to adopt a streamlined approach will mire the DACT in obsolescence far before the DACT is ultimately fielded. This is precisely what happened to the DACT's predecessor, the Digital Message System or Digital Communication Terminal (DCT)—a complete failure that was based on an obsolete Z-80 based architecture with a simply unusable user interface. There is essentially no way the DACT can be fielded in a timely manner given the chaotic environment that characterizes today's computer industry. It is very illustrative that the Marine Corps has scheduled eight years from the initial request for information (RFI) on the DACT in 1993 to its fully operational capable status in the year 2001. Computer performance/price ratios will double about five times during this interval, meaning that new machines will be two orders of magnitude more powerful than those available at the start of this cycle. These eight years may well as be a light year in the dynamic computer industry that confronts system planners today.

Further compounding the difficulties in building the DACT are the numerous incompatible standards existing in legacy battlefield systems which the DACT must exchange data with. For example, the TCO OTH-Gold message set and the DCT's MTS-5 message set all must interoperate with the DACT during the transitional period to the emerging Joint VMF message standard discussed in Chapter IV. This Joint VMF standard has significant problems all of its own which will further confound project officers in meeting their deadlines.

On 8 Nov 95, the DACT program entered Phase II of the Defense Acquisition Management process. The initial developmental and operational tests (DT/OT) for the DACT are scheduled for concurrent execution in September and October of 1996. It is recommended that subsequent tests of the DACT include the integration of some of the initiatives proposed in this research to test their viability in a realistic controlled environment.

B. CONCEPT OF EMPLOYMENT

1. Overview

Regardless of the numerous and varied shortcomings annotated above, the vision of the DACT's concept of employment is a laudable one. The Marine Corps desperately needs a highly mobile communication and situational awareness platform in support of the Department of Defense's digitization of the battlefield initiative. Since the DACT is an emerging system, its concept of employment has not been completely finalized. However, a draft concept of employment has been prepared by the Requirements Division of the Marine Corps Combat Development Center (MCCDC) in Quantico, VA with input provided by the author. Much of the following material is based upon this draft DACT concept of employment. (MCCDC, 1996a)

The DACT's primary mission is to extend situational awareness to echelons below the battalion level within the Marine Corps. However, as pointed out in the introduction above, this extension of situational awareness incorrectly stops at the platoon level. The DACT is designed to receive, store, retrieve, create, modify, transmit, and display map overlays, operational messages/reports, and position information via tactical radios, networks, and/or wire lines. Hence, the DACT provides general automated communications and computer support that supports the Command and Control (C2) and C2 Warfare, Intelligence, Fire Support, and Combat Service Support mission areas. (MCCDC, 1996a)

The DACT will use existing and planned tactical communications systems to support the exchange of operational and situational awareness information between other DACTS and MAGTF C4I systems. These systems include, but are not limited to, the Advanced Field Artillery Tactical Data System; the Advanced Tactical Air Command Center; the Combat Service Support Command and Control System; the Intelligence Analysis System; the Initial Fire Support Artillery System; Ethernet Local-Area Networks; and the aforementioned Tactical Combat Operations system. Additionally, through the implementation of jointly approved message formats and transmission protocols, the DACT will be able to seamlessly interface with joint and other service tactical data systems (however, the joint messaging standards on which the DACT's systems are being based may have serious shortcomings--see Chapter IV). (MCCDC, 1996a)

Due to the increased functionality of the DACT, unit DACT allowances will be higher than the DCT allowances that the DACT is replacing. There are 4111 DACTs currently scheduled for acquisition. This number provides for fielding down to the platoon level in each infantry battalion. Unfortunately, since both squad leaders and fireteam leaders also need to maintain situational awareness and communicate among each other, the number of DACTs scheduled for acquisition is woefully inadequate. (MCCDC, 1996a)

When fully fielded, the DACT will provide the MAGTF or Joint Task Force (JTF) with an increased situational awareness capability. The capability of the DACT user to automatically "push and pull" Position Location Information (PLI) to and from our MAGTF C4I systems is a new capability. The dual modem capability of the DACT will enable us to digitally connect separate tactical communications nets, creating a virtual seamless extension of the Marine Corps Tactical Data Network (TDN). (MCCDC, 1996a)

Primary users of the DACT will include: infantry, tank, Light Armored Vehicle (LAV), Amphibious Assault Vehicle (AAV) platoon and company commanders; forward air controllers, artillery and mortar forward observers; reconnaissance team leaders; radio battalion detachment leaders; ANGLICO team leaders; combat service support detachment leaders; Low Altitude Air Defense (LAAD) firing section leaders; LAAD platoon and battery commanders; and other various users requiring situational awareness and digital messaging. (MCCDC, 1996a)

DACT mission execution primarily consists of DACTs sharing the near-real-time "picture" of the battlespace with each other and MAGTF C4I systems. Information management procedures must be implemented to maximize the usefulness to each Marine with the available combat information at hand. An information warfare goal is to provide the right information to the right person at the right time. Unfortunately, the limited number of DACTs on the battlefield preclude this goal from the outset. (MCCDC, 1996a)

Standard security procedures necessary for the safeguarding of classified material/information resident within DACT shall be used. The DACT will facilitate the quick purging of predesignated critical operational information and GPS key to guard against hostile exploitation. The embedded PPS GPS receiver requires an electronic key, but it can be treated as an "unclassified when keyed" device. The DACT is designed to be carried by combat-loaded Marines and transported in tactical military and amphibious

vehicles. When mounted in vehicles, the DACT configuration will allow for the use of a larger external display and keyboard. (MCCDC, 1996a)

2. Method of Employment

Company commanders, platoon commanders, forward observers, and various section/team leaders within a MAGTF will use the DACT to gain situational awareness and access automated messaging that enhances mission accomplishment and combat effectiveness. (MCCDC, 1996a)

Figure 3.1 contains an example DACT employment for those organic units of an infantry battalion requiring basic C2 and situational awareness. At the company level, the existing company tactical radio net using SINCGARS will handle the data and voice traffic generated between the platoon and company commanders. However, at the battalion level, a separate SINCGARS data net will be required due to the many DACT users requiring mobile data communications. This leaves the current doctrinal Battalion TAC-1 net available for critical voice communications. Note that the company commander's DACT digitally links the company tactical and battalion data nets together, allowing for logical network access to all DACT users within the battalion. There also should be a primary and secondary battalion data net (i.e., VHF and HF) so that STA teams (or any DACT user) could access the network using either SINCGARS or HF radios, depending on mission requirements. Likewise, special purpose units/teams operating over-the-horizon can use SATCOM radios to connect DACTs together. This same basic employment concept can be used by all units that employ the DACT (i.e. armored units, reconnaissance teams, LAAD, etc.). Appendix A contains those operational, intelligence, and logistics messages currently planned for implementation in the DACT. One message in particular (the "Op Plan/Frag/Warning/Orders" message) is the primary subject of this research. (MCCDC, 1996a)

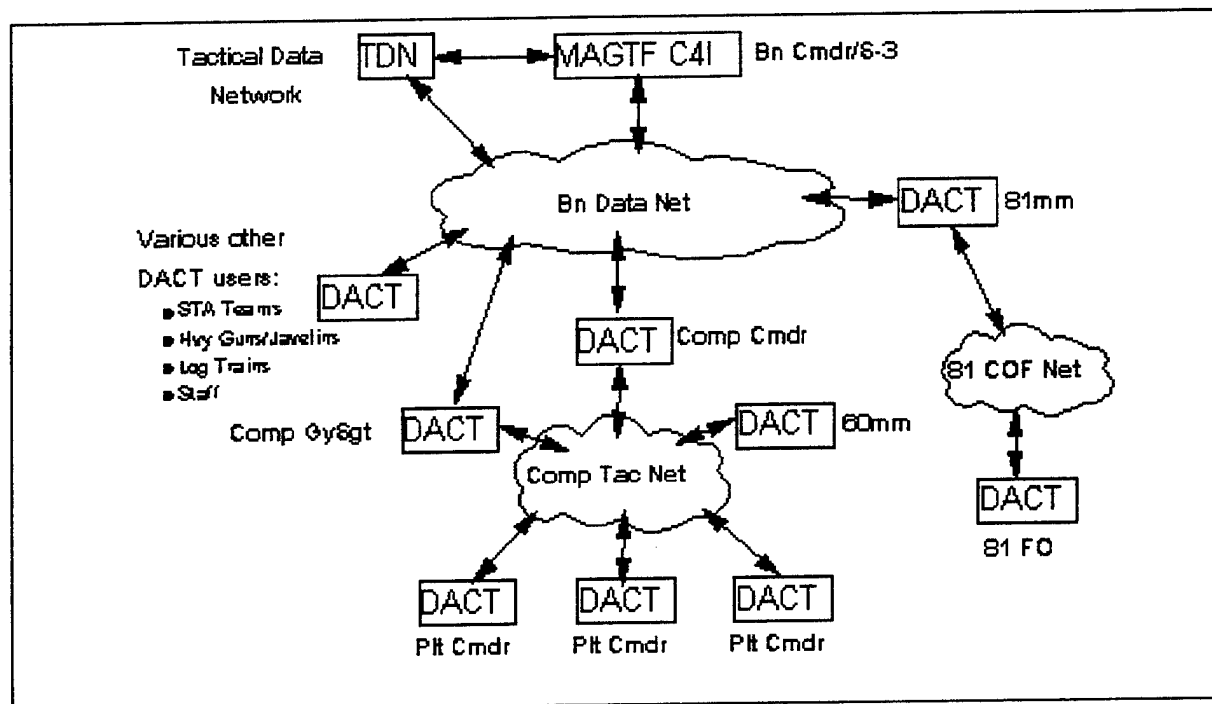


Figure 3.1. DACT Employment. (MCCDC, 1996a)

Due to the required connectivity to separate agencies, the existing doctrinal fire support nets (i.e., 81's COF, Arty COF, TAD, TAR/HR etc.) need to be used for both voice and DACT data communications. The DACT is planned to serve as the primary forward observer's data input/output platform for both the Advanced Field Artillery Tactical Data System (AFATDS) (which will migrate to MAGTF C4I) and the Target Location Designation and Hand-off (TLDH) system. Appendix A also contains those fire support messages currently planned for implementation in the DACT. (MCCDC, 1996a)

3. Mission Effectiveness Criteria

The following mission effectiveness criteria are those mission essential functions that the DACT must perform (MCCDC, 1996a):

- a. Receive, store, retrieve, create, modify, transmit, and display overlays and commanders' critical information via text and graphics.*
- b. Be capable of transmitting and receiving data over two tactical communications channels simultaneously.*
- c. Interface/operate with the current family of VRC-12, SINCGARS, Hand held (AN/PRC-68A), HF (AN/PRC-104), UHF (AN/PRC-113, non-frequency hopping mode), and SATCOM (AN/PSC-3/5) radios (all utilizing their respective cryptographic equipment) for data exchange.*
- d. Be capable of using cable/wire for exchanging information between DACTs.*
- e. Interface with MAGTF C4I systems (defined as systems implementing the MAGTF C4I software baseline on Marine Corps Common Hardware platforms) for purposes of sending/receiving position locations and text/graphics information.*
- f. Interface with the Marine Corps standard tactical data networks. The DACT should be able to log-in to the network and utilize all network user services.*
- g. Utilize the Marine Corps standard switched backbone through tactical phones (KY-68s) for data transfer.*

- h. Display, store, and retrieve digital maps and have the capability to use standard DMA digital mapping products. Map scale options (i.e., 1:25,000, 1:50,000, 1:100,000, and 1:250,000), a zoom in/out capability, route (straight or curved) distance measuring, coordinates of an operator designated point on a displayed map, and polar coordinates between any two user designated points must be supported.*
- i. Compose, edit, transmit, receive, store, and display messages. An acknowledge receipt message or alert (user selectable; on, off, volume) when a message is received at its destination must be available. Preformatted message templates must also be available.*
- j. Internally provide for the receipt and display of positional and navigational data from the Global Positioning System utilizing the Precise Positioning Service (PPS). The ability to transmit and receive GPS derived positional information between other DACTs and MAGTF C4I systems must be supported.*
- k. Be powered by an internal self-contained source, standard military/commercial power, and tactical vehicular power.*
- l. Be easily carried by a combat loaded Marine, be no larger than 9.5" X 6.5" X 3.0", and weight no more than 7 pounds (including internal power source).*
- m. Be operable in most combat/environmental conditions.*

C. DACT SPECIFICATIONS

1. Introduction

The DACT will be a sub-notebook-sized tactical input/output battlefield situational awareness system and communications terminal that will provide seamless MAGTF Command, Control, Communications, Computers, and Intelligence (C4I) digitization capability down to the platoon commander and recon team levels within the Marine Corps. The DACT will be transportable by foot-mobile Marines and mountable in tactical/armored vehicles. The DACT will primarily consist of a computer with an embedded Precise Positioning Service (PPS) Global Positioning Service (GPS) receiver; associated cables for interfacing to radios, networks, wirelines, and power sources; and ancillary devices/mounts associated with various vehicle installations. The DACT's primary features include a passive touch screen input capability (user friendly graphical user interface), color digital map display, dual modem, and internal PPS Global Positioning System (GPS) receiver. (MCCDC, 1996a)

The GPS receiver will indicate a user's location on a screen-display map; transmit the user's location and record his track; display bearing and range to a selected point, and indicate a time to arrive when closing with a selected point. (MCCDC, 1996a) Essentially, the eventual goal for the DACT is to provide a complete graphic-oriented situational awareness environment to allow a Marine to know where he is; where his buddies are; and where the bad guys have been located.

The DACT specifications (unfortunately) do not currently include an infrared data communications capability. This is clearly a mistake because of the superiority of infrared channels in providing low-cost high-bandwidth secure communications under environmental conditions conducive to military operations. Although the requirement for mid- to long-range dynamic mobile communications will also be necessary, many messages and bulky documents can be transmitted over short distances in a confined area. The tendency for Marines to "huddle-up" during combat operations to securely pass convoluted messages can be exploited with the DACT using infrared channels at a very low cost to rapidly transmit combat data inside tentage, shelters and vehicles. Infrared communications do not add significant weight or power-consumption limitations.

Figure 3.2 depicts an early prototype of the DACT. An executive overview of the DACT including an operational concept, description and technical specifications are contained in Appendix B.

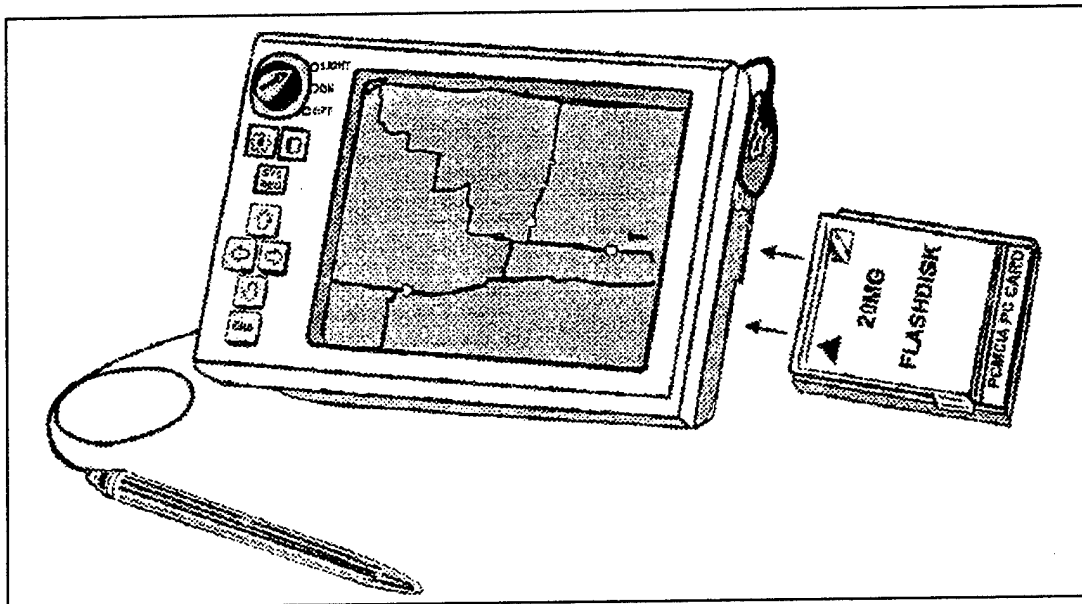


Figure 3.2. Early DACT Prototype. (MCTSSA, 1996)

2. Detailed Specifications

Per the Commerce Business Daily announcement of 15 Nov 95 (NAWC, 1995), the DACT must include the following equipment and specifications:

- a. The DACT must be a hand portable, and have a processing performance not less than that of a 486 33MHz microprocessor.*
- b. The DACT must be compatible with standard IBM personal computers and operate under a DOS/Pen for Windows environment.*
- c. The DACT shall include an operator interface that consists of a pen stylus and VGA color display.*

- d. The DACT must provide the operator with self-positioning information using an internal Precise Positioning Service (PPS) companion GPS receiver.*
- e. The DACT must have at least two externally accessible type II or higher PCMCIA interfaces, and a RS-232 port.*
- f. The DACT must have dual tactical communication interfaces including an analog Phase Shift Keying (PSK) data interface and synchronous/asynchronous digital data interfaces necessary to support military digital radios, standard voice radios, and combat net radio security equipment.*
- g. The DACT's modem will be a dual channel internal programmable modem supporting a variety of communications protocols including MIL-STD-188-220.*

D. SUMMARY

The DACT promises an important warfighting capability for the United States Marine Corps. However, the shortcomings outlined above may prevent its successful integration into our evolving style of warfighting. In the interim, the price/performance ratios of truly mobile palmtop-sized computers can fill in the gaps in the existing DACT program. Those omissions in the DACT program that can be easily and affordably corrected (such as an internal infrared capability) should be integrated into subsequent DACT functional requirements.

IV. COMMUNICATION PROTOCOLS AND STANDARDS IN DISTRIBUTED MILITARY NETWORKS

A. INTRODUCTION

There are currently more than 45 million mobile workers in the United States. Consequently a global wireless infrastructure is being built to accommodate the changing needs of this rapidly growing mobile work force. Both the strengths and weaknesses of mobile computing and wireless connectivity are having a dramatic effect on the direction of networking and communication protocol research. Nowhere do we see this more evidently than in the Department of Defense (DoD). As the military makes the transition towards the use of widely distributed "digital message transfer devices" (DMTDs) on the digitized battlefield, the ability for all these ubiquitous computers to interoperate becomes critical to the success of the mission of the military forces.

Numerous initiatives are therefore underway throughout both the Department of Defense and the commercial sector to provide interim working standards and protocols in support of mobile computing. This chapter discusses some of these critical standards and protocols analyzing their applicability towards the vision of integrating highly mobile computers on the battlefield.

B. JOINT VARIABLE MESSAGE FORMAT (VMF) MESSAGES

1. Overview of Joint VMF

The Department of Defense (DoD) is currently adopting a uniform messaging standard called the Joint Variable Message Format (VMF). This standard is being developed in response to the lack of data communications interoperability between our nation's armed forces so vividly demonstrated during recent military operations in such places as Grenada, Panama and Southwest Asia. A draft version of the messaging standard has been established by the Joint Interoperability and Engineering Organization in the form of the Joint VMF Technical Interface Design Plan - Test Edition (TIDP-TE) (DISA, 1995a). According to this document, "these standards are essential for the design, development, test, certification, fielding, and continued operation of automated tactical data systems which support the requirement to exchange timely, critical command and

control information across joint boundaries.” To date, the U.S. military has invested many millions of dollars in their efforts to accommodate this new messaging standard in emerging communication systems, as well as port over existing legacy applications to work within the constraints of Joint VMF. As the United States military fully transitions into the digitized battlespace of the future, Joint VMF promises to increase the interoperability among our too-often “disjoint” services.

Unfortunately, the current incarnation of the Joint VMF standard has numerous fundamental flaws in its architecture. Primarily, the designers of Joint VMF have violated that most sacred of military maxims: “Keep it Simple, Sir” (KISS). They have failed to provide the flexibility that any successful messaging standard must incorporate in the inherently dynamic business of preparing to fight America’s wars. Moreover, they have crafted an error prone and unreliable system which could significantly delay critical messages being communicated among units on the battlefield. It is evident that any combat information system we design must be noise resistant to the extremely high bit-error rate (BER) of the radio frequency (RF) and satellite communication (SATCOM) channels we will be using. Joint VMF is simply not as noise resistant as many other available encoding strategies. Furthermore, any joint messaging standard must be natural and simple enough to allow a high degree of ease of use from both a designer’s and user’s perspective. Alas, Joint VMF is far from natural and simple.

The designers of the Joint VMF standard have missed the critical point in designing automated systems: it is not the machine that is the expensive part of any system—it is the human being. Saving microseconds and storage space is no longer the driving force in building automated systems. Rather, representing information in a clear, concise, flexible and maintainable structure needs to be the foundation upon which any joint interoperability standard is based. In essence, the designers of the current Joint VMF standard have sacrificed flexibility and robustness on a misguided altar of software optimization. The following commentary will analyze the shortcomings of Joint VMF and offer an alternative methodology for ensuring interoperability between the services on the modern battlefield.

2. The Joint VMF Field Orders Message Format

a. Transmitting Combat Orders with Joint VMF

To specifically illustrate the inherent flaws of Joint VMF, this analysis will focus on a particular message format throughout this chapter. The example message is the K05.15 "Field Orders" message which is designed to "standardize the information format used by commander and staff to issue plans/orders to effect the coordinated execution of an operation." (DISA, 1995a) This particular message was selected primarily because it is one of the largest and most complex of messages to be transmitted using the Joint VMF standard. Furthermore, this message is also among the most important and routine type of messages transmitted on the battlefield. Each battalion-sized and smaller unit could transmit and receive up to three or four distinct "field orders" per day. The hierarchical structure of a military organization perpetuates orders from higher units to subordinate levels of command at almost a geometric rate of progression. A single field order generated by a battalion sized infantry unit, for example, might spawn well over one hundred distinct subordinate level messages delivered to the company, platoon, section, squad, and team levels within a matter of a few short hours. Virtually every one of the almost 900 Marines and sailors in a typical infantry battalion will receive an order of some kind for each mission assigned to that unit. Ergo, it is critical that Joint VMF gets this scenario right. Furthermore, the sheer size and inherent complexity of the ground combat field orders message allows us to gain insight into any strengths and deficiencies that Joint VMF might bring to the messaging requirements of the U.S. military.

b. Bit Orientation of Joint VMF

Data Field Identifier (DFI)	Data Use Indicator (DUI)	DUI Name	= Bits
4093	007	Plan/Order Type	2
4014	002	FPI	1
4003	007	Operation Identification	14
....			
4014	001	GPI	1
4045	001	GRI	1
4014	002	FPI	1
4004	010	Unit Identification	63
4014	001	GPI	1
4045	002	FRI	1
4075	005	Para 1, Situation	1400
4014	001	GPI	1
4045	002	FPI	1
4075	006	Para 2, Mission	1400

Table 4.1. Extract from K05.15 Field Orders Message Format

As can be discerned from Table 4.1, the developers of Joint VMF have selected a bit-oriented approach to the encoding of their messages. Although intended to minimize packet-size and provide a finely-grained resolution to message data components (perhaps to allow messages to be automated parsed, aggregated and routed in an automated fashion), the bit-oriented approach has several fatal flaws. A cursory glance at the message description for K05.15 reveals the essence of the design failure: it is far too complex. There are 48 individual single-bit fields in the message format called Group Presence Indicators (GPIs), Group Recurrence Indicators (GRIs), Field Presence Indicators (FPIs), and Field Recurrence Indicators (FRIs) within K05.15 (interesting, there are only 132 total fields in the message itself—hence, these overhead fields constitute over 35% of the entire record structure). These indicator fields indicate the presence or absence of a corresponding group or field immediately after the GPI/FPI, and also convey (with the GRI/FRI) whether or not the group or field is the last occurrence in its class. Hence, relatively complex parsing routines must be written to accommodate these convoluted data structures. Since the data transmission channels that these message packets will be traveling over have typically high BERs, the robustness of each message packet comes into question. A single swapped or dropped bit on one of these critical presence indicators would trash the remainder of the message's information content. Since the length of each Joint VMF message packet is by definition "variable," it is

difficult to write higher level error detection and correction routines based upon a fixed packet size. Hence, a dropped bit or two could easily be missed by an error checking protocol.

Interestingly, restructuring Joint VMF as a character-based format does not preclude the possibility of compression. Public-domain and shareware utilities like PKZIP can achieve compression ratios similar to correspondingly bit-oriented formats.

c. Bit Error Rates and Joint VMF

The emerging MIL-STD 188.220A transmission protocol (which provides the Internet and Transport protocols required for seamless communications across different network media) contains relatively robust Forward Error Correction (FEC) functionality. Although Digital Messaging Transfer Devices (DMTDs) of the future will work with both Joint VMF and MIL-STD 188.220A, no error detection and correction scheme can be perfect. Hence, bit and burst errors are bound to be occasionally introduced into the data stream. The net result of these unresolved errors will be garbled messages, along with a dramatically decreased throughput when the inevitable retransmission is requested. Will a unit commander fighting for his unit's survival prefer to receive a message that is mostly flawless and readable immediately, or a message that remains unusable until it is received 100% accurately, perhaps too late? Although it is commendable to attempt to automate as much of the routine message traffic as possible, we must never lose sight of the fact that the terminal node at both ends of every communication channel is a human being. Bit-orientation in the domain of military tactical messaging standards is destined for decreased performance since it places unrealistic demands on the real world constraints of data communication. We need the lossy redundancy in the bit stream that character orientation provides--especially in the noisy and bursty channel conditions over which the military traditionally operates.

d. Lack of Flexibility in Joint VMF

As stated in the chapter's introduction, Joint VMF fails to recognize that the messages being transmitted with its established formats are not content immutable. As each military agency adjusts their doctrine to fit their continually evolving techniques of warfighting, there will be a compelling need to adapt the message standards which convey information among the warfighters. Since the bit-oriented approach of Joint VMF requires convoluted parsing routines, we have placed yet another obstacle in the path of

rapid software development: system designers will now have to get consensus among all the services to allow any mutation in the Joint VMF standards to occur. After all, if a single bit is out of place in a Joint VMF message, every existing parsing routine written to handle that message must be redesigned, recoded, and retested. Imagine the political resistance to making even the most minor of changes in the message formats once we have hundreds of millions of dollars in legacy code based around this Joint VMF standard in place. We can gain insight into the potential catastrophic effects such shortsightedness can have by examining the "Year 2000" debacle looming on the immediate horizon (estimated by the Mitre Corporation to be over \$600 billion dollars worldwide to correct the deficiencies in legacy code date routines). (Mitre, 1996)

To illustrate Joint VMF's extraordinary lack of flexibility, let's examine the data structure of K05.15. The message's first field, Data Use Identifier "Plan/Order Type" can currently accommodate only four possible values (either Operations Plan, Operations Order, FragO, or Warning Order). Since the designers of Joint VMF have emphasized saving space in their misguided struggle for optimization, they have allocated only 2 bits to this field. In the event DoD decides in the future to add an additional order type, everyone's parsing software is instantly broken.

Joint VMF ignores the real world implications of imposing its standard on U.S. military systems analysts and programmers. After all, someone is going to have to code the parsing routines for these message packets. Since the messaging standard adopts a extremely fine-grained and convoluted bit-twiddling approach, these parsing routines will be difficult to develop (and also to debug the inevitable errors which will creep into the routines as a result of their inherent complexity). The lines are blurring between user and programmer in the rapidly evolving world of software engineering. Our world is making the transition to the enlightened methodologies of Rapid Application Design (RAD) and to the empowerment of users with easily mutable software.

The "Father of Forward Error Correction," Richard Hamming, says it best when he advises designers to "avoid 'bit-picking' and stay as close as possible to the human being when designing a system. We need to always ask ourselves the questions, 'Is it easy to use?' 'Is it easy to learn?' 'And is it easy to debug?'" (Hamming, 1996) In the case of the Joint VMF standard, any impartial analysis has to conclude with a resounding "No" for each of these critical issues.

e. Packet Sizes of Combat Order Messages Joint VMF

Message Packet Type	Uncompressed File Size (bytes)	PKZIP Compressed File Size (bytes)
Joint VMF Encoded K05.15 Field Order ¹	58,445	19,952
Paradox Version 4 REDMAN Order Database Packet ²	43,008	6,840
Plain Text Operations Order (8-bit ASCII)	12,480	4,493
HTML Version 2.0 REDMAN Order Packet	13,984	4,774

Table 4.2. File Sizes for Typical Field Orders Message Packets

Joint VMF purports to optimize packet size for minimal transmission time of message packets. For the K05.15 message, however, this is clearly not the case as depicted in Table 4.2. The field order that these statistics were based on represents an actual order transmitted by a U.S. Marine infantry battalion on a live-fire exercise at the Marine Corps Air Ground Combat Center (MCAGCC) in 1991. Appendix C contains the exact text and format of this order. It is typical of the orders transmitted in both training and combat environments for units of this size. Note that one of the reasons the results presented in Table 4.2 are so striking is that the data is skewed by virtue of K05.15's lack of normalization. K05.15 is simply a terribly designed message format. Particularly objectionable is its complete lack of conformance with the real-world model regarding the assignment of subordinate tasks. The Joint VMF data structure inexplicably requires the repetition of redundant information for each of the subordinate units' tasks. For example, if a commander can describe the enemy situation in eight free text fields of 1400 bits each (11,200 total bits or 1600 seven bit characters), and he has fifteen subordinate units he is conveying this message to, Joint VMF requires him to encode this same data fifteen separate times. Hence, he has wasted 156,800 bits (11,200 * 14). This sort of needless

¹ Filesizes are based upon the author's selection of the minimum number of repeated fields to convey order information.

² The Paradox Version 4 Order Database Packet is based on a Second Normal Form database structure optimized for the Field Orders format. It consists of six independent physical files with the following file sizes: OUT_ORD.MB-12,288 bytes; OUT_ORD.DB-4,096 bytes; OUT_ORD.PX-4,096 bytes; TO1.DB-10,240 bytes; TO1.PX-4,096 bytes; and TO1.MB-8,192 bytes.

duplication of data subverts the Joint VMF goal of optimizing message packets for minimal transmission times.

f. Other Joint VMF Design Flaws

The data model design of K05.15 further suffers from many other significant problems. For example, Index Nos 9.9 "Earliest Movement," 9.10 "Latest Movement," 9.11 "Mission Start," and 9.12 "Mission Stop" are all clearly not applicable to most field orders issued for ground combat operations. Although the GPIs allow one to bypass these fields (after wasting the space to transmit the GPI bits), the designers' decision to include them in the high level data model of K05.15 requires even more complexity (as the Joint VMF parsing routines must be written to accommodate them). Data fields such as these should be subsumed within a repeatable single free text grouping perhaps called "Coordinating Instructions."

Another problem posed by Joint VMF involves the fixed length free text fields within the data model. Numerous bits are obviously going to be wasted due to the overhead of too-large fixed-size 1400 bit (200 character) fields in the message format (similar to MS-DOS' File Allocation Table (FAT) file cluster sizes on high capacity hard disk partitions). On the average, approximately 700 bits will be wasted per free text message field (Data Field Identifier 4075) in the K05.15 structure. Since there are eight of these fields in the data structure (seven of which can be repeated infinitely), there is a potential waste of tens of thousands of bits as a result of a poor choice for field size for these fixed length fields.

Furthermore, several fields seem too constrained to address the real-world data elements of an operations order. For example, only one 200 character comment line is currently provided for. Any commander wishing to make additional comments beyond that would be frustrated by the artificial constraints of Joint VMF. Position information is also inexplicably stored in Latitude/Longitude format even though K05.15 is a "Land Combat Operations" report. The great majority of positioning systems for ground combat operations are based on the Military Grid Reference System (MGRS).

3. Joint VMF Alternatives

a. The Risks of Imposing Artificial Constraints on Message Content

Much of the commentary above regarding the deficiencies of the Joint VMF data model is oriented towards a Marine infantry centric approach. Other services might argue vehemently that the existing standards (such as Latitude/Longitude positioning) are the correct ones. That is precisely the point. The bottom line is that there will always be widely varying opinions about any messaging standard. Even if we can jointly settle on a standard acceptable to all, changing doctrine and techniques will necessarily dictate changes to that snapshot of concurrence. The hard-coded, bit-oriented approach of Joint VMF simply does not provide the flexibility we need in our messaging standard. Any successful joint messaging standard must be designed to be adaptable enough to accommodate continuous change in its messaging formats. If the Defense Information Systems Agency (DISA) attempts to impose artificial constraints on message content via its overly restrictive format for military messages, there will be understandable resistance to adopt these formats by both the user and the development communities. Clever designers will find ways to "customize" message formats with special codes for their particular service requirements thus subverting the entire purpose of Joint VMF interoperability.

b. Traditional Industry Standard Data Structures and HTML

The bit-orientation and convoluted structure in the present form of Joint VMF must be eliminated. Although plain flat-file ASCII text messages could be used, their lack of relational structure results in a reduced ability to manipulate data in an efficient manner. A much more robust approach would be to manipulate data using a character-oriented, standardized data structure. Several traditional relational databases are available with proven robust data structures which could be employed to transmit field orders message packets. Although these database structures are much easier to work with when designing and building application programs, we still have to deal with the problem of the constant flux in the messaging requirements of each service. Furthermore, these database structures are often hardware- and OS-platform dependent.

Consequently, our world is beginning to turn towards the Hypertext Markup Language (HTML) as the premier data presentation standard for internetworking

between diverse entities on widely varying platforms. HTML has numerous advantages, not the least of which is its platform independence. HTML is also extremely easy to work with when developing information systems as there are numerous development environments and code libraries readily available to support HTML and the Hypertext Transport Protocol (HTTP) used for serving these documents across a TCP/IP based network. Users report similar enhancements in the usability of applications due to the intuitive "point and click" functionality built into the hypertext model upon which HTML is based.

Furthermore, the ability to integrate multimedia data, to include graphics, video and sound, opens up new opportunities for the communication of meaningful messages on a battlefield. The 7-bit character orientation of the existing Joint VMF standard does not allow any data to be transmitted except the standard 128 ASCII character set—binary data simply cannot be communicated using Joint VMF unless we use another layer of cryptic text manipulation (such as UUENCODE or MIME encoding). There would be no requirement to develop convoluted parsing routines to handle HTML-based Joint VMF messages since every modern computing device has a World Wide Web browser available for it. Furthermore, the hypertext aspect of HTML works perfectly with the hierarchical paradigm of most military messages. Granularity of message data components for complex messages such as K05.15 can be achieved for automated routines using a variety of techniques. These techniques might include finely grained separation of HTML pages into physical files; standardized anchor link naming standards for critical data fields; and the possibility of hidden tag fields for particular HTML data components. HTML encoded documents provide redundancy of communication. HTML encoded messages are typically still human-readable in their "marked-up" state--most Joint VMF standards formats are not. This could be important if a critical parsing system went down—messages would still be viewable in their native format.

DISA needs to reevaluate its decision to make Joint VMF bit-oriented and overly convoluted in an effort to optimize message packets for size. In an computer industry which virtually reinvents the hardware every eighteen months, the rapid pace of software development designed to run on this hardware mandates both ease of use and ease of maintenance for encoding military messages. The military needs to lose its preoccupation with developing proprietary messaging standards which serve no effective purpose other than obscuring the information content of the message. An HTML-based

approach to Joint VMF messaging standards is a more sensible approach towards attaining robust data communications interoperability for the U.S. military.

C. FORWARD ERROR CORRECTION (FEC)

1. Overview

Wireless communication channels are the key to making mobile computers successful. However, wireless channel quality varies constantly and is prone to corrupt data due to multi-path and potential motion of the transmitter. Furthermore, frequency reuse in cellular systems increases the probability of channel interference. Hence, transmission unparities in both random and burst errors increase the bit-error rate (BER) of the communication channel. These errors can be corrected either with the traditional Automatic Retransmit reQuest (ARQ) methodology or through forward error correction (FEC). ARQ limits the number of stations to two and is slow since the entire packet must be retransmitted when errors are detected. FEC can communicate one to many and is much faster on a noisy link. Hence, most wireless communications channels use FEC to ensure a reasonable degree of throughput in the typical noisy wireless environments. (CDPD, 1996)

Essentially, FEC segments data into blocks with each block containing redundant bits to assist in correcting errors at the receiver without retransmitting the corrupted packet. FEC allows the receiver to both detect and correct errors in the data stream. At bottom, FEC is a trade-off between throughput and the probability of a error (since adding redundant error correction bits to a block increases transmission time). The output of FEC is a data stream of n -bit block where each block has k information bits and $(n-k)$ redundant error correction bits. (CDPD, 1996)

The degree of error that can be detected and corrected is proportional to the number of redundant bits added. A variety of FEC algorithms are available. The choice of algorithm is typically determined by the probabilistic characteristics of the noise distribution which might garble the channel. Error-correcting codes continue to be the subject of active research.

2. Reed Solomon Algorithm

One of the more common types of FEC used in wireless channels is the Reed Solomon algorithm which was developed in the 60's. For example, the increasingly popular Cellular Digital Packet Data (CDPD) transmission protocol discussed in Chapter V uses a Reed Solomon (63,47) to pre-encode messages with error correction bits. "(63,47)" means that the data is transmitted as 63 symbols, however only 47 of them carry user data. In this instance, 16 of these symbols contain the error correction and detection information. Hence, in CDPD (where the symbols are 6 bits in length), it is possible to correct up to 8 of these 6-bit symbol errors, or up to 48 corrupted bits in each 378 bits transmitted. (CDPD Forum, 1995)

The Reed-Solomon algorithm specifies that the transmitter will gather data into the aforementioned symbols, and then assemble these symbols into blocks. The error-correction code is computed by dividing the data string of the block by a polynomial. The block and the error correction code is then concatenated and transmitted. At the receiving end, the entire block (consisting of data and code) is re-encoded using the same process as the transmitter in order to produce what is called a "syndrome". If all the codes match, no errors occurred. Otherwise, a system is established of N equations with N unknowns to extract the roots of the equations (using the Chien search method) to produce the locations of the errors within the block. Next, using the polynomial selected earlier, the correct values of the erroneous symbols are computed and the erroneous symbols are replaced with the correct values. In CDPD, an undetected error probability performance better than 2.75×10^{-8} results (given typical BER of CDPD transmission channels). (CDPD Forum, 1995 and PCSI, 1996)

In the late 1960's, Kodak Berkeley Research founder Elwyn Berlekamp moved the entire problem from Euclidean space to a finite field. This made Reed-Solomon practical (vice theoretical) and greatly reduced the hardware cost of performing the reed-Solomon computations. In January 1996 Son Le-Ngoc at Memorial University, Newfoundland announced a method for vastly increasing the data throughput of Reed-Solomon FEC by proclaiming a factor-of-200-times speed improvement over the Chien search algorithm. Le-Ngoc claimed that his algorithm's error-locating time remains virtually constant as the number of bits per symbol increase. Consequently, circuit complexity, memory space and decoding time can be minimized to reduce cost. (Le-Ngoc, 1996)

D. MOBILE INTERNET PROTOCOL (MOBILE IP)

Mobile IP is the Internet Engineering Task Force (IETF)'s approach to location-independent access to computing resources in a wireless environment. Mobile IP can provide the illusion that a virtual local network extends over the entire Internet to provide seamless roaming and application transparency. Mobile computers violate the assumption that IP addresses define the topological relationship of a network's computers. Hence, Mobile IP transforms the problem of implementing mobility into a routing problem. Since Internet routing is conducted on a dynamic, unpredictable, and best-effort basis, Mobile IP provides a solution in the form of two-tier IP addressing. (Perkins, 1996)

As mobile computers move from one IP subnet to another, the mobile computer keeps its static IP address, but borrows the service of a "care-of" address on whatever IP subnet it happens to be visiting. This care-of address, when offered by a mobility agent at the mobile node's new point of attachment, can be used by many visiting mobile nodes. (Perkins, 1996). These services are also incorporated in the next-generation Internet Protocol (Ipv6) (Nierle, 1996).

E. IEEE 802.11 WIRELESS LAN AND INTER-ACCESS POINT PROTOCOLS

1. Overview

The goal of wireless LANs is essentially to minimize expenses associated with restructuring the topology of a LAN, as well as to accommodate the mobility of a network's users. Naturally, a primary concern about wireless LANs is interoperability. Because wireless equipment is still relatively expensive, users want to ensure that wireless applications they use today will be interoperable as the technology changes.

There are three major consortiums of multiple vendors all competing for the standardization of wireless LAN protocols. The Inter-Access Protocol Group, the Wireless LAN Interoperability Forum, and the Wireless LAN Alliance (a group aiming to collectively educate the industry on the benefits of wireless LANs) all want to dominate the market with their proposed standards—most of which do not interoperate among each other. (Wexler, 1996)

The IEEE 802.11 Wireless LAN Standards committee has been working six years to develop a common standard. Physical and MAC-layer proposals are completed, however 802.11 is restricted to over-the-air wireless LAN communications. Even though the standard is stable at this point, it probably won't be fully endorsed by the IEEE until early 1997. Approval of this standard appears certain. The 802.11 draft defines the physical layer information, such as the 2.4-GHz frequency range, frequency hopping, direct sequence spread-spectrum techniques, infrared and general connectivity. It also defines Layer 2 of the OSI stack, which is the actual MAC protocol. But what the draft standard does not and will not address is interoperability between access points of different vendors. (Wexler, 1996)

2. The Quest for Interoperability

Network users and administrators want to have interoperability yet retain a single wireless infrastructure that can support a variety of platforms. For example, both desktop PCs and mobile computers should be supportable with the same access point. However most of the products manufactured by the different wireless LAN vendors are proprietary. This proprietary approach has resulted in slow acceptance in the market, except in vertical markets such as warehouses, universities and health-care settings. In order for a wireless LAN to reach its ultimate potential, it must be able to interconnect seamlessly with all other equipment in the network. (Velasquez, 1996)

In May 1996, wireless LAN vendors Digital Ocean, Lucent Technologies (formerly Bellcore) and Aironet Wireless Communications announced that they will be defining an Inter-Access Point Protocol (IAPP) specification to support interoperability between their respective products. These vendors, which together have over 50 percent of the market share, have agreed on a common way to implement an inter-access protocol. The group plans to publish the specification and share the API software so that other wireless LAN vendors may adopt it. Access points are the bridges or base stations that tie wireless LANs into wired networks. With IEEE 802.11 compliant hardware running on an access point, the IAPP can be defined in software. IAPP establishes multi-vendor access-point communications since 802.11 does not go far enough to accommodate inter-access-point communications. (Velasquez, 1996)

The Wireless LAN Interoperability Forum (WLI Forum) has proposed yet another standard based on proven 2.4GHz radio technology. There is limited support by major

wireless LAN providers for this WLI Forum standards and it is expected that the IEEE 802.11 with IAPP extensions will supersede it. (Velasquez, 1996)

F. MIL-STD-188-220A WIRELESS COMMUNICATION PROTOCOL

1. Introduction

The Defense Information Systems Agency (DISA) has released a number of interoperability standards for the lower protocol layers (physical through transport). These standards have been developed in order to ensure effective command, control, communications, computers, and intelligence (C4I) systems can continue to operate over combat net radio (CNR) on the battlefield. The emerging MIL-STD-188-220A protocol provides the Internet and transport protocols required for seamless communications across different network media. This MIL-STD will soon be mandated to be used for all inter- and intra-DoD DMTDs and C4I systems that exchange information with DMTDs. It provides mandatory system standards for planning, engineering, procuring, and using DMTDs in tactical digital communications systems. This section will analyze the emerging MIL-STD (which has not been yet been formally approved) and contrast wherever appropriate with the existing IEEE 802.11 standard for wireless RF LAN products. (DISA, 1995b)

Typically, many of the of the wireless Medium Access Control (MAC) layer protocols such as IEEE 802.11 include a signaling and acknowledgment scheme to maintain transmission reliability. This overhead is in addition to the normal Ethernet, Token-Ring, IPX, or TCP/IP acknowledgments. For example, IEEE 802.11 incorporates three signaling packets for each data packet; the first is a request to send (RTS), the second is a clear to send (CTS), and the third trailing the actual data packet is the acknowledgment specifically for the wireless transmission. According to Martin Nemzow, this scheme typically reduces "wireless channel performance" by at least 50% due to increased overhead of these signaling packets. (Nemzow, 1996)

2. Flow Control

The MIL-STD protocol provides a much more flexible degree of flow control by offering "Type 1" acknowledged or unacknowledged control. Hence, bandwidth utilization can be dramatically improved under optimal channel conditions. The IEEE 802.11 committee proposal also includes Carrier Sense Multiple Access With Collision

Avoidance (CSMA/CA), a variation on Ethernet's CSMA/CD (Collision Detection). CSMA/CD is a communications protocol in which nodes contend for a shared communications channel, and all nodes have equal access to the network. Simultaneous transmissions (which will result in a collision) results in a random restart of those transmissions. CSMA/CA attempts to avoid these collisions before they occur due to their high cost in a wireless environment. (DISA, 1995b)

3. Open Systems Interconnection (OSI) Model

The functional reference model of MIL-STD-188-220A protocol is based on the International Standards Organization (ISO) 7498 Open Systems Interconnection (OSI) seven-layer model. However, it confines itself to the lower three layers: the physical, data link (divided into the network access and link level control sub-layers), and network layers. The MIL-STD protocol assumes that MIL-STD-2045-14502-1 (or standard Transaction Control Protocol/Internet Protocol [TCP/IP]) is implemented at the juncture between the Network and the Transport Layer. This protocol provides a connectionless or a connection-oriented transport service over a connectionless network service. The OSI model is essentially used only for discussion and reference purposes as TCP/IP is dominant. (DISA, 1995b)

4. Forward Error Correction (FEC)

The MIL-STD-188-220A ensures end-to-end interoperability by supporting a entire range of capabilities to include: transmission in a half-duplex mode over radio, wire-line, and satellite links; point-to-point, multi-point, relay or broadcast connectivity between stations; and error control for maintaining data integrity over the link (including frame check sequence (FCS), forward error correction (FEC), and time dispersal coding [TDC]). This forward error correction, in particular, is critical to the success of the protocol over wireless channels. This FEC is not provided for in standard IEEE 802.11. Networks with high bit-error rates (BER), such as those with wireless links and mobile hosts, violate many of the assumptions made by TCP, causing degraded end-to-end performance. Communication over wireless links is characterized by limited bandwidth, high latencies, high BER and temporary disconnections that must be dealt with by network protocols and applications. In addition, protocols and applications have to handle user mobility and the handoffs that occur as users move from cell to cell in cellular wireless networks. The major drawback of standard (wired) TCP is that it assumes all packet losses over a network are due to congestion, and it reacts by dropping its transmission

windows size before retransmitting packets. These measures result in an unnecessary reduction in the channel bandwidth utilization, yielding poor throughput and very high latencies. With wireless networks, instead of slowing down, sometimes you have to try harder to get your packet through. FEC resolves some of the wireless channel bit errors before they can become a serious problem. (DISA, 1995b)

The FEC method used within MIL-STD-188-220A is the Extended Golay (24, 12, 8) Coding algorithm. FEC capable systems allow the receiver to detect and automatically correct errors in a received block of information. This algorithm is a linear, block, perfect, and cyclic code capable of correcting any combination of three or fewer errors in a block of 24 digits. Although the standard doesn't specify that the code be implemented in software or hardware, the Golay code can use shift-registers if done in hardware, yet must use a generator matrix and conversion table if done in software. Hence, the software version is much less efficient. The Golay coding can work in conjunction with the optional Robust Communication Protocol (which is applicable to HAVEQUICK II compatible radio systems). (DISA, 1995b)

This Robust protocol provides the additional processing to aid the transfer of up to 67,200 data symbols in a single transmission with better than a 90% probability of success. This protocol can be achieved entirely in software and capable on running on low-power DMTDs. Further, if Robust Frame Synchronization is implemented in Asynchronous mode operation, a final coding algorithm called Bose-Chaudhari-Hocquenghem (BDH) (15, 7) must be used to provide a linear, block, cyclic code capable of correcting any combination of two or fewer errors in a block of 15 bits. All of these coding algorithms are described in detail in the MIL-STD. (DISA, 1995b)

5. Communications Security (COMSEC)

Another interesting feature of the MIL-STD-188-220A protocol is the incorporation of Communications Security (COMSEC) into the very lowest layer within the protocol data unit (PDU). Currently, this COMSEC is provided by external devices to include either stand-alone equipment (such as VINSON) or communications equipment with embedded COMSEC (such as SINCGARS). Eventually, a forward-compatible mode may apply to future DMTD subsystems that will embed COMSEC within the system. The three physical means of applying COMSEC to PDUs are as follows: the transmission frame structure includes either a COMSEC preamble and postamble wrapped around a

synchronization component and the data field; an embedded COMSEC message indicator into the synchronization component (without a preamble, but with a postamble), or provides no COMSEC whatsoever (neither preamble nor postamble). The preamble and postamble are only used when link encryption is used. When the upper layers interact with the physical layer, they can request that the physical layer encode its PDU using any combination of FEC, TDC, or scrambling. (DISA, 1995b)

6. Intranet Topology Updates

Appendix B in the MIL-STD-188-220A provides a procedure for active intranet topology updates. The "intranet" is defined as all the processors and CNRs within a single transmission channel. Since nodes within the network know nothing about neighbor nodes that are more than one hop away, they need to continually exchange connectivity information as the topology changes. A topology update packet is used to exchange topology information to build up a more complete view of the intranet's topology at every individual node. Sparse spanning trees are used to build topology tables to avoid the overhead of full spanning trees. All topology update packets are transmitted exclusively using a global multicast address whenever a particular node detects either a failed link, a new link, a change in the quality of the link (if link costs are used), or upon receipt of a topology update from another node. (DISA, 1995b)

One of the most important aspects of the military protocol is to provide a procedure for relaying packets across a CNR intranet using very efficient source-directed routes. Intranet relaying is required when nodes in an intranet need to communicate, but are not close enough to their neighbors to be capable of hearing each other's radio transmissions. Source Directed Relay provides a simple non-dynamic procedure for relaying a packet from an originator to one or more destinations. The source calculates the path through the intranet network to reach each destination, and this route is encoded into the intranet header. Paths along a source directed route are expected to never have common nodes. (DISA, 1995b)

7. Net Access Control

MIL-STD-188-220A also provides for a mandatory net access control (NAC) algorithm which is used to detect the presence of active transmissions on a multiple-subscriber-access communications network. It also provides a means to preclude data transmissions from conflicting on the network. Specifically, all stations will interoperate

within the following subservices and constraints: net busy sensing, response hold delay (RHD), a time period (TP), and net access delay (NAD). Net busy sensing is used to establish the presence of a data or voice signal at the receiving station due to activity on the net. An RHD period and an individual RHD value are calculated to determine the time that an addressed receiving stations delays before sending a response PDU to a command PDU. The TP is the time all stations must before they can schedule the NAD. The NAD is defined as the time a station with a message to send shall wait to send a frame after the TP timer has expired. NAD discipline is based on an infinite sequence of "slots" that begin when the TP timer has expired. (DISA, 1995b)

The military protocol also provides for the management process associated with the data-link layer via the ISO 8885, General Purpose Exchange Identification (XID) Frame Information Field Content and Format. This format is used to request or disseminate data-link information. Although it is possible to pre-configure DMTDs with link addresses and operating parameters, the XID messages can be used to dynamically allocate addresses and parameters. XID command messages such as "Join Request," "Join Accept," "Join Reject," "Hello," and "Goodbye" all allow stations to communicate with a network control (NETCON) station to request link operating parameters. For military applications, it is especially important that any station be capable of assuming and performing the functions of NETCON (due to the possible battlefield neutralization of the primary NETCON station). (DISA, 1995b)

8. MIL-STD-188-220A Summary

In summary, MIL-STD-188-220A provides a robust means of interoperating with digital message transfer systems in a battlefield environment over combat net radio. It provides numerous areas of functionality and services not found in industry protocols (to include forward error correction, communications security, and numerous enhancements to data-link layer services to optimize channel utilization over traditional battlefield radio circuits). The incorporation of this important emerging standard into a widely distributed network of highly mobile computers into the existing and future infrastructure of USMC command and control is critical to the success of the digitized battlefield concept. Robust and secure communication protocols are the bedrock of attaining the vision of the Global Mobile Command and Control System envisioned by military planners, as well as the distributed connectivity supporting the mobile Marine proposed in this thesis.

G. SUMMARY

The numerous standards and protocols in support of mobile computing are bound to be in a state of flux for many years as the military plots its course through the rocky shoals of competing technologies. The Joint VMF messaging standard is a step in the right direction, but its significant shortcomings render it an unacceptable standard for ensuring joint interoperability. Mobile IP, wireless LANs, and ad-hoc networking will all play a role in defining the military mobile infrastructure of the future. In particular, the MIL-STD-188-220A wireless communications protocol will provide a flexible extension to standard TCP/IP to enable the U.S. military to interoperate on tomorrow's battlefields.

V. COMMERCIAL CONNECTIVITY TECHNOLOGIES FOR MOBILE COMPUTING

A. INTRODUCTION

One of the most dynamic areas in the field of mobile computing is the rich array of competing communication channels and technologies emerging as research develops the infrastructure for tomorrow's mobile world. Building accessible networks without wire is the first step in creating a global, ubiquitous communications infrastructure. Hence, the interconnectivity of all of our mobile computers is critical to their success—this is especially so in the military domain. The REDMAN suite of applications discussed in later chapters of this thesis is designed to communicate over a broad array of redundant communication channels in the event one or more paths are rendered untenable. For military applications, wireless communication using radio frequency should always be optimized for Low Probability of Intercept (LPI) or Detection (LPD). Although commercial communication channels often do not incorporate LPI/LPD into their systems, there are numerous alternative techniques for achieving communications security of data transmissions in military environments.

From a military perspective, the commercial sector provides a remarkable number of communication channels and emerging networking standards that we can exploit. We can no longer afford to rely on stovepipe military-specific communication circuits to pass our data packets. The United States Special Operations Command, for example, relies almost exclusively on commercial SATCOM services for even the most sensitive of critical TOP SECRET communications (USSOC, 1994). This chapter will compare and contrast existing commercial communication technologies, as well as present futuristic communication strategies that may one day be used on our battlefields.

B. MILITARY APPLICATION OF COMMERCIAL COMMUNICATION SYSTEMS

A critical factor to be considered when designing the mobile computing infrastructure is the broadcast medium to be used when communicating from one computer to another. Both the Marine Corps and the Army are primarily focusing on using Radio Frequency (RF) over either SINCGARS or the Position Location and Reporting System (PLRS) to transmit and receive messages. Although these transmission

means are viable under certain circumstances, they add a tremendous amount of weight to our Marine's already too heavy rucksack. Furthermore, they ignore the dynamics of how information is often communicated on the battlefield. Marines and soldiers are often called forward to "huddle up" to issue operation orders and the like. Hence, we need to fully investigate wired, infrared (IR) and short-haul COTS wireless equipment to create Personal-Area Networks (PANs) in addition to the longer haul networks. Furthermore, Low Earth Orbit SATCOM, paging, and cellular telephone technologies can augment RF in communicating our critical data during the execution of our military missions. These alternate technologies are typically light-weight, high performance, and create far less of an electromagnetic signature--yet they can satisfy a significant percentage of our communication requirements.

The Commandant's Warfighting Lab in Quantico, Virginia has taken a leading role in examining the potential benefits of exploiting and augmenting the existing commercial cellular phone infrastructure to move information around the Area of Operations (AO). Some may recall the story of the Marine officer in Grenada who used his AT&T Calling Card to adjust fire while in combat (due to the lack of interoperability of communication devices with the other services in the fight). It is evident that this officer's inspired exploitation of commercial communication networks is going to be the rule, not the exception, on the battlefields of tomorrow. We can not simply discount the commercial communication infrastructure just because we can't control every aspect of it. After all, don't we employ civilians to deliver our Maritime Pre-positioning Ships (MPS) equipment to foreign lands? If the tactical circumstances permit, why not adopt this same attitude towards the delivery of our message packets?

Clearly, many combat situations require traditional means of communication on the battlefield. However, the commercial infrastructure can help us get on top of the hill in many Low-Intensity Conflict (LIC) scenarios. According to Major Paul Gibbons of the Warfighting Lab, "The direction of the Marine Corps is oriented towards every individual Marine having his own palmtop computer along with a cellular phone to communicate voice and data." (Gibbons, 1996) In the not-so-distant future, Marines will be able to have the equivalent of a Pentium-powered machine in a truly portable palmtop-sized form factor--this "desktop in your pocket" will allow capabilities beyond the wildest dreams of our military visionaries. But, as Chapters II and III exhort, we've got to keep it mobile!

C. WIRED COMMUNICATION CHANNELS

1. RS-232

There will always be a need for a straightforward means of communicating data on a point to point basis. The ubiquity of the RS-232 standard of serial communications in virtually every modern computing product is its strength. RS-232 is an Electronic Industries Association (EIA) standard that provides a physical description of a serial asynchronous communications link. This description includes the voltages, connectors, pin names and purposes of each pin. The speed of RS-232 is typically limited to 115Kbps. Error detection protocols such as the popular *Zmodem* can be used to ensure data integrity when transferring data over RS-232 null-modem cables.

2. Universal Serial Bus

A more modern standard is the Universal Serial Bus (USB) which is a direct replacement for the RS-232 standard. Although the IEEE 1394 (a.k.a. FireWire—see below) is a faster complementary serial bus standard, Intel's USB will appear on most new PCs by the end of 1996. USB is a 12-Mbps standard which will work in conjunction with a variety of USB peripherals, drivers, and applications. Unlike the single connection of RS-232, USB can connect up to 126 daisy-chained devices. (Brown, E., 1996)

The USB Implementers Forum has promoted this technology as a replacement for today's serial and parallel ports and a path away from tangled mouse cords and constant device swapping. USB telephony devices are also in the early stages of development. In the meantime, development continues on IEEE 1394. While increased speed will likely assure IEEE 1394 its place as the serial bus for video and other high bandwidth applications, it will also make 1394 more expensive and likely slow its widespread use on the PC. (Brown, E., 1996)

3. IEEE 1394 FireWire

Microsoft's announcement of the Simply Interactive PC (SIPC) initiative has added momentum to the IEEE 1394 (FireWire) standard. Firewire will be a key technology for the SIPC initiative. Among FireWire's advantages is its ease of connecting devices (up to 63), done in a manner similar to plugging into a phone jack. Current specifications allow for transfer rates of up to 400 megabits per second, although 1 Gbps

speeds are already in development. As stated above, the much slower USB is viewed a complementary to FireWire and is to be used for peripherals that do not require high speed, such as keyboards and mice. (Groz, 1996)

4. Ethernet Wired LANs

Wired Ethernet LANs are the backbone of existing computer networks and will continue to be so for the immediate future. Although the current standard of 10Mbps is rapidly giving way to 100Mbps and faster speeds, any mobile computer must interoperate with wired Ethernet networks in order to facilitate its integration into existing automated information systems. The Personal Computer Memory Card Interface Association (PCMCIA) or "PC Card" format Network Interface Card (NIC) is quickly becoming the interface card of choice for connecting mobile computers to wired networks. The wireless Ethernet LANs described below offer yet another solution to this critical interconnectivity requirement.

D. WIRELESS ETHERNET LANS

1. Overview

No one class of user more typifies the intended audience of a wireless LAN than a member of the military. Wireless LANs (WLANs) allow users to stay connected to their usual network resources while retaining their freedom of movement. WLANs allow systems to be added, moved and removed in an easy and flexible way. These LANs normally consist of user units (a small and light radio and an interface card in an ISA bus slot or PC Card packaging) and an "access point" (a larger external radio unit connected to a custom PC—the access point provides connectivity to an Ethernet segment with built-in transparent bridging functionality). Within a cell, the system replaces the Ethernet physical and Medium Access Control (MAC) layer functionality in a way that is fully transparent to the user. Wireless LANs such as the PortLAN product depicted in Figure 5.1 can extend the distance between user units and access points up to 700 meters in free space while providing up to 1 Mbps bandwidth. (Solectek, 1994)

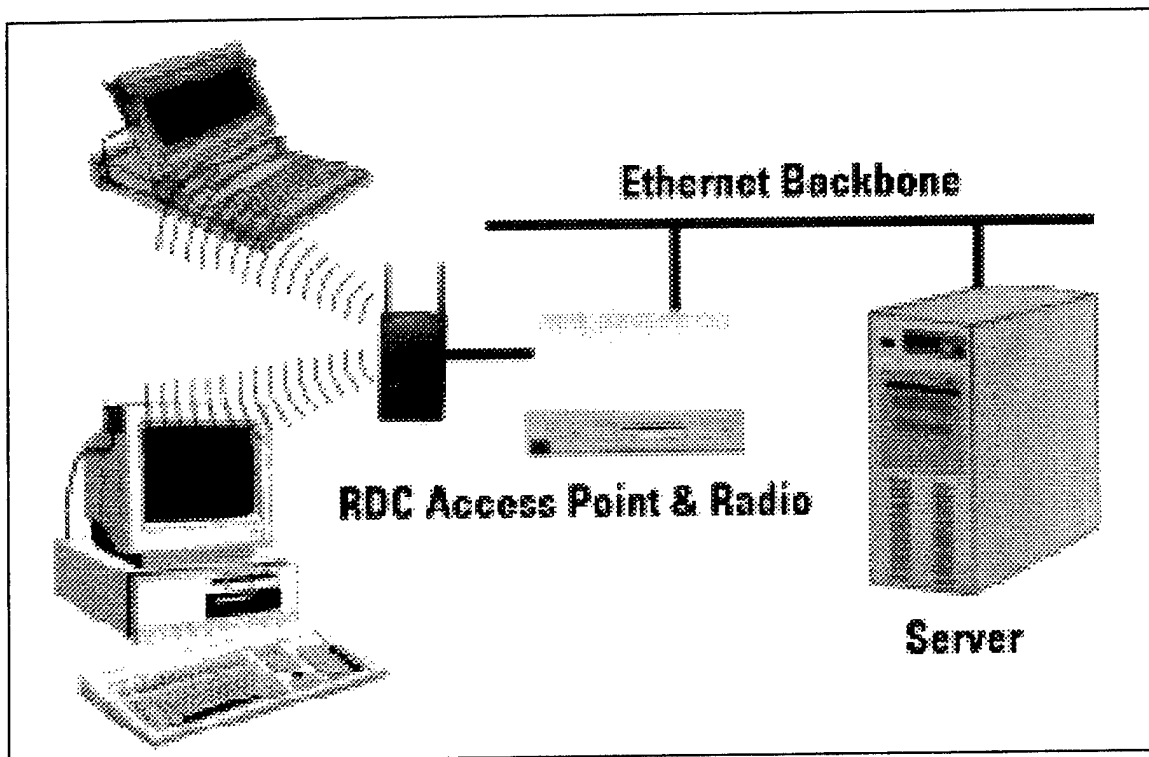


Figure 5.1. PortLAN Systems. (RDC, 1994)

MAC Bridges are also critical components for the successful integration of WLANs into wired environments. A Radio Frequency (RF) MAC Bridge is a transparent system which collects the data packets it receives from one LAN and retransmits them as soon as possible to the intended recipient on another LAN. Hence, the bridged system functions like a single large LAN. The bridge operates at the Data Link Layer of the OSI model and is protocol independent (it can forward multiple protocols), but is media dependent (it cannot link two topologies like Ethernet and Token Ring). The purpose of a bridge is to allow the network administrator to manage the traffic which travels to various segments of his network. A MAC Layer Bridge remembers the ports that packets from a particular source address came from so it will know where to send the reply packets. If a MAC Bridge knows a packet's destination address in advance (that is, if it is stored in the "Learn Table"), it will not forward a packet to a port it knows will not lead to the packet's correct destination. On the other hand, if the packet's address is not known, it will forward packets to all available ports for propagation throughout the network. (Solectek, 1994)

2. AIRLAN Wireless Bridges

Some of the more popular wireless LAN products are made by the AIRLAN company. The Direct Sequence Spread Spectrum (DSSS) modulation used in the AIRLAN products spreads the transmitted energy across a wide band of the radio portion of the electromagnetic energy spectrum. This type of robust radio modulation offers several benefits, including a greatly increased ability to work through significant interference and the ability to penetrate through walls and other obstructions. DSSS divides each element of the digital data transmitted by the system into multiple segments to improve its chances of successful transmission without retries. The AIRLAN equipment should be used in areas which provide for the physical separation of the AIRLAN products from other radio frequency devices which are operating in the same frequency range (902-928 MHz). Other spread spectrum communication frequency bands include 2.4-2.484 GHz and 5.725-5.85 GHz. Interestingly, the 2.4-2.4835 GHz spectrum is also used by microwave ovens--hence, RF wireless units should not be used close to an operating microwave oven. In general, the higher the frequency, the easier it is stopped by drywall, glass, etc. No special licenses are required to operate in these frequencies. The high frequency range and spread-spectrum frequency agility provides an LPI capability. The typical bandwidth is 2Mbps. (Solectek, 1994)

The AIRLAN Bridges can use an optional indoor omni-directional antenna, however such antennas are limited to a range of 500 feet. Point to point links from one Yagi directional antenna to another can be established so that the antennas are oriented with horizontal polarization (the cross elements are horizontal so that the vibrations of a radio wave are confined to a single plane). If an omnidirectional antenna is used with a Yagi directional antenna, one should set the directional antenna in vertical polarization (with the cross elements of the antenna vertical with similar confinement of the radio wave vibrations). (Solectek, 1994)

The AIRLAN equipment can operate in either Hub mode, Peer-to-Peer mode, multi-point mode, and base station/sub station mode. Hub mode is the fastest and most reliable, however it requires that all the bridge units which you want to talk to are within range of another and that you don't require the MAC Layer to provide packet acknowledgment. Peer-to-peer is used when you need the MAC Layer to provide packet acknowledgment for a reliable link. Multi-point Mode is used when all units are within radio transmission range of one another. If there are bridges on the network that cannot

communicate directly with one another, yet can talk with one common unit, one employs the base station/sub station methodology. The bridge designated as the base station will retransmit packets it receives which are intended for one or more of the other units, designated as substations. (Solectek, 1994)

F. INFRARED

1. Overview

Infrared is one of the most important transmission technologies available to the military mobile computing community. Lightwave transmission in the 3×10^{14} Hz range is not vulnerable to remote jamming, interception or interface, however, it does rely on a line-of-sight path. Network designers can overcome this drawback by using mirrors and carefully placed reflectors. Infrared is low-cost, has abundant, unregulated bandwidth and promotes security since it does not pass through walls or around corners. Disadvantages of infrared include short range, shadowing and interference. Highly directed IR performance can match wire channels, however users have the ability to roam from room to room and access network services in a location independent manner. Diffuse or reflective IR can provide bandwidths up to 4 Mbps over 30-60 feet in the best IR transponder systems. (Brodsky, 1996)

2. Ongoing Infrared Research

Infrared research is also among the most active of topics in the mobile community. Exciting technologies such as AT&T's REDNET—Infrared over Asynchronous Transfer Mode (ATM) are being developed which have numerous implications for mobile computing (Condon, 1996). ATM is well suited to low-speed links due to its fine grain multiplexing of small 53-byte cells. Some are predicting that ATM will be the primary technology for supporting mobile communications in the near future (Armbruster, 1995). Infrared offers low access latency and a very low component cost for inexpensive integration into existing topologies. Infrared speeds of 2-5 Mbps with a low BER of 10^{-8} are currently possible. The Infrared Data Association (IrDA) consists of the Serial Infrared (SIR) physical layer standard; the Infrared Link Access Protocol (IrLAP); the Infrared Link Management Protocol (IrLMP); and the Infrared Transport Protocol (IRTP). (Mathias, 1996)

Recently, a new infrared data-transfer standard 2.0 was announced by the IRDA which expands the existing Serial Infrared (SIR) protocol to accommodate transfer speeds up to 4 megabits per sec (Mbps), a significant improvement over the 115 Kbps rate of IRDA 1.0. To put this performance boost into perspective, consider that, at 4 Mbps, you can now transfer a 1MB file in about 3 seconds (due to overhead of the protocol). The IRDA expects further speed improvements to make rates of up to 10 Mbps possible. At these faster speeds, infrared (IR) data transfer is a viable option for PC-to-PC transfer, printing, and, most important, network access. Windows 95 ships with IR drivers that preserve partial data from loss or corruption if the connection is interrupted. (Simone, 1996)

3. Commercial Infrared Connectivity

No matter what data link mechanisms are used, all infrared transfers share a few key benefits such as bidirectional data exchange. Since IR transfers are effective at distances of up to only 1 meter, the one-to-one connection requirement of SIR means that only one notebook can be connected to a network access device at a given time. Multiple connections are possible sequentially, meaning that any number of notebooks can access the same IRDA device in succession. (Simone, 1996)

Several commercial IR products are currently being sold, such as the JetEye Net from Extended Systems and NetBeam IR from Hewlett-Packard. These systems allow any notebook with an IR port to access a network via the IR beam of light. The JetEye Net Plus works in conjunction with any Ethernet network running TCP/IP, IPX, or NetBIOS protocols. Unlike the JetEye, Hewlett-Packard's product doesn't support direct printing to a network printer. The JetEye Net Plus offers up to 4-Mbps throughput. It works up to about 3 feet away from the computer. (Rigney, 1996)

G. PACKET RADIO WIDE AREA NETWORKS (WAN)

Packet Radio WAN technologies are derived from specialized mobile radio (SMR) and use licensed bandwidth. They are built especially for two-way data transmission and they do not carry voice traffic.

1. Advanced Radio Data Information Service (ARDIS)

ARDIS is a division of Motorola and consists of more than 1,300 base stations nationwide. ARDIS covers more than 10,700 U.S. cities and towns which represents

approximately 90 percent of the business activity in the United States, and more than 80 percent of the total U.S. population. ARDIS allows interactive and virtually instantaneous two-way messaging to and from anyone with an Internet Email address. It is optimized for in-building service and is currently being upgraded to a layered and trunked system. It transmits at 4800 bps at 25 KHz, however, it will soon be increased to 19,200 bps. At 4,800 bps, however, user throughputs are often limited to 1,200 bps. Hence, ARDIS is generally not suitable for file transfers over 10K bytes. ARDIS also supports faxes and voice to text messaging services. (Ardis, 1996)

2. RAM Mobile Data

RAM Mobile is a trunked radio data network based on the international Mobitex specification. The system consists of over 800 base stations nationwide operating at 12.5 KHz over SMR channels. RAM Mobile has a 8,000 bps RF bit rate. User throughputs, however, are often limited to 2,000 bps. The RAM infrastructure focuses on covering airports, convention centers and other business-oriented locations. The RAM Mobile network is generally not suitable for file transfers over 20K bytes. (Ram Mobile, 1996)

3. Metricom and the Ricochet Wireless Network Service

The Ricochet Wireless Network Service provided by Metricom is a network system that enables high-speed, low-cost, wide-area access to on-line services, the Internet, LAN applications and peer devices. A Ricochet Wireless Network employs frequency hopping, spread spectrum and packet radios, installed geographically in a mesh topology. (Metricom, 1996)

Through a combination of Ricochet's wireless network and its wired backbone, data packets travel through the network at high throughputs. Ricochet packet radio repeaters (poletops) are shoe-box size and require low power levels. They can be deployed on utility poletops, roof-tops, or cosmetically disguised for interior building use. These poletop radios provide between 10 and 45 Kbps of continuous aggregate user-data throughput (depending upon the software, hardware, and applications being used). User-data burst speeds of up to 70Kbps are possible. (Metricom, 1996)

Poletop radios are deployed in a cluster from one-half mile to two miles apart in a mesh topology. This topology makes the Ricochet system adaptable and scalable. Each poletop radio is periodically made aware of the identity of, location of (longitude and

latitude), and the capability of each radio to transmit to other poletop radios within range. In addition, each serves as a gateway to the network, so data packets can enter the network at any poletop radio location (Figure 5.2). Once in the network, data packets can pass around busy or non-functioning radios. Expanding coverage, increasing capacity, or achieving communications in "dead spots" is simply a matter of installing one or more additional poletop radios. Clusters of poletop radios are interconnected with a high-speed data network running on a frame relay or a similar wired service. Data packets move from poletop radios to this high speed wired network through a Wired Access Point (WAP) or a Campus Access Point (CAP). (Metricom, 1996)

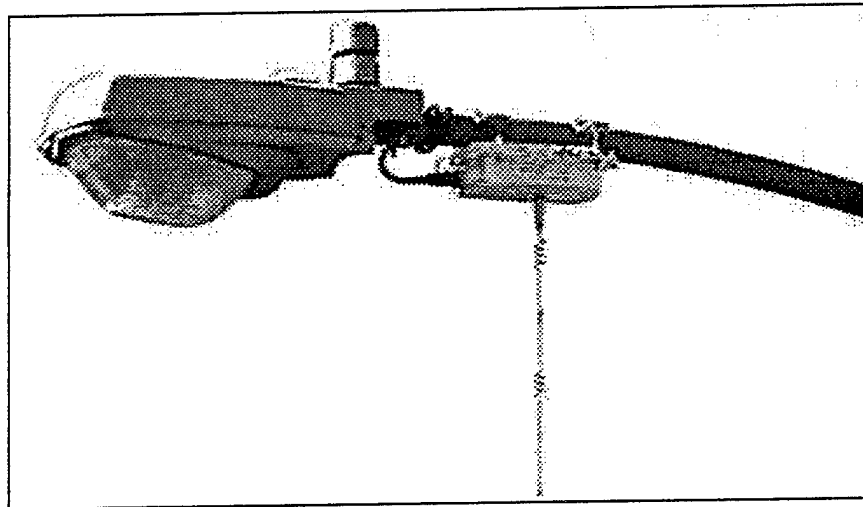


Figure 5.2. Metricom Street Lamp PoleTop Radio. (Metricom, 1996)

H. PAGING AND NARROW-BAND PCS

One of the more exciting areas in low-cost data communication channels is available in the form of paging services and products offered by a wide range of vendors. Paging technology has several advantages over other wireless transmission mediums. One of the most important is availability. Paging is available virtually everywhere in the continental United States. Coverage areas are generally larger than cellular telephone systems in a given area. With several companies offering national paging coverage, it is possible to send messages to recipients wherever they might be, anywhere in the country.

A primary advantage of paging over cellular technology is the fact that paging is a broadcast radio network. This means that it is just as easy to send a message to thousands of recipients as it is to send it to one recipient. With cellular, you would have transmit the

message to each recipient individually. With paging, you only have to send it once, and it can be received by an unlimited number of recipients. (Data Critical, 1996)

There are over 20,000,000 paging subscribers in the U.S. alone. There are voice, tone, numeric and alphanumeric pagers—building penetration is excellent. Existing systems transmit data at up to 2400 bps. Emerging protocols such as ERMES, Flex and pACT offer higher speeds and two way services. FLEX has the ability to transmit up to 25,600 bps and receive at 9600 bps. The SkyTel two-way paging system described below uses this protocol. AT&T has its own advanced paging protocol known as pACT which is a CDPD based protocol optimized for two-way paging. pACT uses an email based limited size messaging protocol at 8,000 bps full-duplex.

The concept of “asymmetrical communications” is important for better understanding of paging and PCS services. Since the pager is a very small, low power device, its range is limited. Thus, the base station is big and it has the power to reach out to many users. Microcells are the obvious answer to ensuring global coverage of future wireless networks.

1. Data Critical Corporation

The Data Critical Corporation was founded by a medical doctor who desired to transmit medical ECG (electrocardiogram) data to a handheld device wirelessly. The device needed to be capable of displaying high-resolution data yet small enough to be carried in a physician’s pocket. The company selected the Hewlett Packard palmtop computer (see Chapter VI) as an ideal platform to accommodate these requirements. Figure 5.3 depicts the HP palmtop used in conjunction with a Motorola Advisor pager. (Data Critical, 1996)

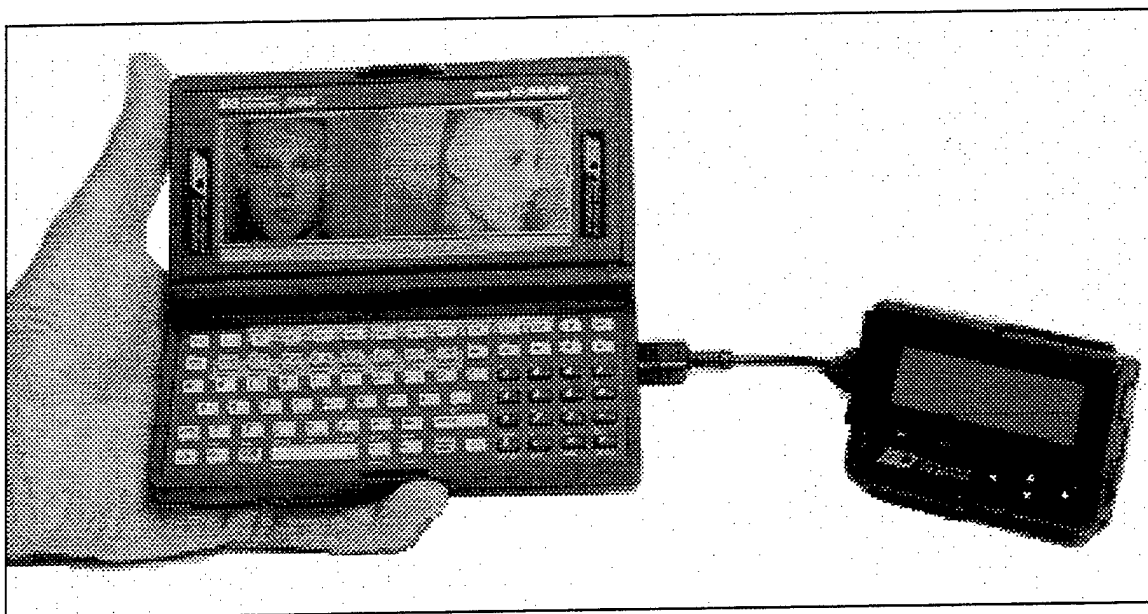


Figure 5.3. Data Critical Critical Link Connectivity. (Data Critical, 1996)

Although the HP palmtop was already capable of supporting a PCMCIA paging receiver, this paging system was only designed to receive short messages consisting of 7-bit ASCII data. Furthermore, the existing paging networks had a limit on the number of characters that each page could consist of. The solution was to encode the data and divide it into packets that can be sent as individual pages, and reassemble and decode the data on the receiving client. The result was Data Critical's patented DTP (data-through-paging) protocol. A redundancy option can be enabled to transmit a duplicate copy of every page during transmission, greatly increasing reception reliability. (Data Critical, 1996)

Data Critical also supports NIST-certified DES encryption which allows a secure way to send messages and data. Each recipient can initially be assigned an encryption key and the system will automatically use the proper key for each recipient. Data Critical's "Image APB" application was specifically designed to acquire digital images from various sources and process them for transmission. Image APB was specifically designed to acquire images from digital cameras and scanners and simplify the process of formatting and compressing them for transmission via paging. The software can also be used to transmit other types of binary data to provide a complete solution to the needs of military mobility systems. (Data Critical, 1996)

Data Critical's "Critical Link" product is a pager belt-clip adapter that turns a standard Motorola Advisor pager (see below) into a fully capable data receiver. The

Critical Link measures 1.5 X 0.5 X 0.2 inches and weighs less than 0.5 oz excluding the cable. The system only consumes 5 milliamps when operating. With Critical Link, the Advisor can receive a standard alpha-numeric or binary file and transfer it to a hand-held mobile computing device via a cable connection. Up to 6,400 bytes of interim data can be stored on the Advisor (due to memory constraints of the pager). This data can then be downloaded to a mobile computer when desired by the user. (Data Critical, 1996)

2. Motorola Advisor Pager

The ADVISOR (Figure 5.4) is a full featured alphanumeric pager that can be used to receive and store text and numeric messages as well as conventional numeric messages. The ADVISOR can receive a full-text message possibly eliminating the need for the user to return telephone calls. Designed for the heavy alphanumeric data user, ADVISOR displays messages over a large four line display. The ADVISOR has a total character memory of 6400 bytes and can handle up to 32 individual message slots. The ADVISOR shows the current time and date and timestamps each message received. Users can be alerted of incoming messages in a noisy environment, or discreetly alerted with the ADVISOR's vibration option. The ADVISOR has a low battery indicator, as well as a battery backup. The standard AAA battery powers the device for weeks with normal use. (Motorola, 1996)

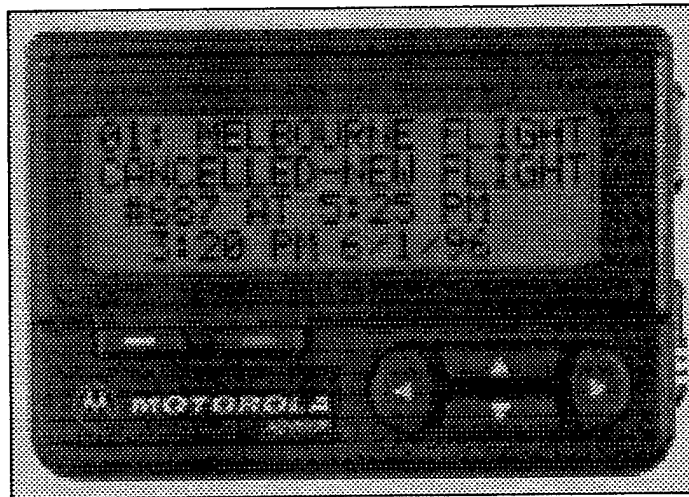


Figure 5.4. Motorola Advisor Pager. (Motorola, 1996)

3. Motorola Tango Pager

One of the more exciting areas in wireless messages is the advent of narrowband PCS with Skytel's original two-way messaging service in September 1995. With the arrival of two-way paging, users now can respond immediately to messages using either preprogrammed stored responses or by creating their own messages in real-time. When the message is received, the unit can respond with a confirming message receipt if desired. (Bryant, 1996)

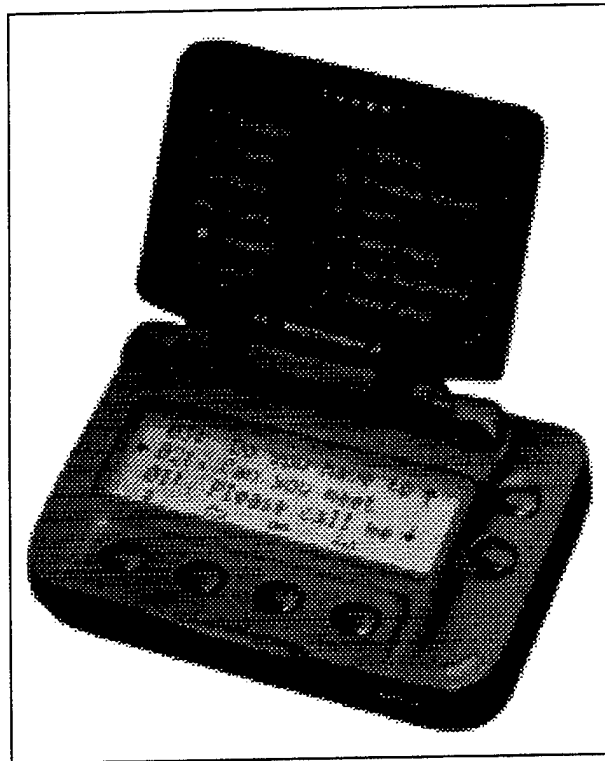


Figure 5.5. SkyTel Tango 2-Way Pager. (Motorola, 1996)

SkyTel is owned by Mobile Telecommunications Technologies (Mtel). It sends data at 24,000/9600 bps on the 50/12.5 KHz channels in the 9000 MHz band. The two-way transmission protocol, ReFLEX[tm], adds a 12.5 KHz response channel to traditional paging systems. ReFLEX protocols work on either 25 KHz or 50 KHz channels. The Tango two-way personal messaging unit accepts messages and data files at a bit rate of 6400 bps. This information can be sent in numeric, alphanumeric, or binary data formats. The Tango is also available with a data port, which allows connection, via an RS-232 interface, to various computers. The pager can receive and store up to 100 Kbytes in memory (more than sixteen times the capacity of the Advisor pager). (Motorola, 1996)

Users can respond by selecting a reply from a menu of preprogrammed messages. Alternately, the user can respond from a list of multiple choice replies included in the sender's message. When connected to a personal computer through the data port, it acts as a wireless modem by transmitting computer created messages back to the service provider where they are routed to other wireless or wireline destinations. The Tango pager has a programmer's guide which is intended for independent software and device vendors. This guide documents the serial communication protocol with a Tango two-way Personal Messaging Unit (PMU). A Software Developer's Kit including an API, documentation, and sample applications will soon be available. The API is intended to overlay this serial interface. (Motorola, 1996)

I. CELLULAR TELEPHONE SERVICES

1. History

The first mobile telephone system was introduced in 1946 by AT&T and reached over 25 cities in one year. It used a single powerful FM transmitter-up to fifty miles line of sight range. The probability of being blocked by another user was high since the service only allowed one user at a time. As late as 1970, New York's Bell System allowed only 7 simultaneous users. Furthermore, the service quality was terrible in this first system Mobile Phone System (MPS).

2. Advanced Mobile Phone System (AMPS)

Sophisticated cellular work began in the late 60s and early 70s and was based on analog modulation in the form of the Advanced Mobile Phone System (AMPS). AMPS is a full duplex, analog voice, wireless communication technology. Each service area is divided into many small regions each served by a low-power transmitter allowing frequency reuse. Naturally, frequencies are not re-used in adjacent cells to avoid interference. The number of cells in a given geographic region dictate the number of channels that are available due to frequency reuse. In general, a 50% reduction in cell radius will quadruple the number of channels per megahertz per square mile (Brodsky, 1996). Handoffs between adjacent transmitters are possible for those users who communicating on the move. Today, the existing infrastructure in America (and 55 other countries) is principally AMPS. Western Europe and many parts of Asia have settled on a different (i.e. incompatible) digital system known as Global System for Mobile

communication (GSM). By most estimates, more than 90 percent of the traffic on the U.S. cellular phone network is voice, but data transmission on cellular systems is growing rapidly (RySavy, 1996).

3. Multiple Access Techniques

Cellular systems are fundamentally based on the principles of multiple access—sharing the same electromagnetic spectrum. The three primary techniques for spectrum sharing are Time-Division Multiple-Access (TDMA), Frequency-Division Multiple-Access (FDMA), and Code-Division Multiple-Access (CDMA). FDMA has both an analog and digital variant, whereas TDMA and CDMA are digital-only standards. The recent push for digitally based cellular is being implemented primarily for the convenience of the carriers, not the consumers. There is little quality-of-service advantage to using a digital based network over an analog network from a consumer's perspective, however, channel capacity is increased significantly in digital networks (hence, the carriers are pushing it). Of course, improved capacity benefits users also.

Both TDMA and FDMA use Frequency-Hopping Spread-Spectrum (FHSS) to transmit its data packets. FHSS involves varying the frequency of a narrowband carrier in a pattern known to both transmitter and receiver. The advantages of FHSS include the possibility of multiple simultaneous distinct channels; lower power requirements; and less potential for interference from noise.

Two primary TDMA standards have been released by the Telecommunications Industry Association (TIA): IS-135 (TDMA Services, Async Data and Fax) and IS-130 (TDMA Radio Interface, Radio Link Protocol 1). TDMA has three times the spectral efficiency of FDMA analog, and will transmit data around 9,600 bps. However, both TDMA, CDMA and GSM support the concept of channel aggregation for higher data rates. TDMA will allow three channels to be combined for user data rates of up to 28.8 bps. (RySavy, 1996)

Code Division Multiple-Access is a spread-spectrum technique which has several significant advantages including the lack of a timing coordination requirement; better performance under fading conditions; resistance to interference; improved security and capacity; and soft handoffs. Soft handoffs are effected by continuing the communication with a new base station at the periphery of a cell's range using an identical frequency

between the old base station and the new. Spread spectrum technology consumes more bandwidth than "narrowband" in exchange for security and integrity. The downside of CDMA is increased complexity and the requirement for precise transmission power control. CDMA does not use FHSS, but rather employs Direct-Sequence Spread-Spectrum (DSSS) to create a redundant bit pattern (called a "chip") from each bit in the data stream. Large chipping codes provide greater reliability at the expense of bandwidth. Security is enhanced for military applications since the signal looks like low-level background noise to unintended receivers. Redundancy provides error checking and correction through statistical techniques. The advantages of DSSS are better multipath rejection and performance in noisy environments. (Mathias, 1996)

CDMA (IS-95) has up to 10 times the spectral efficiency of analog. Users share the same wide-band channel. Users are differentiated by unique spreading codes to enable frequency reuse in adjacent cells. CDMA permits a graceful degradation of capacity, yet requires a dynamic power control to effectively employ the technology. CDMA will transmit data around 14.4 kbps, but theoretically will eventually transfer data up to 48 Kbps. Since CDMA is a spread-spectrum technology which effectively mitigates multipath propagated due to its wide bandwidth and frequency diversity. At the Joint Warrior Interoperability Demonstration in September 1995, a COTS system called the Mobile Cellular Communications System (MCCS) was demonstrated which employed broadband CDMA with up to 64Kbps data channels and a very high user density. The system has inherent LPI/D performance as well as a 8 km line-of-sight propagation range at 1.9 GHz. This is a good example where we don't need the civilian infrastructure in order to support military communication requirements. If infrastructure isn't already installed where needed, we can build it very rapidly using COTS technology and techniques. (Mathias, 1996)

Table 5.1 represents a comparison between different cellular technologies.

	Analog FDMA	FDMA	TDMA	CDMA
Channels/MHz	30	30	5	1
Calls/MHz	15	30	20	12.5
Reuse Pattern	7	7	3	1
Calls/MHz/Cell	2	4	7	12.5
Capacity Gain	1	2	3.5	6

Table 5.1. Cellular Technology Comparison. (Brodsky, 1996)

4. Packet vs. Circuit Switched Cellular

Cellular systems operate either on a packet or circuit switched basis. In packet switched systems, there is no call setup delay since the circuit is continually connected. Charges are based on the data volume and type of data transmitted. Packet switched systems are usually ideal for short, bursty messages and can't typically handle large file transfers since they are often restricted to store-and-forward systems. Packets normally are an autonomous unit with address, payload and error checking internal to the packet. Packet streams can follow multiple paths to get to a single destination host.

Circuit-switched systems, on the other hand, often have lengthy setup time (over 30 seconds in some instances). Charges are based strictly on airtime, hence, this is a good solution for larger file transfers and interactive access. Circuit-switched system are very similar to landline Plain Old Telephone System (POTS) as they follow a single logical path through a network. Circuit switched systems allow a quality of service that is often not found in packet switched systems.

5. Cellular Digital Packet Data (CDPD)

There are numerous shortcomings of transmitting data over an analog system such as AMPS. Nonetheless, the AMPS infrastructure is the dominant architecture in North America and several other countries. The advantages of digital radio include spectral efficiency, noise immunity and privacy. However, the primary motivation for digital cellular deployment is not enhanced services, but rather one of capacity from the service providers perspective. This is why the use of digital phone handsets are heavily subsidized by the service providers. Spread spectrum techniques can be used to overcome interference inherent in radio channels. Essentially, a spread spectrum signal occupies more bandwidth than is strictly required by the user information. It is accomplished by

combining a pseudo-random signal with the user signal to combat multi-path, fading, and other problems. Mobile computers will connect to digital cellular telephones using a simple serial connection rather than with a modem required by the analog systems. Ultimately, the modem pools at Mobile Switching Centers will be extended to support internetworking with ISDN and other wide area network technologies. Digital cellular services will also support a short messaging service similar to paging but providing much more reliable delivery. Digital services are a jumble of incompatible standards—especially in the United States.

In 1992, a consortium of U.S. Cellular carriers was formed to create Cellular Digital Packet Data (CDPD). CDPD is an open system which extends the AMPS RF infrastructure by exploiting the characteristics of transmission idle time across cell channels (> 30% of this bandwidth otherwise goes unused). The goal of CDPD is to integrate within the AMPS structure without affecting voice capacity. (CDPD Forum, 1996)

CDPD leverages the existing AMPS infrastructure and spectrum by requiring only the addition of relatively low-cost Mobile Data Base Stations (MDBS) to the AMPS infrastructure. Voice and data can be transmitted on the same frequency simultaneously. Users are charged on a per-packet basis vice time on the channel which allows inexpensive continuous connectivity to minimize setup time. Internet support is built in to CDPD since it extends existing data networks via a connectionless, multi-protocol open design. It provides wireless extensions to the Internet via native IP support, and allows for graceful evolution to future extensions. CDPD uses Gaussian Minimum Shift Keying modulation at 19,200 bits per second (although 9,600 bps is more realistic throughput) and Digital Sense Multiple Access in the MDBS as its media access control method. There is contention for a single radio channel per cell, hence CDPD uses a busy/idle status (similar to Carrier Sense protocols) and a decode status (similar to collision detection). CDPD also uses a mobility management scheme to maintain a location information database of mobile users. CDPD routes packets based upon these locations. Hence, for military applications, CDPD would have to encrypt or otherwise secure this database to prevent enemy interception. CDPD also uses a scheme called "Radio Resource Management" to discover, select, and use the most optimal radio channel available. Like most wireless protocols, CDPD makes extensive use of Forward Error Correction (FEC) (See Chapter III) to ensure high throughputs are maintained on noisy analog circuits. (CDPD Forum, 1996)

Wireless network operators have been slashing the prices of CDPD services and shifting to flat-rate pricing models. Industry wide, there is full CDPD coverage in 44 markets today (Brown, B., 1996). The Wireless Connect Company of Santa Clara, CA sells a CDPD software development kit for use in building CDPD applications.

6. Personal Communication Services (PCS)

The emergence of Personal Communications Services (PCS) in the United States includes up to six or more competing incompatible digital standards (based on FDMA, TDMA and CDMA). PCS is a new frequency spectrum for digitally based cellular and paging services located in frequency bands recently auctioned by the FCC. PCS is normally broken down into narrowband (2-way paging and short messaging) and broadband (digital cellular). Broadband PCS is the logical successor to cellular telephony. B-PCS transmits in the 1.850 - 1.990 GHz band. It is a micro-cellular system which allows a great number of users (due to increased frequency reuse), greater capacity and longer battery life (due to smaller power requirements). However, according to the Farpoint Group, the future of wide-area data communications is in high-energy wireless-LAN technology, not BPCS. (Mathias, 1996)

7. Global System for Mobile (GSM) Communications

GSM is a European digital cellular system which is rapidly being adopted in many other world markets. It is expected to be eventually deployed in over 70 countries. GSM has the capability to carry voice, short-message service, and links to X.400 electronic mail messaging and Group III Fax. However, GSM is incompatible with the North American TDMA and CDMA infrastructure that is being built. In North America, it's currently available only in Washington, DC and parts of Canada.

8. Personal HandyPhone System (PHS)

PHS is a Japanese all digital system incompatible with other standards. It does not offer any significant advantages over the other approaches.

9. Using a Mobile Computer with Circuit Switched Analog Cellular

Data has been successfully transmitted over cellular networks for years. However signal strength variation and other issues have presented special problems to mobile data users. To address these issues, modem manufacturers have developed special protocols to include AT&T's Enhanced Throughput Cellular (ETC), Microcom's MNP 4 / V.42, Microcom/Rockwell's MNP 10, etc. to maintain reliable high speed connections in the cellular environment. These protocols use a variety of techniques including the following: (Brodsky, 1996)

- Multiple connect attempts - due to the fact that the initial handshake is the most fragile period during the modem session
- Speed shifting – enables modem to change speeds up or down in response to current link conditions while maximizing the average throughput
- Dynamic packet sizing – starts with small packets; increases packet size only if error rate remains sufficiently low
- Forward Error Correction (FEC) – sends redundant bits to enable the receiver to repair corrupted data without retransmission (See Chapter IV)
- Bit interleaving – scrambles bits prior to transmission to randomly distribute burst errors upon unscrambling for more effective FEC performance
- Auto Repeat Query (ARQ) – resends packets that are received with irreparable bit errors.

Unfortunately, these protocols are only efficient if the modems on both sides of a transmission have the protocol available. This is seldom the case with existing land-line modems. If one side of the connection uses a conventional modem, throughput can suffer. Most cellular service providers offer a solution in the form of special data access services known as “modem pools.” By dialing a special code prior to initiating communication, the cellular service provider automatically translates the enhanced cellular modem protocol on the wireless side to the standard, non-cellular modem protocols on the landline side.

The result is reliable and efficient data transmission without having to upgrade all the landline modems in the world.

On many cellular services, this access is accomplished by sending a ***DATA** message to the provider before making the call. The service responds with a few beeps to let the user know he is in the digital data transmission mode. When the user wants to switch back to regular analog calls, he needs to send a ***VOICE** message to reset this option.

For purposes of establishing connectivity between the primary palmtop computer used in this thesis research and a DACT-level machine, the AT&T Keep-In-Touch (KIT) modem card was used in conjunction with a Motorola Ultra-Lite cellular phone, a HP 200LX, and a HP OmniBook 425 (simulating a DACT-level device). The AT&T Keep-in-Touch (KIT) Card (Model 3762) is a PCMCIA Release 2.0, Type II PC Card fax/modem. The card's firmware contains AT&T's proprietary cellular protocol ETC. ETC was designed to allow for higher throughput and more reliable connections on the cellular network. The protocol consists primarily of enhancements to both the V.42 protocol and V.32bis modulation.

The AT&T modem was tested with the HP 200LX and HP OmniBook 425 and it worked well with both (Cumiskey, 1995). Data throughput rates of 14.4Kbps were achieved when an exceptionally strong cellular signal was available. However, 9,600bps is the best users can hope for in most situations. The modem detects improvement or deterioration in the signal and automatically switches to the most appropriate transmission rate. Although occasional dropped cellular connections were experienced during transmission, these were not experienced any more often than when the cellular phone was used for voice calls.

The one major drawback with this wireless solution had to do with the limited battery life of the 200LX. The modem card draws its power from the palmtop, and this shortens the life of the batteries dramatically. You can connect the palmtop to an AC adapter to eliminate the battery drain problem. However, this requires that you connect up to an outlet, subverting the entire purpose of wireless communication. A pocket cellular-ready modem with its own battery might be worth considering for use with the palmtop.

J. SATELLITE COMMUNICATIONS (SATCOM)

Geosynchronous satellites have been around since 1979. These satellites are generally fixed at a geosynchronous orbital altitude of approximately 22,000 miles to provide 24 hour per day coverage. In general, SATCOM employs transmission frequencies over 1 GHz to minimize atmospheric losses, however the distance between the receiver and transmitter typically cause excessive propagation delays. Furthermore, the costs associated with the launch of each satellite in a particular constellation can be prohibitive. (Brodsky, 1996)

1. International Maritime Satellite Organization (INMARSAT)

INMARSAT consists of mobile terminals from several manufacturers. The system was originally designed to serve maritime users in 1979. It is now generalized for terrestrial use as well. INMARSAT uses the L band in the 1.5/1.6 GHz range which is allocated worldwide for mobile up/down links. The coastal service has evolved into land mobile applications to include four standards: Standard A (analog channel with a one meter diameter antenna); Standard C (600 bps, full duplex store and forward); Standard M (2.4kbps digital); and Standard B (16kbps digital). Other companies providing similar service include COMSAT and INTELSAT. (INMARSAT, 1996)

2. Qualcomm's OmniTRACS

The Qualcomm OmniTRACS system is used for truck fleet management. The system consists of an electronically steered, mechanically driven antenna and conventional satellite constellation. The downlink from the satellite is 10,000-15,000 bps TDMA, however the uplink is only 55-165 bps. (Qualcomm, 1996)

3. American Mobile Satellite Corporation (AMSC) SKYCELL

AMSC's SKYCELL's services consists of multiple, fixed-spot beams with 4,200 5 KHz channels transmitting over the L band at 4,800 bps. AMSC plans to market and distribute SKYCELL Mobile Telephone services through existing cellular carriers in North America. SKYCELL will have 1900 circuits and each earth station will cost in the neighborhood of \$2,000. (AMSC, 1996)

4. Low-Earth-Orbit Satellites (LEOS)

This class of satellites is just being fielded and has great promise for mobile military communication requirements. LEOS is sometimes described as “cellular with moving cells.” The low orbits of the satellites reduce propagation delay and power consumption. LEOS can have anywhere between 2 and 840 satellites per constellation at an altitude of 400-10,000 kilometers. LEO satellites are generally broken down into two categories: “Little LEOS” and “Big LEOS.” The distinguishing features between the two categories are that little LEOS usually transmit below the 1 GHz frequency range and have less than 5 MHz bandwidth. Little LEOS also generally allow only low-bandwidth data transmission. Big LEOS, on the other hand, usually transmit above the 1 GHz range and have over 30 MHz bandwidth, which allows much greater data transmission rates making multimedia and other technologies possible. (Mathias, 1996)

In most cases, the constellations are designed to provide user visibility to at least one satellite all the time. The lack of redundancy in these constellations can cause some reliability problems. Most LEOS systems are compatible, but not interoperable--an earth station that talks to one system won't hand off to another constellation. (Buddenberg, 1996)

Some of the major satellite projects are listed below. Additional LEOS being built include Odyssey, Aries, Constellation Communications, Ellipso, Marafon (from Russia), Hughes SpaceWay, and Loral Cyberstar.

a. The Iridium Project

Begun in 1990, Iridium is a \$4 billion Motorola-sponsored big LEOS project designed primarily for world-wide cellular telephone coverage. The project has been scaled back to 6 polar orbital planes of 11 satellites each. Iridium data channels will be able to transmit at 2,400 bps, and the system promises to provide wireless fax, paging, data, and voice services from anywhere on earth. Iridium is scheduled to be in full operation by the third quarter of 1998. While Iridium currently has an FCC license to operate in the U.S., it still must secure licenses for trans-border roaming. Once in orbit, the 2.4-Kbps constellation will be optimized for voice and two-way alphanumeric paging. (Levin, 1996)

b. Globalstar

Globalstar is a joint venture between Qualcomm and Loral Corporation consisting of 48 CDMA-based LEOS at an altitude of 875 miles. The system has no inter-satellite links and is limited to 4.8 kbps voice and 2.4 kbps data. Unlike Iridium, Globalstar is claiming that their system will offer in-building coverage. They initially will deploy only 24 satellites. (Globalstar, 1996)

c. Orbcomm

ORBCOMM is the world's first commercial LEOS data communications and position determination satellite system. ORBCOMM requires 1,000 times less power than the traditional geosynchronous satellites. The system uses two-way data Doppler radio-location and will eventually include 26 satellites in the constellation transmitting in the 138-148 MHz range. Many of Orbcomm's communications products are being built by a company called Torrey Science located in San Diego, California. Torrey Science is a small, rapidly growing, high technology company formed to provide wireless communications and sensor products for worldwide commercial and US government markets. (Pause, 1996)

The major product areas of Torrey Science include satellite systems, underwater acoustic communications, and high-resolution active sonar. The ORBCOMM constellation will be fully operational in 1997 (two satellites are in the sky today—they need 24 more for continuous world-wide coverage, to include the poles). They also are developing the world's first mobile-to-mobile voice and data LEO satellite system. The obvious advantages of LEO satellites for military applications is the relatively low altitude involved (approximately 400 miles up) yielding very low power requirements for successful data communications, lower latency and propagation delay, and numerous simultaneous users. The current generation of OrbComm LEO ground transponder (the Comcore 300B) is approximately 3 x 8 x 1.5 inches. A developer's kit is also available which allows the rapid development and integration of the Comcore 300B into custom applications. (Pause, 1996)

A PC-Card format of the Comcore 300B (which will also interface with the HP family of palmtops) is scheduled to be ready by the summer of 1997. Once the constellation is full, the unit will have full GPS capability (yielding an extremely compact combined GPS and over-the-horizon communication capability for the mobile user). The

transmission rate of Torrey Science's equipment is relatively limited--approximately 4800 downlink and 2400 uplink. By next year, it is supposed to double in speed to 9600/4800 respectively. The system is designed to support both small burst messaging of millions of world-wide users simultaneously and provide geo-positioning data. (Pause, 1996)

Torrey Science is currently building the Tracking and Reporting System (TRS) based on their LEO equipment for the U.S. Army. This system involves the use of their LEO equipment for tracking and logistical purposes. In support of this initiative, Torrey Science has selected the HP palmtop as the communication terminal and plans on having it interfaced to their current generation LEO transmitter by the end of 1996. (Pause, 1996)

d. Teledesic

Bill Gates and Craig McCaw are funding the \$9 billion Teledesic project that will create a broadband network in the sky. Teledesic, a LEOS network optimized for high-bandwidth traffic, is planned to become available in 2001. Teledesic is designed to transmit data at 16 Kbps to 2 Mbps inexpensively with low error rate and low delay. Traditional LEOS systems such as the Iridium Project are lower bandwidth and are not competitive with Teledesic. Since the economics of fiber installation drops off rapidly outside urban areas (as user density declines and the intensity of usage drops), Teledesic is expected to provide an affordable technology to provide Internet access to remote areas. (Levin, 1996)

Teledesic has planned for 840 satellites in 21 orbital planes, of which users will be able to see at least two at elevation angles of >40 degrees from anywhere in the world. Teledesic satellites will have crosslinks and provide up to 155 Mbps aggregate throughput. (Levin, 1996)

e. ICO Global Communications

The London-based ICO Global Communications was formed last year as a privatized arm of INMARSAT--the international mobile satellite organization comprised of 79 member countries. ICO Global plans to launch an "intermediate circular orbit" network to provide global phone, data, fax, and paging services to hand-held devices, at \$2 a minute. Since ICO Global is intergovernmental, it may provide a significant challenge to the Iridium project. (Levin, 1996)

5. Sky Station—Stratospheric Geostationary Dirigibles

The Chantilly, Virginia based Sky Station International has filed applications with the Federal Communications Commission proposing a new global wireless communications system that will compete with GlobalStar, Iridium, and other satellite services. Using a network of 250 stratospheric geostationary dirigibles, Sky Station hopes to offer wireless services to more than 80 percent of the world's population by 2002. Each lighter-than-air dirigible will comprise two blimp-like aircraft and a large antenna system (approximately the size of a football field) suspended between them. This assembly will remain aloft 18 miles above the earth for up to 10 years at a time. (Somers, 1996)

Once completed, the network (which reportedly costs \$4.2 billion) will provide 64-Kbps digital broadband service to 1.5 billion customers worldwide. The network will provide services such as wireless Web access and picture-phone connectivity. Sky Station has asked the FCC to designate frequencies at 47 GHz for the system's operation in the U.S. The company also will request several global policy adjustments at the 1997 World Radio-Communications Conference designed to make these frequencies consistently available for the new technology, which is now termed the "Global Stratospheric Telecommunications Service" (GSTS). (Somers, 1996)

At the heart of the technology is a technology called the "Corona Ion Engine," which enables the aircraft to remain in a fixed position relative to the Earth's surface. Previously, only satellites that escaped the Earth's atmosphere were able to achieve such a geostationary position. The on-board phased-array antenna and data-processing equipment will enable each unmanned dirigible to process an estimated 60,000 simultaneous calls. (Somers, 1996)

K. SUMMARY

The commercial sector provides a remarkable number of communication channels and emerging networking standards that the military can exploit. Wired communication channels will continue to play an important role in interconnecting our mobile devices. However, the domain of wireless channels such as Infrared, packet radio, paging, cellular, and LEOS offers the most promise in realizing the potential of a robust and reliable mobile infrastructure on the battlefield.

VI. THE HEWLETT-PACKARD FAMILY OF PALMTOP COMPUTERS

A. INTRODUCTION

Beginning with the release of the HP-35 in February 1972, the Hewlett-Packard (HP) Corporation has made successful handheld calculator products for almost twenty-five years. With the combined experience developed by designing over seventy different models, HP's family of financial and advanced math and engineering calculators represent the de facto standard for engineers and mathematicians internationally. HP's in-depth knowledge of miniaturization technology, open architecture standards, communication and desktop connectivity have allowed them to create the current family of HP LX palmtops preferred by most mobile computing industry insiders. This chapter will discuss this family of palmtop computers and some of the ancillary products that work in conjunction with them.

As discussed in Chapter III, the DACT initiative is designed to stop at the platoon leader's level of command primarily owing to the extraordinary expense of the DACT hardware. The integration of inexpensive, COTS palmtop-sized computers into the emerging DACT network would allow the Marine Corps to transmit critical combat information all the way up and down the chain of command. This chapter will also examine some of the specific MS-DOS and HP-specific developmental tools used in building mobile computing software for military applications. A sample application written in the C language is developed to demonstrate the potential power of the palmtop on the battlefield. To many mobile computing users, the most important aspect of a mobile computer is its relatively tiny form factor. The ultraportability and long battery life of the HP palmtop is compatible with the high mobility requirements of a modern Marine.

1. The HP95LX

Leveraging off its experience with calculators, HP's first foray into a general purpose MS-DOS palmtop computer was the HP95LX in 1991. This Intel x86 real-mode machine was in essence the equivalent of the original IBM PC-XT, yet it weighed 11 ounces and ran for weeks on a single pair of AA batteries. The HP95LX included a full suite of Personal Information Management (PIM) software as well as a streamlined version of Lotus 123 spreadsheet. The HP also had the capability of running thousands

of MS-DOS programs from its ROM-based version of DOS. The HP95 was rugged, reliable and has proved itself in many battlefield situations. In the jungles of Okinawa, the 3d Reconnaissance Battalion has employed the palmtop as a data entry platform using the TRQ-43/PRC-137 HF radios to demonstrate seamless E-mail exchange between mobile forces and the supporting establishment. The author has used the same palmtop in the deserts of Kuwait (Cumiskey, 1993) with great effect. However, the HP95 was far from perfect. Its primary shortcoming was the HP95's 40 x 16 LCD display which limited its utility for full-sized screen applications. Users were forced to scroll left and right for DOS-based applications requiring 80-column output. Furthermore, the HP95LX did not have a serial port completely compatible with standard RS-232 devices. (HP, 1992)

2. The HP100LX and HP200LX

In 1993, HP released the HP100LX which resolved both shortcomings with a full-sized 80 x 25 character screen, as well as a standard serial port. This was soon followed in 1995 with the 200LX which contained 3mb of ROM including software enhancements such as Pocket Quicken, Laplink Remote, and both functional and cosmetic changes to the existing PIM suite. The HP200LX (Figure 6.1) is the current state of the art for MS-DOS based general purpose palmtop computers sold in the United States. Hence, we will focus on the 200LX for much of the remaining sections of this chapter.

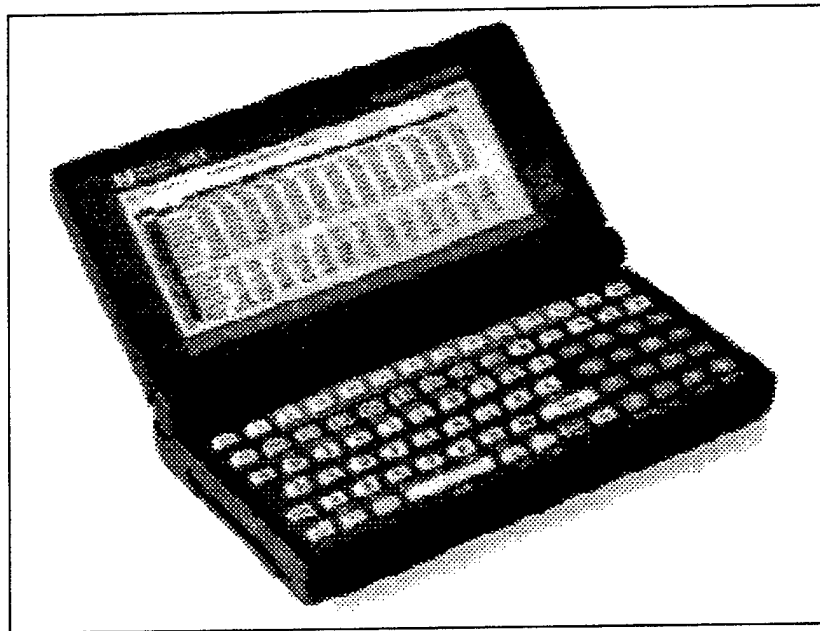


Figure 6.1. HP200LX Palmtop. (HP, 1994)

3. The OmniGo 100 and 700

In 1996, HP announced the Omnigo 100 (Figure 6.2)—a radical departure from the purely DOS-based HP palmtops. The Omnigo is based on a 80186 Intel processor running at 16Mhz. The big change between this system and the LX family of machines is that the OmniGo 100 is based on the GEOS graphical operating system (which includes the integral Graffiti handwriting recognition). The Omnigo 100 comes standard with 1 MB of RAM and 3MB of RAM. Unfortunately, the current version of the OmniGo 100 can only use a type I SRAM card—the more popular Flash RAM cards (see below) are not supported. The display screen is a square 240 X 240 pixels owing to the ability of the user to dynamically reorient the display image 90 degrees. The device weighs 11.6 ounces (only slightly more than a 200LX). Recently, there has been a new version of the MegaHertz RadioModem announced for the OmniGo 100 (see below).

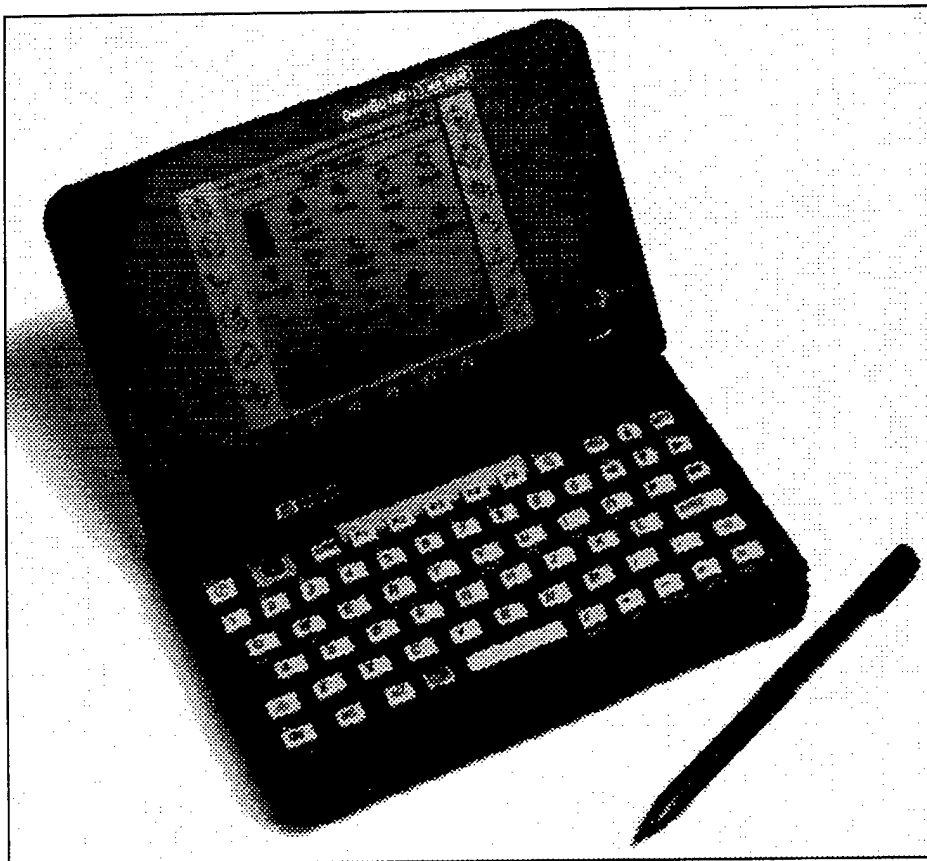


Figure 6.2. HP Omnigo 100. (Ace, 1996)

The GEOS operating system is developed and marketed by the Geoworks Corporation as an easy-to-use graphical environment for what they term the Consumer Computing Device (CCD) market. GEOS is designed for Intel X86 compatible semiconductors based on a layered object-oriented component architecture. The operating system is preemptive multithreaded and multitasking so that multiple tasks can be executed concurrently. Since memory and power management are so key to mobile computing platforms, GEOS manages both functions internally in its kernel. The GEOS architecture is designed to be highly flexible and extensible, and the operating system is being incorporated into a variety of portable products from smart phones to higher end palmtops. Unfortunately, as a first generation product, the product is still immature, and has relatively poor development tools for the aspiring mobile computing system designer. (Geoworks, 1996)

The HP OmniGo 700 (Figure 6.3) is essentially a HP200LX with an integrated Global System for Mobile Communications (GSM) Nokia 2110 phone which can be smart-docked with the OmniGo 700. The OmniGo 700 includes short message management with a built-in Short Message System (SMS) application (SMS is a subset of the GSM standard). The device also includes auto-dial connectivity between the GSM phone and the standard HP200LX phone database PIM application. Hence the OmniGo 700 is a self-contained data communicator. Unfortunately the GSM infrastructure is not widely available in the U.S. (see Chapter V) and the OmniGo 700 is not expected to be a viable option within the U.S. for as many as five years. Furthermore the Omnigo 700 already has a number of competitors including the GSM based Nokia 9000 which combines Geoworks running on an Intel processor. Nonetheless the Omnigo 700 offers a look into the future as mobile computers are rapidly becoming completely integrated with cellular phones. (HP, 1995)

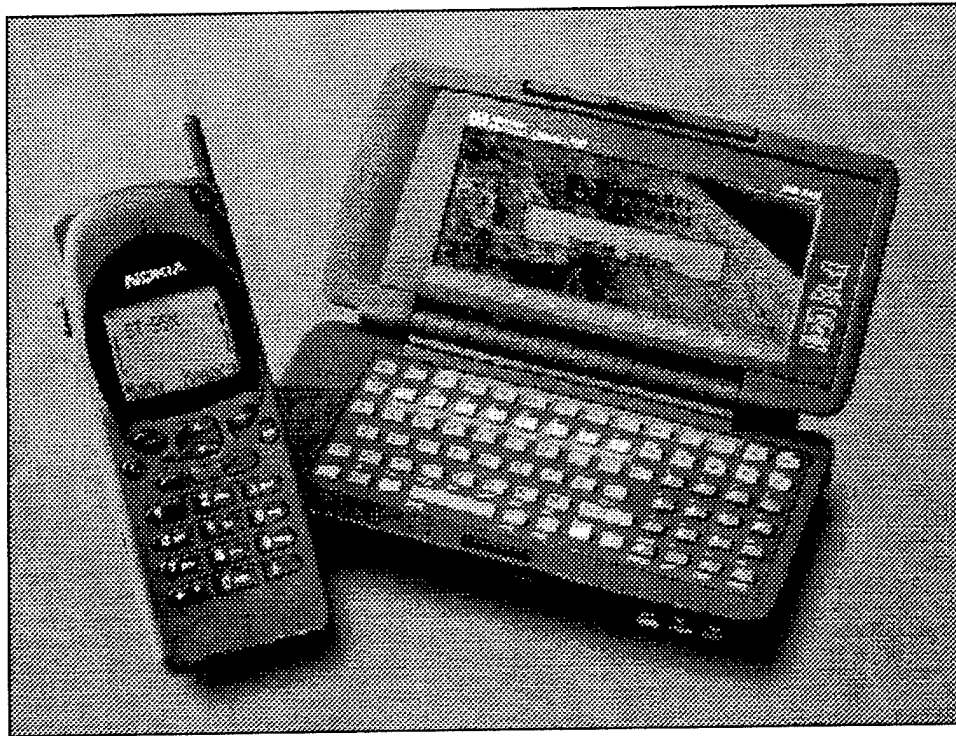


Figure 6.3. HP Omnigo 700. (HP, 1995)

4. Strengths of the HP 200LX Palmtop

The strength of the HP200LX palmtop (Figure 6.1) lies in both the quality of HP's engineering and in the choice of MS-DOS as the HP200LX's operating system. By using an industry standard OS, HP has assured the availability of literally thousands of DOS-based programs, libraries and programming environments for the military developer. However, the strength of the HP palmtop for developers is also its weakness for users. The DOS command line interface can be a real stumbling block for new users. Although the PIM-based applications are penetrable for most users, the thought of running programs from the command line leaves most users cold. Nevertheless, the HP200LX remains very popular. Although there have been dozens of various GUI-based palmtops and PDAs available for years (such as the Motorola Envoy and the Apple Newton), most industry insiders (such as Andrew Seybold--principal organizer of the Portable Computer and Communications Association) prefer the HP200LX as their mobile computing platform of choice. It is simply the all-around best choice in the current marketplace for many users. Furthermore, the HP200LX enjoys the most third-party vendor hardware and

software support than any other palmtop (especially with communications products such as the Megahertz AllPoints card discussed below).

The instant-on capability in conjunction with the power-on password protection of the HP is consistent with the requirements of most military users. Furthermore, most of the integral PIM applications save user data automatically—enhancing the HP's ease of use. The HP200LX has the most customizable and powerful scheduling system available for scheduling convoluted repeating appointments and to-do list items. The HP also enjoys one of the longest battery lives of all similar palmtop—lasting up to 2-4 weeks on a single pair of AA batteries. Rechargeable Nickel Cadmium (NiCd) or Nickel Metal Hybrid (NiMh) batteries are available along with an optional A/C adapter which recharges batteries without removing them from the device. The HP200LX also has one of the best warranties in the industry—offering a no-questions asked over-night replacement policy for a period up to three years (with the purchase of an 2-year extended warranty contract for the nominal fee of \$85.00).

B. THE ANATOMY OF THE 200LX

1. Hardware Features

The HP200LX consists of the following hardware features:

a. Flexible Memory and Storage Options

The HP200LX contains either 1 or 2 MB of user RAM. A portion of this memory (usually 640KB) is reserved for program execution. The remainder is designated as the "C:" drive for the storage of data files and user programs. As stated earlier, all the built in programs are stored in the 3MB of ROM that comes standard on the HP200LX. This drive is designated as drive "D:". Additionally, the HP200LX supports both PCMCIA 2.0 type I or type II Static RAM (SRAM) and Flash Disk PC memory cards for increased storage capabilities up to 160MB with compression. When a memory card is inserted, the card is referenced as drive "A." SRAM cards are generally faster than flash and have less of a power drain, however they are expensive and require a battery back-up. Flash memory is cheaper per megabyte, available in larger sizes, and can hold data forever without power. Flash memory is also very slow for writing (on the order of the speed of a floppy disk), requires significant power for writing, and will eventually wear out when written to many times. Although the HP200LX does not support extended or enhanced

memory natively, there is a program in the public domain called "EMM100" which provides a paging file on C: and supports EMS 4.0 almost completely. (Cochran, 1996)

b. Flexible and widely available batteries and optional A/C power

The HP200LX can run on any brand of 1.5 volt, size AA Alkaline or Nickel Cadmium rechargeable batteries. Although HP doesn't officially support NiMH batteries, many users (including the author) have used them successfully for years without incident. The state of the art batteries for use in the HP are NiMH batteries in the range of 1200 mAH which provides a long-lasting, rechargeable solution with a minimal memory effect of NiCd batteries.

Since RAM (including user data) can only be preserved by continual electrical current, the HP200LX includes a backup 3-volt CR2032 lithium coin cell which lasts about a year with normal use. This backup battery preserves memory when main batteries become exhausted (or are being changed).

HP provides an optional AC to DC power converter (HP #F1011A) which provides regulated power and allows "in-machine" recharging of NiCd and NiMH batteries. Optional third-party power adapters are available for "cigarette lighter" sockets. These power adapters can be easily converted to other power sources as well.

c. Rugged "clamshell" design which protects the keyboard and display

Approximately 16cm x 8.5cm x 2.5cm (6.25" x 3.3" x 1") closed and about 11 ounces in weight.

d. A CGA-compatible, 80 * 25 character display with a "zoom" capability for text-based applications

The display screen can be zoomed between 80 * 25, 64 * 18, and 40 * 16 to enhance readability under low-light conditions (and to ensure compatibility with earlier HP95 applications which used a 40 * 16 user interface). Most built-in applications use the 64 * 18 mode for enhanced usability.

e. A PCMCIA (or PC-Card) release 2.0, type II plug-in card slot

Compatible with both memory cards and numerous I/O cards such as various modems, Network Interface Cards (NICs), pagers, Geo Positioning System (GPS) cards, etc. It supplies 5 or 12 volts. It can only supply 150mA, so it has trouble with certain cards which attempt to draw high current (such as older 28.8K modems). Most modems, SRAM cards, and ATA (Sundisk-style) flash cards work fine. Most Ethernet adapters and GPS receivers draw too much current to work, however there are solutions optimized for both applications that will work on the HP family of palmtops. Intel-style flash cards (like the ones the Apple Newton uses) are not supported. (Cochran, 1996)

f. An Infrared (IR) port for wireless communications

Limited to extremely close (1 to 2 inch distances) for serial communications, but can control IR devices from up to 20 feet away.

g. A serial port and UART for communications to a PC, modem, printer or other serial device

Serial to parallel converters are available from third party vendors to allow the direct printing of documents to standard parallel printers and devices. The onboard serial port uses a 10 pin HP proprietary pinout—a standard 9-pin RS-232 device can be interfaced to the HP200LX with an HP F1015A cable. Pinouts for this serial port are as follows:

HP200LX Pin #	Signal	RS-232 Connector Pin #
1	Data Carrier Detect	1
2	Receive Data	3
3	Transmit Data	2
4	Data Terminal Ready	6
5	Signal Ground	5
6	Data Set Ready	4
7	Request to Send	8
8	Clear to Send	7
9	Ring Indicator	9
10	Shield	5

Table 6.1. HP200LX Serial Port Pin Outs to RS-232.

The maximum speed for the internal UART (8250 serial port controller implemented in the 100/200LX) is 19,200 bps. With a buffered UART (NS16550, implemented in many PCMCIA modems) speeds up to 115,200 bps work fine. An important note: this does not mean that 100/200LX can actually send data at 115,200 bps. It means that palmtop can receive data at 115,200bps, without data loss.

h. Instant On capability

There is no requirement for a booting process everytime the machine is used since the CPU is placed in sleep mode with the previous session suspended.

i. A typewriter keyboard layout with "sticky" shift keys

The majority of standard PC keyboard keys are available. Although most users find it cramped, continued practice can generate typing rates up to 60 words per minute.

2. Software Features

The HP200LX consists of the following software applications built into ROM running as Execute-In-Place (XIP) applications:

a. A subset of MS-DOS 5.0

An optimized version of the popular PC operating system. Compatible with all programs that are designed to run on a CGA adapter.

b. CGA GUI-based task-switching operating system extension known as the "System Manager"

The System Manager allows the user to context switch among programs in memory without exiting applications. It is the core software under which all of the PIM applications run. It allows the suspension of one application to run another, keyboard macros, and data exchange between applications via a clipboard. The alarm clock and appointment manager will only wake up if the system manager is active. Certain third party applications (*.EXM files) are "System Manager Compliant." Applications which are not system manager compliant can still be run, either by exiting the system manager entirely, or by opening up another DOS shell under the system manager. (Cochran, 1996)

c. Filer Program

Provides basic file management functionality to include viewing, moving, copying, renaming and deleting files. Also provides a gateway to IR based communications; backup and restore operations; shelling to MS-DOS; etc.

d. Pocket Quicken

Intuit's popular personal financial management software.

e. Appointment Book

Powerful means of tracking appointments at specific dates and times; as well as lists of things to do. The ability to generate repeating appointments and to-do list items is one of areas which sets apart the HP from other similar products.

f. Phone Book

A custom instantiation of the built-in general database utility with predefined (yet customizable) fields for establishing a database of contact information.

g. Memo Editor

A text editor with automatic word wrap with both ASCII and a proprietary binary file format for formatting information such as typefaces, font sizes, tabs and margins, etc. The program also includes an outliner component.

h. Lotus 1-2-3

Generally compatible with the full version of 1-2-3 Release 2.4 except that some features like the install utility; landscape printing; color and mouse support; tutorial; translate utilities; etc. are not included in the palmtop version.

i. Calculator

A powerful and extensible HP calculator with predefined applications including time value of money and amortization; interest rate and currency conversions, one and two variable statistics, date calculations, graphing, and general mathematics.

j. Applications Manager

Provides a user-adjustable icon-based menu to allow a customized view of application programs to ease navigation and execution of user and built-in programs.

k. Data Communications

A general-purpose text-based telecommunications program with built-in VT100 and ANSI emulation as well as popular file transfer protocols such as Kermit, Xmodem, Ymodem, and Zmodem. Data communications reliability and speed has been significantly improved over the HP100LX predecessor of the HP200LX.

l. General Database

One of the more powerful PIM applications which allows the construction of custom database structures by non-programmers optimized for the palmtop platform. Field types include text, numbers, date/time, labels, group boxes and option buttons which correspond to the standard look and feel of the built-in applications. Programs are available in the public domain which allow the data interchange between the proprietary HP database format and various database structures such as comma-delimited text.

m. Note Taker

Allows the creation of a "Post-It" note oriented database on miscellaneous topics. Each memo can be up to 32KB in size along with pre-defined data fields for searching and editing short notes.

n. World Time

A database of cities including local time and Daylight Savings Time, graphic representation on a map with latitude and longitude, and telephone dialing prefixes. This database can be extended by the user by adding additional locations.

o. Miscellaneous Applications

Including a user options setup utility; a standard clipboard for cutting, copying and pasting; a stop watch; a macro program; cc:Mail for interfacing with its popular Email cousin; Laplink Remote for transferring files to other machines; and various games and demonstrations programs.

3. Connectivity with Desktop Platforms (Cochran, 1996)

For the 100LX, HP sells the connectivity pack, (HP F1021A) which comes with a serial cable (HP F1015A), a package of various adapters for different serial connections (HP F1023A), and software for the PC. The PC software includes versions of the 100LX PIM software (but not Lotus 1-2-3). The filer applications let you transfer files back and forth, and the redirector program lets you slowly use one machine's disk drive from the other machine.

A similar connectivity pack is available for the 200LX (F1021B includes English documentation and F1021C contains multilingual documentation). The 200LX connectivity pack includes software to integrate Pocket Quicken with Quicken for DOS or Windows, in addition to updated versions of the software in the 100LX connectivity pack. A "software-only" version of the 200LX connectivity pack is also available, for those users who already have cables and adaptors.

C. SHORTCOMINGS OF THE HP200LX

Like most mobile computers, the HP has a number of shortcomings besides the endemic cramped keyboards and hard-to-read screens typically found in this class of computing devices. Although the DOS-based software offers a plethora of programs and development tools, DOS offers only a relatively crude text-based user interface in contrast to the modern GUI platforms now emerging in the mobile computing industry. The lack of a pointing device also limits the utility of the HP200LX to work with graphic-oriented applications.

The HP has a relatively narrow operating temperature (0 to 50 degrees centigrade) which limits its use in a cold-weather environment without a heated harsh environment case. Like any low-cost consumer palmtop, the HP is vulnerable to moisture damage. Techniques such as water-proofing in ziplock containers offer a reasonable degree of protection against the elements.

Although the HP has proved its mettle in a variety of harsh environments, the device does suffer from a few endemic design flaws that include the tendency for the screen hinge to droop, and for columns of pixels to disappear on the left side of the screen. Both of these problems are relatively minor flaws in an otherwise great machine. Nonetheless, they can be extremely annoying to many users. Fortunately, HP offers one

of the most comprehensive and inclusive warranty programs in the industry. A palmtop which suffers from one of these maladies can be swapped out overnight at no cost.

D. HP PALMTOP PROGRAMMING ENVIRONMENTS

1. HP200LX Built-In Programming Tools

The HP200LX contains numerous programming tools and environments for the development of surprisingly sophisticated applications. For example, as the author's experience in Kuwait (Cumiskey, 1993) demonstrates, you can do quite a lot with the macro language built into Lotus 1-2-3. Furthermore, users have the keyboard macro application, the calculator's solver application, as well as the standard DOS programs such as DEBUG.EXE to compile assembly language programs and the DOS batch file interpreter to automate sequences of tasks.

2. General Programming Languages for the HP200LX

In general, anything that will run on a PC-XT including various flavors of C, C++, Pascal, Basic, LISP, Prolog, etc. will run on an HP200LX. Microsoft's QBASIC.EXE is not included in ROM, but will run if it's copied from a MS-DOS 5.0 machine. (Cochran, 1996)

The sample application developed in conjunction with this thesis was written in the C Language and compiled using the Borland C++ compiler Version 4.0 on a desktop development platform. Numerous other flavors of C and C++ will run on the HP200LX itself including Borland's Turbo C++ Version 2.0 (version 3.0 of Borland's compiler requires an Intel x286 level processor or higher).

In general, C is probably the best choice for general program development for the HP200LX (especially so if one is developing System Manager compliant programs). The Palmtop Developer's Guide being distributed by Thaddeus Computing contains full documentation on the internals of the HP Palmtops, plus software for developing system-manager compliant applications. Users will need to supply their own compiler and/or assembler to use PAL. (Cochran, 1996)

3. The HP Program Applications Library (PAL)

PAL is a set of C functions that will enable programmers to emulate the look & feel of the built-in applications of the HP100/200LX. PAL is copyrighted freeware and may be used to develop public domain, shareware, or commercial applications. A commercial version of PAL is currently being developed and should be released by late 1996.

PAL supports most of the popular C compilers including Borland Turbo C, Borland C++, Microsoft MSC, Microsoft QuickC, MIX PowerC, and Symantec C++. PAL includes a set of utilities that simplify development including a PalMake utility, a PAL dialog Editor, the PAL font editor, an HTML viewer, some conversion utilities, and the CGAGRAPH (TSR) that enables you to run any PAL written application on your desktop for debugging. (Kohl, 1996)

4. HP200-LX Database-Centric Programming

Numerous older environments (such as the classic Nantucket Clipper database-centric development environment) and libraries exist which extend the flexibility of the HP palmtop platform. Specialized libraries for C and other languages offer a great wealth of robust code to integrate into RAD projects. The Summer 87 version of Clipper offers a powerful capability to develop full-featured database applications on the palmtop. Unfortunately, the current release of Clipper 5.3 cannot run effectively on the underpowered 80186 CPU. Plug-and-play software for the HP abounds. Customized programs can be designed and burned into ROM XIP storage for execution on the palmtop without consuming precious internal RAM.

E. CRITICAL SOFTWARE FOR THE HP200LX

The HP200LX is the premier platform for many users options owing principally to its MS-DOS based operating system. Literally thousands of older programs from the glory days of DOS are available which enable the palmtop to perform wonderful things running on a pocket-sized platform. Foremost of these is the ability of the palmtop to communicate with other computers over a rich range of communication channels. Many of these programs have been updated and optimized for use with the HP palmtop. The following text will examine several of the options available to those desiring to use their HP palmtop for applications outside the domain of personal information management.

The HP200LX family of palmtops will only allow one MS-DOS session to run in conjunction with the XIP ROM-based System Manager suite of PIM applications. The Sunshine Software Company markets the Software Carousel memory management product which allows context switching with up to 12 simultaneously loaded DOS programs—allowing each program to use the maximum amount of memory (640K) without having to close the other programs. Even multiple System Manager sessions can be run in order to use more built-in applications simultaneously.

1. General Telecommunications Programs

Although the HP200LX comes with its own internal general communications program in ROM, that program is inadequate for most users. A program named *Commo* (currently in version 6.6) has worked for the author on a variety of platforms for almost a decade. The program is generally considered the premier telecommunications program among HP palmtop users. *Commo* is written entirely in assembly language, so it is very tight and fast with a broad array of internal file transfer protocols (including Zmodem, Ymodem and Xmodem) and a free-form dialing directory which can hold an unlimited number of entries.. It also has a rich scripting language which allows the creation of macros to automate online tasks. The program has excellent VT102 emulation which allows interfacing to UNIX based systems as well as other PCs. *Commo* supports up to eight serial ports with completely configurable port addresses, IRQ's, and speeds up to 115,200 bps. Automatic support also includes support for the 16550A buffered UART chip. *Commo* includes very nice features such as detailed usage logging, a 64Kb scrollbar buffer which gives instant replay of the text that has scrolled off the screen, and a capture file which saves all text from the screen to a disk file. (Drucker, 1996)

2. Internet Direct Connection Software

In order to avail yourself of the full range of Internet services (including access to the RedWeb system developed in this thesis research), it is necessary to obtain either a Point-to-Point Protocol (PPP) (Simpson, 1994) or a Serial Line Internet Protocol (SLIP) (Romkey, 1988) program for temporary direct connection to the Internet—usually via a dial-up communications channel. PPP and SLIP programs can either have internal dialer functionality, or use a general telecommunications program (such as *Commo*) to dialup an Internet Service Provider (ISP). The PPP or SLIP program is then run separately to establish the connection to the Internet once the account name and password have been

sent. There are numerous programs available that work well for the HP200LX such as “EtherPPP” for PPP connectivity and “Slipper” for SLIP connections. (Siig, 1996)

3. HTML Viewing Software: HV

One of the important strengths of HTML is that it is reasonably straightforward to write viewing software for a variety of platforms. This feature is especially important considering the HTML-centricity of the REDMAN application developed in this research (see Chapters VIII and IX). The excellent public domain software HV written by Andreas Garzotto in C and the PAL library were selected for use. HV allows the viewing ASCII text documents marked up with HTML tags for the specific needs of the REDMAN and RedWeb systems. It should be noted that this HV viewer does not contain a full-featured browser in the sense of conducting Hyper Text Transfer Protocol (HTTP) transactions with a server. Rather, it is strictly used for viewing saved HTML documents which have been transferred to the palmtop using a variety of communication channels discussed in this research. HV supports the display of CGA-based images on the palmtop's screen. (Garzotto, 1995)

4. World Wide Web (WWW) Browsers

The most popular web browser for MS-DOS systems is probably *DosLynx* developed by the University of Kansas. The program works well on the HP family of palmtops. *DosLynx* is restricted to just displaying text—not images. In addition to HTTP, *DosLynx* supports the File Transport Protocol (FTP), Gopher, UseNet newsgroups, the Wide Area Information Service (WAIS). *DOSLynx* requires an external dialer program to establish the shell-based connection before it can request HTTP documents from a web server. Each of the Internet software suites (discussed immediately below) also contain their own WWW browser. Lynx is public domain software available for many different operating systems. (University of Kansas, 1996)

5. Internet Software Suites

An Internet software suite is a package of client applications with a consistent user interface to offer access to the most popular Internet functions and services. The two leading suites for the HP family of palmtops include the “Minuet” and the “Net-Tamer” programs.

a. Minnesota Internet User's Essential Tool (MINUET)

Minuet works in conjunction with one of the direct connection PPP or SLIP programs discussed above. It includes a POP mail client, a Gopher client, a Telnet, FTP, and UseNet reader, and an internal WWW browser. (University of Minnesota, 1996)

b. Net-Tamer

By far the most popular Internet suite available to palmtop users, Net-Tamer includes an internal dialer program making it the only full-featured encapsulated Internet suite available which executes on the HP palmtop platform. Although the author prefers the HTML viewer provided by the modified HV program discussed earlier for viewing HTML documents (the Net-Tamer program is very slow to render HTML images), future versions of the Net-Tamer program will be better optimized for speed of execution on the HP palmtops. Net-Tamer provides access to Internet Email, WWW, Usenet newsgroups, FTP, and telnet. (Colston, 1996)

F. EXTENDING THE 200LX WITH ADD-ON HARDWARE

The HP family of palmtops has the most extensible architecture of all the available mobile computers. Some of the leading palmtop products are highlighted below in each of their applicable categories. These vendors have provided either HP palmtop specific products, or created extensions to their products are available to make them work well with the HP palmtops.

1. RAM and Clock Doubling Extensions

Although both increasing RAM and the clock rate of the HP200LX palmtops is possible, both of these techniques void the warranty of the product. Several companies including TechRAM Corporation and EduCalc sell 4-6MB internal RAM upgrade kits as well as internal clock doubling chips which increase the speed of the palmtop dramatically in conjunction with a RAM resident utility to keep accurate time. (EduCalc, 1996)

2. Parallel Port Adapters

The Quatech corporation offers a product called the SPP-100 which when placed into the HP200LX's PC Card slot allows you to connect the palmtop to parallel port printers, external storage drives, etc. (Quatech, 1996)

3. Flash Memory Cards

Numerous brands of PC Card-based flash cards are available, with the SanDisk (formerly SunDisk) Corporation being the most popular. S-RAM cards also work in the HP200LX, however, the price/performance ratio of these cards limits their popularity. Flashcards are available up to 85MB (170MB compressed) with even larger capacities expected. (Sandisk, 1996)

4. External Floppy Drives and Hard Drives

The Accurite Travel Floppy drive is a lightweight external system for accessing 3.5 high capacity floppy diskettes from the HP200LX. The Travel Floppy connects to the palmtop with a PassPort PC Card and includes all necessary device drivers to establish connectivity to the external drive. (Accurite, 1996)

The DISKDOCK system from Greystone Peripherals and MagicRAM, Inc. allows the interface of a high-speed and capacity (currently up to 340MB), low-cost 2 1/2" hard drive via the PC Card slot. Greystone also markets a PA-70D PalmTop Adapter which allows the HP200LX to interface with a variety of new 1.8 inch PCMCIA hard drives (such as the 130MB Maxtor MXL-105-III). (Greystone, 1996)

5. Geographic Positioning System (GPS) Sensors

The leader in mobile computing GPS sensors is Trimble Navigation. Although Trimble sells a PC-Card based GPS sensor, the card draws 850 mW in active mode, and therefore it is unsuitable for the HP family of palmtops without external power. A better solution is offered in the form of the Mobile GPS Locator 110—an external GPS sensor which connects to the palmtop via its standard serial port (Figure 6.4). This sensor tracks up to eight satellites with a one second update rate. There is a Trimble Mobile GPS Software Development Kit (SDK) available which has both DOS and Windows product design tools. Adding GPS functionality to the palmtop using this SDK is expected to be straightforward. (Trimble, 1996)

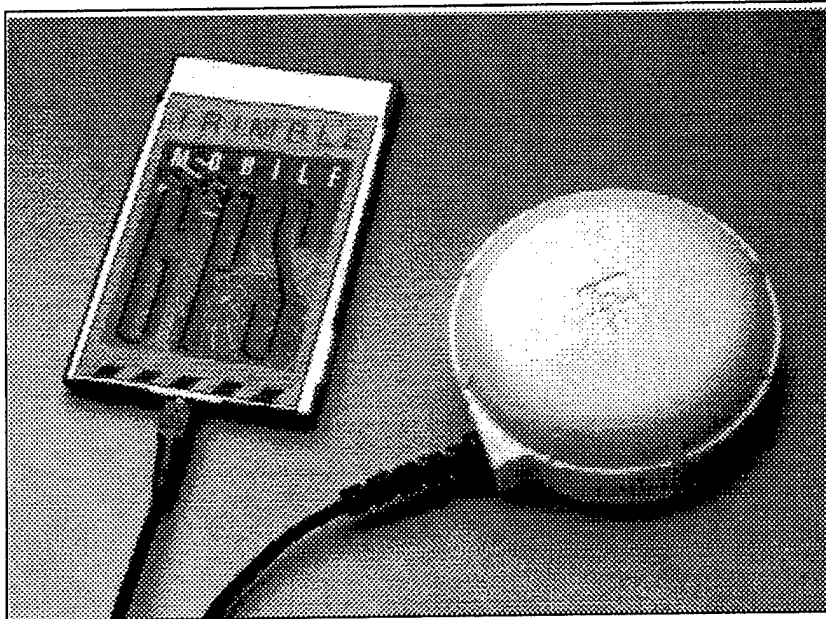


Figure 6.4. Trimble Mobile GPS Locator 110. (Trimble, 1995)

6. Wireless Packet Radio Modems

a. Megahertz Allpoints card

The Megahertz Allpoints card (Figure 6.5) is a Type II PC Card that works fine in conjunction with the HP family of palmtops for connectivity to various wide area Packet Radio service providers to provide two-way, real-time access to critical data. The PC Card bulges at the end to allow room for the card's 9-volt battery and 5-inch retractable and removable antenna. Each has a rechargeable nickel cadmium battery that lasts for 40 hours per charge. The card also comes with Wynd Communications Corp.'s WyndMail and RadioMail, wireless-service providers with individual gateways for one- and two-way communications. The system can connect to the radio network with low signal strengths. With even a minimal signal, the card works from moving vehicles and from a variety of stationary locations. The RAM Mobile Data wireless packet data network is available in over 90% of the U.S. population centers, as well as offering some limited international coverage (see Chapter V for a discussion of the RAM network). The card is low power—only requiring 12 mA in standby mode; 25 mA in receive mode; and 75 mA in transmit mode. The card transmits at two watts of power on the 896-902 MHz frequency and receives at 935-941 MHz. The card emulates the NSC 16550 UART chip

and used the Mobitex Asynchronous Communications (MASC) protocol to transmit data efficiently. (Megahertz , 1995)

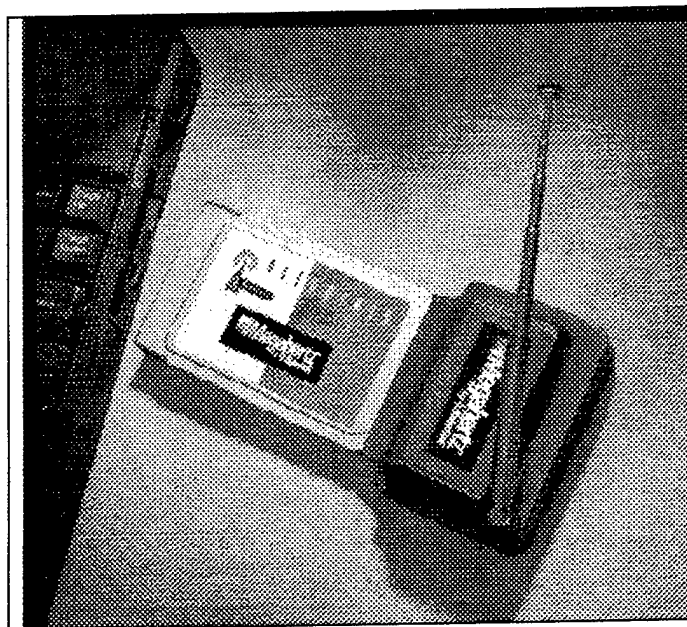


Figure 6.5. Megahertz AllPoints Card. (Megahertz, 1995)

b. Motorola 100D Personal Messenger Card

Motorola makes the Personal Messenger (PM) 100D card (Figure 6.6) which runs over the ARDIS network. The PM 100D is a similar card to the Megahertz AllPoints with RD-LAP 19.2 support. The radio transmits in the 806-825 MHz range and receives in the 851-870 MHz range with a transmission power of 1W and 8KB of RAM. The product weights only 0.34 lbs with the battery. (Motorola, 1995)

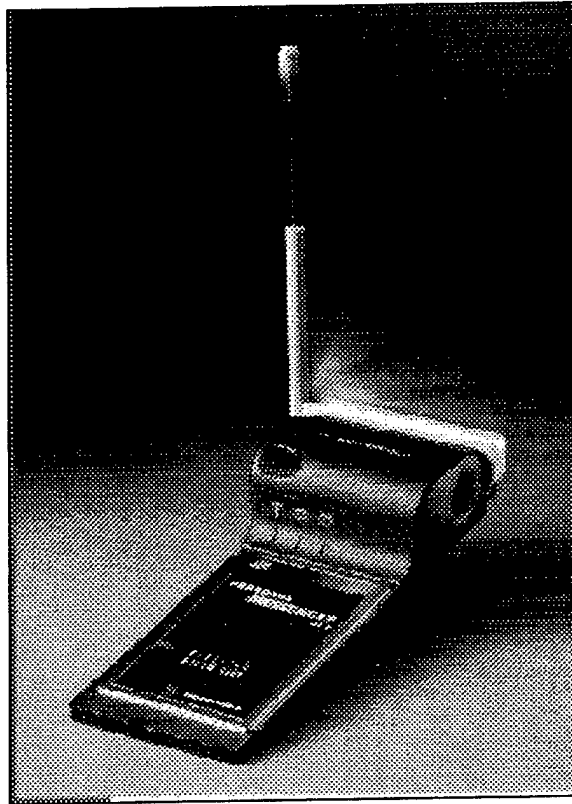


Figure 6.6. Motorola Personal Messenger 100D. (Motorola, 1995)

c. EnBlock R/F PalmStation Plus

The EnBlock R/F PalmStation Plus (Figure 6.7) is an integrated palmtop connectivity solution complete with docking station (similar to the older Sparcom Corporation devices), as well as a built-in wireline modem. Since wireless communication is such a battery draining application, the PalmStation Plus includes a rechargeable battery to allow from 10-12 hours of continuous use. The system also leaves the PC-Card slot free for flash memory and other products (such as a LAN adapter). The PalmStation Plus also includes an standard RS-232 serial I/O port of ease of connectivity to serial devices. Like the Motorola 100D card, the PalmStation Plus uses the Ardis RadioMail service for connectivity. This is an excellent product for palmtop use in stationary environments. (Enbloc, 1996)



Figure 6.7. EnBloc R/F PalmStationPlus. (Enbloc, 1996)

7. Pagers

As discussed in Chapter V, the Critical Link “Secure Page Receive” software allows the HP200LX palmtop to receive files (including images and REDMAN combat order packets) via existing commercial paging networks (Figure 6.8). It would also be possible to establish private networks without reliance on established civilian infrastructure.

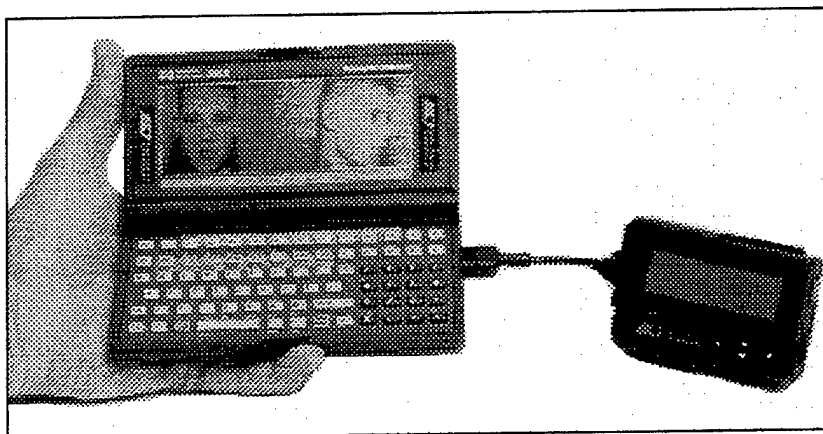


Figure 6.8. Data Critical Critical Link Connectivity. (Data Critical, 1996)

Both the Motorola Advisor and Tango 2-way pagers work well with the HP200LX to allow a broad range of connectivity options. These pagers are also described in Chapter V.

8. Ethernet LAN Adapters

The only Ethernet LAN card which works reliably in the HP200LX is the Silicom Ethernet IEEE 802.3 LAN 10-BaseT adapter includes an onboard ROM based software driver which does not take up any memory space on the palmtop. This PCMCIA adapter comes complete with a wide selection of ODI, NDIS and Packet software to support all major Network Operation Systems, including Novell Netware, Microsoft Networks, Banyan VINES, Artisoft LANtastic and most importantly TCP/IP. (Silicom, 1996)

9. PC-Card Modems

Virtually any external modem and most PC card based modems will work with the HP200LX. Since circuit-switched cellular connectivity is becoming popular, modems such as the AT&T KIT modem (discussed in Chapter V), as well as new modems from companies such as MegaHertz (Model XJ3288M and XJ3144M) are becoming available. Typically, the power requirements of 28.8 modems has made them impractical for palmtop usage—even when connectivity to AC adapters was employed. The modem would simply shut down. More recent 28.8 modems have been optimized for power consumption and work fine in the 200LX. Multi-function PC Cards have appeared such as EXP Computer's ThinFax 1414LXM modem which provides a fully integrated Fax/Modem with onboard fax software as well as up to 8MB of low power 5 volt flash RAM storage. (Boardman, 1996)

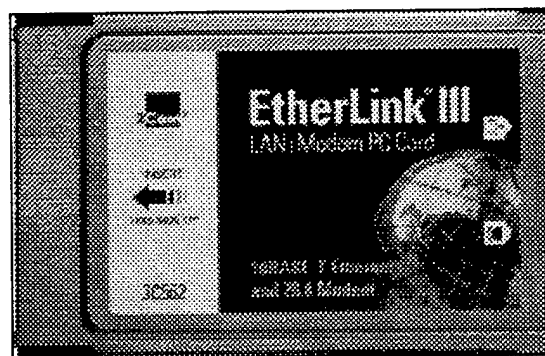


Figure 6.9. EtherLink III V.34 Modem / Ethernet LAN Card. (Boardman, 1996)

10. Cryptographic Cards

The current market leader in cryptographic products for mobile computers is the Fortezza Plus/LP card (Figure 6.10). This card is a low power 3.3 volt PCMCIA Type II card which provides 64Kbps single channel performance along with STU-III compatibility. Since the card only consumes 200 mW of power, it will run successfully in the HP family of palmtops to ensure data integrity, authentication and confidentiality of sensitive information on mobile computers. (Allied Signal, 1996)

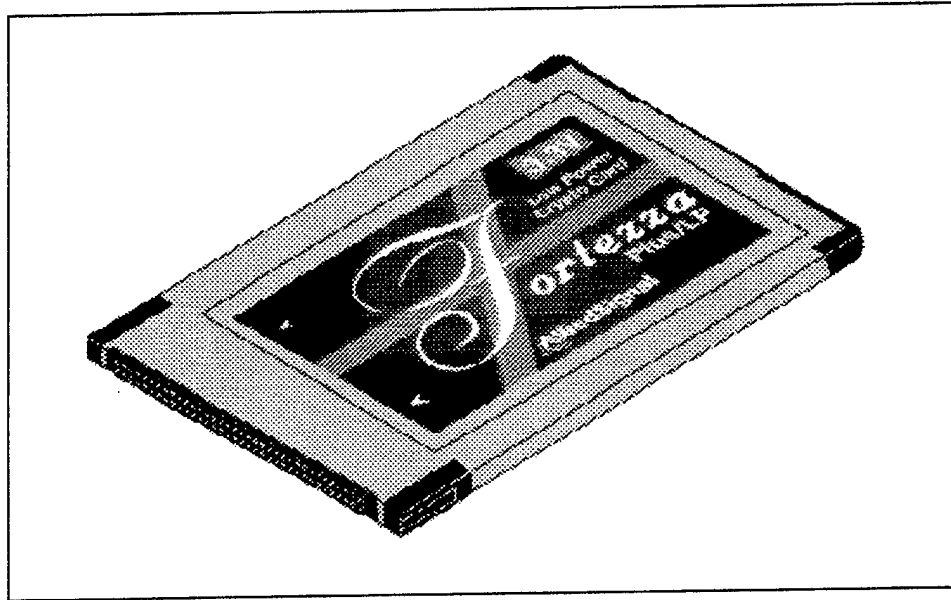


Figure 6.10. Allied Signal Fortezza Plus/LP Crypto Card. (Allied Signal, 1996)

11. PCMCIA Tactical Communications Interface Module (P-TCIM) Modem

The Science Applications International Corporation (SAIC) makes a PC-Card based variant of their popular TCIM tactical modem which interfaces with virtually all existing radios in the USMC inventory. The TCIM is a front-end communications processor with two high-performance Digital Signal Processors (DSPs) and provides up to 16Kbps throughput with the KY-57/SINGARS radio suite. The P-TCIM also interoperates using the emerging MIL-STD 188-220 protocol discussed in Chapter III. Although a software driver has not been made available for DOS in the PC-Card format, it can be done if RF connectivity becomes an important objective when integrating palmtops on the battlefield. (SAIC, 1996)

HP palmtops have proven themselves in military operations since their first appearance on the computing landscape. Palmtops have been used successfully in the jungles of Okinawa as data entry platforms for messaging systems working in conjunction with wireless LANs and High Frequency (HF) radio in reconnaissance battalions. As a member of the 1st Marine Expeditionary Force (MEF), the author has personally used the original HP palmtop (the HP 95LX) in the searing heat of Kuwait (Figure 6.11) (Cummiskey, 1993). During this period, a crude yet effective system for land navigation and target engagement was built using the internal programming capability of the palmtop computer. As a component part of this thesis, a much more full-featured and efficient version of this land navigation software has been developed in C and the PAL development library discussed above. This software appears in Appendix D.

[illegible]

103

In 1991 in the Republic of Kuwait, 2d Battalion, 7th Marines was assigned the mission of securing the port of Kuwait City; off-loading vehicles, weapons, logistical supplies and ammunition stored aboard Maritime Pre-Positioned Ships (MPS); and finally, conducting various tactical exercises in the remote, windswept desert to the northwest of Kuwait city. This exercise was conducted in conjunction with the retaliatory bombing of Baghdad for the aborted assassination attempt on our former President Bush. Kuwait was experiencing its "shamal" or windy season with winds up to 60 knots. 125° temperatures were not uncommon as these strong winds blew Kuwait's fine, gritty white sand everywhere. Despite the best efforts to protect the system, this palmtop eventually became covered with the sand. Sand was literally everywhere: in the serial port and AC power recesses, the PC card slot, and all over the keypad area. However, all this dirt didn't seem to faze the palmtop in the slightest. The HP palmtop never failed in the slightest way throughout the entire ordeal.

Later, it became apparent just how shock-proof the HP palmtops are after dropping my computer a few times while bouncing along the rough terrain in our Marine Amphibious Assault Vehicle (AAV) and High-Mobility Multi-purpose Wheeled Vehicle (HMMWV). The system was even dropped from a height of approximately eight feet onto the hard tarmac road while getting off a 5-ton truck. The HP took all this abuse, and came back for more.

Another obvious concern before deploying to Kuwait was the unavailability of electric outlets to recharge the palmtop's batteries (especially during the extended field training). Since Kuwait uses industrial-type 3-prong outlets as well as 220-240 Alternating Current (AC), the limited number of 110 AC converters the Marines had on-hand left my palmtop without its life-sustaining electricity. One thing Kuwait didn't lack, however, was a steady supply of sunshine. The possibility of harnessing the sun's power with a solar-cell AA battery charger was explored with a nicely designed and compact unit (part #22310) from the camping supply outfit, CampMor, located in Paramus, New Jersey. This three-ounce, \$12.00 charger allowed the author to tap into a continuous source of power for recharging the Millennium Nickel Cadmium (NiCad) 700 mAH batteries used in the palmtop. Although the recharger's instructions indicate that two batteries can be charged in as little as "4 to 6 hours," two full days of continuous exposure to the sun was necessary to fully power-up the batteries. A solar charging system can be an excellent way to provide both reliable and environmentally conscious power for any

palmtop--especially important for those of us who often find ourselves in remote parts of the world.

2. Land Navigation and Target Engagement with the Grid Calculator

The revised C program (called GridCalc) developed in conjunction with this research makes uses of standard trigonometric functions to calculate azimuth and distance from one 6-digit grid coordinate to another. Furthermore, as illustrated above, it can calculate a "target grid" given a range, bearing, and observer location. It can calculate compass directions in either degrees or mils, and will also accept distances in kilometers or nautical miles. The program also calculates resection and intersection coordinates and a time-speed-distance module. More importantly, the program will accept a default Grid-Magnetic Angle for ease of converting from grid-North to magnetic azimuths during the calculation process. Appendix D contains the source code for the GridCalc application.

Figure 6.12 depicts the opening screen of the GridCalc application. The ten button areas on the bottom of the screen correspond to the function keys on the HP200LX palmtop. The look and feel of the internal PIM applications are maintained by the PAL function library.

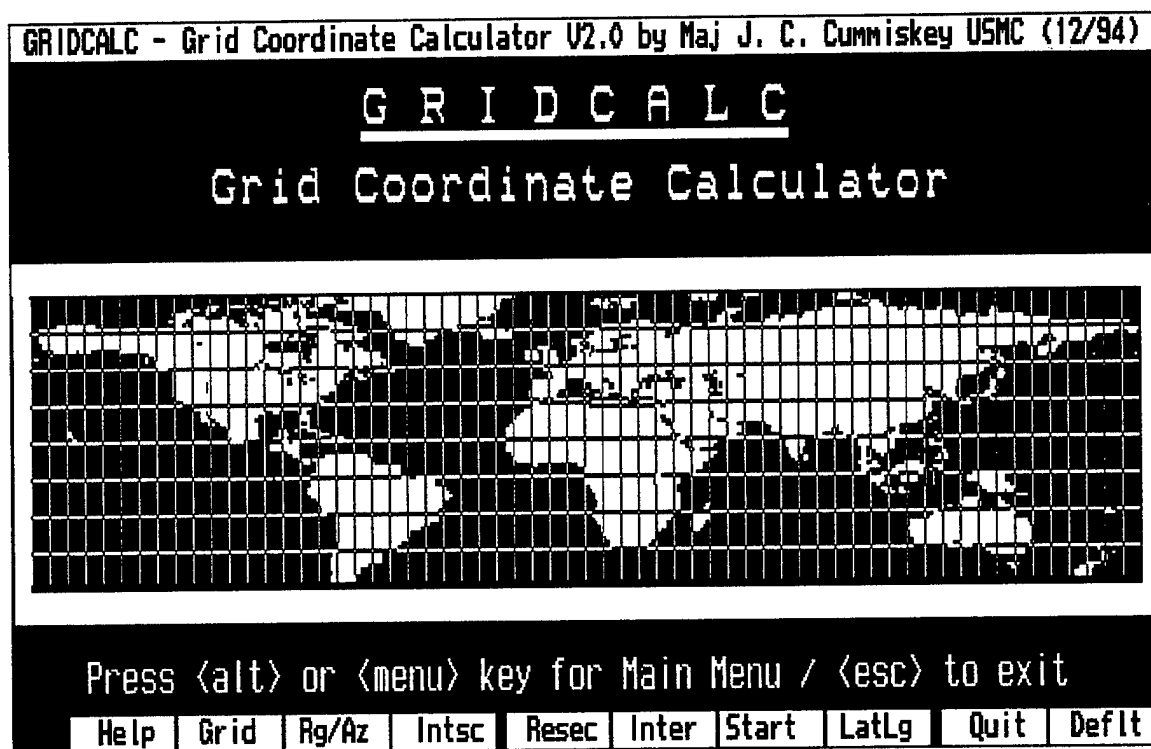


Figure 6.12. GridCalc 2.0 Opening Screen.

Figure 6.13 depicts the selection of the Range/Azimuth functionality from the main menu. This module calculates the target location of a enemy position given a range to the position; a magnetic azimuth towards the position; and a distance from the position.

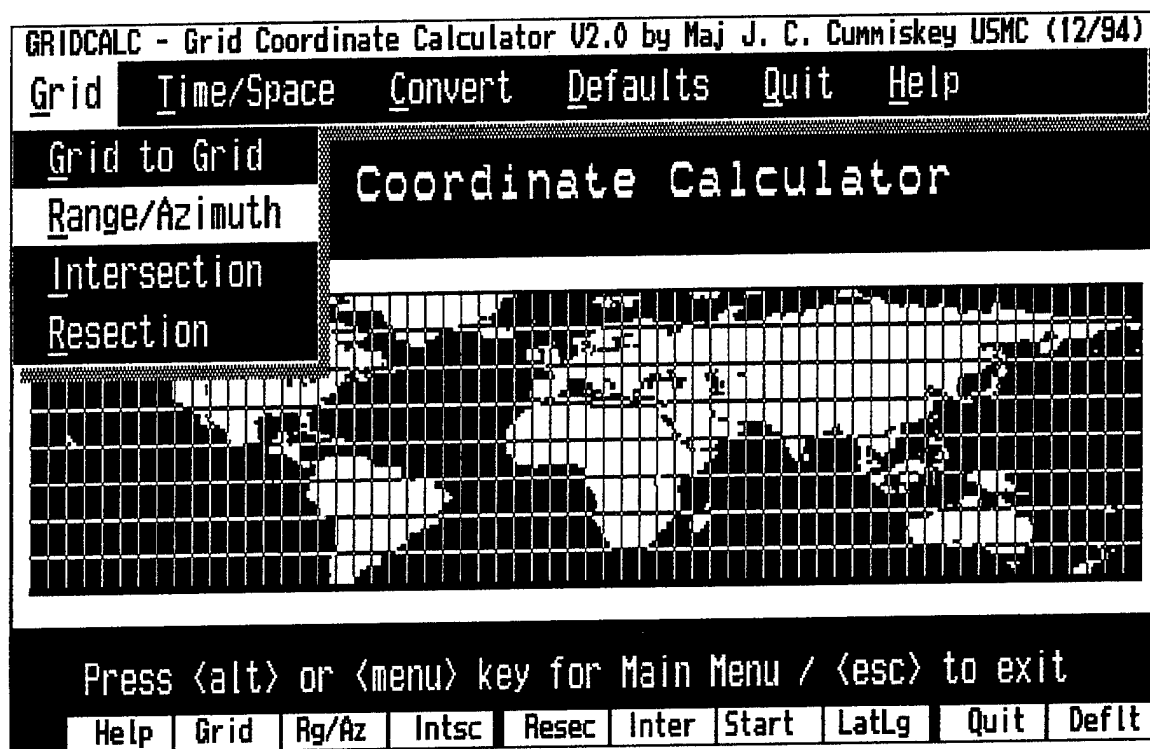


Figure 6.13. GridCalc 2.0 Range/Azimuth Menu Selection Screen.

Figure 6.14 depicts the data entry subform in the Range/Azimuth module. The user tabs through each of the three data entry fields before selecting the <OK> button to continue. In this instance, the user has selected a particular six digit (nearest 100 meters) grid coordinate for his current location. Since the user selected kilometers in the program defaults (vice nautical miles), the range to the target is entered in kilometers. Finally, the magnetic azimuth towards the target is entered in degrees. The user had the option of entering it in mils as selected by another program option. The Grid-Magnetic angle (representing the difference between grid north and magnetic north for the particular location in the world that the user is using the program) is also set in the program options module and appropriate adjustments are made to the calculation based upon this value.

GRIDCALC - Grid Coordinate Calculator V2.0 by Maj J. C. Cumiskey USMC (12/94)

GRIDCALC

Range / Azimuth Calculation

Enter GRID of your location: 123456

Enter range to target in (KM): 5.6

Enter mag azimuth in (Degs): 137

OK

Cancel

Press TAB to move from field to field

Press <alt> or <menu> key for Main Menu / <esc> to exit

Cancel OK

Figure 6.14. GridCalc 2.0 Range/Azimuth Data Entry SubForm Screen.

Figure 6.15 depicts the results of the Range/Azimuth calculations module. The target location is calculated to the nearest 8 digit (nearest 10 meters) grid coordinate for precision adjustment of fire if required.

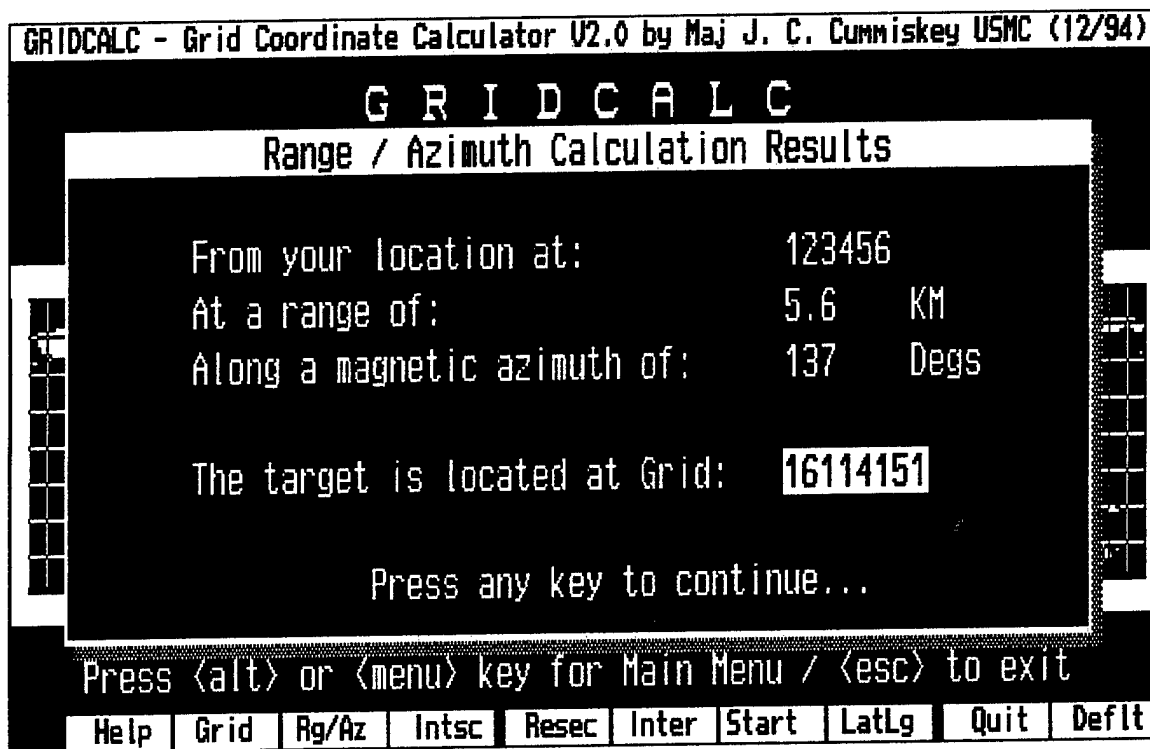


Figure 6.15. GridCalc 2.0 Range/Azimuth Calculation Results Screen.

The GridCalc system was found to increase the responsiveness of artillery units to requests for grid call-for-fire missions (vice the previously submitted polar missions) by more than 75% in actual live-fire exercises conducted in Kuwait and Twentynine Palms, California in 1993 and 1994.

H. SUMMARY

The HP family of palmtops offers a wealth of functionality and capability for employment on modern battlefields. Ample tools and programming environments exist to allow the rapid development of numerous applications to increase the efficiency of Marine Corps units. The GridCalc application, in particular, represents an example of how sorely needed tools in the hands of trained Marines can make a significant difference in the combat effectiveness of a unit.

VII. THE PEGASUS MACHINE AND OPERATING SYSTEM

A. INTRODUCTION

1. Non-Disclosure Agreement Specifications

The bulk of the contents of this chapter is protected by a legal agreement between the author and the Microsoft Corporation. The author is an authorized Independent Software Vendor (ISV) developer for the Pegasus Operating System and has agreed not to release certain proprietary and confidential aspects of Microsoft's unannounced products to persons outside the Naval Postgraduate School and the United States Department of Defense. It is expected that the formal release of Pegasus on or about 17 November 1996 will make the restrictions applicable to the contents of this document null and void. The author has received special authorization from the Microsoft Corporation to release the contents of this chapter in this thesis (Merwick, 1996). Until the formal release of Pegasus is finalized, no part of this document may be reproduced or transmitted in any form to unauthorized persons outside the domain of the Naval Postgraduate School and the U.S. Department of Defense.

2. The Pegasus OS: "32 bit Mobility with a Familiar User Interface"

Pegasus is the code name for the current 32-bit operating system (OS) under development by the Microsoft Corporation for the emerging hand-held Pegasus device. In September 1996, Microsoft will announced that the final product name for this OS will be "Windows CE" (where CE is believed to stand for "Consumer Electronics"). Nonetheless, "Pegasus" will be used throughout the remainder of this chapter since that was the name of system that was evaluated at the time of this writing. The Pegasus OS is based on a subset of the Win32 API. It currently consists of a Microsoft Windows operating system optimized for mobile palmtop platforms, a shell, and a suite of personal information manager (PIM) applications that are designed to run on a pocket-sized mobile computer weighing under one pound.

Microsoft intends to market its OS as a "mobile companion to a personal computer running Microsoft Windows 95 or Windows NT 4.0." According to Microsoft, there are more than 32 million mobile professionals in the United States alone. These mobile

professionals are away from their desktop computer more than 20% of the time. Furthermore, over 90% of these 32 million workers who own PCs use a Windows operating system. Clearly Pegasus will be an important and potentially very successful product. (Microsoft, 1996a)

The Pegasus OS has a consistent user interface that will be readily familiar to any user of the Windows 95 platform. It includes a taskbar, explorer, recycle bin, shortcuts, dialog boxes, and standard Win95 controls. Hence, the learning curve for users is expected to be minimal. The importance of this familiarity with the user interface cannot be overstated. In an industry that reinvents microprocessor hardware every eighteen months, Microsoft is being careful to ensure that users will not face the daunting prospect of having to learn a completely new operating system in order to use their new mobile computing device. Furthermore, the powerful and complete suite of development tools that Microsoft is providing to its 100 authorized developers will be instrumental in the success of Pegasus as a general purpose computing platform. It will be possible to port existing 32-bit applications to Pegasus for true Graphic User Interface (GUI) power on a mobile handheld platform.

Pegasus will come bundled with several PIM applications to include pocket versions of Microsoft's industry leading word processor and spreadsheet applications (Microsoft Word and Excel), a calendar, address book, and a task manager. Microsoft will provide desktop versions of each of these PIM applications with data conversion filters and synchronization functionality to allow the easy file transfer and reconciliation of information back and forth between platforms. These file and graphic filters will be in the form of dynamic-link libraries (DLLs) on the desktop PC to automatically convert files as they are transferred.

Pegasus will also ship with a built-in TCP/IP stack, various sockets including WinSock 2.0, Point-to-Point Protocol (PPP), and an internal HTML browser (Pegasus Internet Explorer) to facilitate the transfer of documents from the Internet to the Pegasus device using a variety of network protocols and channels. Subsets of the Telephony Application Programming Interface (TAPI) and Unimodem will also be provided for dialup communications (Unimodem is Microsoft's universal modem driver and telephony service provider which supports hundreds of the most popular fax/data modems). (Microsoft, 1996b)

B. PEGASUS HARDWARE

1. Hardware Overview

Microsoft has entered into agreements with numerous hardware manufacturers to develop the hardware which will run the Pegasus operating system. Their goal is to produce a device costing under \$500 running at least 100 MIPS on a single pair of AA batteries. Microsoft does not wish to repeat the mistakes of Pen Windows running on a single architecture: the ill-fated WinPad computer. Hence, they have designed the current Pegasus Software Development Kit (SDK) to support three distinct hardware platforms: the MIPS R4000, the Hitachi SH3, and the Intel x86 microprocessors. Currently, only the MIPS R4000 and Hitachi SH3 architectures are used in the early prototypes which have been made available to the developer community. According to Infoworld magazine, Casio Computer Corporation, Compaq Computer Corporation, Hewlett-Packard Co., NEC Technologies, Inc., Toshiba America Information Systems, Inc. all plan to unveil devices based on Pegasus this year (April, 1996).

The key to the success of Pegasus will be the powerful reduced instruction sets of these microprocessors. The simplified instruction sets of RISC based processors allow for a pipelined, superscalar design. Hence RISC processors often achieve many times the performance of traditional processors using comparable technology and the same clock rates. Because the instruction set of a RISC processor is relatively simple, it uses up much less space. Smaller chips allow a manufacturer to place more parts on a single silicon wafer, which can lower the per-chip cost dramatically. In the case of Pegasus machine, the ratio of cost to MIPS is extraordinarily low-on the order of \$0.20/MIPS. Furthermore, the low power requirements of Pegasus RISC microprocessors can be as small as 9mW/MIPS. (Microsoft, 1996b)

2. Power Management

The Pegasus microprocessor is designed to maximize battery life by offering different operation modes internal to the architecture (these modes include full speed, standby, and suspend). The first-level power management transparently suspends the CPU without application programs being aware of it. Microsoft estimates that when the average application is running, the CPU will be sleeping 90% of the time in standby mode. However, various alarms and interrupts such as a key press can wake up the device

instantly. The second level of power management turns off all hardware devices. Instant-On functionality is provided to eliminate the requirement for booting upon restart. Applications are not notified on suspend (however, driver software is notified to prevent critical processes from being interrupted). (Microsoft, 1996b)

According to the Pegasus SDK Reference Guide, "Standby mode uses less than one-tenth of full-speed power ... [and]... suspend mode uses less than one-thousandth of full speed power." (Microsoft, 1996a) It is expected that the average user will get weeks of use with a single charge of their batteries. Furthermore, a power failure during routine transactions (such as a database APJ call) theoretically will not corrupt the database. (Microsoft, 1996b)

3. Memory

Each Pegasus machine will offer a minimum of 4MB of ROM and 2MB of RAM. The SH3 architecture requires 4MB of ROM, whereas the MIPS architecture requires 5MB. PC Week Magazine reported in May 96 that some Pegasus machines may require as much as 5.5MB of ROM (Leach, 1996). The ROM or persistent memory will contain the operating system and the execute-in-place (XIP) suite of software applications mentioned previously. Hence ROM-based applications will launch quickly and require minimal memory overhead in RAM. (Microsoft, 1996b)

RAM will be used by the operating system for the stacks and heaps for running applications. It will be also be used to store other external application programs and user data. RAM on the Pegasus is also known as the "object store." It is expected that some manufacturers will offer upgradeable ROMs (either by replacing chips in sockets or via programmatic means), as well as larger RAM capacity. RAM, of course, will be volatile. Hence automated backup of data will be initiated whenever connectivity is established between the mobile computer and its parent desktop unit. (Microsoft, 1996b)

Like most modem computers, the Pegasus device will maintain two hardware timers-one for high resolution scheduling and timer events, and the other for low resolution requirements (such as maintaining the system date and time). Each Pegasus machine will have a hardware reset button to trigger a non-maskable reset. Cold boots of the Pegasus device can only be performed by removing both batteries for approximately one minute to ensure the object store is completely erased. (Microsoft, 1996b)

4. Pegasus Input Devices

The designers of the Pegasus machine feel that any successful mobile machine must incorporate a keyboard. The alternative input devices such as handwriting and voice recognition which are available today are still too slow and error-prone to provide a robust means of data entry. However, the keyboard represents a fundamental dilemma in the design of any mobile computing device. Users want a large full-sized keyboard with all their familiar keys in the same place as their desktop. However, they also want to be able to slip their mobile computer into their pocket. Pegasus compromises by offering a "QWERTY" style very similar to its desktop counterpart. Nonetheless, a few of the less common keys familiar to desktop users will not be available on the Pegasus due to space constraints.

A continuous resistive touch panel is embedded in the LCD panel which will support the use of an alternative input device (typically a plastic stylus) for menu navigation and graphic applications. A mouse is not supported on the Pegasus as the plastic stylus is designed to replace it. Although the Pegasus does not natively support handwriting recognition, Microsoft intends to build this into subsequent releases (see "Pegasus Operating System" below for a discussion of future handwriting recognition capability).

Pegasus machines are not currently anticipated to include an audio input device.

5. Pegasus Output Devices

One of the shortcomings of the Pegasus machine in its current incarnation is the limitations of its 480 * 240 liquid crystal display (LCD). Due to power considerations, the screen is not backlit and can be very difficult to read under certain lighting conditions. Furthermore, the LCD is monochrome and will support only four levels of gray. Microsoft is working with its hardware partners to develop finer resolution LCDs and color displays as power management technology permits their use. Low power backlighting or reflective LCD technology is also a possibility in future generations of the machine.

Each Pegasus device will have a light emitting diode (LED) on the exterior of its case which can be used by software applications to signal an event (such as the arrival of a

new tactical message on the battlefield). The user can press a programmer defined button to turn off the blinking LED and continue operation.

All Pegasus machines will offer audio output hardware (including a loudspeaker) which will support industry standard WAV formatted sound files as well as the possibility of other formats.

6. Pegasus Serial Ports

All Pegasus machines will have a proprietary serial port which supports Win32 serial communications. This port will be the primary means of communication between the desktop machine and the Pegasus device in order to install software and data, and synchronize and back up PIM data. Some manufacturers will offer a docking cradle similar to Palm's "Pilot" PDA for ease of use which can also be used to power the Pegasus device and recharge its batteries.

A serial cable will be the most common means of connecting the Pegasus machine to another PC. The serial port can be used for connecting to the myriad of devices that a mobile professional may want to use such as external modems, printers and other computers. Oddly, the current release of the Pegasus OS does not support Pegasus-to-Pegasus communications through the serial port.

7. Pegasus Infrared Communications

Each Pegasus machine will have an Infrared Data Association (IRDA) compatible port which implements the lower two layers of the IRDA protocol. This will be the primary means of communicating between two different Pegasus devices. The currently supported release of the Pegasus OS does not allow communication between other infrared enabled devices from the Pegasus IRDA port. The IRDA physical layer specification provides communication over a distance of one meter, with a minimum 15 (+/-) degree cone from the center of the transmitting device. The specification also defines the wavelength of the transceiver. Note that while serial ports are capable of full-duplex communication, the IRDA interface is limited to half-duplex operation due to interference from the reflected transmit signal. (Microsoft, 1996b)

8. Pegasus Hardware Expansion Slot

Although some Pegasus manufacturers may elect to build internal modems, cell phones, and pagers inside their units, most will opt to employ the standard PC-Card or PCMCIA slot found in the Pegasus. All of the prototypes this author has seen to date include a Type II expansion slot for communication devices. This slot supports a subset of the standard PCMCIA card/socket services for modems. However, the slot does not currently support flash card memory for increased storage capacity. Microsoft promises flash card support in their next version of the OS.

Support for two-way pagers, wireless networks, cellular modems and Geographic Positioning System (GPS) navigation and location devices are all expected to be available using the PC Card peripheral slot. Multi-function cards popularized in the notebook computer market will almost certainly find a place on the Pegasus device.

9. Hardware Preview Program

Microsoft has recently announced a hardware preview program to allow developers an opportunity to perform complete testing and development of Pegasus applications. The suite of platforms included in this preview include Pegasus machine prototypes manufactured by Casio, LG Electronics, and NEC. (Microsoft, 1996a)

C. PEGASUS OPERATING SYSTEM

1. Software Overview

The Pegasus OS supports the multi-threaded Win32 programming model and uses the standard Windows Application Programming Interface (API) model common to 32-bit development. However, any OS must fit within the minimum 4 megabytes of ROM of the Pegasus. Hence there are fewer functions and structures offered in the Pegasus OS in contrast to the rich array of Win32 API functions supported by Windows 95 and NT. (Microsoft, 1996b)

Although Windows 95 supports over 1000 function calls, Pegasus supports less than 500. Major missing APIs include ActiveX, DirectX (which allows the direct access of the enhanced features of computer hardware such as video and sound cards--an important capability for Pegasus use as a game playing platform), Object Linking and

Embedding (OLE), Component Object Model (COM), Messaging API (MAPI), Open Database Connectivity (ODBC), Dynamic Data Exchange (DDE), and multimedia support. (Microsoft, 1996b)

The hardware constraints of the Pegasus machines dictate many of the compromises made in the Pegasus OS. The Pegasus machine's limited memory, power, small monochrome output display, and touch-screen pen input all contribute to the challenges facing the Pegasus software designer. Additionally, several Win32 API function calls have been modified to accommodate the new platform. Of course, some new API calls have been added to access some of the Pegasus machine's new functionality.

The Pegasus OS tries hard to emulate the Windows 95 look and feel by providing common Win32 controls such as the scroll bar, button, check, radio, static, edit, group, combo, listbox, gauge, status bar, image list, listview, treeview, tab, trackbar, progress, and property sheets. (Microsoft, 1996a)

2. Operating System Components

The Operating system consists of several primary components:

a. Kernel

The real-time kernel is small, fast and portable across the three classes of microprocessors mentioned above. Porting among the different architectures is done internally--there is no exposed API. Virtual memory is supported, but there is no demand paging yet. The kernel takes up less than 64K of ROM. It has a single protected address space with a limit of 32 concurrent processes. The number of threads, however, is limited only by RAM. (Microsoft, 1996b)

b. Windows and Event Managers

The Window and Events Managers are Pegasus's equivalent of the Win32 USER APIs. However, unlike USER, there is no support for owned or topmost windows with the Pegasus windows manager. Also, cascading menus and non-client messages are not supported. The event manager uses the standard Win32 messaging model, however no cursor support is provided. Furthermore, events like key presses cannot be hooked by the OS (in order to avoid degradation of performance since messages are sent across multiple process streams). (Richter, 1995)

c. Graphical Device Interface (GDI)

There are two types of device contexts (DC) including the screen DC with 2 bits per pixel (BPP) and the memory DC with either 1 or 2 BPP. Printing is not supported from the GDI, nor are coordinate transformations. (Microsoft, 1996b)

d. File System, Registry and Database

These components are discussed below under "Object Store."

3. Characteristics of PEG/OS

The Pegasus OS (Peg/OS) offers true preemptive multitasking with its capability to manage multiple processes concurrently (with the possibility of multiple threads within each process). Peg/OS gives the appearance of running programs simultaneously by allocating processing time to each thread based upon a priority. Peg/OS recognizes seven different levels of priority grouped into three classifications as shown in Table 7.1. It is recommended by the Microsoft Pegasus development team that priority levels never be explicitly set by the programmer unless absolutely necessary. This is especially true because threads which run at a higher level or priority are never preempted by a thread running at a lower level. (Microsoft, 1996b)

Priority Level	Classification
Time Critical	Interrupt
Highest	Interrupt
Above Normal	Main
Normal	Main
Below Normal	Main
Lowest	Main
Idle	Idle

Table 7.1. Peg/OS Thread Priority Levels. (Microsoft, 1996b)

Peg/OS can use dynamic-link libraries to allow multiple applications to simultaneously share common routines and minimize code size in light of the severe constraints on storage. The Peg/OS Kernel (NK.EXE) can also run routines in re-entrant mode, hence some of the Windows 95 DLL functions have been incorporated into the kernel to eliminate the overhead of a library function call. (Microsoft, 1996b)

4. Object Store

Any portion of RAM not used by the OS and executing programs is called the "object store." Since a Pegasus machine does not have a traditional direct-access storage device (DASD), Peg/OS uses the object store to store external applications and the runtime data they require (as well as the PIM data created by the ROM-based PIM applications). The applications stored in the object store are decompressed into RAM and executed. The entire RAM area is persistent and if there is a memory fault, the object store will be rolled back to a consistent state. (Microsoft, 1996b)

The balance between storage and the portion of RAM used to run Peg/OS processes can be dynamically partitioned by the user using the System Control Panel. If the user selects the default partition of 1MB for both object store and program memory, the system and shell together will use approximately 600KB of this RAM, leaving approximately 424KB for applications and the full 1MB for the object store. The Pegasus supports four main groups of APIs. (Microsoft, 1996b)

a. System Registry API

The Pegasus system registry is similar to that used in the Windows NT and 95 operating systems.

b. File System API

An application program can use the file system API to create folders and data files in the object store, as well as read and write data to these files. File system integrity is "guaranteed" across power failures. The directory structure is similar to standard Windows, however, there is no concept of "path." (Microsoft, 1996b)

c. Database System API

Peg/OS contains its own unique flat-file database API which contains functions that an application can use to create and manipulate databases. Each database is a collection of objects consisting of properties and values. Objects are assigned a unique object identifier and can be sorted on any of four properties. Databases contain addresses, user information, system configuration, and application-specific objects. Field types are currently limited to integer, string, time/date, and a binary-large-object (blob) array of bytes. (Microsoft, 1996b)

d Remote Procedure Call (RPC) API

PC applications communicate with the Pegasus machine via RPC running over Sockets, PPP and TCP/IP. Furthermore all database, file system and registry calls are remote capable. The RPC can only be initiated from the desktop PC in version 1.0 of Pegasus. Hence, there is currently no remote browsing from a Pegasus machine available. (Microsoft, 1996b)

5. Memory Management

The relatively severe memory constraints of the Pegasus device make memory management critical to the success of any application. Clever use of the limited RAM allows multiple applications to run in less than 250K of memory. Peg/OS employs virtual memory which is always allocated in page sizes dependent on the architecture of the device (1 to 4KB). 1KB pages should be used whenever possible to eliminate wasted space due to rounding. Memory pools available in Peg/OS include: (Microsoft, 1996b)

a. Stack

Small data items that exist for the life of a specific function should be placed on the stack.

b. Static Data

Data items which last the lifetime of the program should be placed in the static data section of the application (providing there is sufficient room).

c. Virtual Memory

Programmers should select virtual memory for a single allocation of multiple pages.

d Default Heap

For small items with random overlapping lifetimes, the default heap should be used.

e. Separate Heaps

For a set of small data items with the same lifetime, a separate heap should be used.

Since RAM is always at a premium on a Pegasus machines, handling low memory situations is critical to the success of the OS. The system uses WM-HIBERNATE messages as the primary mechanism for asking applications to free memory. Hence, the system supports shutting down and restarting applications without the user knowing when memory is low. (Microsoft, 1996b)

6. Peg/OS Messages

Similar to Windows 95 and NT, Peg/OS is an event-driven operating system. Applications do not make calls to Peg/OS to obtain input; rather they wait for the operating system to pass input to them. The OS waits for input from the keyboard, touchscreen, serial port, or PC card slot to arrive and then redirects it via the notification API to an event handler looking for that message. That is why the object orientation of C++ is well suited to programming for the Win32 API. Each message is communicated in a standard message structure called "MSG." This API message structure consists of the following fields: (Microsoft, 1996b)

- a. The window handle of the window receiving the message-*
- b. The message identifier which determines how the message will be processed*
- c. Message parameters specifying data used for processing the message*
- d. The time the message was placed in the queue*
- e. The coordinates of the last position touched on the touch-screen*

7. Future Directions for Peg/OS

Microsoft is currently expecting to add the follow functionality to Peg/OS in subsequent releases of the product. Pegasus 1.5 (with ActiveX support) is expected to be delivered by the 1st quarter of 1997. Some of the following functionality may be in the release version, however it is not currently supported in the pre-release version. (Microsoft, 1996b)

a. Demand paging for non-ROM applications

b. PC card File Allocation Table (FAT) support

Flash and Static RAM (S-RAM) cards will eventually be added to minimize the storage constraints of the current Pegasus machine prototypes. These FAT file systems will be mountable and hot-swappable.

c. Minicard support

Minicards are half-size PC Cards for storing and transferring digital files. The Compact Flash (CF) Association is the first group to announce a standard for these cards based on technology from SanDisk. Measuring 1.7 by 1.4 by 0.13 inches, CF cards can be used in a traditional PC Card slot with an adapter. The only device currently using the half-size CF standard at the time of this writing is the IBM Palm Top PC II 0. A competing half-size form factor--1.5 by 1.3 by 0.13 inches--is under development by a consortium made up of Intel, Advanced Micro Devices, Fujitsu, and Sharp.

d. Multiple screen sizes and color support

Microsoft estimates that support for color screens will be available by mid to late 1997.

e. OLE, COM, DirectX, and ActiveX support

f. Printing from IR and over the network

g. Remote network browsing from the Pegasus

h. Inking and handwriting recognition

Although Pegasus Release 1.0 will not incorporate native hand-writing recognition, Microsoft acquired Aha Software in early 1996. It is anticipated that the innovative handwriting technology of Aha will be built into subsequent releases of Pegasus. The Aha InkWriter handwriting engine lets you write naturally and expressively with a pen on your mobile or desktop computer while letting you insert paragraphs, lists, mark-ups, and drawing without stopping to apply styles or switch modes. In essence, it lets you edit your handwriting like a word processor.

i. Better interprocess synchronization mechanisms (mutexes, etc.)

j. Inclusion of Universal Serial Bus support

Adding a Universal Serial Bus (USB) port on the Pegasus machine will eliminate the restrictions imposed by the single PC-Card expansion slot. The USB is the emerging standard for Plug and Play connectivity that uses a single standard connector for all external peripherals. USB supports 12Mbps throughput (as opposed to the bottleneck of 115Kbps of RS-232 serial communications) and provides for up to 127 simultaneous devices daisy-chained off a single port.

D. MILITARY USE OF PEGASUS INTERNAL APPLICATIONS

1. Word Processing, Spreadsheet and Personal Information Management

For the same reasons that Microsoft is hoping that the Pegasus machine will be successful in the mobile professional market, the Pegasus is likely to prove equally successful in the military market. After all, who is more mobile than a soldier, sailor, Marine or airman? The Pocket Word and Pocket Excel applications will both enhance

productivity for military users of Pegasus-especially for those who use the desktop equivalents of these products already on their larger desk-bound computers.

Managing personal data is the cornerstone of any successful military leader. The Pegasus' ability to automate much of the drudgery of maintaining appointments, tasks, and address book information will be a critical component of its functionality. Pegasus' automated synchronization with the desktop PIM applications will encourage continual backup of a user's data-an important aspect of automated data processing security that is often neglected by most users employing traditional methods of managing their personal data.

2. Pegasus Mail (PMail)

PMail is provided in the ROM of each Pegasus device. It is a Simple Mail Transport Protocol (SMTP) and Point of Presence (POP3) mail client with remote mail functionality. It includes pluggable transports acting as the middleware between the WinSock layer and the Pmail application. Although it currently does not support binary file attachment, this functionality is expected to be added soon. (Microsoft, 1996b)

Applications of PMail both in garrison and on the battlefield are clear. Using standard protocols, the Pegasus device will be used to access mail servers over the entire spectrum of communication channels. The low cost, small power consumption, and high MIPS performance makes the Pegasus machine an attractive option as a communications platform for transmitting military communications among widely dispersed units.

3. Pegasus Internet Explorer (PIE)

PIE supports all current Hyper Text Markup Language (HTML) Level 2.0 tags including forms support, tables, and audio files. Dynamic JPEG/GIF scaling and color conversion is also provided for the display of inline images. "Mailto" support for <A HREF> links is provided with hooks to the PMail application. Support for registering limited external helper applications is also available.

Microsoft plans to enhance PIE generally in lockstep with the development of their flagship HTML browser product, Microsoft Internet Explorer. The release of version 3.0 of the Internet Explorer on 28 May 1996 suggests that future versions of PIE will include support for HTML 3.0 tags, tables, frames, math and style sheets. ActiveX and Java

embedding should also be on the horizon. This will open up new vistas of opportunity for an 11-ounce pocket computer. Client side scripting for Visual Basic and JavaScript is also planned. However, the limited memory and multimedia capabilities of a current Pegasus machine dictates that there will be only limited OLE embedding in PIE.

If the Pegasus machine is fully integrated into the military way of warfighting, PIE will almost certainly become a fundamental part of communicating military messages owing to the superiority of the platform independence of HTML. As discussed in other sections of this thesis, the use of Hyper Text Transport Protocol (HTTP) servers in conjunction with back-end military databases (such as the REDMAN application) will be the foundation of military communication on tomorrow's battlefields.

4. Pegasus Help

Pegasus uses an application called "Pocket Help" to display help files for Peg/OS applications. These help files are stored in standard HTML format. Hence, the Pocket Help program is really an extension of the PIE HTML viewer which is generalized for the browsing of external HTML files via wired and wireless networks. Help files are authored using standard ASCII editors or specialized HTML authoring tools (such as the *Hotdog* HTML tag editing program preferred by the author).

E. PROGRAMMING FOR THE PEG/OS

1. Introduction

Applications for the Pegasus device are built with a Windows-based IDE using a special cross-development edition of Visual C++ for Pegasus running on the desktop. The Pegasus Software Development Kit (SDK) and Device Driver Kit (DDK) work in conjunction with a standard version of Microsoft's Visual C++ version 4.0 by optimizing the Visual C++ Developer Studio for Pegasus machine development. Curiously, although Pegasus applications can (and should) be written in C++ to take advantage of the object orientation of the Win32 model, all the demonstration programs provided with the SDK are written in C. (Microsoft, 1995)

As of this writing, the desktop platform for Pegasus development is Windows NT 4.0 Beta II. However, currently Win NT 4.0 will only support the x86 architecture for Pegasus development. The July release of Windows NT is expected to also support the

MIPS architecture. The Windows 95 platform offers only limited Pegasus emulation support.

The cross-development edition of Visual C++ provides multiple targets for Pegasus machines and allows the majority of development to be conducted on the desktop via the SDK emulation package. The development environment includes a remote debugger via a dedicated debug port on each device. Future releases may use WinSock over the standard serial port to eliminate this special debug port.

Porting existing Win32 applications will involve extensive UI work, elimination of non-supported API calls, and conversion of all strings to Unicode format (16 bit characters vice 8-bit) mandated by the Pegasus OS. Furthermore, all resources such as bit map images and icons must be converted to their Pegasus equivalents. Perhaps half of these tasks can be automated, so nontrivial programmer effort is needed to perform conversions to Pegasus.

2. Pegasus SDK and DDK Utilities

Tools provided with the SDK include a device display examination utility (Zoomin), a device registry editor (RegEdit), an application message viewing utility (Spy), an application resource tracker (Memview), and a system process view and control (pView). Conspicuously absent in the emulator is a internal means of simulating the severe memory constraints of the Pegasus device. Hence, final testing of each application must always be done on the device itself. This is a significant SDK limitation.

3. Pegasus Programming Fundamentals

All Peg/OS programs are written for execution in the adapted Pegasus GUI popularized by Windows 95. The Peg/OS memory model requires each application to fit entirely within memory. Applications should be designed to run in no larger a space than 100 MB of RAM whenever possible.

The area available to an application is called the client area. The remainder of the window is taken up by the command bar and the task bar. This single command bar replaces the separate nested menus and tool bars of Windows 95 in order to maximize the client area available to the user. Cascading menus or menu bitmaps are also not currently supported.

Although most Peg/OS applications are designed to run full-screen, smaller Pegasus windows can be moved around the client area (currently, however they cannot be resized). Since one of the principal features of the Win32 API is device independence, Peg/OS uses the Graphic Device Interface (GDI) to manipulate objects on the screen. Features such as metafiles, TrueType fonts, custom palettes and other functions are not supported due to the limitations of the hardware.

The non-sizable windows, tiny screen, limited memory and API set all dictate a different approach to user interface (UI) design for the Pegasus. The pen stylus serves as the left mouse button. Currently, the right mouse button is not directly supported, although the Alt + <tap> convention can be used to simulate a right mouse click. Cursor icons cannot be changed to prompt the user.

The dynamics of how the Pegasus will be used is also critical to understanding how applications ought to be built. Use of the Pegasus will often be "on the run" as soldiers or mobile workers move from place to place. Hence, complicated keyboard input must be minimized to allow users to successfully interact with the application. Lighting conditions will often be dim, precluding the use of convoluted graphic images or tiny fonts. The lack of screen real estate may dictate a shift in design principles when crafting the UI for a Pegasus application. Functionality must be minimized whenever possible. Reading information is much more important than writing--the Pegasus is designed to be primarily a data access device.

F. SUMMARY

The emerging Pegasus machine and its optimized Peg/OS offers an exciting alternative to any existing mobile computing platform currently on the market. Pegasus retains the portability and extended battery life so critical to these devices, yet offers an unparalleled price/performance ratio. The power of a commercial off-the-shelf (COTS) 100-MIPS RISC microprocessor in conjunction with the obvious benefits of the Win32 API and Win95 UI raises the bar on the utility of these pocket sized devices. Nowhere will this be more evident than in the domain of military mobile data communications. The Pegasus machine, and its successors, offers a level of functionality to the military professional that will be significant in realizing the promise of tomorrow's digital

battlefield. Pegasus is the Digital Messaging Transfer Device (DTMD) of choice on today's military landscape.

VIII. THE RAPID ELECTRONIC DELIVERY OF MESSAGES OVER ASYNCHRONOUS NETWORKS (REDMAN)

A. INTRODUCTION

The Rapid Electronic Delivery of Messages over Asynchronous Networks (REDMAN) software developed herein is the cornerstone of the mobile computing research contained in this thesis. It is a proof of concept application, and represents a prototype to serve as the foundation for future development per the remaining work identified in Chapter X. Although REDMAN is intended to be a generic system for transmitting general messages among military units on the joint battlefield, this thesis focuses on the specific transmission of combat order message packets from the U.S. Marine Corps infantry battalion level to the fireteam level via a wide range of digital communication channels. REDMAN was developed in part based upon numerous interviews and briefings conducted with infantry Marines at the School of Infantry, Camp Pendleton and the Marine Corps Security Forces, North Island in 1996.

Figure 8.1 depicts the flow of combat information from DACT to DACT integrating palmtops at the lower levels of command in an infantry battalion. The REDMAN discussion contained in this chapter focuses on the design and implementation of software to move platform independent combat orders over these varied transmission channels. REDMAN also demonstrates the viability of integrating low-cost palmtop computers with the emerging USMC DACT program discussed in Chapter III. Mobile computers can effectively augment a DACT infrastructure. As discussed earlier, the DACT is heavy and requires RF connectivity to realize the emerging vision of military radios. REDMAN is designed to exploit commercial communication channels whenever possible, as well as take advantage of cutting-edge technologies such as diffuse and directed IR in tactical huddle scenarios.

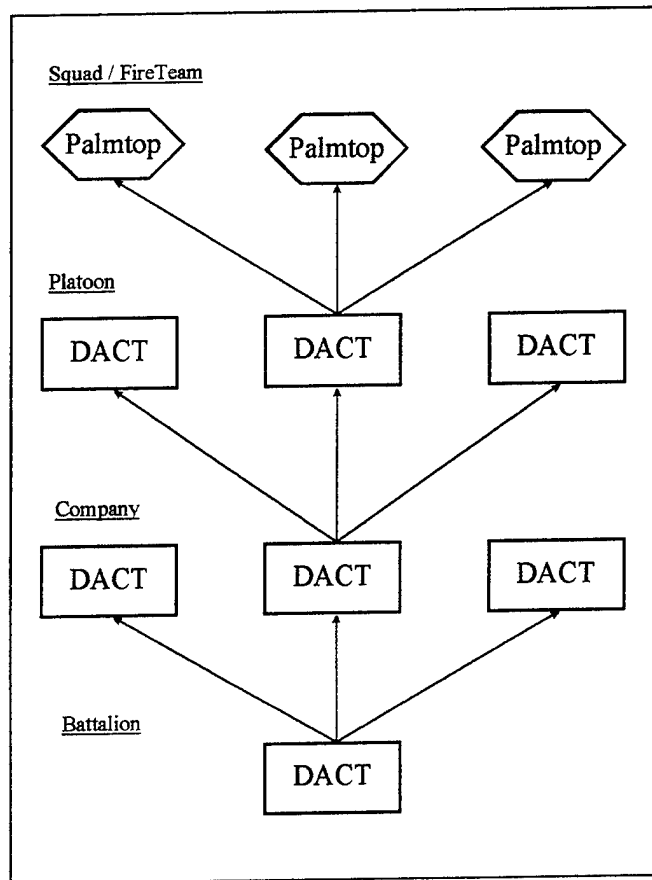


Figure 8.1. DACT/Palmtop Combat Information Flow.

REDMAN embraces the all-important KISS principle in its user interface—the software is written by grunts for grunts. REDMAN also understands that any Marine on the battlefield might become the battalion commander at any time due to combat attrition. Hence, REDMAN is malleable in the sense that any leader can pick up another Marine's DACT or palmtop; rapidly adjust the system defaults to his preferences; and issue an order or report for any arbitrary level of command.

Primarily, REDMAN allows the leader at each level of command to receive a combat order from his higher commander. The data format of each message transmitted is HTML due to its device independence and universality as discussed in Chapter IV. At those higher levels of command which possess the processing power of a DACT (or equivalent future generation palmtop such as the Pegasus), the order processing software will also let the leader rapidly select a course of action, and deliver his own order to his

subordinates using wired and wireless LANs, IR, combat net radio, 2-way pagers, cellular phones, or LEOS satellite communication. There are a myriad of "value-added" components to the REDMAN order processing software that optimize the often extreme time constraints combat leaders are required to undergo during the preparation of a subordinate order. These components are discussed in the following chapter.

REDMAN is designed to ultimately become a family of integrated products used in transmitting message packets across various networks while interoperating among widely varying hardware and operating system platforms. These integrated products currently include a combat order processing system; a combat order Web Server with front end security; and a variety of COTS message viewers optimized for various platforms. Following the stated direction of the DACT program, REDMAN exploits COTS software solutions whenever possible. If a product is available "off-the-shelf," REDMAN makes use of it whether it be in the form of complete applications, an application framework, or a rapid application development environment.

REDMAN embraces sensible commercial and industry standards whenever possible in order to keep development and hardware costs at a minimum. As stated above, combat message packets are primarily transmitted in HTML format to provide interoperability among legacy and emerging systems. Filters to translate GCCS orders in Joint VMF message format into REDMAN database format can be built to run on the TCO UNIX desktop systems found at the battalion and higher levels of command. A Windows 3.X (and follow-on Win95/NT versions) of the software runs on the DACT DMTDs discussed in Chapter III. MS-DOS versions of REDMAN HTML message viewers can run on the HP family of palmtop computers discussed in Chapter VI. Versions of HTML browsing software for GEOS-based platforms are also becoming available. Finally, Pegasus machines can run REDMAN HTML combat packets on its native HTML/Help browser. Future work will involve porting over a full REDMAN message processing system to the Pegasus machine to take advantage of its Win32 API.

REDMAN also encapsulates the interaction and connectivity between a database of operation orders and a World Wide Web server using a state-of-the-art application framework known as WebHub (HRef, 1996). The REDWEB application provides a secure front end to a particular battalion's Intranet and allows dynamic web pages to be created on the fly based upon search criteria entered by the user. Combat orders are served in easy-to-read HTML format for universal browsing on a web browser and

hardware platform of the user's choice. As the military makes the inevitable transition towards HTML as the lingua franca of military communications, the REDMAN and REDWEB systems will be positioned to rapidly evolve to meet the changing requirements of the U.S. Marine Corps.

B. WHY REDMAN?

Why do we need mobile computers running a system like REDMAN on the battlefield? What's wrong with the old manual way of communicating combat orders? Primarily, the answer involves the slow speed of transmitting combat orders in the current structure. Any infantryman can tell you that the state-of-the-art reproduction capability in the field at the battalion level is traditionally carbon paper. Often, even this carbon paper isn't available. Hence, combat orders often become a tedious exercise in note-taking—especially at the company level and below. Subordinate leaders are forced to take notes manually, often under adverse weather conditions and error-prone environments. The leader says “Phase Line Yellow” and somehow the subordinate writes down “Phase Line Green.” The end result is that the Marines at the lower levels aren't getting accurate information. By the time the critical data gets filtered down to their level, there rarely is any time to deliver a formal order as dictated by the 5-Paragraph operation order paradigm (see below for a definition of this universal USMC format). At the squad and fireteam level, the unit's “order” often becomes simply “Get up and follow me!”

As pointed out in Chapter III, the high development costs of the DACT program does not make fielding DACT machines feasible below the platoon level. The DACT is designed to be a situational awareness and communications platform. However, don't infantrymen at the squad and fireteam level need situational awareness and communication capability as well? Integrating low-cost palmtops into the fight allows everyone to share in the wealth of combat information available on a digital battlefield. This approach extends the capabilities of DACT computers as shown in Figure 8.1.

Moreover, every infantryman knows that the “2/3 Time Management Rule” doesn't work in the real world using manual means of communicating combat orders. This management rule tells us that unit commanders should never take more than 1/3 of the available time in planning and issuing their order (reserving 2/3 of the time for their subordinates' efforts). Time is something that is always in short supply on the modern battlefield. According to the Tactical Exercise and Evaluation Controller, Marine Corps

Air Ground Combat Center, Twentynine Palms, California, platoon leaders receive a formal order less than 70% of the time during a Combined Arms Exercise (CAX). Squad leaders receive a formal order less than 25% of the time, and fireteam leaders less than 5% of the time. (Keenan, 1995) The bottom line is that Marines are often going into battle without detailed knowledge of their mission and commander's intent. Of course, combat orders should always be delivered in person whenever possible—every leader wants to look into the eyes of his subordinates and ensure they understand exactly what he wants them to do. Even so, do leaders need to write down every word coming out of the commander's mouth? As Don Brutzman has said, "Stenography is not a prized battlefield skill." (Brutzman, 1996)

REDMAN removes the administrative burden of communicating combat orders manually. It will dramatically speed up the accurate flow of combat orders down to the lowest level of command, improving comprehension and enhancing unity of command at all levels throughout the infantry battalion. REDMAN and its eventual successors have the potential to be the "killer app" of the battlefield—reason alone to adopt highly mobile computers to the Marine Corps' way of warfighting.

C. REDMAN COMBAT ORDER PROCESSING SUB-SYSTEM

The REDMAN Combat Order Processing Sub-System consists of four primary modules as depicted in Figure 8.2 and the following paragraphs.

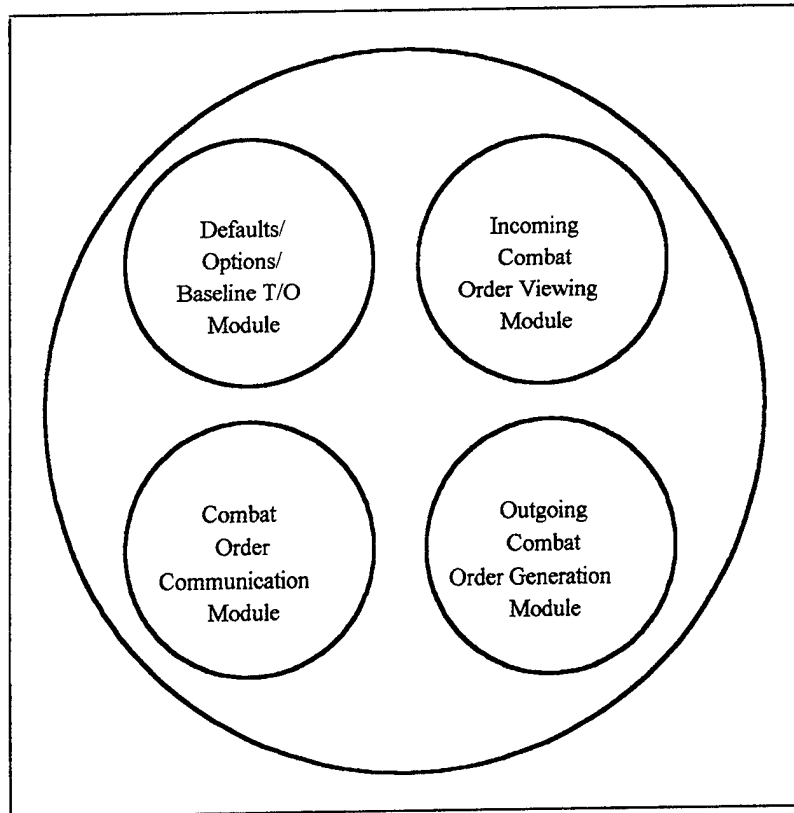


Figure 8.2. REDMAN Combat Order Processing System Modules.

1. Defaults/Options/Baseline T/O Module

The Defaults/Options/Baseline T/O module allows the user to change system options and defaults and allows the update/maintenance of unit Table of Organization (T/O) data. T/O data typically includes the name of the unit and the unit commander for each internal sub-unit within the organization.

2. Incoming Combat Order Viewing Module

The incoming combat order viewing module provides for an easy to use display of higher level combat orders. The incoming T/O is the foundation for the generation of the outgoing T/O by asking which subordinate leader you are. If this subordinate leader doesn't match your baseline, the program will allow the user to rebuild his baseline (with T/O table as input).

3. Outgoing Combat Order Generation Module

The outgoing combat order generation module allows the user to create a new order from scratch or generate one from an incoming higher level order. Previous orders can be used as templates for subsequent orders. This module automatically integrates components of the higher level order into the generated order. For example, the higher unit mission in Paragraph 2 of the incoming order is moved to the Paragraph 1 of the outgoing order (Situation/Friendly/Higher). The system provides an easy-to-use user interface to allow the selection of boilerplate text for all unit missions. The system generates each subordinate mission in Paragraph 3 based on an existing baseline T/O.

4. Combat Order Communication Module

The combat order communication module will ultimately provides transmission of combat order packets over a wide variety of communication channels to include point-to-point RS-232 wire; Ethernet LANs (both wired and wireless); Infrared (with the possible addition of transponder amplifiers); RF (P-TCIM and SINCGARS); Cellular and Paging technologies, and LEO SATCOM.

D. COMMUNICATION CHANNELS USED IN REDMAN/REDWEB

1. REDMAN File Transfer Connectivity

The intent of the REDMAN order processing system is to transfer files in compressed native Paradox database form for easy integration into subsequent levels of the order processing software. Paradox was chosen as the REDMAN database format since it is representative of a widely available and compatible class of commercial PC database formats. Clearly, any other popular database format (such as xBase) could be used with equivalent results. Uppercase field names of 10 characters or less have been maintained throughout to ease the process of migrating the data to other formats if desired (since xBase formats only support 10 character uppercase field names). In the event the receiver of a combat order message packet does not have the combat order processing software on his target machine, the user can request the order in HTML form as described below.

If the user elects to receive the order in native database format, there are numerous alternative channels and protocols that can be exploited to effect the data transfer. Figure

8.3 includes a partial list of these alternative channels as discussed in Chapter V. Users are expected to select the channel which affords the most reliable and secure transmission path given the tactical circumstances. For example, in a tactical huddle scenario requiring wireless high-bandwidth transmission, the order could be passed via diffuse infrared access points established inside a standard General Purpose (GP) tent during a battalion operations order brief. Hence, everyone in the room can receive the order concurrently at speeds up to 2-4 Mbps.

In a defensive scenario, the order can be passed via Ethernet over a previously established wired or wireless LAN. In a point-to-point scenario, an infrared SIR connection, direct RS-232 null modem connection, or even manually copying the compressed combat order to a temporary file on a PCMCIA Flashcard might be employed.

For long-haul transmission requirements, SINGCARS combat net radio can be used in conjunction with the P-TCIM PC Card digital modem discussed in Chapter III. Civilian infrastructure supporting either circuit switched cellular, CDPD, digital cellular, Narrow or Broadband PCS, GSM, or any of its variants can be used in conjunction with cellular-ready modems (if necessary) to effect connectivity to terminal servers which might conduct Z-Modem file transfers to ensure data integrity. If the civilian infrastructure was unavailable or inoperative, a network of mobile base-stations could be installed (JITC, 1995). Ultimately, LEOS communication will allow over the horizon and robust transmission of combat order packets regardless of geographic location.

Figure 8.3 depicts the data flow of Paradox-based combat orders along the myriad of communication channels available to REDMAN users.

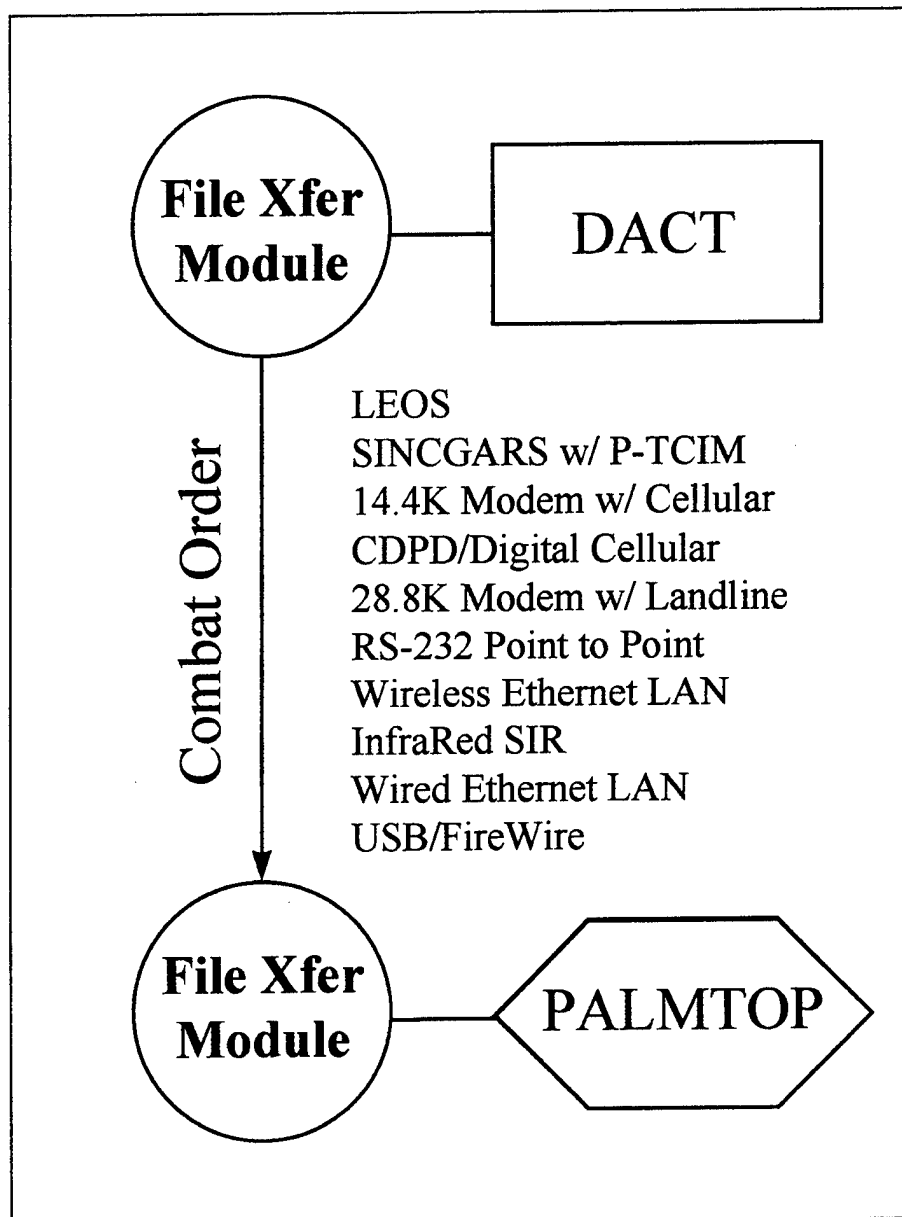


Figure 8.3. REDMAN File Transfer Connectivity.

Table 8.1 reflects comparative speeds and characteristics of the various communication channels which can be exploited by REDMAN (using the existing threshold performance objectives for DACT and the HP200LX palmtop and ignoring real-world throughput, call setup time, etc.

Communication Channel	Theoretical Maximum Throughput	Time (ms) to Xmit 7K Compressed Paradox Packet	Time (ms) to Xmit 14K HTML Packet
ORBCOMM LEOS	4.8Kbps	1493.3334	2986.6668
SINGARS w/ P-TCIM	9.6Kbps	746.6667	1493.3334
14.4 modem w/ ETC (analog cellular)	14.4Kbps	497.7779	995.5558
CDPD/Digital Cellular	19.2Kbps	373.3334	746.6668
28.8K modem (Landline)	28.8Kbps	248.8889	497.7778
RS-232 w/ Null Modem	115Kbps	60.8695	121.739
DSSS Ethernet Wireless LAN	2 Mbps	3.4179	6.8358
InfraRed SIR	4 Mbps	1.7089	3.4178
10-BaseT Ethernet Wired LAN	10Mbps	0.6835	1.367
Universal Serial Bus	12Mbps	0.5697	1.1394
IEEE 1394 FireWire	400Mbps	0.0171	0.0342

Table 8.1. REDMAN Communication Channel Theoretical Throughputs.

2. REDWEB HTML Connectivity

Users are notified (via server pull) that a new order is available for download (either via narrowband-PCS paging or traditional military messaging channels). The user can then employ a traditional web browser either native to the operating system (as in the case of Pegasus) or as a COTS application software (as in the case of DACT or the HP palmtop). Network connectivity is established with the Web server using any of the established connection-oriented technologies discussed above. Finally, after the RedWeb application's front-end security has validated the authenticity of the user, the user is allowed to download any combat order currently in the database of the server.

Real-time tracking of which subordinate units have downloaded which orders can be automatically generated in the form of an order transmission log to provide a means of allowing those units initiating order to validate the status of all subordinate units who had received the order to date. Confirmation messages can also be sent and the system can automatically update a table to acknowledge receipt of each order per unit standard operating procedures (SOP).

E. THE USMC OPERATIONS ORDER FORMAT

The U.S. Marine Corps uses a doctrinal format for their operation orders as defined in Fleet Marine Field Manual (FMFM) 3-1, Command and Staff Action (USMC, 1982). This format has been slightly adjusted owing to modifications to tactical principles embodied in more recent publications such as FMFM-1, Warfighting (USMC, 1989). Different organizations interpret these modifications in slightly different ways. For the purposes of this research, a composite of numerous formats was used to produce the following order format to design the REDMAN data model. REDMAN is a highly malleable system and RAD adjustments can easily be made to the following structure to reflect inevitably changing requirements.

A USMC operations order is formatted into five different paragraphs per the naming convention depicted in Table 8.2. The major paragraph names are "Situation," "Mission," "Execution," "Administration and Logistics" and "Command and Signal." The standard sub-paragraph "numbering" has been retained throughout to include the field names for the data model discussed below. For an example of an actual operations order, see Appendix C.

Paragraph Number	Paragraph Name
1.A	General Situation
1.B	Enemy Forces
1.C.1.A	Friendly Higher Mission
1.C.1.B	Friendly Higher Intent
1.C.1C	Friendly Higher Scheme of Maneuver
1.C.2	Friendly Adjacent
1.C.3	Friendly Supporting
1.D.1	Attachments
1.D.2	Detachments
1.E	Assumptions
2	Mission
3.A.1	Commander's Intent
3.A.2.A	Scheme of Maneuver
3.A.2.B	Fire Support Plan
3.B.1 through 3.B.n	Tasks (dependent on Task Organization)
3.C	Reserve Mission
3.D	Coordinating Instructions
4	Administration and Logistics
5A	Command Relationships
5B	Command
5C	Signal

Table 8.2. USMC 5-Paragraph Order Format (USMC, 1982)

F. REDMAN DATA MODEL

1. Introduction

The data model in support of REDMAN was divided into seven primary database tables and two support database tables as depicted in Table 8.3. The filename of each table, a definition of the contents of the table, and a plain language description of the records contained within each table are included. Detailed explanations of each table follow.

Table File Name	Table Definition	Table Description
ORDERS.DB	Orders Table	A table of all orders in the REDMAN system
TO-<n>.DB	Task Organization Table	A related table of all the units who have a mission statement for a particular order (for example, the first order's T/O would be named "TO-1.DB"; the second "TO-2.DB"; etc.
TO-BASE.DB	Baseline Task Organization Table	A table consisting of all the units INTERNAL to a particular unit—used for auto populating the task organization for orders generation
MISSIONS.DB	Missions Table	A table of all missions that can be assigned an infantry battalion
MSN-AREA.DB	Mission Areas Table	A table of all the mission areas that can be assigned an infantry battalion. Each Mission in the MISSIONS.DB table has a mission area. This is used for filtering the Checklst and Elements tables during program execution.
CHECKLST.DB	Checklist Items Table	A table of checklist items so that users writing orders are reminded of key elements to consider when assigning tasks.

Table 8.3. REDMAN Data Model and Table Composition (Part 1/2).

Table File Name	Table Definition	Table Description
ELEMENTS.DB	Element Items Table	A table of boilerplate text used for storing standard mission statements, coordinating instructions, and other oft-repeated verbiage for use in creating orders.
OUT-ORD.DB	Outgoing Order Table	Support table used for the temporary storage of an outgoing order prior to compression and transmission.
TO-EMPTY.DB	Empty Task Organization Table	Support table used for initializing new task organization tables when creating new orders (either manually or by autogenerating an order).

Table 8.3. REDMAN Data Model and Table Composition (Part 2/2).

2. ORDERS.DB Table

This table contains all the orders received, created, or generated for a particular unit. It is intended that users retain all received and generated orders to use as templates for subsequent order generation, as well as serve as the foundation for the future development of intelligent agents in recommending courses of action (see Chapter X). During the data model design and subsequent refinement, numerous fields were added to the fields above in order to accommodate all the data elements included in the sample order of Appendix C. Many of these fields are included in the FMFM 3-1 format guidance, although they are not strictly part of the “5-Paragraph Order” format established as doctrine in numerous USMC publications. These fields are depicted in Table 8.4. The memo field type is equivalent to an alphanumeric field of arbitrary length.

Field Name	Field Type	Field Description/Purpose
ORDER_ID	Short Integer	Primary Key for Internal Use
MISSION_ID	Short Integer	Foreign Key into Mission Table
INIT_UNIT	Alpha (50)	Unit who initiated Order
LOCATION	Alpha (50)	Geographic Location of Order Issuance
DTG	Alpha (13)	Date Time Group of Order
SERIAL_NO	Alpha (5)	Order Serial Number
ORDER_NAME	Alpha (80)	Name of the Operation
REFERENCE	Memo	List of all references for use with the order
TIMEZONE	Alpha (1)	Time Zone (usually Zulu)
ORIENT	Memo	Orientation

Table 8.4. REDMAN Fields in Data Model.

As shown in Table 8.5, the “paragraph” fields in the standard 5-Paragraph order structure discussed above are then translated into compatible field types with appropriate field names to produce the following fields which make up the final ORDERS.DB schema.

Field Name	Field Type	Field Description/Purpose
PARA1A	Memo	General Situation
PARA1B	Memo	Enemy Forces
PARA1C1A	Memo	Friendly Higher Mission
PARA1C1B	Memo	Friendly Higher Intent
PARA1C1C	Memo	Friendly Higher Scheme of Maneuver
PARA1C2	Memo	Friendly Adjacent
PARA1C3	Memo	Friendly Supporting
PARA1D1	Memo	Attachments
PARA1D2	Memo	Detachments
PARA1E	Memo	Assumptions
PARA3A1	Memo	Commander's Intent
PARA3A2A	Memo	Scheme of Maneuver
PARA3A2B	Memo	Fire Support Plan
PARA3D	Memo	Coordinating Instructions
PARA4	Memo	Administration and Logistics
PARA5A	Memo	Command Relationships
PARA5B	Memo	Command
PARA5C	Memo	Signal

Table 8.5. REDMAN USMC 5 Paragraph Order Fields.

3. TO-<n>.DB Table

This table contains all the units who are assigned subordinate missions (to include the unit which initiated the order) for a particular order. The reason the initiating unit is also contained in this table is a peculiarity of the USMC order system. Each unit which initiates an order also assigns itself (representing the unit as a whole) a specific mission as directed by Paragraph 2. Hence, it was possible to abstract "mission" related fields in the USMC 5-Paragraph order format out of the ORDERS.DB and simply reflect them as additional records in the TO-<n>.DB table. The sub-paragraphs which fall in this category include Paragraph 2 (Mission); Paragraphs 3.B1..3Bn (Subordinate Unit Tasks); and Paragraph 3.C (Reserve Mission). This provides a one-to-many relationship between the order record and the multiple units which are assigned tasks by that order.

Furthermore, the initiating unit is also listed in the T/O at the top level of the task organization structure. Encapsulating the initiating unit information in the Task Organization streamlines the process of auto-generating HTML for a specific order, printing of orders, etc.

The "<n>" in the table name corresponds to the ORDER_ID field of the table ORDERS.DB. Hence, the fourth order in the ORDERS.DB table would have a corresponding TO-4.DB table associated with it. All fields are shown in Table 8.6.

Field Name	Field Type	Field Description/Purpose
UNIT_CODE	Short Integer	Primary Key for Internal Use
UNIT_TEXT	Alpha (50)	Name of the Unit assigned a subordinate mission (unless the unit who initiated the order)
UNIT_CMDR	Alpha (40)	Name of the commander of the Unit
REPORTS_TO	Short Integer	Which unit this unit works for (is 0 if the unit initiated the order)
RESERVE	Logical	Whether or not the unit's mission is part of the reserve mission
TASK	Memo	A description of all the tasks associated with this unit's mission
PASSWORD	Alpha (8)	RedWeb Password to allow access via the World Wide Web

Table 8.6. T/O Table Data Schema.

4. TO-BASE.DB Table

This table is identical in structure to the TO-<n>.DB table shown in the preceding Table 8.6. However, it serves its purpose only during auto-generation of orders. For each unit in the TO-BASE table, REDMAN will generate an entry in the outgoing order TO-<n>.DB table. This allows a user to preload his database with his internal subordinate units in order for the program to autogenerate the outgoing T/O.

5. MISSIONS.DB Table

This table contains a listing of all missions that can be assigned an infantry battalion, currently 57 separate missions as specified by USMC Operational Handbook (OH) 6-1 (USMC, 1986). This will be used during REDMAN execution to allow the user to select the exact mission type of the order he is generating. Also, this table will allow subsequent refinement for checklist and element table filtering once these particular database tables are sufficiently large. It is intended that the user will eventually be able to add/edit/delete the records in this Missions table to customize it for his specific unit's purposes. The table format is shown in Table 8.7.

Field Name	Field Type	Field Description/Purpose
MISSION_ID	Short Integer	Primary Key for Internal Use
AREA	Short Integer	Foreign Key into Mission Area Table
TYPE	Alpha (50)	Name of the Mission

Table 8.7. Missions Table Data Schema.

6. MSN-AREA.DB Table

A table of all the mission areas that can be assigned an infantry battalion (Table 8.8). Each Mission in the MISSIONS.DB table has a mission area. This is used for filtering the Checklist and Elements tables during program execution to allow a user to rapidly drill down into these tables to only show those checklist and order element items pertinent to the mission in question (see below for a definition of Checklist and Order Elements).

Field Name	Field Type	Field Description/Purpose
AREA_ID	Short Integer	Primary Key for Internal Use
AREA_NAME	Alpha (30)	Name of the Mission Area

Table 8.8. Mission Area Table Data Schema.

7. CHECKLST.DB Table

A table of checklist items so that users writing orders are reminded of key elements to consider when assigning tasks (Table 8.9). A maximum length string variable (255 characters) was used to hold the Checklist item text.

Field Name	Field Type	Field Description/Purpose
CHECK_ID	Short Integer	Primary Key for Internal Use
MISSION_AREA	Short Integer	Foreign Key into Mission Area Table
CHECK_TEXT	Alpha (255)	The text of the Checklist Item

Table 8.9. Checklist Item Table Data Schema.

8. ELEMENTS.DB Table

A table of boilerplate text used for storing standard mission statements, coordinating instructions, and other oft-repeated verbiage for use in creating orders (Table 8.10). A maximum length string variable (255 characters) was used to hold the Order Element text, and to facilitate the copying of the boilerplate text to the unit's mission statement being worked on.

Field Name	Field Type	Field Description/Purpose
ELEMENT_ID	Short Integer	Primary Key for Internal Use
MISSION_AREA	Short Integer	Foreign Key into Mission Area Table
ELEM_TEXT	Alpha (255)	The text of the Element Item
AUTO_INCL	Logical	Whether or not to auto-include this text in everyone's mission for a particular mission area

Table 8.10. Order Elements Table Data Schema.

9. OUT-ORD.DB Table

This table is identical in structure to the ORDERS.DB table shown in Table 8.4. However, it serves its purpose only during the transmission and compression of orders. Hence, this table is a support table used for the temporary storage of an outgoing order prior to compression and transmission.

10. TO-EMPTY.DB Table

This table is identical in structure to the TO-<n>.DB table shown in Table 8.6. However, it serves its purpose only during the creation of new T/Os in conjunction with new orders. Support table used for initializing new task organization tables when creating new orders (either manually or by auto-generating an order).

G. CRITICAL RAD DESIGN DECISIONS OF THE REDMAN APPLICATION

The decision was made early on in the design process to make the REDMAN application a Multiple Document Interface (MDI) program. MDI programs, although much more difficult to write, encapsulate all child windows into a single framed document so that windows can be cascaded, tiled, etc. Since part of the appeal of REDMAN is the ability to instantly access all previous orders generated by a particular unit commander, the ability to have multiple orders on the screen simultaneously, allowing the user to cut and paste at will between windows, is a powerful component of the system.

Furthermore, the decision was made to limit the screen resolution to a standard VGA 640 * 480 model since that is the threshold for DACT hardware at the time of this writing. Furthermore, since the eventual goal of REDMAN is to port the order processing software to a more compact and powerful mobile computer (such as the Pegasus), users will often have to live with a relatively low screen resolution in their user interface.

“Ease of use” has been incorporated into all facets of REDMAN, considering that the intended user audience for this product may have limited computer experience. The familiar 5-Paragraph paradigm has been retained with a tabbed “notebook” which allows the user to simply select the area of the order he wishes to consider. A hierarchical database enabled outline paradigm was used to represent the Task Organization to great effect. Drag and drop functionality is enabled with this component for ease of manipulating this most convoluted aspects of user orders.

Furthermore, the ability to “autogenerate” orders was a critical part of the user functionality design. Since almost 40% of any unit commander’s order is simply the restructuring of information received at higher levels of command, REDMAN avails itself of the power of the computer to automate the reformatting of higher unit data from the incoming order to produce a schematic of the outgoing order. This process will save many tactically-critical minutes, and aid in comprehension of the order by lower unit

commanders by automatically extracting the pertinent information from an incoming order to the outgoing order generation screen.

The REDMAN implementation discussed in Chapter IX avails itself of numerous commercial components whenever possible. Some of these components have been extended in order to achieve the degree of programmatic control desired in the REDMAN system. Code reuse has been emphasized whenever possible throughout the system. Clear comments and publication of source code in Appendix E make this proof-of-concept software ready for operational testing and long-term support.

H. SUMMARY

The REDMAN discussion contained in this chapter focuses on the design and implementation of software to move platform independent combat orders over varied transmission channels. REDMAN also demonstrates the viability of integrating low-cost palmtop computers with the emerging USMC DACT program. Integrating low-cost palmtops into the fight allows everyone to share in the wealth of combat information available on a digital battlefield.

The REDMAN Combat Order Processing Sub-System consists of four primary modules: the Defaults/Options/Baseline T/O Module; the Incoming Combat Order Viewing Module; the Outgoing Combat Order Generation Module; and the Combat Order Communication Module.

Message packets can be transmitted with REDMAN using a rich array of communication technologies including: LEOS; SINCGARS w/ P-TCIM; 14.4K Modem w/ Cellular; CDPD/Digital Cellular; 28.8K Modem w/ Landline; RS-232 Point to Point; Wireless Ethernet LAN; InfraRed SIR; Wired Ethernet LAN; and USB/FireWire.

REDMAN bases its data model on the U.S. Marine Corps' doctrinal format for their operation orders as defined in Fleet Marine Field Manual (FMFM) 3-1, Command and Staff Action. This data model was divided into seven primary database tables and two support database tables in native Paradox format.

REDMAN is a standard VGA resolution, Multiple Document Interface (MDI) program allowing the user to easily cut and paste between various orders. Autogeneration of orders saves users significant time in issuing orders in tactical scenarios.

IX. REDMAN IMPLEMENTATION USING RAPID APPLICATION DEVELOPMENT (RAD) TOOLS

A. INTRODUCTION

This chapter discusses the details of the REDMAN order processing system and the RedWeb HTML processing module working in conjunction with DACT based and palmtop based web browsers. All annotated source code for these programs is provided in the appendices. Extracts and snippets of code will be used throughout this chapter to illustrate programming techniques and discussion points.

Sample operations order data will also be used to illustrate the functionality of the REDMAN suite of application software. This sample order data is contained in Appendix C and is based upon a realistic USMC infantry battalion operations order written by the author while serving as the S-3A of 2nd Battalion, 7th Marines during a live-fire exercise at the Marine Corps Air Ground Combat Center (MCAGCC) in 29 Palms, California in 1991. This order remains typical of combat orders used in coordinating infantry operations by Marine Corps battalions.

The prototype applications depicted in this chapter will focus on the dissemination of a combat order from the battalion to the fireteam level. It is recommended that this prototype serves as a model for the development of other critical mobile applications and inspire additional mobile computing research initiatives. A major feature of this approach is that all operations order functionality can reside identically (and even simultaneously) on operation order servers and mobile computer clients.

1. REDMAN, REDWEB and Web Server Interactivity

As discussed in Chapter VIII, the REDMAN system primarily receives information as HTML from a World Wide Web HTTP server. Figure 9.1 demonstrates the interactivity between the primary modules in the REDMAN system. These modules will be subsequently decomposed into more granular modules in later parts of this chapter.

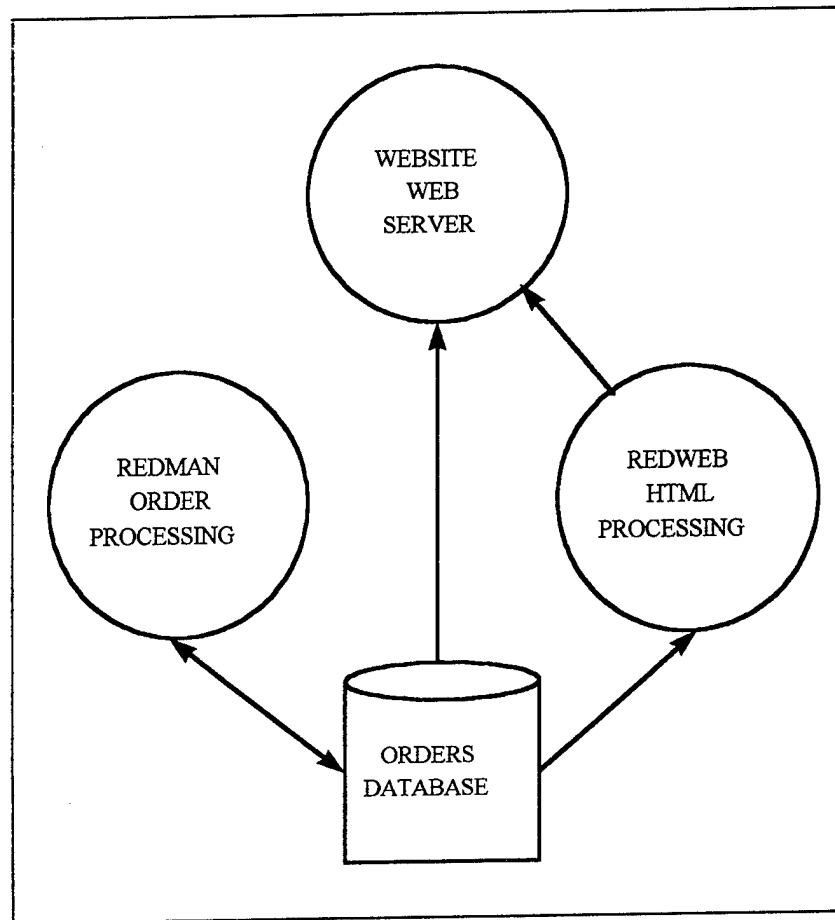


Figure 9.1. REDMAN, REDWEB, Web Server Interaction.

B. DELPHI - THE 16 AND 32-BIT RAD DEVELOPMENT ENVIRONMENT

The DACT version of the REDMAN system was primarily coded using Borland's International Delphi programming environment.

1. What is Delphi?

Delphi is an object-oriented, visual programming environment for Rapid Application Development (RAD) of general purpose and client/server applications for the Microsoft Windows operating system. The core of the environment is based on the Object Pascal language. Delphi provides a comprehensive library of reusable components and a suite of RAD design tools (Borland, 1996). Delphi 1.0 was released in 1995 and creates optimized code for execution on any Windows 3.x platform. Delphi 2.0 was released in 1996 and creates code for the Win32 API operating systems only (Windows 95 and NT).

Delphi embodies the emerging RAD philosophy in designing software. In an industry which reinvents itself so quickly, RAD is critical to compressing the development time of any new system. The Delphi environment shines when used to quickly field prototypes and then eventually turn them into robust applications based upon detailed user feedback. Hence, if things are working as they should, the REDMAN system of 1996 might only superficially resemble the REDMAN applications of the year 2000. Technology and military tactics are changing too rapidly to struggle through the traditional ten-year development cycles of automated information systems. In these modern times, good software essentially means throwing it away every two weeks and fielding a more refined product based upon feedback from the user base. Delphi provides a flexible environment to support this iterative development requisite in any successful software development project. Integrating the customer (in this instance, the USMC Fleet Marine Force) into the process of developing applications is critical to the success of the DACT and REDMAN projects.

Since the REDMAN implementation discussed in this thesis was designed for the DACT, the Delphi 1.0 16-bit environment was selected to create the REDMAN prototype which executes on any WinTel platform PC with at least 4MB of memory. Although the Delphi 1.0 environment is missing several features in contrast to Delphi 2.0, the code created with it can be easily ported to Delphi 2.0 when the DACT operating system changes to a Win32 API based platform.

2. The Borland Database Engine (BDE)

Both the REDMAN order processing system and the REDWEB HTML processing system require the services of the BDE. The BDE integrates with data-aware components found within the Delphi environment to allow the rapid development of database enabled applications. The BDE sits between a database server and a database client (all database enabled Delphi applications are BDE clients). In order to reduce the footprint of the REDMAN and REDWEB applications, these programs access local tables in Paradox form when modifying either orders or related task organizations for that specific order. One of the nice features about Delphi is that the application is completely scalable to a multi-tiered architecture by changing just a few lines of code regarding the data source for the data-aware components. Hence, in a production environment where multiple clients might be accessing the database at one time, it is relatively trivial to convert the REDMAN system to a true client/server application. The BDE also supports

a Open Database Connectivity (ODBC) socket working in conjunction with various ODBC drivers for those database products not natively supported by Delphi. Figure 9.2 represents the BDE data flows and interaction between the Delphi architecture components.

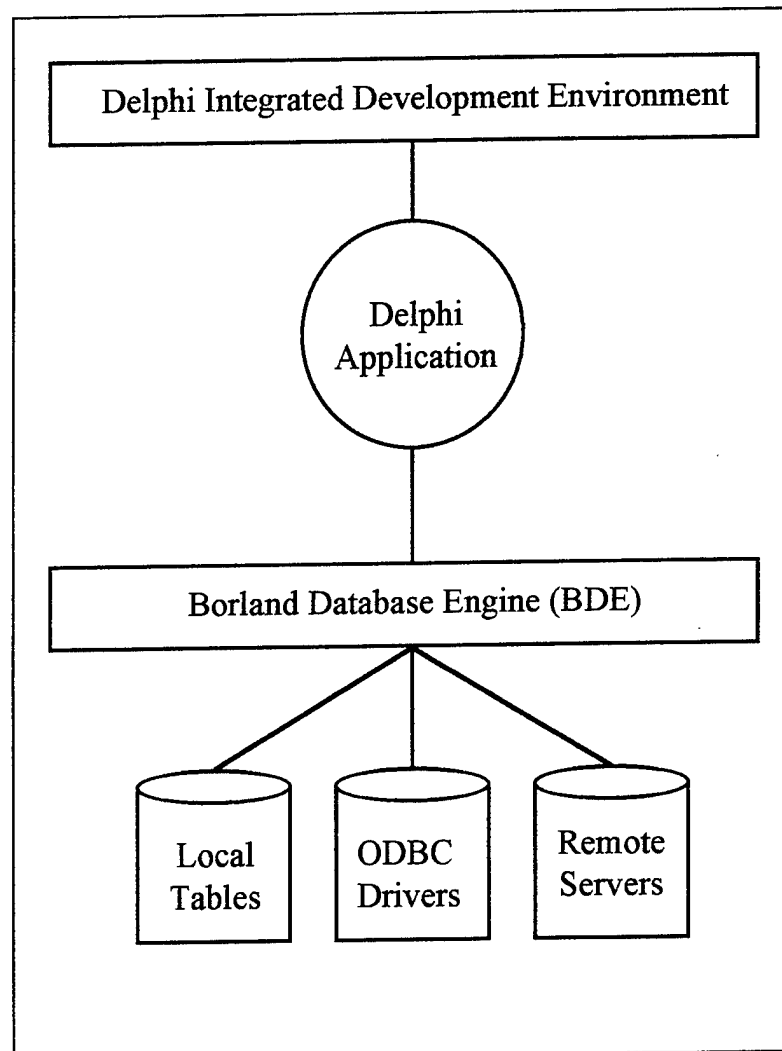


Figure 9.2. Borland Database Engine Connectivity.

The database structure for the tables used in REDMAN are documented in Chapter VIII along with the design decisions made in creating the data model.

C. WEBHUB HTML APPLICATION FRAMEWORK

1. Introduction

WebHub is a Common Gateway Interface (CGI) framework for building dynamic, database-driven web sites with Delphi for deployment on Windows-based web servers. CGI is the mechanism by which Web browsers interact with programs controlled by a Web server (Gundavaram, 1996). Although WebHub is designed to work in conjunction with the Win95 or NT operating systems, an earlier version supports 16-bit Win 3.11 with the Win-HTTPD web-server. Hence, the decision was made to use the 16-bit version of WebHub working in conjunction with the Website Web Server as discussed above. WebHub works equally well with Spry and Netscape Commerce Web servers and more recently the Microsoft Internet Information Server with ISAPI support. The ability to develop applications on fast machines that are directly portable to slower machines is a major benefit needed for effective RAD. (HRef, 1996)

WebHub is a immature product—in fact, it has still not been produced in “release form” and is only available in an “Early Experience Package (EEP).” The documentation is weak. The demonstration programs in particular are much too convoluted to be easily followed by programmers learning WebHub for the first time. Nevertheless these demonstration programs can be very useful later. Furthermore, WebHub’s reliance on INI files for configuration information is crude and the programs which manipulate these initialization files are similarly immature. Hence, the learning curve for WebHub can be daunting, but once the mysteries of the framework interface have been penetrated, WebHub provides a powerful tool available for developing database enabled web applications in a RAD environment. (HRef, 1996)

WebHub constrains its behavior to the rules of CGI and lets the application developer focus on writing the application rather than on the interface to the web server. The framework includes the WebHub System (HUB.EXE) and over 30 web-specific Delphi components that make it easy to build CGI custom applications which serve web pages dynamically. All CGI programs create web pages on the fly. The real issue is how responsive those pages are to the client. Any WebHub application can easily be made aware of client preferences and take them into consideration when assembling a page. This customization can range from the cosmetic level of including/excluding the graphics

on a page to features like specifying the way a search is done and how the resulting answer set is presented. (HRef, 1996)

WebHub provides an easy mechanism to allow the core content of the web pages to come from a database. The web application has a live connection to whatever desktop or SQL database it wants, and it can use that database in its processing at any time. This includes publishing information from the database as well as posting information to it. (HRef, 1996)

2. Benefits of the WebHub Framework

WebHub provides numerous advantages over traditional methodologies of serving database enabled web pages. Specifically, WebHub has the following features built into the framework:

a. Saving State

If your dynamic web site has more than a couple pages, applications must handle saving the surfer's data from one page request to the next. WebHub handles that automatically, using anonymous Session IDs and keeping the data privately on the web server machine. Each surfer is assigned a unique session ID which is accessible programmatically via the `%=session=%` macro. All form data entered by a surfer is preserved automatically, as well as the current record in a database or level of expansion of a particular outline. (HRef, 1996)

b. Process Control

The WebHub system queues page requests instead of repeatedly invoking time-consuming executions of the .EXE or .DLL file. If adequate CPU power and memory is available, multiple instances of a WebApp can be running on the server for simultaneous database queries and maximum use of SQL licenses. (HRef, 1996)

c. Integration with Delphi Debugging Environment

The Delphi debugger can be used on projects built with WebHub components. The runner application (see Figure 9.3) takes care of the interface to the web server. Hence project can be tested from inside Delphi using breakpoints, etc. to determine the source of program flaws. By contrast, easy debugging is not possible if you

build your own ISAPI DLLs, or if you write CGI EXEs using the traditional model. (HRef, 1996)

d. Easy Site Maintenance and Flexibility

WebHub is based upon a foundation of reusable chunks of HTML and TWebAction components. Macros and the separation of the code from the HTML make WebHub an ideal environment for larger web sites as they encourage code reuse and streamlined maintenance. Conditional expansion can be used within the WebHub web pages to optionally display HTML chunks based upon programmatic control. (HRef, 1996)

e. Scalability and High Performance

WebHub is designed to support simultaneous surfers against full-featured, high-traffic web sites served by a single machine, or a cluster of state-of-the-art multi-processor boxes. By pre-loading the hub and WebHub enabled applications in memory, WebHub offers a performance level not found in traditional CGI programs (such as the *PERL* interpreter). These traditional programs must be loaded from disk whenever the surfer invokes the call to the CGI program. (HRef, 1996)

3. WebHub Connectivity and Data Flows

Figure 9.3 represents the flow of control and data within a WebHub-based application. Note that the Web Server-specific “runner” program is the only executable which must be loaded dynamically.

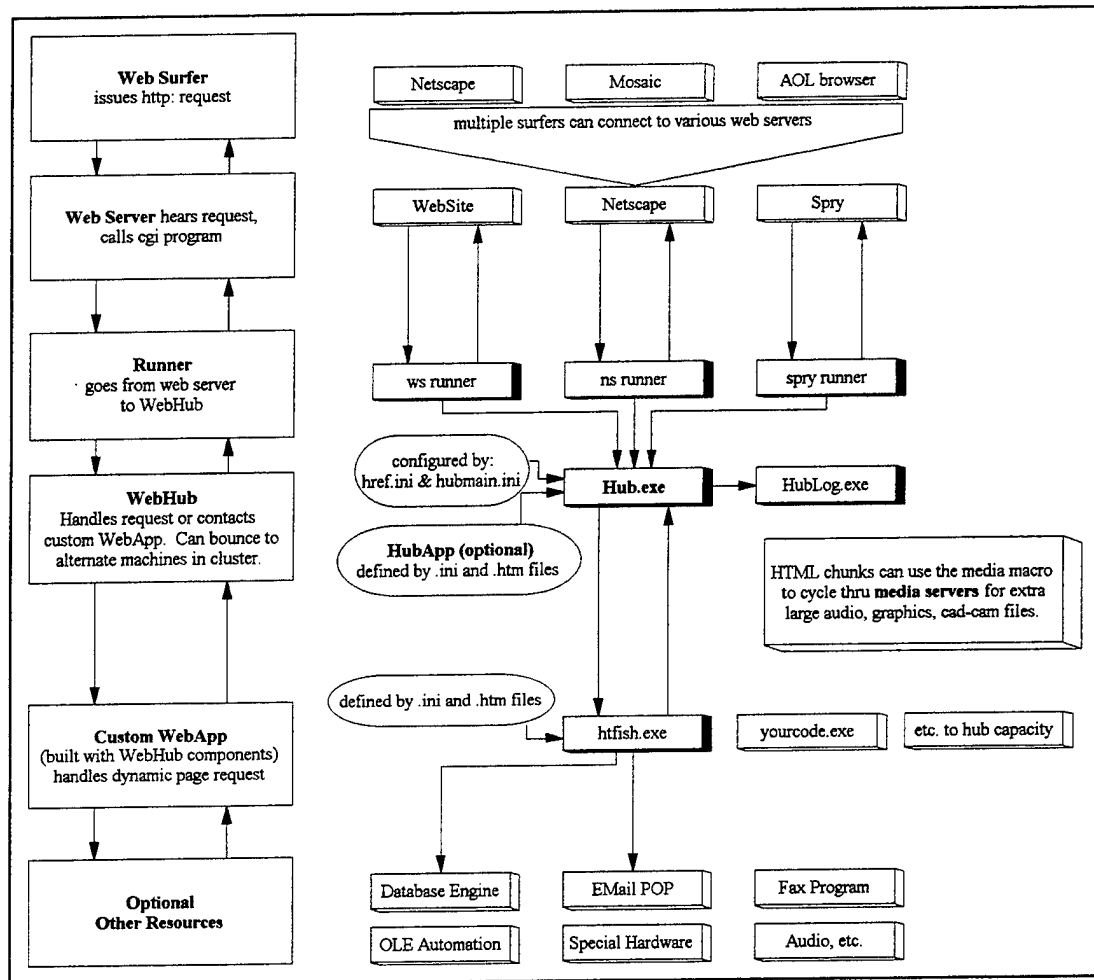


Figure 9.3. WebHub Connectivity and Data Flows. (HRef, 1996)

4. WebHub Uniform Resource Locator (URL) Syntax

The syntax for calling a WebHub application (or “WebApp”) is straightforward. The path to the server name is normally used in conjunction with the directory of the WebHub “runner” program. Then the WebHub application identification number is suffixed with numerous qualifiers to indicate to the Hub system which specific Web page, session number, and specific sub-command the surfer is interested in retrieving. Normally, the session number and sub-commands are provided dynamically by the Hub engine itself during programmatic control inside the WebApp. Hence, a fully qualified URL for the “homepage” of the “REDMAN” WebApp on the Website server located at IP address “1.2.3.4” might be:

http://1.2.3.4/cgi-win/hubws.exe?REDMAN:homepage

D. COMMERCIAL DELPHI COMPONENTS USED WITHIN REDMAN

Numerous commercial components were using in the creation of REDMAN. The following discussion highlights some of the more significant ones, and discusses extensions or tweaks that were made to the component to achieve a certain degree of functionality:

1. Maelstrom TDBOutline Component

TDBOutline is a native Visual Component Library (VCL) Delphi component designed to allow the loading, navigation, and manipulation of hierarchically organized data contained in a database. TDBOutline can be used to display, manipulate and save the hierarchic structure of recursively related (self-referencing) data. Per the TDBOutline User's Manual, TDBOutline supports automatic loading of recursively related data into the outline, preserving the hierarchic structure; automatic drag & drop facility that moves a node and all its children to be children of the dropped-on node; and automatic record-pointer synchronization and outline synchronization. (Maelstrom, 1996)

TDBOutline is not perfect, however. There is no way to easily resequence the physical order of nodes at the same level in the hierarchy (although there is a work around by simply moving the nodes to a temporary location, and then adding them back in the order you wish them to be listed in the hierarchy).

A much more serious problem involves the deletion of nodes. If the user deletes a node with child nodes, these child nodes become isolated and are lost in the database table. TDBOutline does not support recursive delete of all the nodes in a subtree. Hence, it was necessary to implement a customized recursive delete procedure based on a sequential search algorithm with the following two functions:

The function `AnyNodesThatReportToThisNode()` (shown in Figure 9.4) determines if the node pointed at is a leaf node, or whether it has any children. The procedure `DeleteTOTree` calls `AnyNodesThatReportToThisNode` and if the function is true, recursively calls itself for all the nodes that report to the original node. This code will function for any arbitrary nesting of subtrees.

```

function TfrmOrderView.AnyNodesThatReportToThisNode(Sender: TObject; NodeNum : Integer): Boolean;
var
  Found : Boolean;
Begin
  {This can be replaced by a FindKey if we can dynamically create secondary indices}
  With tblTO do
  Begin
    Found := False;
    First;
    While (Not EOF AND Not Found) Do
    Begin
      If (FieldByName('REPORTS_TO').AsInteger = NodeNum) Then
        Found := True
      Else
        Next;
    End;
    FindKey([NodeNum]); {Go back to Node we started at}
  End;
  AnyNodesThatReportToThisNode:= Found;
End;

procedure TfrmOrderView.DeleteTOTree(Sender: TObject; NodeNum : Integer);
Begin
  With tblTO Do
  Begin
    DisableControls;
    If (NOT AnyNodesThatReportToThisNode(Self, NodeNum))Then
      Delete
    Else
    Begin
      First;
      While NOT EOF DO
      Begin
        If (FieldByName('REPORTS_TO').AsInteger = NodeNum) Then
          DeleteTOTree(Self, FieldByName('UNIT_CODE').AsInteger)
        Else
          Next;
      End;
      FindKey([NodeNum]);
      Delete;
    End;
    EnableControls;
  End;
End;

```

Figure 9.4. TDBOutline Recursive Delete Code Modules.

2. Xceed Zip Compression Library

The Xceed Zip Compression Library is a data compression library allowing the manipulation of the latest industry standard Zip files, including those created by PKZIP Version 2.04g. The library comes with support for 16-bit Visual Basic, Borland Delphi 1.0. and Delphi 2.0. For Delphi, you get 100% native 16-bit and 32-bit VCL (Visual Component Library) controls. These VBX and VCL controls provide the user with all the properties, methods and events necessary in order to accomplish various data-compression tasks. The library makes use of fast and compact external Dynamic Link Libraries (DLLs) that perform the compression and the decompression. (Xceed, 1996)

In order to minimize packet size and data transmission times, REDMAN compresses a total of six Paradox tables (consisting of OUT_ORD.DB with a single record in it, the corresponding memo file [OUT_ORD.MB], and the index file [OUT_ORD.PX]; and the TO-<n>.DB, the TO-<n>.MB, and the TO-<n>.PX for the corresponding task organization of the subordinate units assigned tasks by that order).

Before compression, the average size of the six tables with associated memo files and indices is 43,008 bytes. After compression, the single file compressed by the Xceed Zip library is only 6,840 bytes providing a compression ratio of approximately 6.25:1. The procedure ZiptheOrder (shown in Figure 9.5) copies and compresses the appropriate files into a single zipped file for subsequent transmission.

```

procedure TfrmOrderView.ZiptheOrder(Sender: TObject);
var
  strOutgoingOrderPathName : String;
  strTOFileNumber : String;
begin
  qryOutgoingOrder.ParamByName('Current Order').AsInteger :=
    tblOrders.FieldByName('ORDER_ID').AsInteger;
  qryOutgoingOrder.Open;
  bmvOutgoingOrder.Source := qryOutgoingOrder;
  bmvOutgoingOrder.Destination := tblOutgoingOrder;
  bmvOutgoingOrder.Mode := batCopy;
  bmvOutgoingOrder.Execute;
  tblOutgoingOrder.AddIndex('ORDER_ID', 'ORDER_ID', [ixPrimary]);

  { add outgoing orders pathname from defaults here }
  strOutgoingOrderPathName := 'f:\delphi\redman3\';

  {Set up zip component and zip outgoing order and matching TO}
  strTOFileNumber := tblOrders.FieldByName('ORDER_ID').AsString;
  zipOutgoingOrder.ZipFilename := strOutgoingOrderPathName + strTOFileNumber + '.ZIP';
  zipOutgoingOrder.FilesToProcess.Add(strOutgoingOrderPathName + 'OUT_ORD.*');
  zipOutgoingOrder.FilesToProcess.Add(strOutgoingOrderPathName + 'TO' + strTOFileNumber + '.*');
  tblTO.Close;
  zipOutgoingOrder.Add(xecAll);
  tblTO.Open;
end;

```

Figure 9.5. Xceed Outgoing Order Compression Code Module.

3. TcsNotebook Tabbed Notebook Component

As discussed in Chapter VIII, one of the primary goals of REDMAN is ease of use while offering a consistent 5-Paragraph order paradigm to the user. The commercial TcsNotebook native VCL component is an enhanced tabbed notebook which does precisely that. The TcsNotebook component integrates seamlessly into the Delphi development environment just like the standard components supplied with Delphi. The component is similar to the TTabbedNotebook component but differs in that you can attach the tabs to the top, bottom, left or right of the notebook using the TabOrientation property. You can also specify a different bitmap to be displayed on each tab and specify the color and alignment of various properties. Figure 9.6 depicts the component as modified for use in the REDMAN application. (Classic, 1996)

The screenshot shows the 'RedMan Alpha 1.0' application window. The title bar reads 'RedMan Alpha 1.0 - [Frag Order 5-92 (OPERATION DEEP BANDINI)]'. The menu bar includes 'File', 'Edit', 'OrderAssistant', 'Xmit/Rcv', 'Window', and 'Help'. A toolbar with various icons is located below the menu bar. The main area is a tabbed notebook with the following tabs: 'Admin/Logistics', 'Command/Signal', 'Situation', 'Mission', 'Execution', 'General Info', 'Task Organization', and 'Orientation'. The 'Mission' tab is currently selected. It contains several data entry fields: 'INITIATING UNIT' (2d Bn (-) (Rein), 7th Mar), 'LOCATION' (MCAGCC, 29 PALMS), 'MISSION TYPE' (Movement to Contact), 'DTG' (141800Z OCT 91), 'SERIAL NO' (MTR-5), 'TIME ZONE' (T), 'ORDER NAME' (Frag Order 5-92 (OPERATION DEEP BANDINI)), and 'REFERENCE' (a. Twentynine Palms EastWest, Edition 3-DMA, Series V7953 1:50,000). The 'ORDER #1 of 3' indicator is visible in the top right corner.

Figure 9.6. TDBOutline Tabbed Notebook Component Screen.

4. Infopower

InfoPower is a library of several native Delphi, data-aware components that are automatically installed into Delphi's component palette in the integrated development environment (IDE). InfoPower's fifteen components includes numerous extensions to the standard data-aware components—many of which are incorporated within REDMAN. In fact, REDMAN exclusively uses Infopower components for all data sources, tables and queries used throughout the application. (Woll2Woll, 1995)

For example, the TwwTable component contains a new property named Filter, which allows you to specify one or more record selection criteria to be used when displaying data from a table. Filter criteria can be specified for any number of fields in a table, giving you more flexibility than Delphi's built-in Range operators and some advantages over performing a single-table query (Woll2Woll, 1995). This component's filter property was exploited in the checklist and order elements functionality with the following code snippets. This functionality also illustrates the pointer logic used in accessing multiple child windows in an MDI application. The TfrmElement object must know which TfrmOrderView object contains the data value which it must filter on. Hence, since only the framing form (the TfrmMain object) knows which order form has the focus, the Element form calls the GetCurrentChildPointer to return a pointer to the active MDI child. Failure to use this pointer will result in a dreaded run-time general protection fault (GPF) which can be difficult to trace during program debugging. Finally, the FormShow event code of the TfrmElement object is called to set the TwwTable property appropriately to display only those records which has a mission area consistent with the mission of the order in question. The pertinent source code is shown in Figure 9.7.

```

function TfrmMain.GetCurrentChildPointer(Sender: TObject) : TfrmOrderView;
var
    tempPointer : TfrmOrderView;
begin
    TForm(tempPointer) := ActiveMDIChild;
    GetCurrentChildPointer := tempPointer;
end;

procedure TfrmElement.FormShow(Sender: TObject);
var
    ChildPointer : TfrmOrderView;
begin
    ChildPointer := frmMain.GetCurrentChildPointer(Self);
    with tblElement do
    begin
        filter.clear;
        filter.add('MISSION_AREA = ' + ChildPointer.tblMissions.FieldByName('AREA').AsString) ;
        filter.Activate;
    end;
end;

```

Figure 9.7. MDI Child Form Pointer Handling Code Modules.

5. WebHub & TPack Components

The WebHub framework discussed above requires numerous components installed into the Delphi IDE in order to build WebHub-enabled applications. Furthermore, WebHub components use some lower level general functions found within the TPack suite of Delphi components. An analysis of some of these critical components is included in the discussion of the RedWeb implementation discussed below.

E. REDMAN ORDER PROCESSING SYSTEM FUNCTIONAL OVERVIEW

1. Introduction

Figure 9.8 depicts the interactivity between the REDMAN order processing system, the data model, and the input and output files. Appendix E contains the complete source code for all REDMAN modules.

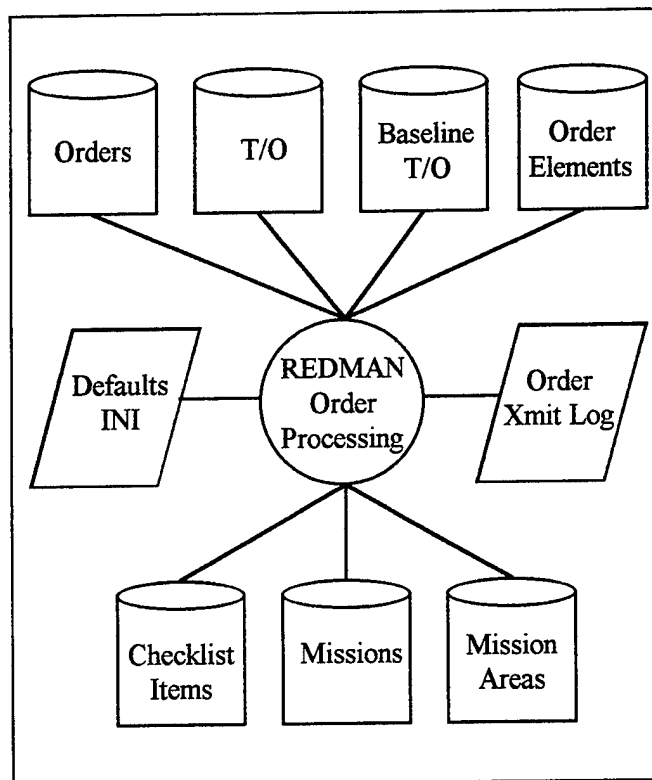


Figure 9.8. REDMAN Order Processing System Interactivity.

2. REDMAN Combat Order Processing

Figure 9.9 represents a screen shot of the REDMAN order processing system's view of a unit's task organization. The task organization sub-module supports an arbitrary nesting depth of units. It should be noted that for the purposes of USMC order processing, it is irrelevant to include mission data on units farther than one level below the initiating level of the order. The drag and drop of units (with user confirmation) in addition to the easy editing, insertion and deletion of units is accomplished via the integration of the functionality of the TDBOutline component discussed earlier and custom code developed in conjunction with the REDMAN implementation of this thesis.

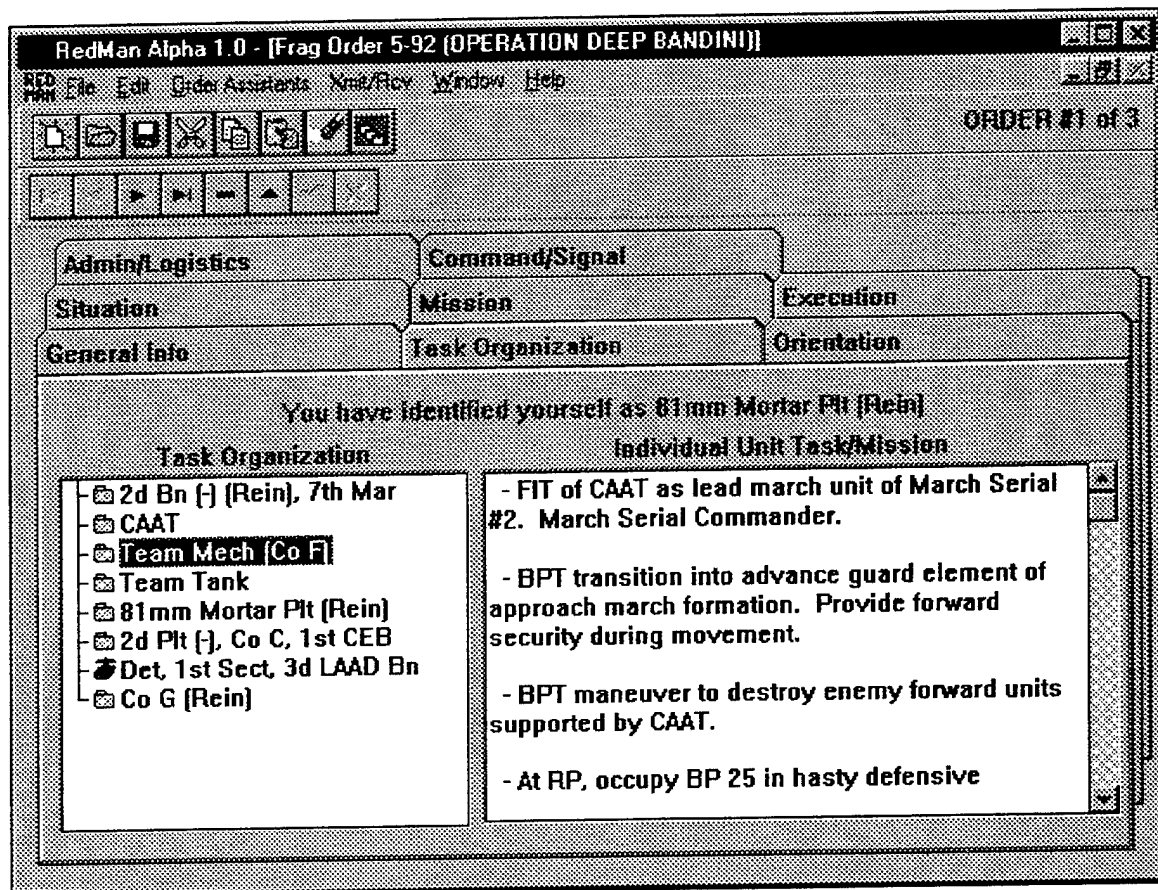


Figure 9.9. REDMAN Task Organization Screen.

Figure 9.10 shows the expansion of a subordinate level of command (with a double mouse click on the unit desiring to be expanded). Node units are represented with a USMC insignia bitmap icon, whereas internal nodes are represented with the standard Windows “folder” icon. In order to automatically generate outgoing orders, the REDMAN software needs to know which subordinate unit the user is representing. Hence, Figure 9.10 indicates that this user has identified himself as “Team Mech [Co F].” A user selects which unit he is for the purposes of each incoming order by pressing the “F10” function key when the task organization pointer is on his unit.

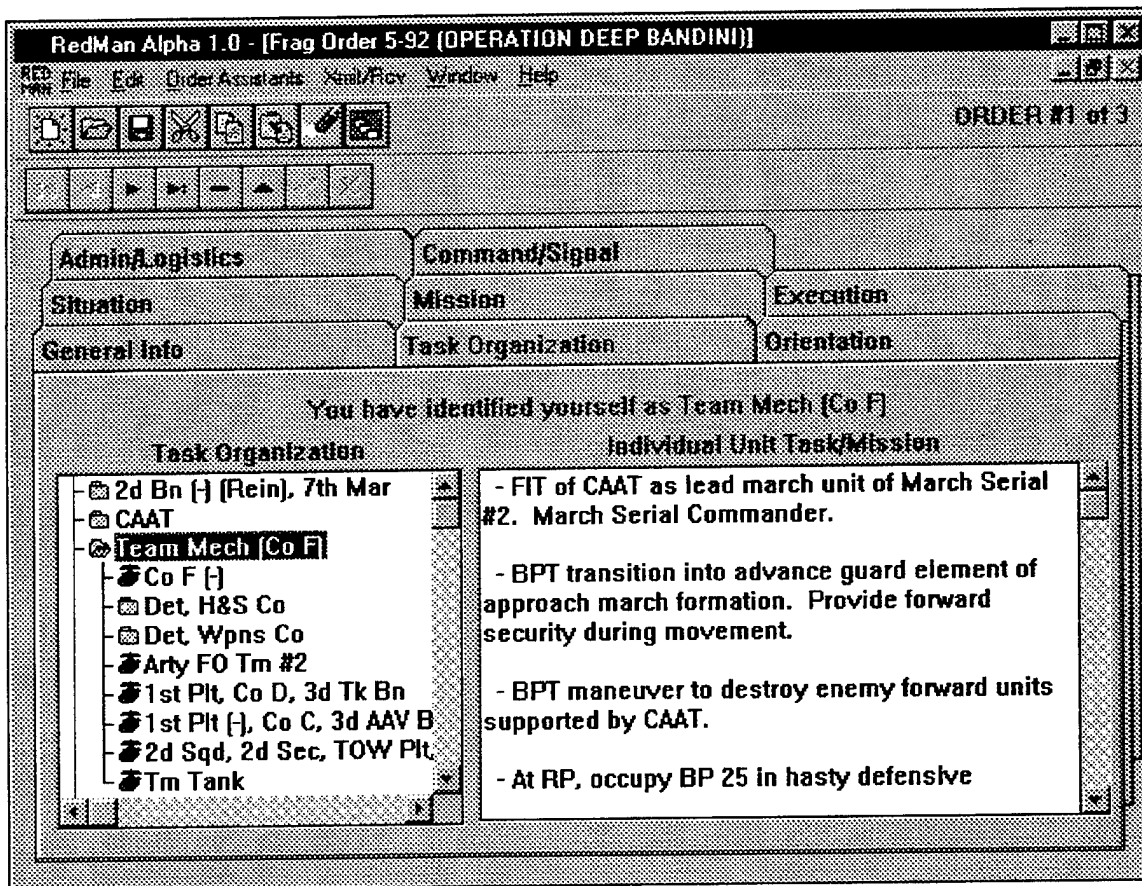


Figure 9.10. REDMAN Nested Task Organization Screen.

Figure 9.11 depicts the user interface for specifying the internal units to a particular level of command. This module can be customized to an arbitrary number and nesting of sub-units. When outgoing orders are generated, this list of internal units will be merged with all subordinate units given to the unit by the higher level commander in his order.

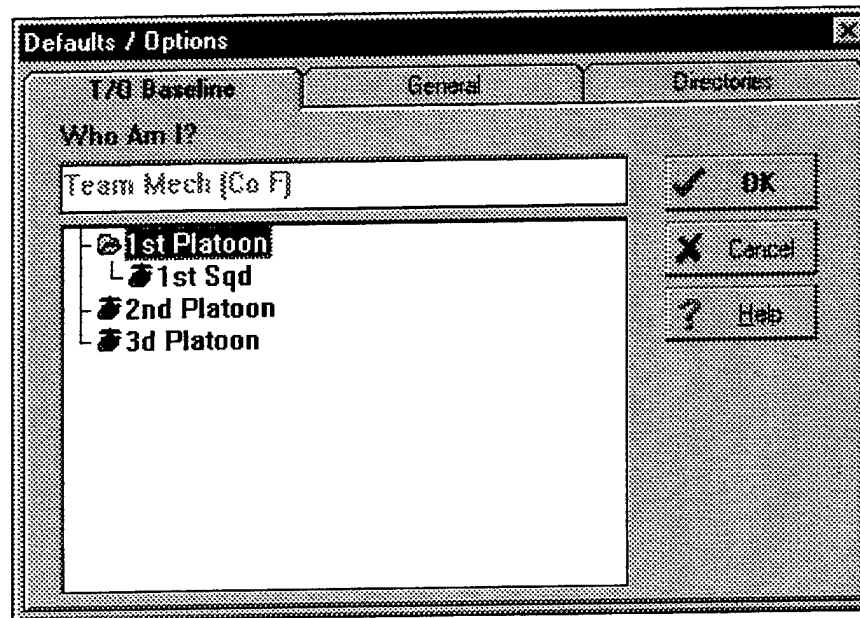


Figure 9.11. REDMAN Defaults/Options T/O Screen.

Figure 9.12 depicts the user interface for the Execution paragraph of the 5-paragraph operations order format. This particular aspect of the program demonstrates the use of an internal tabbed form embedded within another tabbed form. This allows extensive operations order structures to be represented to the user in an intuitive fashion.

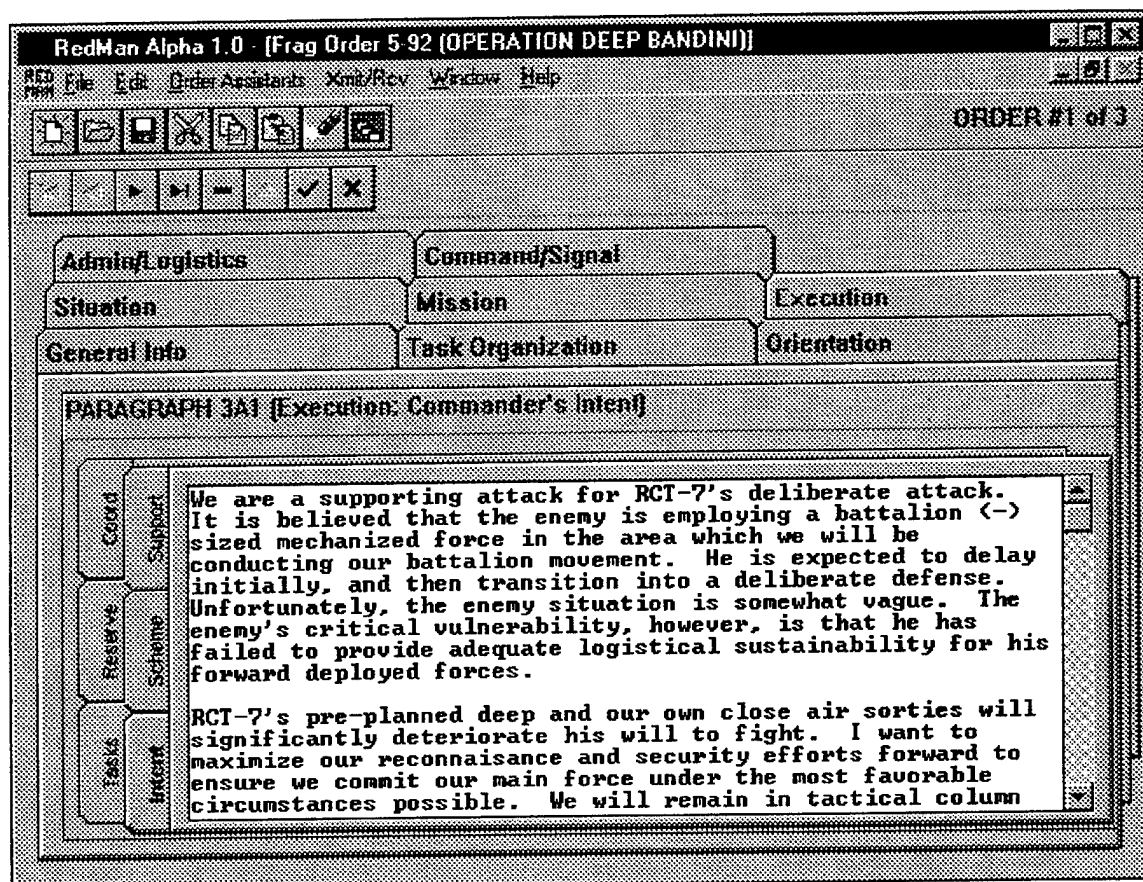


Figure 9.12. REDMAN Execution Commander's Intent Screen.

Figure 9.13 represents the user about to generate an outgoing order from his higher level commander's order. The user has properly identified himself to the software, and he has ensured his current internal task organization is current.

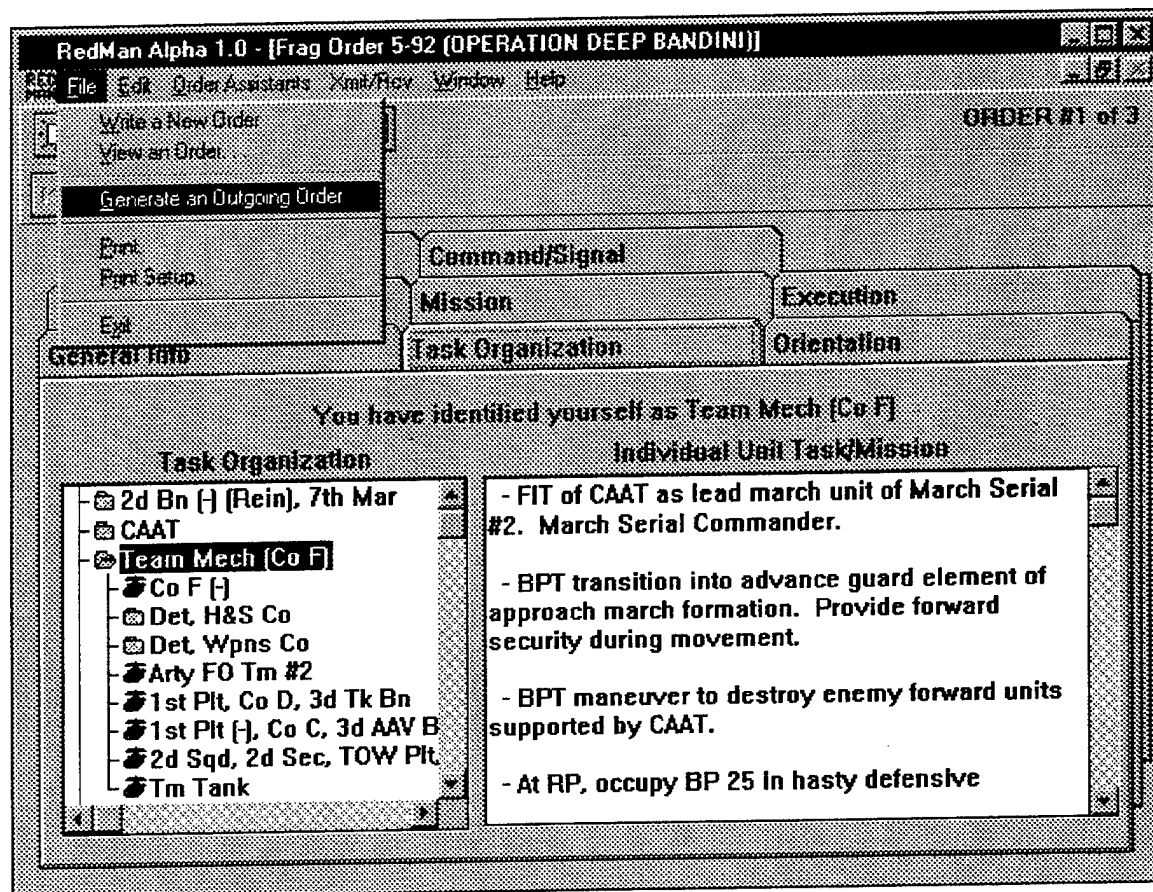


Figure 9.13. REDMAN Outgoing Order Generation Menu Screen.

Figure 9.14 represents the resultant outgoing order “General Info” screen. Every field which will be reused in the subordinate outgoing order is placed in the appropriate field unchanged, ready for user editing as required.

RedMan Alpha 1.0 - [New Order]

File Edit Order Assistants Xmit/Rcv Window Help

NEW ORDER

Admin/Logistics Command/Signal Situation Mission Execution General Info Task Organization Orientation

MISSION TYPE: Movement to Contact

INITIATING UNIT: Team Mech [Co F]

LOCATION: MCAGCC, 29 PALMS

DTG: 190833TAUG 96

SERIAL NO: MTR-5

TIME ZONE: T

ORDER NAME: Frag Order 5-92 [OPERATION DEEP BANDINI]

REFERENCE: a. Twentynine Palms East/West, Edition 3-DMA, Series V7953 1:50,000

Figure 9.14. REDMAN Outgoing Order General Info Screen.

Figure 9.15 depicts the generated outgoing order screen shot. Note how the default internal T/O units have been merged with all the units given to the subordinate unit by the higher level commander. These units can be further task organized as desired by the unit commander using the REDMAN software.

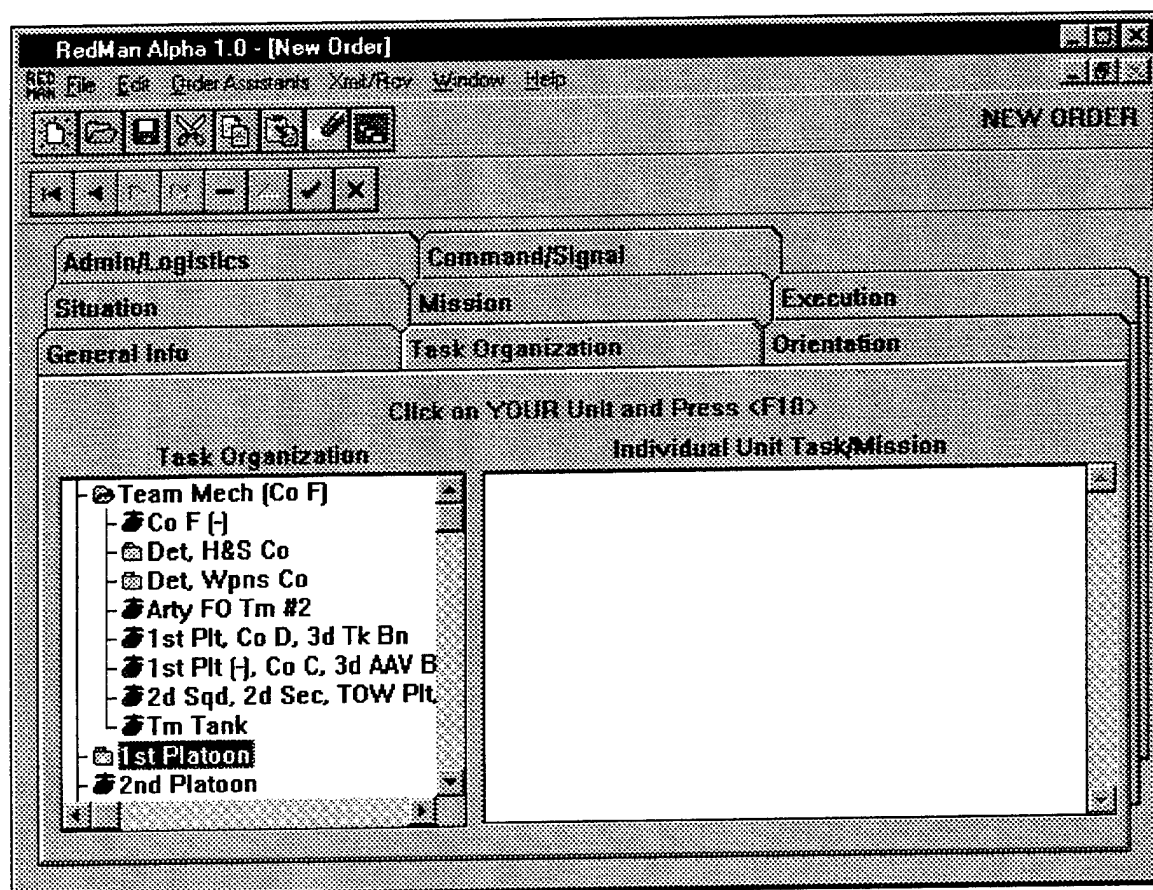


Figure 9.15. REDMAN Outgoing Order T/O Screen.

Figure 9.16 depicts the generated outgoing order mission statement. This exemplifies the ease of use of the REDMAN software since this mission was directly copied over from the higher unit commander's order (by definition a higher unit commander's mission statement to each subordinate unit will never be modified by the subordinate unit).

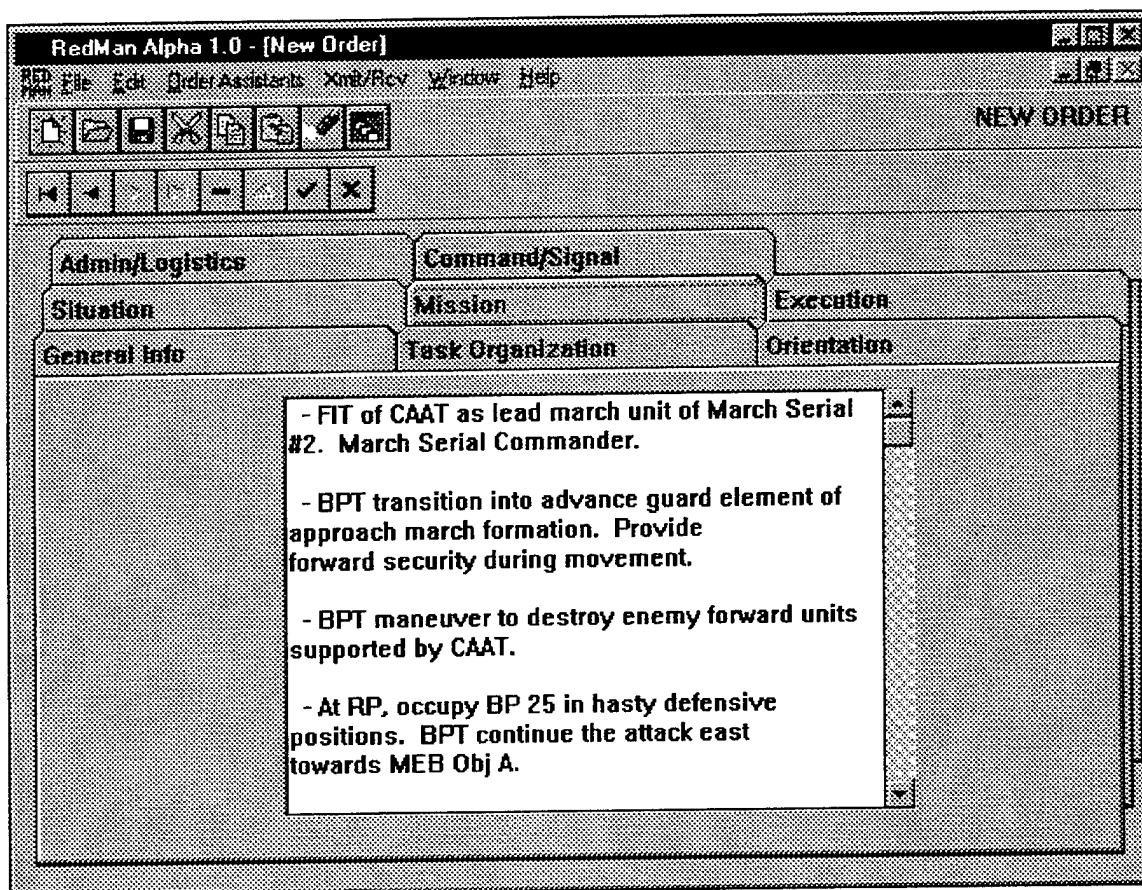


Figure 9.16. REDMAN Outgoing Order Mission Screen.

Figure 9.17 depicts the Checklist Items form which provides an easy-to-use table of editable checklist items which the user can use to ensure he has considered all standard factors in determining a course of action when constructing his order. This table is malleable and customizable by the user. Depending on the mission area of the operation selected by the user, the checklist items database will be searched to constrain entries to just those checklist items corresponding to a particular mission area.

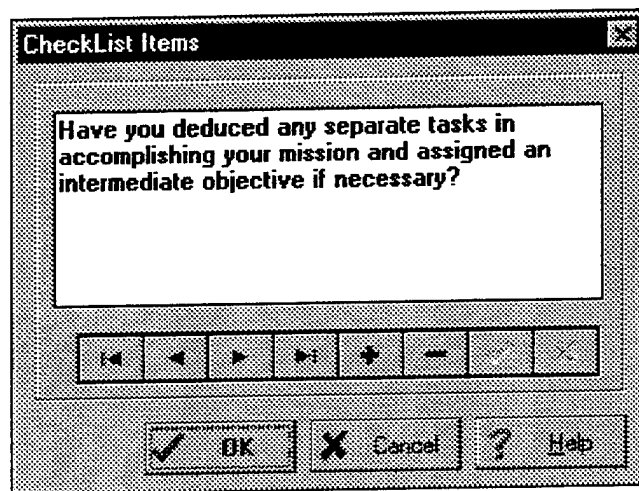


Figure 9.17. REDMAN Order Checklist Items Screen.

Figure 9.18 depicts a similar interface to the order elements. This table consists of boilerplate text which can be incorporated into the user's order with a simply click of the "copy" button. Like the checklist table above, the database is completely configurable.

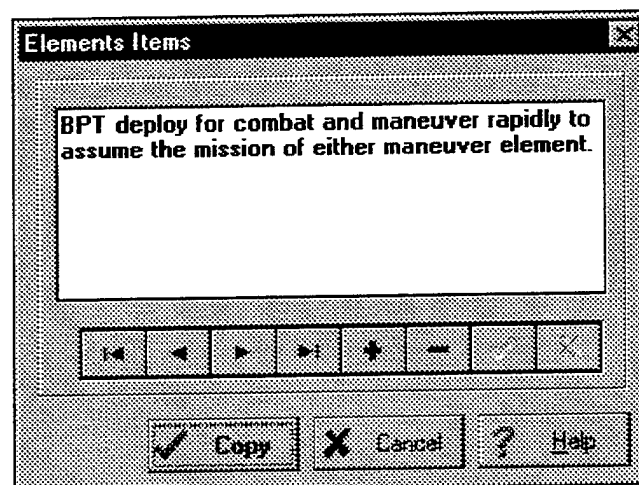


Figure 9.18. REDMAN Order Elements Screen.

Figure 9.19 depicts the result after the user clicked on the “copy” button when the system was in the state represented by Figure 9.18. Since the task organization unit pointer was sitting on 2nd Platoon, the boilerplate text was copied to the unit task/mission memo field corresponding to the current unit.

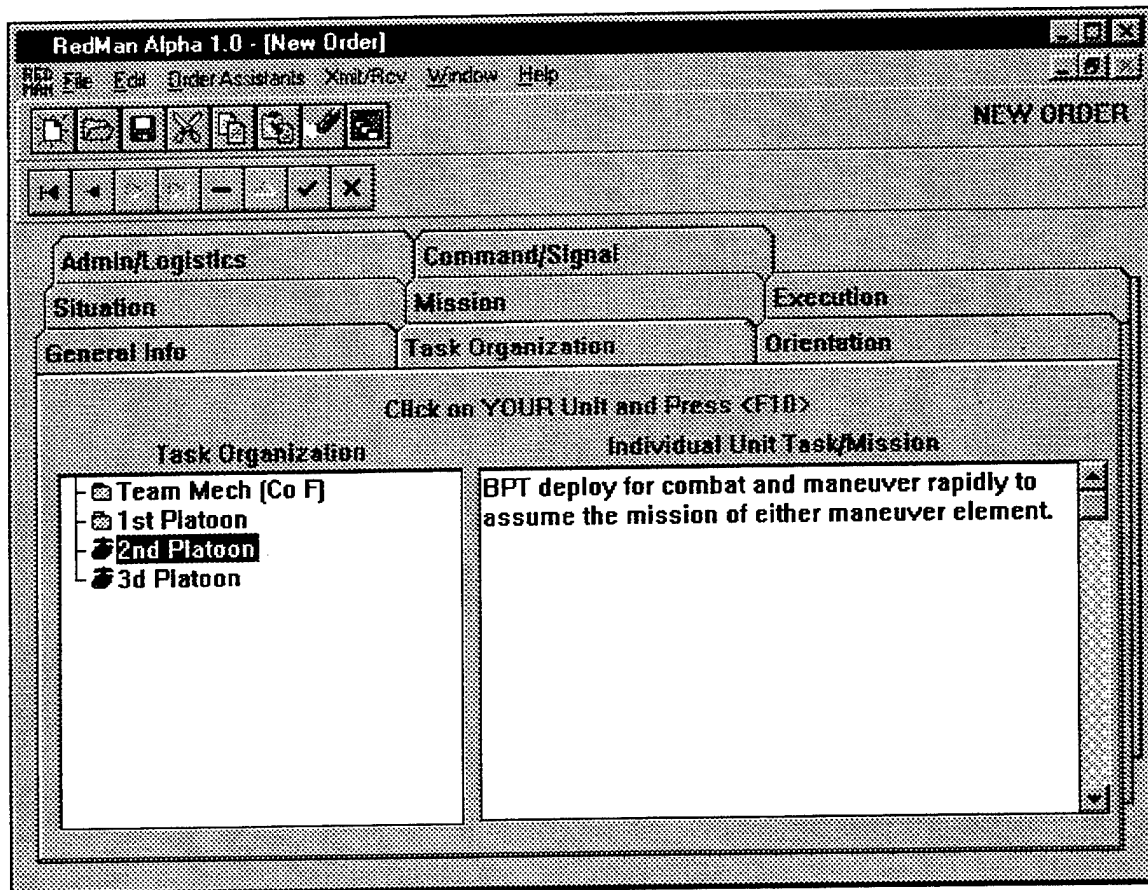


Figure 9.19. REDMAN Order Elements Screen.

Figure 9.20 reflects the user interface for order message packet transmission and reception. Use of the “AutoSend” and “AutoReceive” functionality in conjunction with an intelligent agent would be the preferred methodology for effecting transmission of message packets. This interface is provided for future growth of the system as it would be relatively straightforward to encapsulate the various communication technologies within the REDMAN software itself. Currently, this various technologies are accessed through independent programs running in conjunction with the REDMAN application. Selecting “Transmit Orders-AutoSend” from the menu currently isolates and compresses the correct order in a consolidated integral order packet for eventual transmission to other DACTs or later generation palmtops running the REDMAN order processing software.

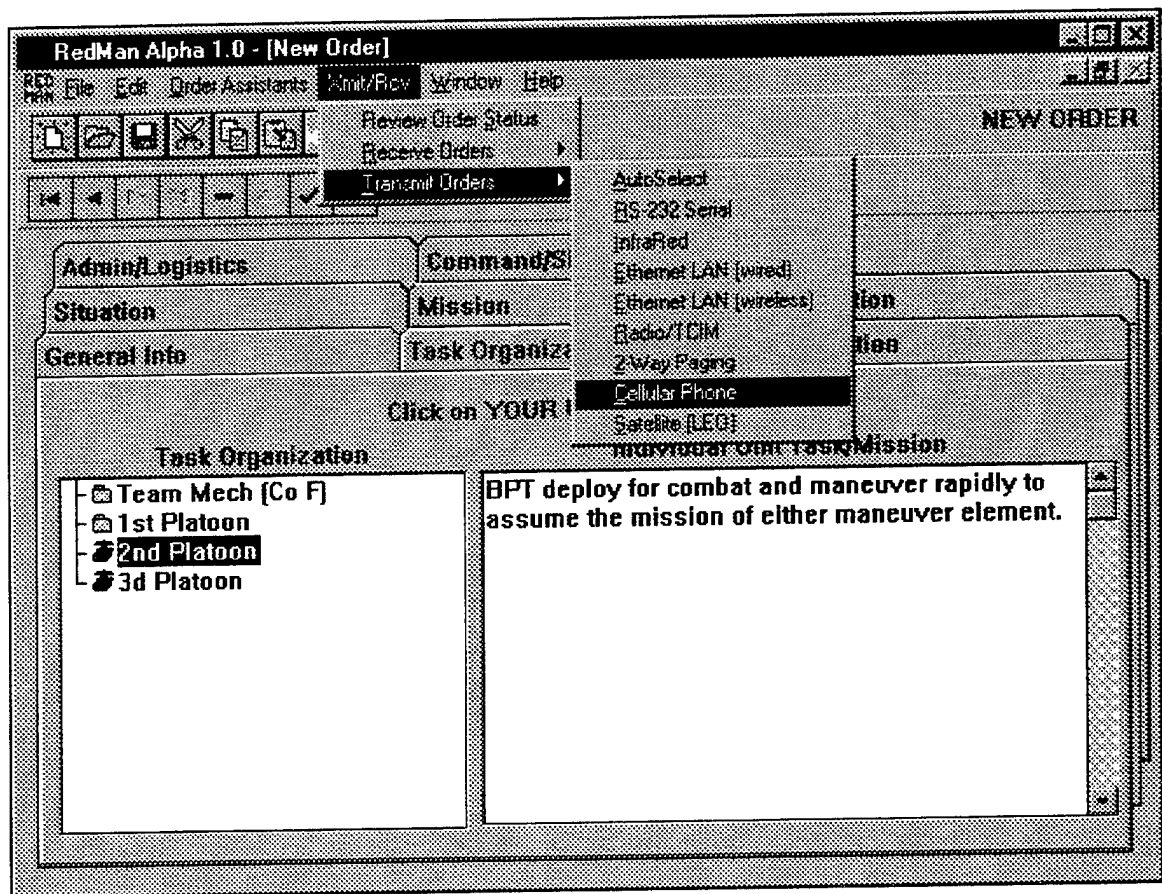


Figure 9.20. REDMAN Transmit/Receive Screen.

F. WEBSITE WEB SERVER

Although any number of World Wide Web servers can be used in conjunction with the RedWeb HTML processing system, Website 1.1 Service Release 4 (version 1.1e) was selected for RedWeb due to its easy of use and wide compatibility with various operating system platforms and the WebHub application framework. Although WebSite is a 32-bit product (hence, it requires either Win95 or NT to run), RedWeb could also be run strictly on a 16-bit platform by using the 16-bit Win-Httpd product). However, it is anticipated that the eventual DACT system will be Win32 API compatible, hence, WebSite or similar 32-bit product offers better performance and closer compatibility to the eventual form of a RedWeb enabled web site. Figure 9.21 depicts the default WebSite Server properties screen.

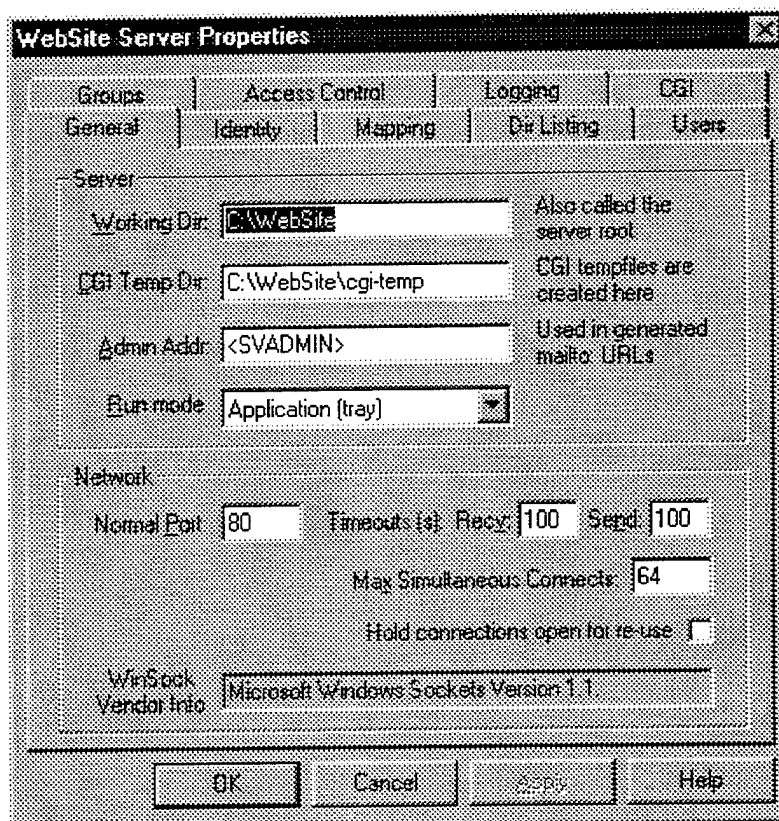


Figure 9.21. WebSite WWW Server Properties Interface Screen.

G. REDWEB IMPLEMENTATION

1. Introduction

Figure 9.22 depicts the data and programmatic flow of the RedWeb HTML processing system implementation. After loading the WebSite server and the 16-bit version of the WebHub, a single web application (or "WebApp") called REDWEBP.EXE is then executed. The WebHub senses its presence and prepares to serve pages as defined by the initialization files (REDWEB.INI and HUBMAIN.INI) for the RedWeb application. When a web surfer clicks on a link which is mapped to the CGI program representing the WebSite runner (along with a corresponding WebApp name and page id), the runner is dynamically loaded from the WebSite web server. The runner then passes its arguments to the WebHub system for dynamic construction and display of the appropriate HTML pages.

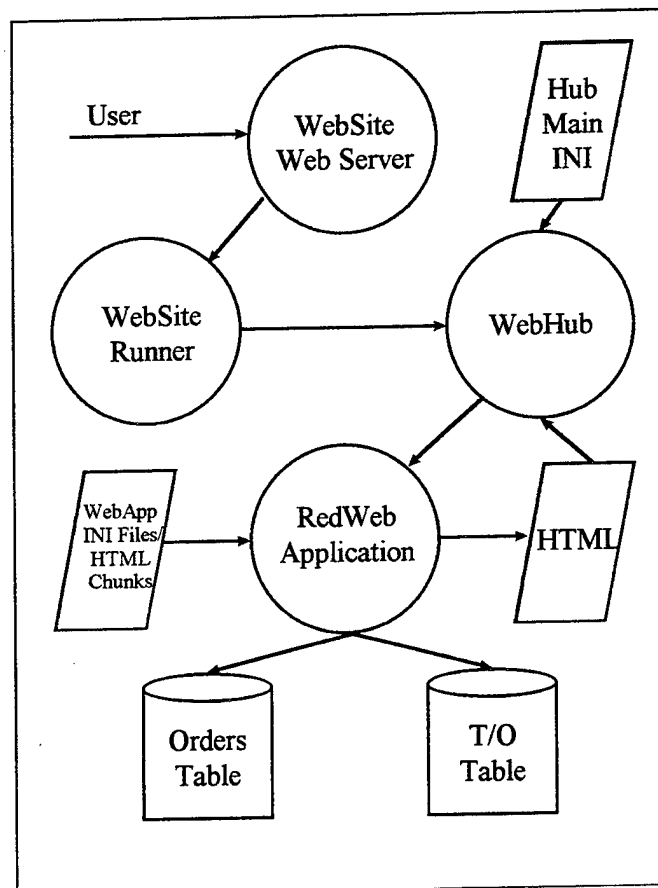


Figure 9.22. RedWeb Connectivity and Data Flows.

After validating that the WebApp and page id requested are valid, the arguments are passed to the WebApp for execution. Based upon the INI file entries corresponding to the page ID requested, HTML chunks listed sequentially in the INI file and defined by external HTML files are output to a temporary file to dynamically construct the web page. The WebHub system then passes back the dynamically constructed page via the runner to the WebSite server which returns the page to the surfer's browser.

If data is needed from an external database to populate an HTML select box, for example, an "event macro" can be called within the HTML to pass control to a procedure within the RedWeb application. This procedure can also output HTML code, text and data to the dynamic page. Hence, there is really no practical limit as to the level of programming control one can exert over an HTML page. If it can be coded, it can be coded in Delphi, and it can be returned on an HTML page to the surfer. An example of both the unformatted and formatted HTML page produced by RedWeb for the sample order is contained in Appendix F.

All RedWeb INI files, HTML files, and source code for the RedWeb WebApp can be found in Appendix G. Figure 9.23 depicts the 16-bit WebHub system user interface connected to the RedWeb application (with an application ID of "REDMAN").

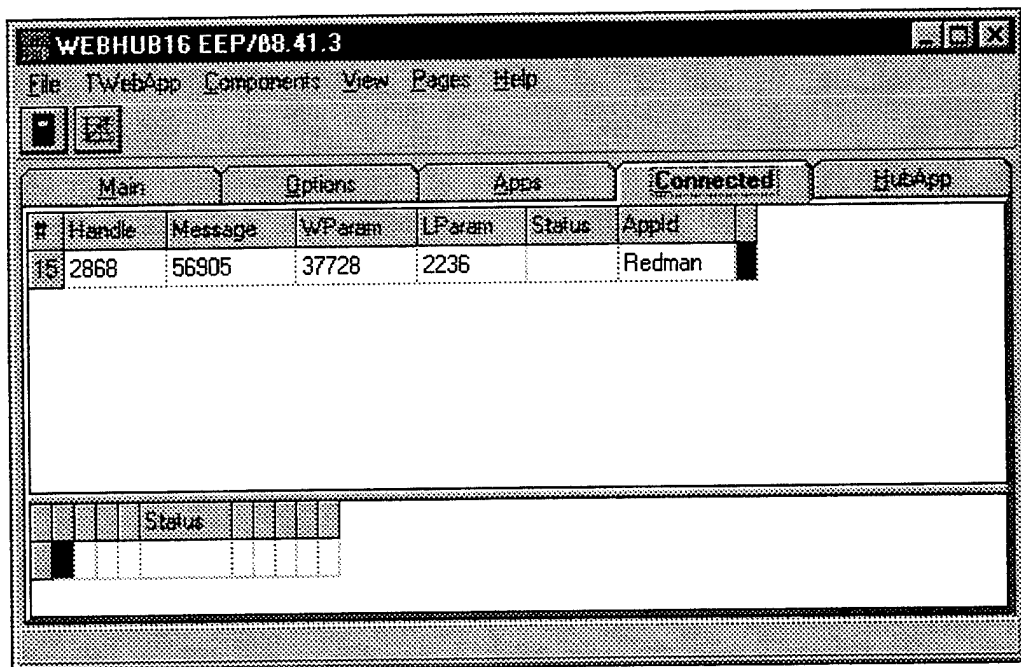


Figure 9.23. 16-bit WebHub Connected Status Screen.

Figure 9.24 depicts a screen snapshot of the RedWeb Application as it is running on the Web Server. This program is meant to run strictly on the Web Server and is never seen directly by users. It will only be accessed directly by system programmers or system administrators who desire low-level debugging or a statistical report from the WebApp while it is executing. The Delphi project which contains RedWeb is based on the “standard gallery project” provided with WebHub. As recommended in the user documentation, this is often the best starting place to build a WebApp since all of the mandatory (and admittedly overly convoluted) components which comprise a WebApp program are built for you with a standardized user interface. Most users find the user interface much too in-depth as well, but it is probable that the WebHub designers wanted to err on the side of including too much information in user applications. In any case, this convoluted nature of the baseline gallery project further complicates the learning curve of WebHub. Note that Figure 9.24 is displaying the HTML corresponding to a particular page selected by the operator. This can be extremely useful for debugging WebHub applications.

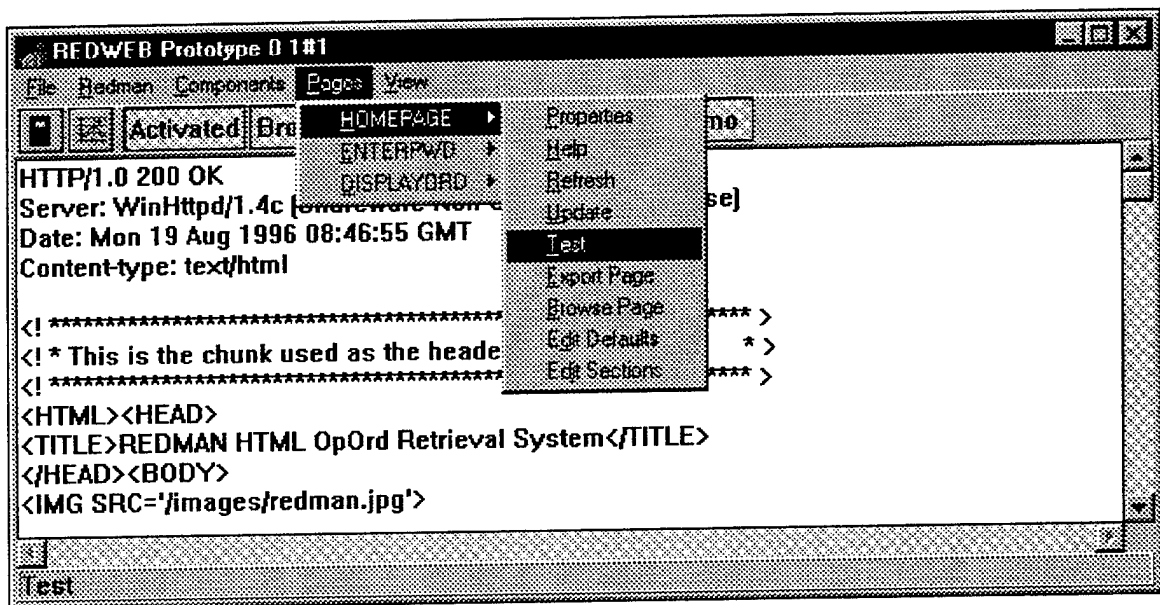


Figure 9.24. RedWeb Application Page Test Screen.

2. RedWeb HTML Order Generation

Figure 9.25 represents the Netscape Web browser rendering of the homepage generated by the RedWeb system. Note the URL for this page is "*http://127.0.0.1/cgi-win/hubws.exe?REDMAN:homepage*". The server IP address of "127.0.0.1" indicates "localhost" mode where the server is serving pages directly to the system which is requesting them. If the RedWeb system is desired to be brought on line with a live Internet connection, this IP address is merely replaced by the actual IP address of the remote server. Furthermore, the Website server properties will need to reflect the IP address assigned to that server. Since most dial-up connections are assigned dynamic IP addresses, this may need to be configured by the user when he brought the RedWeb system up. In most tactical environments, DNS name resolution can be employed with static host names so that browsers can be preconfigured with the appropriate URL bookmarks thereby requiring no configuration by users.

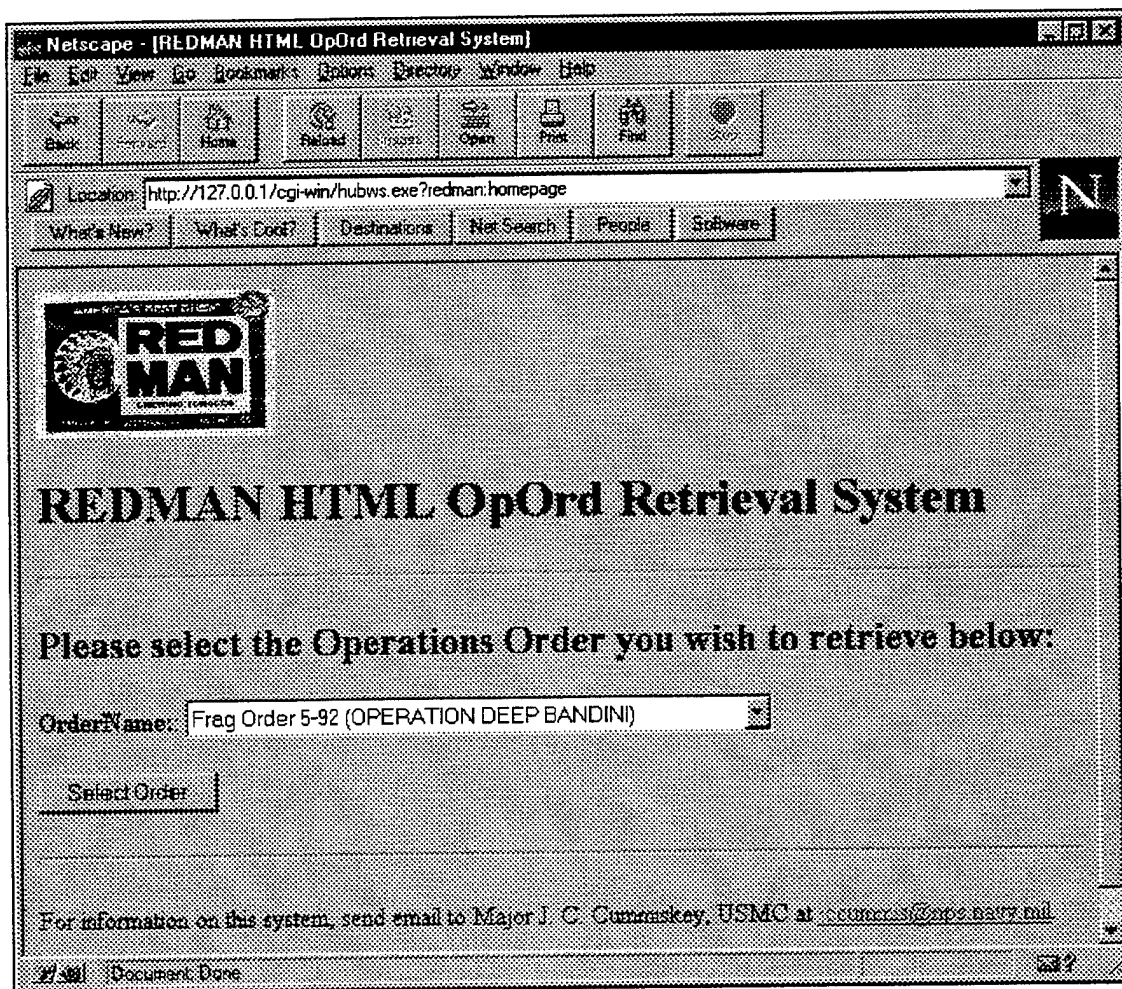


Figure 9.25. Netscape Web Browser with RedWeb HomePage Screen.

Figure 9.26 reflects the potential of the RedWeb application. Rather than being simply a static HTML page, the drop down selection menu HTML widget in the center of the form was dynamically populated from the current REDMAN order database table. This selection menu informs the user there are three orders currently available for viewing.

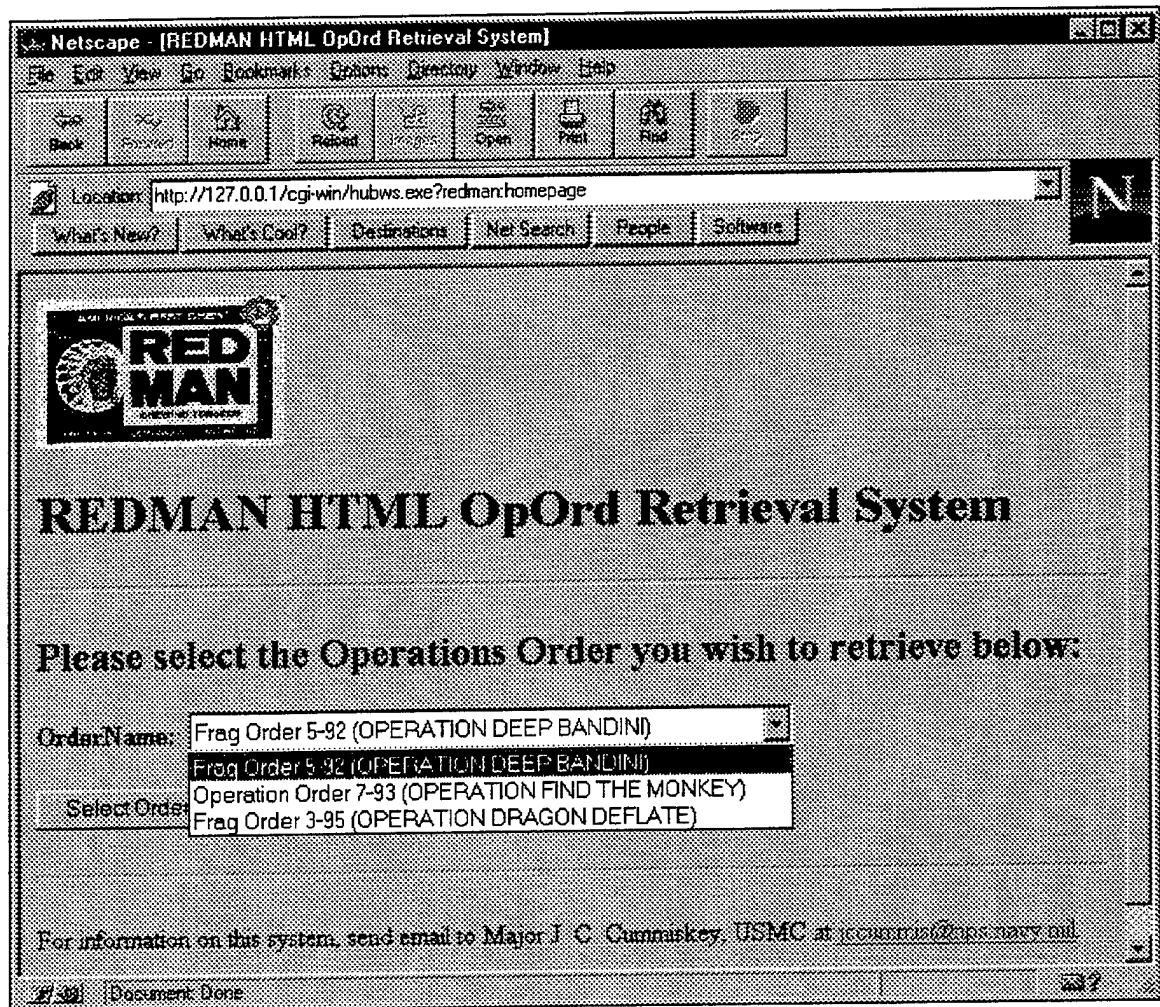


Figure 9.26. RedWeb Order Selection Screen.

Figure 9.27 represents the next page created dynamically by the RedWeb system. Since the user selected the first order in the orders table (OPERATION DEEP BANDINI), the appropriate task organization table corresponding to this order was opened and read, and another selection menu widget was dynamically populated with the unit name values contained within the task organization.

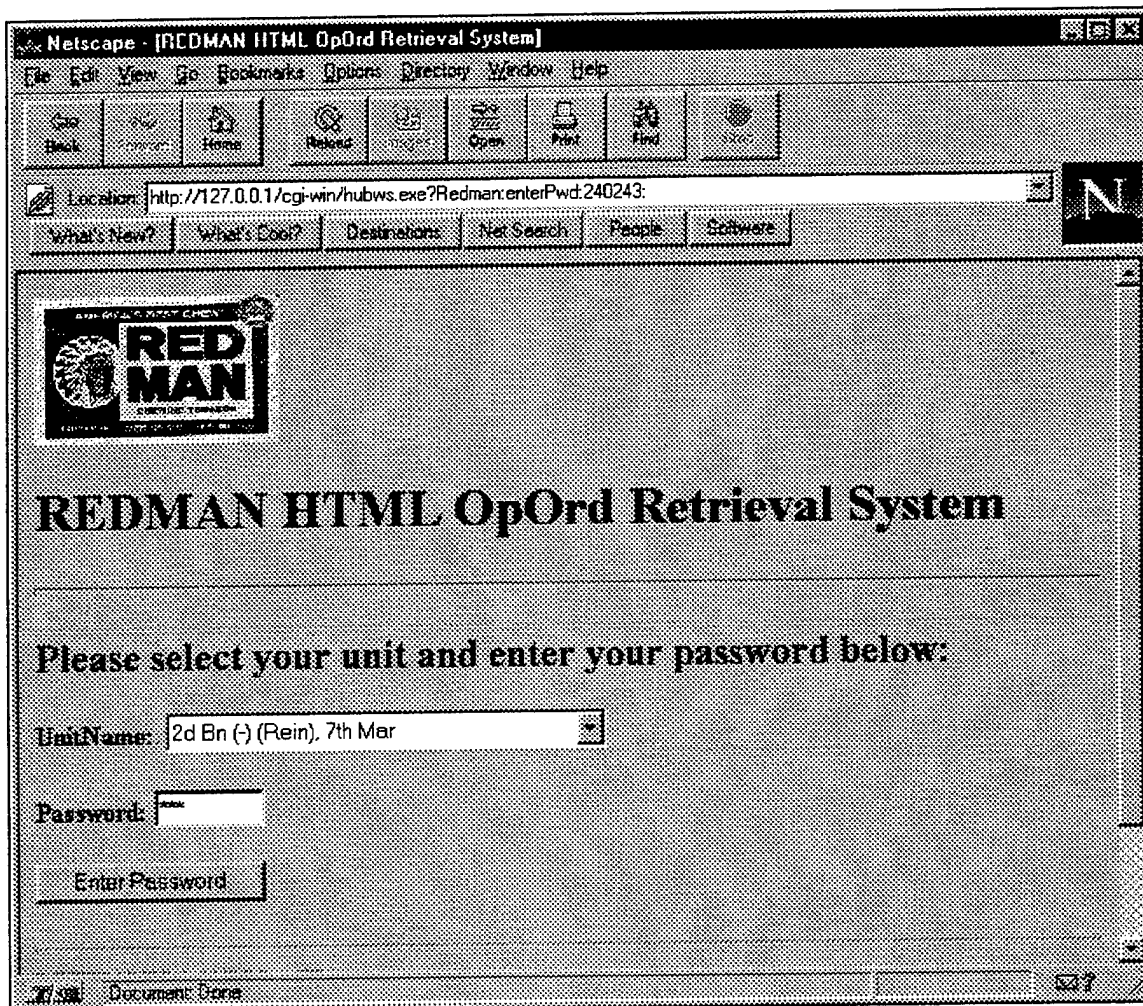


Figure 9.27. RedWeb Unit Selection Screen.

Figure 9.28 depicts the user scrolling through the HTML selection menu searching for his unit. He will then be asked to enter a specific user password corresponding to the PASSWORD field in the task organization table.

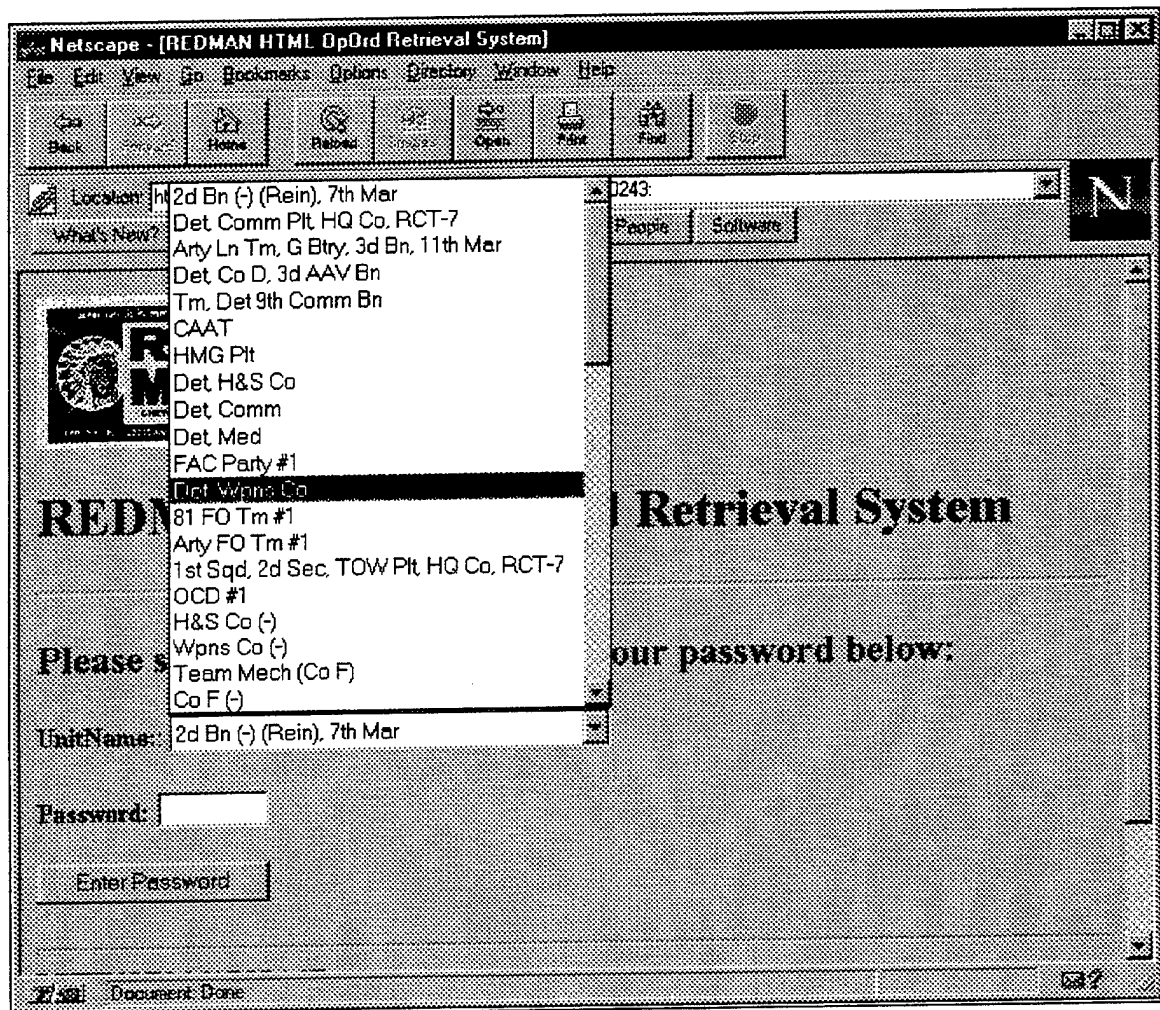


Figure 9.28. RedWeb Unit Selection Scrolling Screen.

Figure 9.29 reflects the page returned to the user if the password he entered is invalid. A HTML hot link is provided to quickly return to the appropriate page in the event he simply made a typographical error. Logging of security violations can be automatically accomplished using internal WebHub logging functionality.

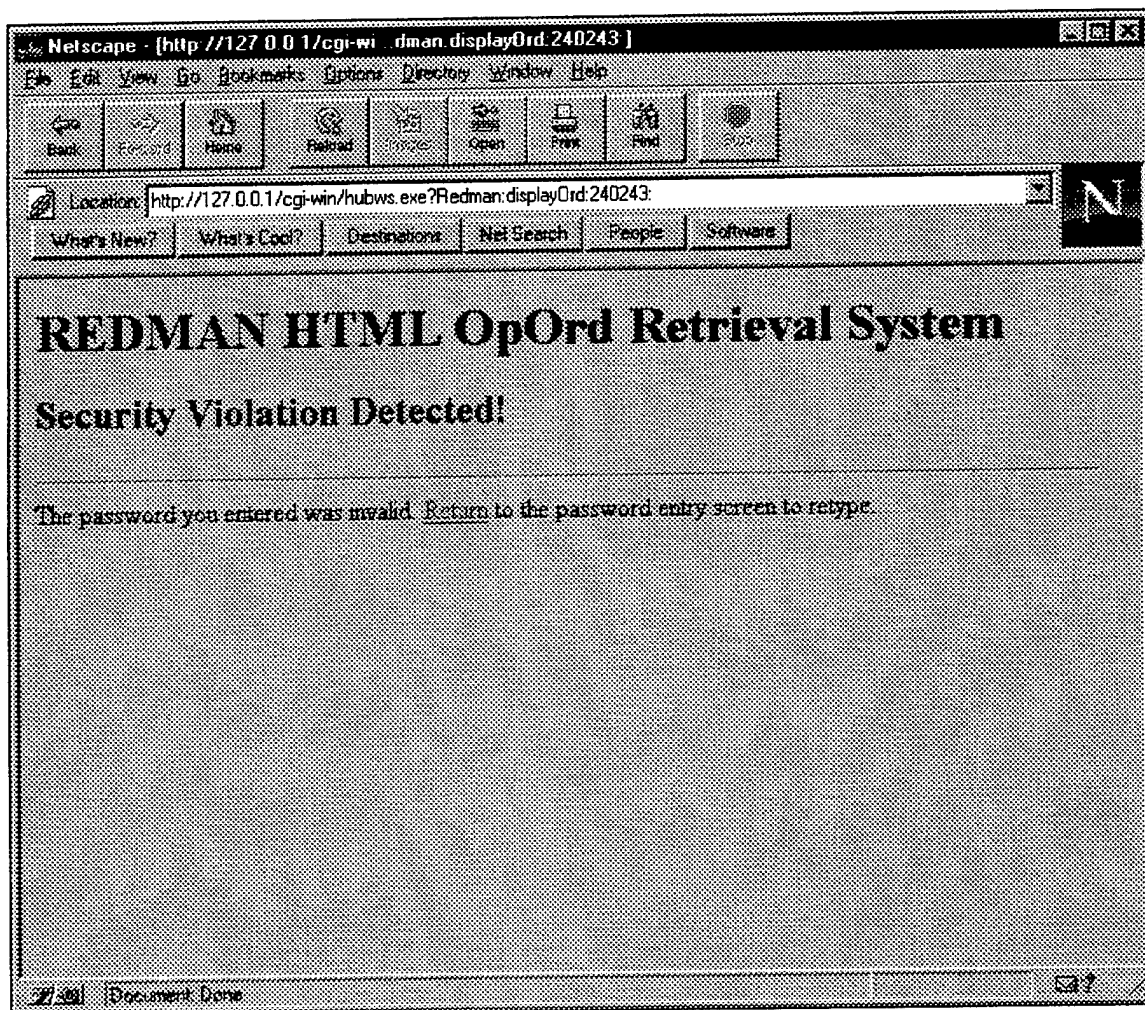


Figure 9.29. RedWeb Invalid Password Screen.

Provided the user enters the correct password, Figure 9.30 depicts the first screen of the operation order the user requested. All datafields have been retrieved from the REDMAN database and formatted appropriately for HTML display. This HTML message has been streamlined to remove all extraneous links, headers and footers to foster complete portability of the HTML order packet among various viewers (some of which may not support graphics, etc.). Furthermore, it is desirable that the HTML order message be as small as possible for subsequent data transmission. Internal document links are maintained with a short table of contents to allow easy browsing of the combat order document.

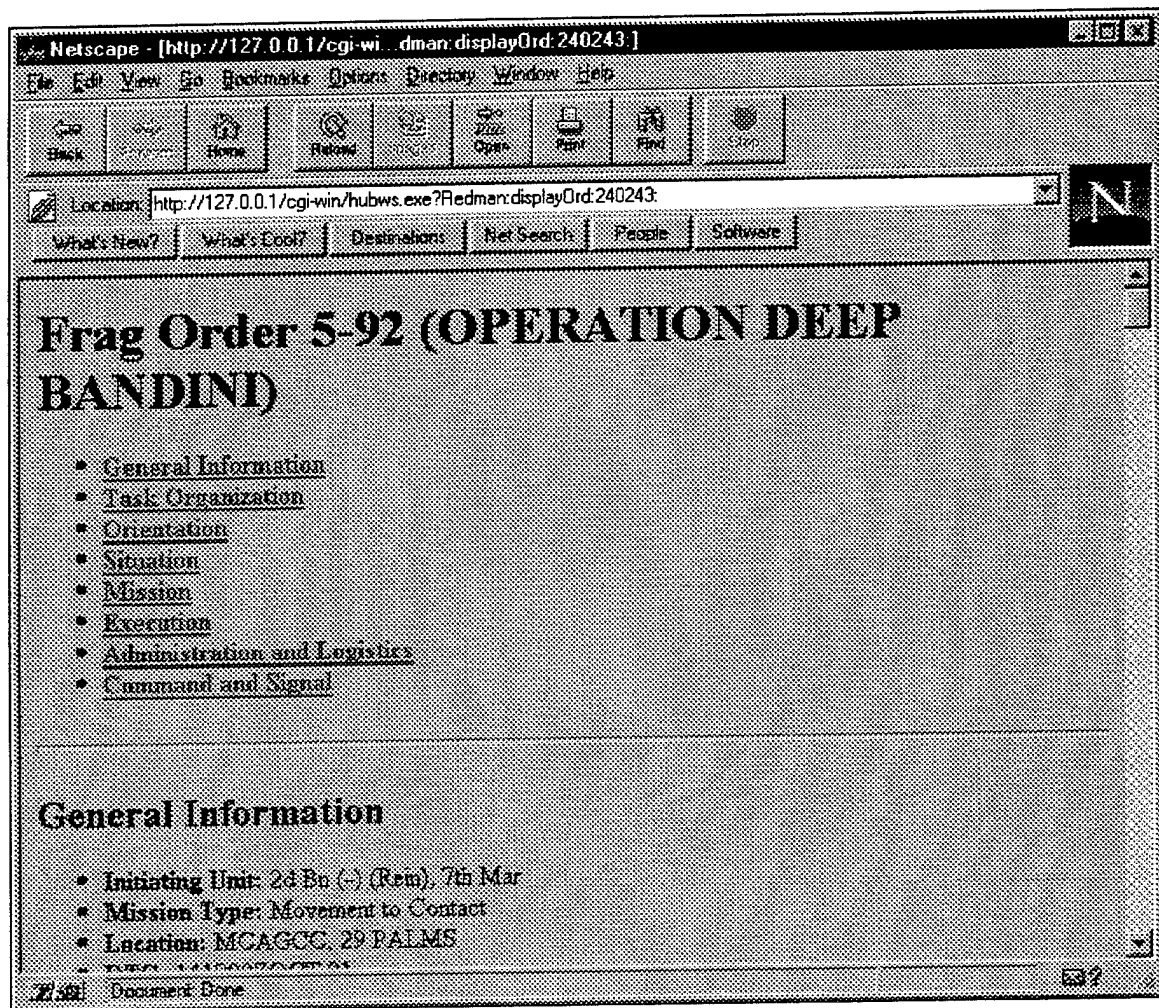


Figure 9.30. RedWeb HTML Operation Order Screen.

Figure 9.31 represents the beginning of the HTML task organization section of the selected operation order. The Delphi and WebHub code to produce this HTML chunk is documented below in Section H of this chapter.

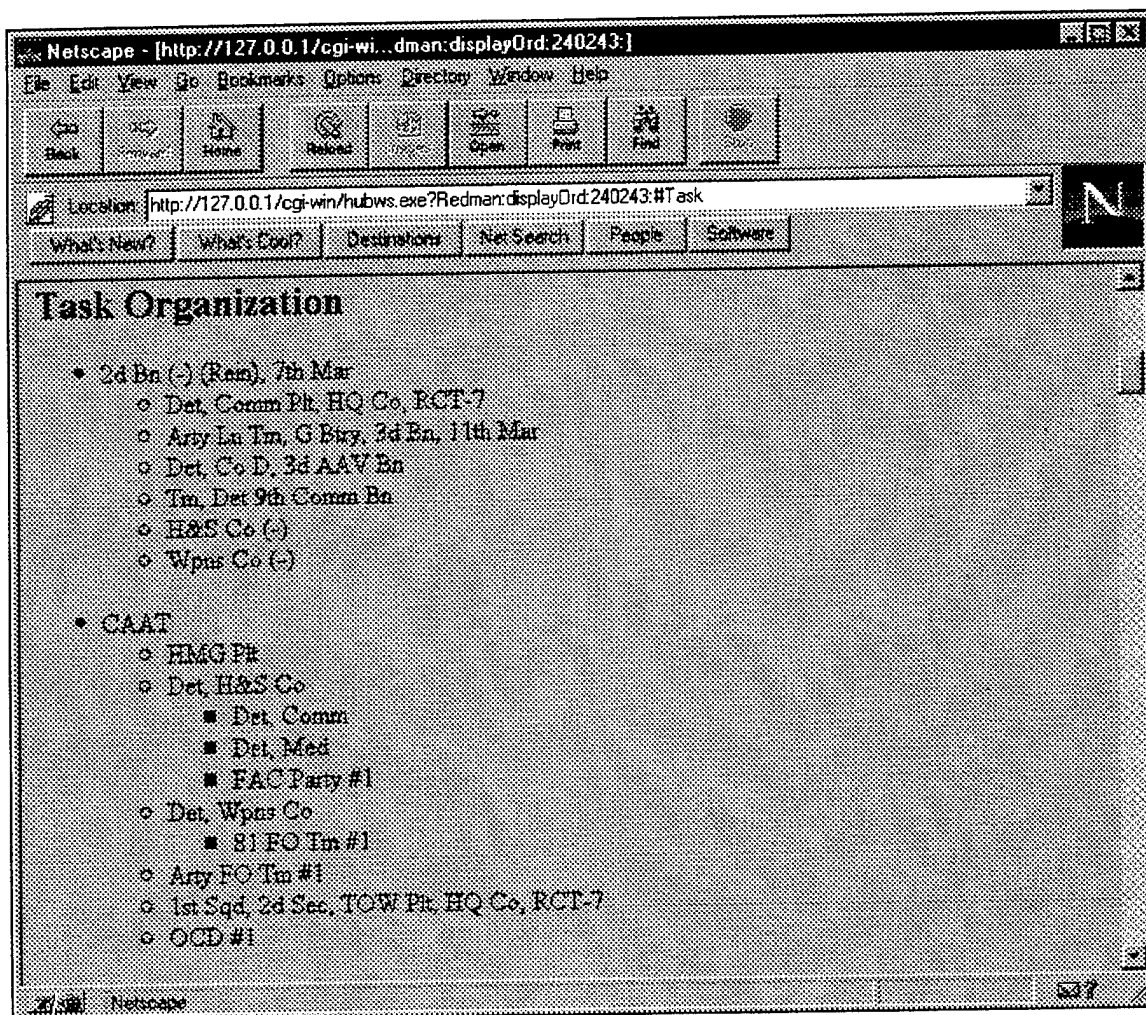


Figure 9.31. RedWeb HTML Task Organization Screen.

Figure 9.32 shows a segment from the Tasks subparagraph of the operations order selected by the user. Note the use of embedded unordered HTML lists to achieve a very easy to read, yet portable interface. Whenever tabs or hard returns are found in the database source memo fields, an appropriate HTML tag is inserted to maintain the neat and orderly appearance of the data.

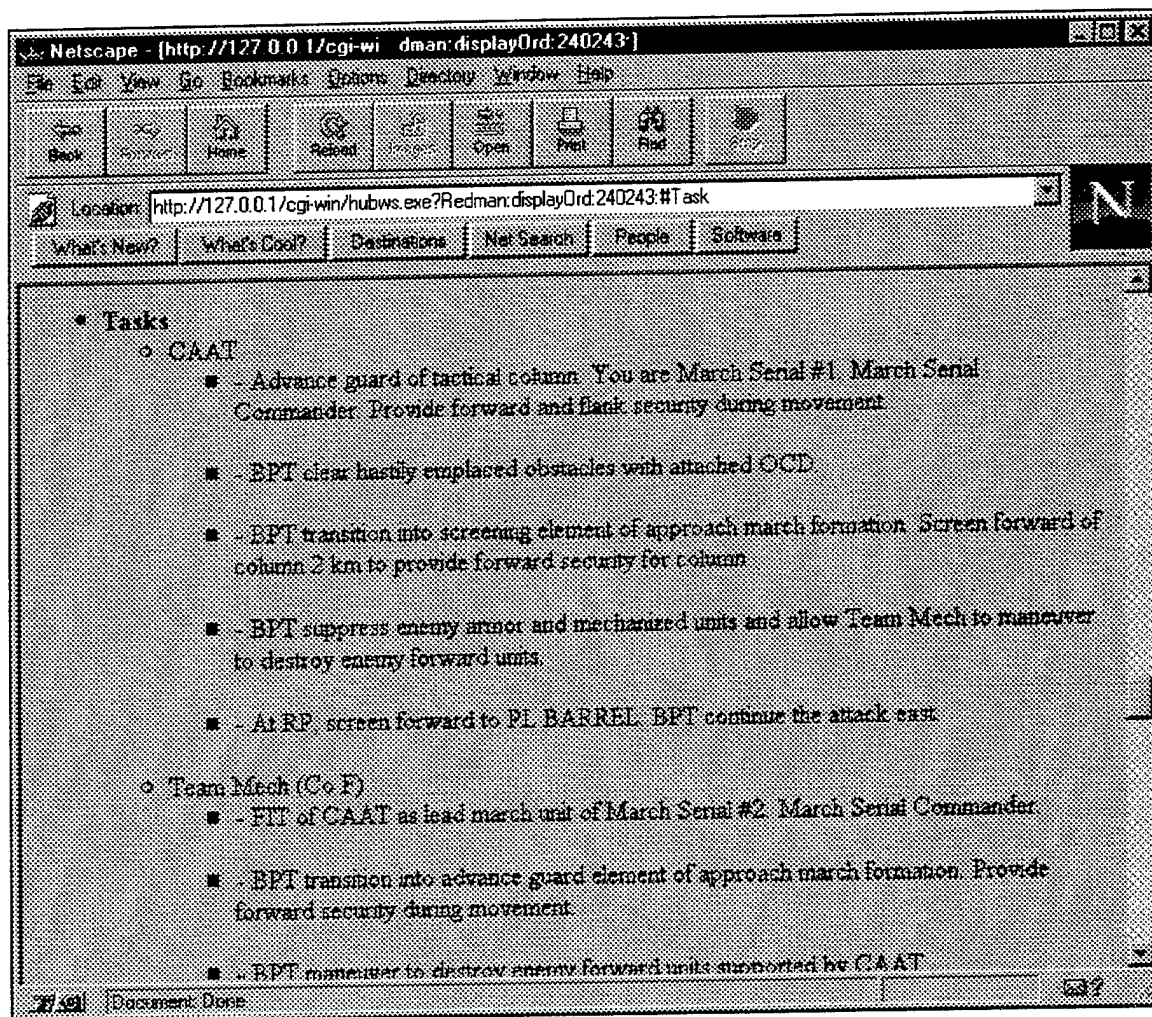


Figure 9.32. RedWeb HTML Tasks Screen.

Figure 9.33 reflects the “Command and Signal” paragraph for the selected operations order. Note the use of the “Return to Top” hotlink for ease of internal document navigation by the user. Since this hotlink is used in multiple places in the document, it is a perfect candidate for a macro which is maintained in the REDWEB.INI file in the [TWebApp.Macros] section as:

```
ReturnToTop=<P>Return to <A HREF="#Top">Top</A></P>
```

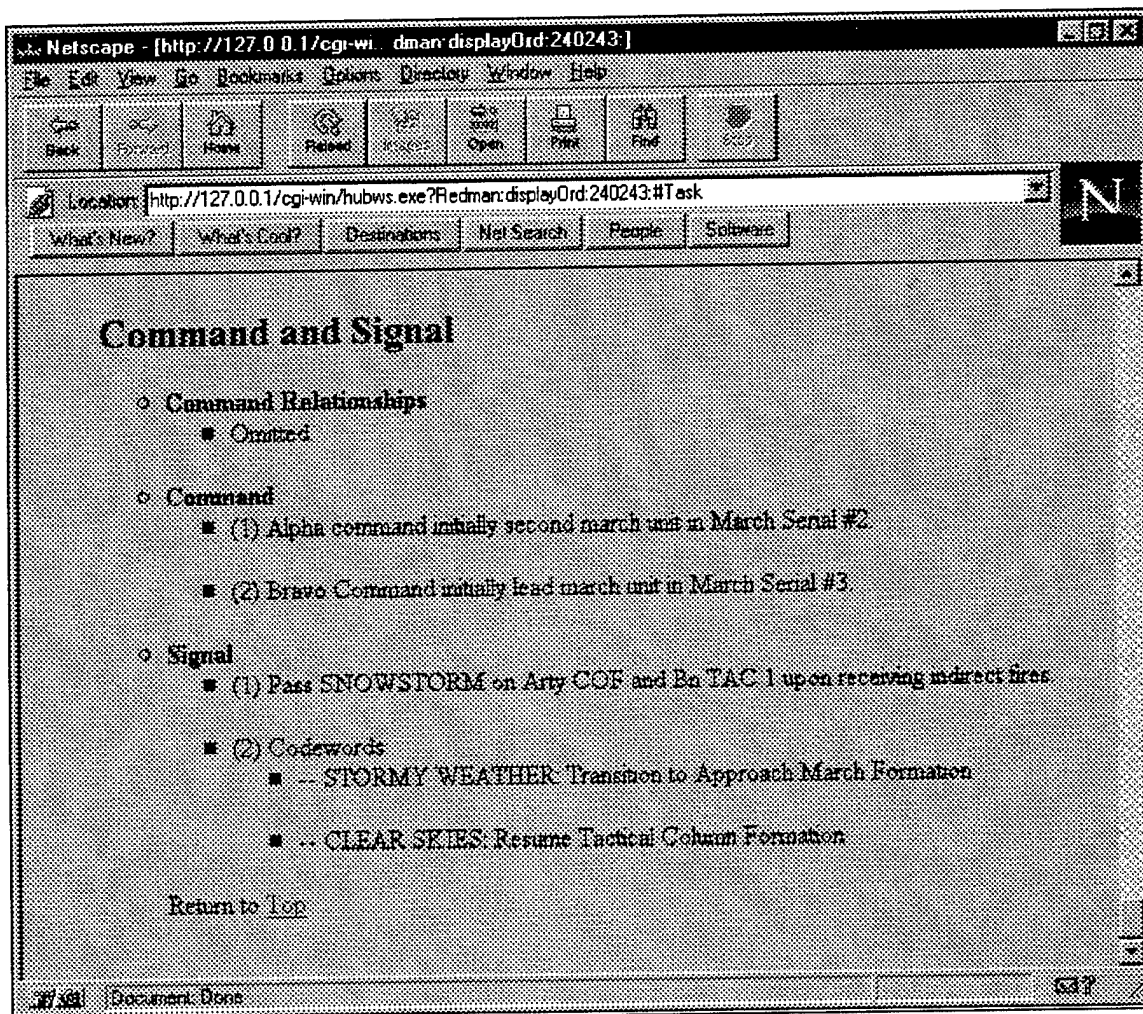


Figure 9.33. RedWeb HTML Command and Signal Screen.

3 Format of the WebApp INI file

The INI files used within the WebHub framework are the key to understanding the system. For example, here is the page entry for the homepage of the RedWeb application:

```
[HOMEPAGE]
1=pageheader
2=HOMEPAGE
3=pagefooter
```

The entry within the brackets indicates the name of the page being constructed (as defined in the [TWebApp.Pages] section of the INI file). The sequential listing of numerals lists the HTML “chunks” which make up the page which will be dynamically constructed by the WebApp when the “homepage” page id is called by the WebHub. Hence, the homepage of the RedWeb system consists of three HTML chunks defined respectively in HTML files which are a component part of the system. The purpose of the INI entry “ChunkDir=c:\redweb\” is to tell the system that all the HTML chunks belonging to the application can be found in a specific directory.

The INI file can also contain macros which provide a streamlined way to conduct maintenance on a web site. Macros in WebHub are used very similarly to how constants are used in a standard programming language. For example, the RedWeb system defines the title of site as follows: “SiteTitle=REDMAN HTML OpOrd Retrieval System.” Whenever, the macro title “SiteTitle” is found in an HTML chunk (with the syntax %=SiteTitle=%), the macro is expanded to “REDMAN HTML OpOrd Retrieval System.”

The last item of significance in the RedWeb INI file is the “[TWebApp.Events]” section. This section lists all the “event macros” which are called dynamically by the HTML chunks (normally to query a backend database) and which map to a corresponding “event handler” procedure within the WebApp itself. For example, the HTML chunk for the “homepage” referred to in the “[HOMEPAGE]” page definition above, is defined as follows in the REDWEB.HTM file:

```
%=PageBegin=%homepage=,,%=SiteTitle=%%=where=%
<! ***** >
<! * This is the Site's HOMEPAGE. It includes the form for * >
<! * selecting the particular Operations Order in the database * >
<! * the user wants to retrieve. Note all images should be * >
<! * stored in the c:\website\htdocs\images directory. * >
<! ***** >
<H2>Please select the Operations Order you wish to retrieve below:</H2>

<FORM METHOD=POST ACTION=%=action|enterPwd,=%>
%=OrderPick=%</P>
```



```
<INPUT TYPE=SUBMIT VALUE="Select Order"></P>
</FORM>
```

The “%=OrderPick=%” event macro is called which in turn calls the event handler within the RedWeb WebApp.

Within the RedWeb code, the following WebAppRedmanEventMacro procedure is found. This procedure is the standard name for the event macro handler within WebHub

```
procedure TfrmRedWeb.WebAppRedmanEventMacro(Sender: TWebOutputApp;
const aMacro, aParams, aID: String);
var
  temp : Tweblist;
  strTemp : String;
  i : integer;
  level1Flag, level2Flag : Boolean;
begin
  .
  .
  .
end;
```

The WebAppRedmanEventMacro contains the following conditional compound statement which is executed when the standard system variable “amacro” is set to the value “OrderPick” by the WebHub system. WebHub sets the system variable, of course, when a previously defined event macro is found within the body of an HTML chunk.

```
if comparetext (amacro, 'OrderPick')=0 then
begin
  temp:=Tweblist.create(self);
  with webappoutput do
  begin
    filldropdown(temp,stlOrderNames,'OrderName:', 'OrderSelected', '',ddmvalueasvalue);
    sendstringlist(temp,false);
  end;
  temp.free;
end
```

Hence, whenever the “OrderPick” event macro is called, a temporary object “temp” is instantiated which serves as the repository for the output from the “filldropdown” method of the Twebappoutput object. The filldropdown method, of course, populates in HTML syntax a drop down with the values contained within the string list variable stlOrderNames. This variable was populated at program initialization with the following code:

```
procedure TfrmRedWeb.FormShow(Sender: TObject);
var
  strTemp : String;
begin
  stlOrderNames := TStringlist.create;
  With tblOrders do
  begin
    Open;
```

```

while not EOF do
begin
    strTemp := FieldByName('ORDER_NAME').AsString + '=' +
        FieldByName('ORDER_ID').AsString;
    stlOrderNames.Add(strTemp);
    Next;
end;
end;
end;

```

This code opens the ORDERS.DB table and fills a dynamically created TStringList object with the values contained in the ORDER_NAME field of this table. Note the “=” sign and the right-hand ORDER_ID field are also sent to the StringList so that the proper HTML will be constructed by the FillDropDown method. Of course, this right-hand value will be used to open the appropriate TO-<n>.DB value to ensure the user has adequate permission to access the database table.

Per the help file accompanying WebHub (HRef, 1996), the filldropdown method has the following syntax:

```

procedure FillDropDown( target, source: TStringList; const prompt, varName, value : string;
    aDropDownMode : TWebDropDownModes );

```

It should be noted that the above is a good example of the paucity of adequate documentation on the emerging WebHub system. This FillDropDown method contains no accompanying explanatory text or sample code whatsoever. Programmers are required to experiment with the various parameters until they discover the effect they want.

H. REDWEB HTML PROCESSING SYSTEM FUNCTIONAL HIGHLIGHTS

The following discussion focuses on some of the more interesting aspects of the RedWeb HTML processing system.

1. Converting a Limited Level Recursive Task Organization to an Unordered HTML list

One of the more demanding modules in RedWeb was the procedure which creates an unordered HTML list from the task organization database table. A close examination of this code illustrates many of WebHub's features as well as some interesting coding techniques used in this powerful module. Unfortunately, there is no native WebHub component which supports automatic embedding of other unordered lists inside an unordered list to achieve the effect of indenting that is desired within RedWeb.

When the "WriteTaskOrg" event macro is called from the appropriate HTML chunk, the following code opens the appropriate table corresponding to the task organization for the order selected by the user. The table is then loaded into a TDBOutline component to achieve the hierarchical sequencing reflected in the table schema (see discussion of the TDBOutline component above). The ordered Unit name strings are then dumped into a temporary TMem object for subsequent processing.

```
if Comparetext (amacro, 'WriteTaskOrg')=0 then
begin
  With tblTO do
  begin
    TableName := 'TO' + WebAppRedman.literal['OrderSelected'] + '.DB';
    Open;
    {Load up TDBOutline StringList from TO data structure}
    dboTaskOrg.LoadFromDataSet;
    mmoTaskOrg.Clear;
    mmoTaskOrg.Lines.AddStrings(dboTaskOrg.Lines);
    Close;
  end;
end;
```

The code continues below with a long loop to process every text line in the temporary TMem object. The loop construct maintains two flags to handle up to three levels of indentation of the unordered list (in other words, three levels of embedding). The complexity of the code is due to cryptic tags required to begin and end unordered lists found within other unordered lists. The remainder of the code is well commented and should be straight forward to follow the logic in parsing the initial tabs in each text line (which correspond to the level of indentation/embedding required in HTML). For example, if there is no tab in the first character position of each text line, than this is a top-level embedded list item. If there is a single tab character, than it is a secondary level embedded list item. Finally, if there are a pair of tab characters, than it is a tertiary level embedded list item.

```
WebAppRedman.SendString('<UL>');
level1Flag := False; {Flag to indicate we are inside the first level of the list}
level2Flag := False; {Flag to indicate we are inside the second level of the list}

{Continue until we have processed every unit in the TO list; handles three
embedded levels
of lists--one of which is hard coded at entry; we can hardcode <CR><LF> combinations
after each SendString with '+ #13#10' if we want to make the HTML more legible,
however, this will increase packet size for transmission}
for i := 0 to mmoTaskOrg.Lines.Count - 1 do
begin
  {we are at the top most level, so simply write out the unit}
  if (Copy(mmoTaskOrg.Lines[i], 1, 1) <> #9) AND NOT level1Flag then
    WebAppRedman.SendString('<P><LI>' + mmoTaskOrg.Lines[i])

  {we are at the top most level, but need to close out an embedded list first}
  else if (Copy(mmoTaskOrg.Lines[i], 1, 1) <> #9) AND level1Flag then
    begin
```

```

        if level2Flag then
            WebAppRedman.SendString('</UL></UL><P><LI>' + mmoTaskOrg.Lines[i])
        else
            WebAppRedman.SendString('</UL><P><LI>' + mmoTaskOrg.Lines[i]);
            level1Flag := False;
            level2Flag := False;
        end

        {this is the first unit in an embedded list, so start the list}
        else if (Copy(mmoTaskOrg.Lines[i], 1, 1) = #9) AND NOT level1Flag then
            begin
                level1Flag := True;
                WebAppRedman.SendString('<UL><LI>' + mmoTaskOrg.Lines[i]);
            end

            {this is NOT the first unit in an embedded list, so check to see if we are at
            level 2}
            else if (Copy(mmoTaskOrg.Lines[i], 1, 1) = #9) AND level1Flag then
                begin
                    {this is the first unit in a sub-embedded list, so start the list}
                    if (Copy(mmoTaskOrg.Lines[i], 2, 1) = #9) AND NOT level2Flag then
                        begin
                            level2Flag := True;
                            WebAppRedman.SendString('<UL><LI>' + mmoTaskOrg.Lines[i]);
                        end

                        {this is another unit in the same list, so just write it out}
                    else if (Copy(mmoTaskOrg.Lines[i], 2, 1) = #9) AND level2Flag then
                        WebAppRedman.SendString('<LI>' + mmoTaskOrg.Lines[i])

                        {this is the first unit in ANOTHER sub-embedded list at the same level,
                        so end the previous list first}
                    else if (Copy(mmoTaskOrg.Lines[i], 2, 1) <> #9) AND level2Flag then
                        begin
                            level2Flag := False;
                            WebAppRedman.SendString('</UL><LI>' + mmoTaskOrg.Lines[i]);
                        end

                        {this is another unit at level one, so simply write it out}
                    else if (Copy(mmoTaskOrg.Lines[i], 2, 1) <> #9) AND NOT level2Flag then
                        WebAppRedman.SendString('<LI>' + mmoTaskOrg.Lines[i]);

                end;
            end;
end;

```

Finally, we write our terminal end of list tags which completes the module.

```

{Close out lists appropriately based on current level we stopped at}
WebAppRedman.SendString('</UL>');
if level1flag then
    WebAppRedman.SendString('</UL>');
if level2flag then
    WebAppRedman.SendString('</UL>');
end

```

I. SUMMARY

The prototype applications depicted in this chapter focus on the dissemination of a combat order from the battalion to the fireteam level. State of the art products such as Delphi, WebHub, and numerous commercial components are exploited to provide a rapid application development environment for both the REDMAN and RedWeb applications.

The REDMAN application is an easy-to-use product which closely emulates the real-world USMC methodology of communicating orders on the modern battlefield. The autogeneration capability of REDMAN allows the rapid reformatting, editing, and delivery of combat orders to subsequent levels of command. The RedWeb HTTP server provides a database-enabled web server to allow virtually all modern computing platforms access to the orders contained in a unit's database.

X. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

1. Introduction

This research has demonstrated that inexpensive palmtop computers can be successfully integrated into the current Marine Corps style of warfighting to effect the rapid dissemination of combat orders on the battlefield. The ubiquitous employment of palmtop computers on the battlefield will allow us to fully realize the promise of providing seamless Digitization of the Battlespace (DotB) below the battalion level without needlessly burdening our Marines with heavy, bulky equipment.

The \$15,000+ highly ruggedized, sub-notebook sized computers the Marine Corps and Army programs are adopting as their mobile computing platforms are simply not a replacement for \$400 palmtop computers. The true utility of a mobile computer is that it is compact enough to be carried everywhere with you. A computer can't possibly be useful if it isn't with you when you really need it, with adequate battery power remaining. As Joseph H. Schmoll writes, "a successful system acquisition program is one that places a capable and supportable. . .[tool]. . .in the hands of a user when and where it is needed, and does so within affordable resources." (Schmoll, 1993). Clearly, the Marine Corps needs to focus on the capabilities of today's technology in the form of inexpensive palmtop computers. After all, we can reap immediate rewards from such a system—and it needn't be all that waterproof, or even that rugged! During those relatively rare moments in the field where we need a ruggedized solution, there are ample "harsh environment" cases available that provide shock and water-resistant storage for our palmtops.

Mobile computers have utility in a variety of domains. Inexpensive palmtop computers can be used for a wide array of purposes to include those represented in Table 10.1.

Combat/Training	Garrison/Administrative
Friendly Weapons Ranges/Tech Details	Training Management
Combat Reports	Morning Reports
Operation/Frag Orders	Legal / NJP
FMs/FMFMs/FMFRPs/etc. Publications	CMC Reading List
Training Lesson Plans	Training Lesson Plans
Tactical Checklists	LAN/Internet Messaging
MOS Specific Material	MOS Specific Material
Range Regulations	Fitness Reports
Land Navigation	To-Do Lists
Tactical Decision Games	Appointments
Tactical/Logistics Report Formats	Unit Rosters/Manifests
Communication Procedures/Brevity Lists	Armory/Supply Inventories
Ammo Handling Procedures/Safety	Leadership Materials
Threat Weapons Ranges/Tech Details	Training Reports

Table 10.1. Uses of Mobile Computers in Combat and Garrison.

2. Where to Go from Here

The success of the REDMAN suite of software developed in this thesis demonstrates that the DACT program must primarily be considered a software system capable of interoperating with all types of digital communication devices both on the battlefield and in garrison. We need to continue to maximize our use of COTS software to keep expenses down and development time short. However, some of the software we need just isn't available off the shelf. The Marine Corps data processing community must focus on providing no-cost, quality, MOS-specific software to everyone who wants it (along with extending the great COTS PIM software that is already built into most commercial palmtops). We need to eliminate our traditional reliance on contracted industry software developers as much as possible—Marines should be the primary builders of Marine software because only Marines understand what the software needs to do. This is possible using today's software development environment. The Marine Corps Tactical Systems Support Activity (MCTSSA) has an ongoing development initiative called the Accelerated Software Acquisition Program (ASAP) that ought to be a model for other agencies. By building on top of existing platforms with modern Rapid Application Development (RAD) tools, it is possible to build quality software for the use of the entire

Corps with a minimal investment in manpower and funding. A well-motivated team of competent Marine programmers can build an entire suite of valuable Marine-specific software for the ground combat and support communities using modern prototyping techniques. Software bug fixes and enhancements need to be continuously developed, and new versions released via electronic means over the Internet. In just a few years, the Marine Corps could realize a significant increase in its combat proficiency—across the entire Corps—all with minimum investment.

3. Summary

The “digital computer revolution” has often been compared to the “Big Bang” theory of the creation of the Universe. The pundits tell us that a few micro-seconds after the Big Bang, there was a lot of smoke and uncertainty in the world. The military (along with just about everybody else out there) seems to be lost in this fog of uncertainty as to where all this digital technology is taking us. What will the Command and Control infrastructure look like in the year 2025? No one can be certain. However, if the Marine Corps is to fulfill its legacy of becoming a “certain force in an uncertain world,” (C4I, USMC, 1995) we must dedicate ourselves towards fully exploiting the power of today’s mobile computers by integrating them into the Marine Corps style of warfare. At the same time, we need always be mindful of the military maxim, “We gotta walk before we can run.” Since the purpose of the Marine Corps is to get the rifleman on top of that muddy hill, we must reorient the direction of mobile computing towards helping the rifleman accomplish that very mission. Palmtop computers will help get us on top of those muddy hills of today—as well as the hills of tomorrow—if only we’ve mentally prepared ourselves to use them.

B. RECOMMENDATIONS FOR FUTURE WORK

Much work remains to bring the promise of mobile computing to the battlefield. We need to begin by aggregating real-world statistics and conducting a thorough analysis regarding the timing and accuracy of disseminating combat orders under battle conditions. Various paging, cellular and SATCOM communication channels need to be exploited and evaluated in wide-ranging geographic areas. This evaluation will include the construction of temporary deployable mobile paging/cellular infrastructures for isolated geographical regions. This field testing is essential to convince commanders of the utility of the REDMAN system and commercial palmtops on the battlefield. This analysis should contrast the existing manual methodology with the various technologies investigated

earlier and the prototype applications developed in this study. These statistics will assist commanders and program managers in determining the viability of the integration of palmtop computers on the battlefield.

Cooperative Research and Development Agreements (CRADAs) with such companies as Microsoft, Hewlett-Packard, Reticular Systems, Torrey Science, Booz-Allen Hamilton, Trimble Navigation, etc. should continue to be explored to provide commercial research partners for further development and support of this work. The creation of a concept video will also be useful in educating both industry and Marine leadership regarding the benefits of mobile computers and the REDMAN system.

Specific recommendations for future work follow.

1. Software Enhancements

The software applications developed in this research should be enhanced in the following areas:

a. REDMAN Order Processing Software

- (1) The REDMAN order processing software should be ported to the Pegasus machine and other suitable platforms once the Pegasus OS and companion hardware are available commercially.
- (2) Add warning order and fragmentary order support.
- (3) Printing of orders and integral HTML output file generation should be supported.
- (4) Refine Element and Checklist searching so that precise mission can be filtered as well as mission area. Both order elements and checklist items will be greatly expanded.
- (5) Implement auto-inclusion of elements by mission area (with check box in program options).
- (6) Implement full defaults module (with path names, etc.).

(7) Build templates for creating orders from scratch in addition to autogenerating them from a higher commander.

(8) Implement a master USMC T/O database to autogenerate T/Os for any unit at any level.

(9) Implement encryption and data security with internal software based encryption or external Fortezza PC Card support.

(10) Further minimize compressed message size through tokenizing and encoding to reduce transmission time.

(11) Add redundant FEC coding schemes to maximize reliability for various wireless channels.

(12) Implement backwards translation of RedWeb HTML packets back into REDMAN Paradox format for subsequent order processing at lower levels of command.

b. RedWeb HTML Processing Software

(1) Increase security to include encryption of order packets integrated with HTML browsers capable of decrypting them.

(2) Enhance memo field unordered list generation to handle an arbitrarily deep nesting of lists.

(3) Integrate the download of REDMAN packets via FTP as well as present RedWeb HTML messages.

(4) Enhance hot-linking to internal anchors to improve navigability of the HTML order pages.

(5) Implement logging functionality to ensure a chronological record is maintained of who downloaded which orders when. Integrate this

logging with the transmission log functionality in the REDMAN order processing software.

(6) Integrate electronic mail connectivity throughout the REDMAN architecture.

c. Palmtop Grid Calculation Software

(1) Implement Time/Space calculations by allowing users to forecast the location of moving objects on the battlefield

(2) Implement Warsaw Pact to NATO grid conversion

2. Additional Research Areas

The following additional research areas need exploration:

a. Interface the DACT with the Army's Handheld Terminal Unit (HTU) and DSSU, and commercial notebook sized computers.

b. Interface the DACT with other USMC systems including the Position, Location and Reporting System (PLRS), Initial Fire Support Automated System (IFSAS), Automatic Target Handoff System (ATHS), Target Location, Designation and Handoff System (TLDHS), the Mortar Ballistic Computer, and the Backup Computer System (BUCS).

c. Develop user interfaces and applications for other major DACT functional areas (e.g., fire support and situational awareness positioning systems).

d. Interface the DACT with the emerging SPEAKEasy software-based radio.

The SPEAKEasy vision is to develop and field a multi-mode, multi-band radio based on a programmable, modular architecture. Projects such as the SpectrumWare project apply a software oriented approach to wireless

communication and distributed signal processing. Advances in processor and analog-to-digital conversion technology have made it possible to implement virtual radios that directly sample wide bands of the RF spectrum, and process these samples in application software (Tennenhouse, 1996).

- e. Develop a version of REDMAN (or filters to translate REDMAN order packets to Joint VMF format) running on the TCO to integrate with GCCS.*
- f. Develop additional DACT functionality interoperating with palmtop-sized mobile computers (e.g., graphic-based navigational aids and situational awareness tools).*
- g. Develop both diffuse and narrow beam Infrared transmission support paying particular attention to the investigation of AT&T's promising ATM over IR "Rednet" protocol*
- h. Develop power management strategies to include providing standard ways for Marines to recharge batteries on every transportation prime mover. Solar energy should continue to be exploited.*
- i. Build filtering algorithms for communication of battlefield packets.*
- j. Incorporate cognitive decision aiding tools to recommend sound courses of action in chaotic environments.*

The cognitive decision system will recommend courses of action—it does not decide for the user. Continue ongoing work with Reticular Systems company to develop an OpenPDA Toolbox to perceive the environment, query a rule-based expert system, and provide recommended courses of action in support of the combat order decision-making process. Reticular Systems is a company involved with the development of cognitive decision aiding products. Intelligent agents can perform such tasks as choosing the correct transmission path based upon environment, as well as sorting pick-

lists and recommendations based on database of past decision making made by commander.

k. Continue to integrate NPS student and faculty research into USMC mobile computing initiatives.

NPS should continue to be a focal point for the development of mobile wireless palmtop battlefield computing applications. This will ensure that the USMC can maximize its technical capabilities in an exponentially improving computer industry.

APPENDIX A. MESSAGES PLANNED FOR IMPLEMENTATION IN THE DACT

A. OPERATIONAL, INTELLIGENCE AND LOGISTICS MESSAGES

The following operational, intelligence and logistics messages are planned for implementation in the latest DACT Concept of Employment (MCCDC, 1996):

1. Spot/Salute Report (Patrol Report info)

2. Op Plan/Frag/Warning/Orders

3. Op/Intel Overlay

4. Situation Report

5. Position Report

6. Free Text message

7. MEDEVAC request

8. Rapid Request/Logistics Support

9. Personnel Strength Report

10. Threat Warning Message

11. Casualty Report

12. TRAP alert

13. Shell Report

14. NBC reports (1-6)

15. MIJI report

16. Logistics Summary

17. Beach report

18. Landing Zone Brief
19. Obstacle report
20. Minefield report
21. Strike Warning

B. FIRE SUPPORT MESSAGES

The following fire support messages are planned for implementation in the latest DACT Concept of Employment (MCCDC, 1996):

1. Artillery/Mortar:

- a. Message to Observer*
- b. FO Notification (shot/splash)*
- c. Fire Cap*
- d. Fire Plans*
- e. Fire Support Coordination Measures (FSCMs)*
- f. Call for Fire (includes Fire Command)*
- g. Adjustment*
- h. Check Fire*
- i. End of Mission and Surveillance*
- j. Suppression of Enemy Air Defenses (SEAD) and Mark*
- k. 81mm and 60mm Call for Fire*

2. Close Air Support Messages:

- a. Tactical Air Request*
- b. 9-line Brief (Used for updates also)*

- c. On Station Report*
- d. Receipt/Compliance*
- e. Depart Initial Point (IP)*
- f. Cleared Hot (voice/data)*
- g. Abort (voice/data)*
- h. Adjustment from Previous Hit/Mark (voice/data)*
- i. Battle Damage Assessment*
- j. LAAD Engagement Report*
- k. Inflight Report*
- l. Hostile Aircraft Sighting Report*
- m. Assault Support Request*

APPENDIX B. EXECUTIVE OVERVIEW OF THE DACT

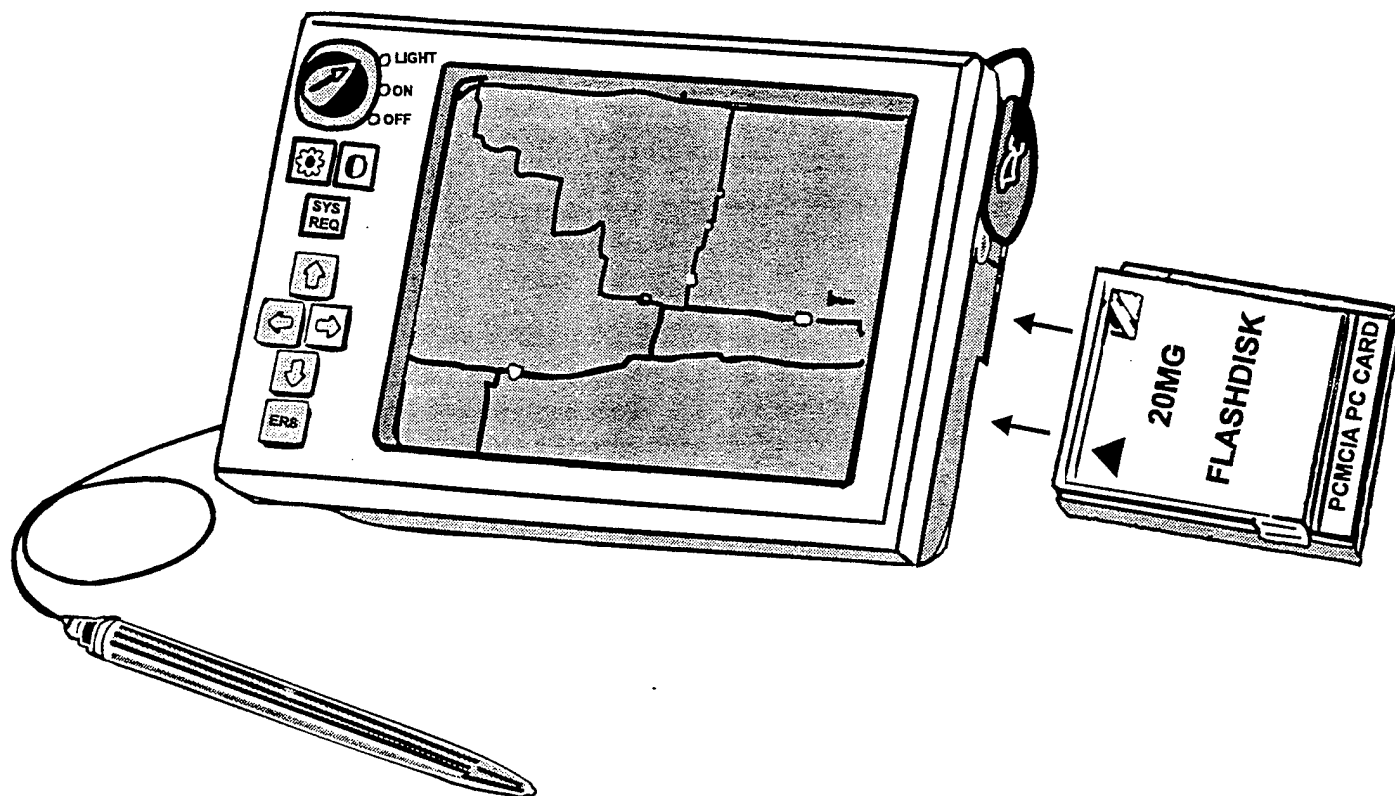
A. OVERVIEW

The attached executive overview provides an operational concept, operational description, and technical characteristics of the emerging DACT. (MCTSSA, 1995)



Marine Corps Tactical Systems Support Activity Camp Pendleton, CA

Digital Automated Communications Terminal (DACT)



OPERATIONAL CONCEPT

The DACT will provide the Marine Corps with a digital communications capability for general purpose communications and a means to interface with tactical data systems. The DACT will use a digital message system to send and receive messages in short digital bursts. This hand-held device will provide the user with a versatile, multifunction capability. In addition to communications, the DACT will function as a Marine Corps Tactical Command and Control Class 4 computing device, determine precise geographic position via Global Positioning Satellite (GPS) calculate target position, and provide the operator the capability to create, display, and manipulate graphic images.

DACT software applications have been developed for field operational assessment which enable the Marine

to input to and receive tactical information from the Global Command and Control System (GCCS) as well as those Marine Corps legacy systems which have been re-engineered to be GCCS Common Operating Environment (COE) compliant. This interoperability is seen as critical in ensuring full integration with the C4I for the Warrior concept.

OPERATIONAL DESCRIPTION

The DACT is a commercial-off-the-shelf (COTS) system. It interfaces to a tactical communications modem and GPS. It also supports mapping and overlay capabilities.

Functionality being demonstrated during JWID-95 is primarily related to the exchange of position, overlay and operational note formats between the DACT and the GCCS COE compliant Marine Air Ground Task Force (MAGTF) C4I Baseline. Specifically, (1) the

GPS feed integral to the DACT is being used to create an OTH-Gold Position Report which is forwarded from a small Marine unit back to the MARFOR Command Element for inclusion in the Common Operational Picture (COP); (2) alternatively, the COP is being transmitted from GCCS down to the DACT; graphic overlays generated on the DACT can be exchanged with the MAGTF C4I Baseline/GCCS as well as freeform narrative text in the form of operational notes.

COMMON OPERATIONAL PICTURE

The DACT will use its GPS generated location information to generate a GCCS ground track for correlation into the Joint Task Force's (JTF) COP. Additionally, both friendly and hostile units within the same geographic region will be disseminated from the GCCS COE to the DACT over single channel radio.

AIR MISSION PLANNING

The overlay/operational note capabilities of the DACT will be used to submit Air Support Requests (ASR); supporting our Forward Air Controller (FAC) function. This request will be processed through the Air Combat Element (ACE) for inclusion in the Air Tasking Order (ATO). Additionally, the DACT will be used to provide Battle Damage Assessment (BDA) in support of validating air mission accomplishment.

COURSE OF ACTION DEVELOPMENT

The DACT will be used to create/transmit enemy positional information as a GCCS overlay for inclusion into the Course Of Action by the MARFOR Current Operations Cell. This hostile track location will be subsequently disseminated to all JTF elements via GCCS.

TECHNICAL CHARACTERISTICS

HARDWARE:

- COTS, 5.5 lbs hand held communications terminal
- 486, 32 bit, 25 Mhz
- Memory
 - DRAM: 4 MB expandable to 16 MB
 - Flash Disk: 2 MB to 32 MB expandable in 4 MB increments. Further expansion via PCMCIA slots
- Display: LCD, high contrast, graphic VGA, 640 X 480 pixels, 7" diagonal, 16 gray levels, backlit illumination, brightness and illumination control
- PCMCIA: 2 type II or 1 type III
- Pointing Device: Pen
- Communications Module: A DSP based tactical communications module supports one tactical channel that can be connected to a radio set or to 2/4 wire telephone set

SOFTWARE:

- Operating System: MS-DOS/Pen Operating System, Windows, Windows for Pen Computing
- Languages: All IBM-PC standard languages
- Communication: Embedded communication package

ELECTRICAL:

- Internal battery> Lithium, BA-5800/U
- Vehicle Power: Via adapter, 24 VDC from vehicle source, MIL-STD-1275
- Consumption: 2.2W using automatic power conservation procedures

INTERFACES:

- Pen Device: with "soft" keyboard available on screen
- Serial Interface: MS-DOS COM 1 & 2 ports, RS-232, asynchronous at 110 to 115,200 bps
- Radio Communication Interface Analog modems:
 - FSK: 75 - 1200 bps
 - PSK: 1200 bps
 - DPSK: 2400 bps
 - Digital Interface: 75 - 16,000 bps
- Field Telephone Interface: FSK or CDP data transfer via field telephone lines

RELIABILITY & MAINTAINABILITY:

- MTBF: > 8000 hours
- MTTR: < 20 minutes for faulty module location and replacement

Point of Contact: Communication Systems Division
 Ms Kathy Houshmand (DSN) 365-2615
 Commercial (619) 725-2615
 Internet address:
 kathy.houshmand@mctssa-gw.usmc.mil

APPENDIX C. SAMPLE USMC INFANTRY BATTALION COMBAT ORDER

A. SAMPLE ORDER TEXT

The following text represents an actual USMC infantry battalion combat order used throughout this thesis as a realistic example of the types of combat messages required to be communicated on the battlefield. This order is unclassified and used for training purposes.

Copy no. ___ of ___ copies
2nd Bn, 7th Marines
MCAGCC, 29 PALMS
141800Z OCT 91
MTR-5

Frag Order 5-92 (OPERATION DEEP BANDINI)

Ref: a. Twentynine Palms East/West, Edition 3-DMA, Series
V7953 1:50,000

Time Zone: T

Task Organization: Annex A (Task Organization)

1. Situation

a. General. (Omitted)

b. Enemy Forces. Wahabian forces in the vicinity of Los Angeles have temporarily halted the advance of allied forces though they have suffered up to 35% casualties. The allied advance will not continue for at least 24 more hours while allied forces wait for their supply lines to catch up with their forward units. The 111th Brigade has taken advantage of the lull in fighting to the West to attempt a push south from Sunshine Peak to Maumee Mine and Emerson Lake in an attempt to disrupt allied east-west supply lines. Though the 111th attack was unsuccessful due largely to allied air superiority, and the enemy has withdrawn the majority of its forces, the 111th Bde has left behind up to one Mech battalion in Maumee Mine and Emerson Lake. These remain-behind forces are likely to establish platoon and company sized defensive positions, with minimal engineer support in an effort to delay any allied counterattack.

c. Friendly Forces

(1) Higher. At 0900 on D-Day, 7th MEB conducts a movement to contact to destroy enemy forces within the GCE's zone of action in order to assume a forward defensive posture and deny enemy forces into the GCE's TAOR. RCT-7 Intent: Argos Pass is the key to the overall effort in order to ensure the northern movement of GCE forces, and deny enemy access to the high speed avenue of approach of Quackenbush Lake. In order to isolate the battle area, I want to prevent enemy forces from escaping and divulging friendly disposition of forces. Overall, I am buying time and distance against the potential for a subsequent enemy attack. Our efforts will provide an eyes forward advantage and prevent enemy indirect firing platforms from reaching the port of 29 Palms.

(2) Adjacent. 1/7 attacks to our east in Quackenbush corridor. 3d Tank Bn (-) follows in trace of 1/7 as RCT reserve.

(3) Artillery.

(4) Air.

2. Mission. 2/7 conducts a movement to contact along Axis Jane IOT clear enemy forces in zone within the Emerson Corridor, control Gay's and Maumee Pass, deceive the enemy as to the location of the RCT main effort, and guard the MEB's left flank.

3. Execution

a. Commander's Intent and Concept of Operations.

(1) Commander's Intent. We are a supporting attack for RCT-7's deliberate attack. It is believed that the enemy is employing a battalion (-) sized mechanized force in the area which we will be conducting our battalion movement. He is expected to delay initially, and then transition into a deliberate defense. Unfortunately, the enemy situation is somewhat vague. The enemy's critical vulnerability, however, is that he has failed to provide adequate logistical sustainability for his forward deployed forces. RCT-7's pre-planned deep and our own close air sorties will significantly deteriorate his will to fight. I want to maximize our reconnaissance and security efforts forward to ensure we commit our main force under the most favorable circumstances possible. We will remain in tactical column as long as possible to enhance the speed and control for which we can deploy for combat. Unit commanders must be prepared to act boldly to seize the initiative, keep the enemy off balance, and to exploit opportunities as they arise on the battlefield. Our first priority is to protect the column against surprise. Our second is to be prepared to bring our superior combat power to bear against the enemy if he should challenge us. Hence, we will employ our air defense in bounds to ensure the column is protected at critical passage points. Engineers will be employed well forward to facilitate our movement by ensuring they are prepared to rapidly breach any obstacles we may encounter. Armor and anti-armor assets will be distributed throughout the column to provide all-around protection, and facilitate our integrated entry into combat. I intend to rapidly suppress and eliminate any forward enemy forces, develop the situation with my lead maneuver units, and deploy my main body to destroy enemy resistance IOT control Gays and Maumee Pass.

(2) Concept of Operations. 2/7 conducts a single-axis movement to contact in tactical column with one CAAT Team and one Team Mech forward, and one Team Tank and one mechanized company back.

c. Tasks

(1) CAAT

- Advance guard of tactical column. You are March Serial #1. March Serial Commander. Provide forward and flank security during movement.

- BPT clear hastily emplaced obstacles with attached OCD.

- BPT transition into screening element of approach march formation. Screen forward of column 2 km to provide forward security for column.

- BPT suppress enemy armor and mechanized units and allow Team Mech to maneuver to destroy enemy forward units.

- At RP, screen forward to PL BARREL. BPT continue the attack east.

(2) Team Mech (Co F)

- FIT of CAAT as lead march unit of March Serial #2.
March Serial Commander.

- BPT transition into advance guard element of approach march formation. Provide forward security during movement.

- BPT maneuver to destroy enemy forward units supported by CAAT.

- At RP, occupy BP 25 in hasty defensive positions.
BPT continue the attack east towards MEB Obj A.

(3) 2d Plt (-) (Rein), Co C, 1st CEB

- FIT of the Alpha Command Group as the third march unit of March Serial #2.

- BPT to displace additional OCDs forward to breach deliberate obstacle belts.

- At RP, occupy an AA at GC-_____ in GS of the battalion. BPT construct hasty obstacles vicinity battalion defensive area. BPT continue the attack east towards MEB Obj A.

(4) 81s Plt (Rein)

- FIT of Engineers as fourth march unit of March Serial #2.

- BPT establish hasty firing positions ISO march column. POF to CAAT initially.

- At RP, establish firing positions at FP 55 in GS of the battalion. POF to Tm Mech. BPT continue the attack east.

(5) Team Tank

- FIT 81s Plt as fifth march unit of March Serial #2.

- BPT transition into lead element of the main body in an approach march formation. Provide forward security of the main body during movement.

- BPT deploy for combat and maneuver rapidly to exploit the success of Team Mech forward.

- At RP, occupy BP 35 in hasty defensive positions.
BPT continue the attack east towards MEB Obj A.

(6) STA/Recon

- Establish surveillance positions:

- STA 1- GC _____

- STA 2- GC _____

- Recon 1- GC _____

- Recon 2- GC _____

- BPT to control supporting arms to engage enemy forces when sighted.

(7) Det, 1st Sect, 3d LAAD Bn (DS)

- Establish firing positions along Axis Jane in support of battalion scheme of maneuver.

- Displace forward in bounds to ensure continuous umbrella of anti-air support for the march column.

d. Bn Reserve.

Co G (Rein)

- FIT Bravo Command Group as second march unit of March Serial #3. March Serial Commander. Provide rear security march column.

- BPT transition into lead element of the rear guard in an approach march formation.

- BPT deploy for combat and maneuver rapidly to assume the mission of either maneuver unit forward.

- At RP, occupy BP 55 in hasty defensive positions. BPT continue the attack east.

e. Coordinating Instructions

(1) MOPP level 0 initially.

(2) Order of Movement in Battalion Zone: March Serial #1: CAAT. March Serial #2: Team Mech, Alpha Command, Engineers, 81 Plt, Team Tank March Serial #3: Bravo Command, Co G (Rein).

(3) Units start engines for movement from assembly areas to SP at _____.

(4) LAAD priority of protection

- Tm 1 to Team Mech
- Tm 2 to Team Tank

(5) LAAD Weapons Control- Yellow (Weapons Tight)

(6) SP, RP, Route, PLs, March Objectives, and BPs per Operations Overlay.

(7) Designate Air Sentries for movement.

(8) All units conduct immediate action drills for ambush situations; disabled vehicle troop and cargo transfer; dismounted close security; and establishment of outpost security during halts.

(9) Dispersion between march units: Tactical Column: 250m. Approach March: 500m. Dispersion between march serials: Tactical Column: 500m. Approach March: 1000m.

4. Administration and Logistics. No change

5. Command and Signal

a. Signal

- Pass SNOWSTORM on Arty COF and Bn TAC 1 upon receiving indirect fires.

- Codewords

-- STORMY WEATHER: Transition to Approach March Formation

-- CLEAR SKIES: Resume Tactical Column Formation

b. Command

- Alpha command initially second march unit in March Serial #2.

- Bravo Command initially lead march unit in March Serial #3.

ACKNOWLEDGE RECEIPT

R. J. MAUER
Lieutenant Colonel, U.S. Marine Corps
Commanding

OFFICIAL:

D. A. NICHOLAS
Major, USMC
S-3

Copy no. ___ of ___ copies
2d Bn, 7th Mar (Rein)
TWENTYNINE PALMS, CA
201200T OCT 91
UYD-5

ANNEX A (Task Organization) to FRAGO 5-92

Ref: (a) Maps: California Series V795S, Sheets Twentynine
Palms East and West, 2-DMA, 1:50,000

(All attachments effective 220700T OCT 91)

2d Bn (-) (Rein), 7th Mar

H&S Co (-)
Wpns Co (-)
Det, Comm Plt, Hq Co, RCT-7
Arty Ln Tm, G Btry, 3d Bn, 11th Mar
Det, Co D, 3d AAV Bn
Tm, Det 9th Comm Bn

CAAT

HMG Plt
Det, H&S Co
Det, Comm
Det, Med
FAC Party #1
Det Wpns Co
81 FO Tm # 1
Arty FO Tm # 1
1st Sqd, 2d Sec, TOW Plt, HQ Co, RCT-7
OCD #1

Tm Mech

Co F (-)
Det, H&S Co
Det, Comm
Det, Med
Det Wpns Co
81 FO Tm # 2
1st/2d Sec, AA Plt
Arty FO Tm # 2
1st Plt, Co D, 3d Tk Bn
1st Plt (-), Co C, 3d AAV Bn
2d Sqd, 2d Sec, TOW Plt, HQ Co, RCT-7

Tm Tank

Co D (-), 3d Tk Bn
1st Plt, Co F, 2/7
Det, H&S Co
Det, Comm
Det, Med
FAC Party #2
Det Wpns Co
81 FO Tm # 3
2nd Sec, AA Plt
Arty FO Tm # 3
OCD #2
1st Sec, 1st Plt, Co C, 3d AAV Bn

81 mm Mortar Plt (Rein)

Det, H&S Co
Det, Med
1st Sec, 3d Plt, Co C, 3d AAV Bn

2d Plt (-), Co C, 1st CEB
Det, Engr Spt Co, 1st CEB

Det, 1st Sect, 3d LAAD Bn

Bn Reserve: Co E (Rein)

Det, H&S Co
Det, Comm
Det, Med
Det, Wpns Co
81 FO Tm #3
3d Sec, AA Plt
Arty FO #3
3d Plt (-), Co C, 3d AAV Bn

ACKNOWLEDGE RECEIPT

R. J. MAUER
Lieutenant Colonel, U.S. Marine Corps
Commanding

OFFICIAL:

D. A. NICHOLAS
Major, USMC
S-3

APPENDIX D. SOURCE CODE FOR THE GRIDCALC APPLICATION

A. GRIDCALC APPLICATION SOURCE CODE

The following code represents the C language source code for the HP200LX version of the GRIDCALC application developed in conjunction with the PAL development library.

1. GRIDCALC.H Header File

This is the header file for the main program:

```
#ifndef __GRIDCALC_H
#define __GRIDCALC_H

void gridToGrid ();
void rangeAzimuth ();
int badGrid(char *grid);
void defaultSet();
int badDefaults();
void intersection();
void resection();

struct Defaults {
    char compass[5];
    char yardstick[3];
    int gmAngle;
};

#endif
```

2. GRIDCALC.C Source File

This is the source code file for the main program:

```
#include "hpclib.h"      /* HPCLIB header file */
#include "gridmath.h"
#include "gridcalc.h"
#include "pal.h"
#include "palsmsvc.h"

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Defaults defaults;

const char DEF_FILE[] = "gridcalc.def";

TOPMESSAGE = " GRIDCALC - Grid Coordinate Calculator V2.0 by
              Maj J. C. Cummiskey USMC (12/94)";

MENUBARMSG = "&Grid   &Time/Space   &Convert   &Defaults   &Quit   &Help ";

MENUITEMS = {

/* menu1 */ {"&Grid to Grid","&Range/Azimuth","&Intersection",
```

```

                                "&Resection"},
/* menu2 */ {"&Intercept","Start Time"},
/* menu3 */ {"Lat/Long","Warsaw Pact"},
/* menu4 */ {""},
/* menu5 */ {""},
/* menu6 */ {"&Help   F1","&About"} };

KEYLABELS={ /* You can define and display multiple key labels */

" Help  Grid  Rg/Az  Intsc  "
" Resec  Inter  Start  LatLg  "
" Quit   Deflt  ";

char keylab2[]={ /* Here's another bottom line key label definition */

"
"
" Cancel  OK
";

/*****
* DEMO WINDOW *
*****/

void stub(char *message)
{
    open_win(1, HP2, 150, 50, 475, 150, "Feature Not Implemented");
    wrtext(260, 90, message);
    wrtext(190, 120, "Press any key to continue");
    getkscan();
    close_win(1);
}

/*****
* HELP SCREEN *
*****/

void help(void)
{
    open_win(1, HLP, 40, 20, 600, 180, "On-Line Help.");
    setfon(MEDIUM_FONT);
    wrtext(200, 80, "This is the Help window.");

    botlin(keylab2); /* display other set of key labels on bottom */
    getkscan();
    botlin(botmes); /* display default KEYLABELS */
    close_win(1);
}

/*****
* ABOUT WINDOW *
*****/

void about(void)
{
    open_win(1, HP1, 175, 85, 465, 140, "A Quality CS4203 Product");
    wrtext(220, 108, "GRIDCALC - Ver 2.00b");
    setfon(SMALL_FONT);
    wrtext(215, 125, "<press any key to continue>");
    getkscan();
    close_win(1);
    setfon(MEDIUM_FONT);
}

/*****

```

```

* INITIALISATION *
*****/

void initialize(void)

{
    char temp[4];
    int i,j;
    IMGHDR *pImg;
    char *hymn = "T200 k1 o3 v3 L16 ce L8 gggg L6 g o4 L16"
                " c o3 L8 g L16 ef L8 gg L16 f L5 d L3 c";

    // Open Defaults Initialization File
    FILE *defaultFile;

    is100();          /* Is it an HP100LX? */
    setgra();         /* Set graphics mode */
    announ(AN_LEFT);  /* place annunciators left */

    if ((defaultFile = fopen(DEF_FILE, "r")) == NULL) {
        strcpy(defaults.compass, "Dgs");
        strcpy(defaults.yardstick, "KM");
        defaults.gmAngle = 0;
    }
    else {
        fgets(defaults.compass, 5, defaultFile);
        fgets(defaults.yardstick, 3, defaultFile);
        fgets(temp, 5, defaultFile);
        fclose(defaultFile);
        defaults.gmAngle = atoi(temp);
    }

    toplin(topmes);   /* Create top line */
    botlin(botmes);   /* Create bottom (key-label) line */

    setfon(LARGE_FONT);
    wrtext(195, 18, "G R I D C A L C");
    rectan(195,30,435,32, SOLID_FILL);
    wrtext(110, 40, "Grid Coordinate Calculator");

    for (i = 65; i < 175; i += 10)
        line(0, i, 639, i); /* draw a horizaontal line */
    for (i = 10; i < 640; i += 10)
        line(i, 65, i, 165); /* draw a vertical line */

    for (i = 0; i < 640; i +=10) {
        rectan(i, 66, i+9, 74, SOLID_FILL);
        for (j=0; j < 2000; ++j);
    }

    for (i = 75; i < 155; i +=10) {
        rectan(631, i, 639, i+9, SOLID_FILL);
        for (j=0; j < 2000; ++j);
    }

    for (i = 630; i > 0; i -=10) {
        rectan(i, 156, i+9, 164, SOLID_FILL);
        for (j=0; j < 2000; ++j);
    }

    for (i = 155; i > 65; i -=10) {
        rectan(1, i, 9, i+9, SOLID_FILL);
        for (j=0; j < 2000; ++j);
    }

    pImg = LoadPcx("world.pcx", 1);
    if(!pImg) FatalExit("WORLD.PCX file not found", 1);
    PutImg(10, 76, FORCE_RULE, pImg);

    m_play(hymn);
}

```

```

        for (i = 65; i < 175; i += 10)
            line(0, i, 639, i); /* draw a horizontal line */
        for (i = 10; i < 640; i += 10)
            line(i, 65, i, 165); /* draw a vertical line */
        line(631,155,639,155); // patch for looping cleanup

        about();
        wrtext(40, 175, "Press <alt> or <menu> key for Main Menu / <esc> to exit");
    }

    /*****
    * EXECUTE FUNCTIONS *
    *****/

    void exec_function(int function)
    {
        switch(function) {

            /* Menu 1, option 1 */
            case 0x11:
                gridToGrid();
                break;

            /* Menu 1, option 2 */
            case 0x12:
                rangeAzimuth();
                break;

            /* Menu 1, option 3 */
            case 0x13:
                intersection();
                break;

            /* Menu 1, option 4 */
            case 0x14:
                resection();
                break;

            /* Menu 2, option 1 */
            case 0x21:
                stub("Intercept");

/*
                setmda();
                puts("This is a test-are we in text mode?");
                getkscan();
                setgra();
*/

                break;

            case 0x22: stub("Start Time"); /* Menu 2, option 2 */
                                break;

            case 0x31: stub("Lat/Long"); /* Menu 3, option 1 */
                                break;

            case 0x32: stub("Warsaw Pact"); /* Menu 3, option 2 */
                                break;

            /* Menu 3, option 2 */
            case 0x41:
                defaultSet();
                break;

            case 0x51: terminate(0); /* Quit menu */
                                break;

            case 0x61: help(); /* Menu 5, help */
                                break;

            case 0x62: about(); /* Menu 5, about */
                                break;

```



```

/* Functions outside menus */

    case 0x13b: help();          /* 3b=scancode of F1 */
                                break;

    case 0x13c: gridToGrid();    /* 3c=scancode of F2 */
                                break;

    case 0x13d: rangeAzimuth();  /* 3d=scancode of F3 */
                                break;

    case 0x13e: intersection(); /* 3e=scancode of F4 */
                                break;

    case 0x13f: resection();     /* 3f=scancode of F5 */
                                break;

    case 0x140: stub("Intercept"); /* 40=scancode of F6 */
                                break;

    case 0x141: stub("Start Time"); /* 41=scancode of F7 */
                                break;

    case 0x142: stub("Lat/Long"); /* 42=scancode of F8 */
                                break;

    case 0x143: terminate(0);    /* 43=scancode of F9 */
                                break;

    case 0x144: defaultSet();    /* 44=scancode of F10 */
                                break;

}

}

/*****
 * MAIN PROGRAM *
 *****/

void main(void)
{
    int func;

    initialize();

    do {
        func = get_function(6);
        exec_function(func);
    } while(1);
}

void
gridToGrid()
{
    int istat, badData;
    float distance;
    int azimuth;
    char temp[10];
    char saveGrid1[9] = "";
    char saveGrid2[9] = "";

    do {
        open_win(1, HP2, 30, 30, 600, 170, "Grid to Grid Calculation");
        wrtext(140, 150, "Press TAB to move from field to field");
        botlin(keylab2); /* display F9/F10 inside key labels on bottom */

        entry_form(1, NUMERIC, HP2, 60, 60, 8, "Enter GRID coordinate #1:");
        entry_form(2, NUMERIC, HP2, 60, 80, 8, "Enter GRID coordinate #2:");
        entry_dflt(1, saveGrid1);

```

```

entry_dflt(2,saveGrid2);

/* Now we process the input screen */
badData = 0;
istat=entry_proc(2, 450, 70, 450, 110); /* process 4 data entry fields */
strcpy(saveGrid1,get_entry(1));
strcpy(saveGrid2,get_entry(2));

if (istat && (badGrid(get_entry(1)) || badGrid(get_entry(2)))) {
    errwin("Invalid Grid(s) ");
    badData = 1;
    getkscan();
}

close_win(1);      /* close window */
botlin(botmes);    /* display default KEYLABELS */

/* Now we check if the user accepted the input (F10 or F9) */
/* input status OK=1 CANCEL=0 */
if (istat==1 && !badData) {
    /* Here we display the input data */
    open_win(2, HP2, 30, 30, 600, 170,
    "Grid to Grid Calculation Results");

    distazim(get_entry(1), get_entry(2), &distance, &azimuth);

    // If we're not in KM mode
    if (strcmp(defaults.yardstick,"KM"))
        distance = convertKMtoNM(distance);

    // If we're not in Degr mode
    if (strcmp(defaults.compass,"Degr"))
        azimuth = convertDegstoMils(azimuth);

    azimuth = addGMangle(azimuth);

    wrtext(100, 60, "From the Grid Coordinate at:");
    wrtext(400, 60, get_entry(1));

    wrtext(100, 75, "to the Grid Coordinate at:");
    wrtext(400, 75, get_entry(2));

    wrtext(100, 90, "The distance between them is: ");
    wrtext(400, 90, gcvt(distance,5,temp));
    wrtext(470, 90, defaults.yardstick);
    revblk(400, 88,460,100);

    wrtext(100, 105, "The magnetic azimuth is: ");
    wrtext(400, 105, itoa(azimuth,temp,10));
    wrtext(470, 105, defaults.compass);
    revblk(400,103,460,115);

    wrtext(100, 120, "The magnetic back azimuth is:");
    wrtext(400, 120, itoa(back_azimuth(azimuth),temp,10));
    wrtext(470, 120, defaults.compass);
    revblk(400,118,460,130);

    wrtext(200, 150, "Press any key to continue...");
    getkscan();      /* wait for a key */
    close_win(2);
}
} while (istat);
}

void
rangeAzimuth()
{
    int istat, badData;
    float distance;
    int azimuth;
    char temp[10];
    char saveGrid[9] = "";

```

```

char saveRange[7] = "";
char saveAzimuth[5] = "";

do {
    open_win(1, HP2, 30, 30, 600, 170, "Range / Azimuth Calculation");
    wrtext(140, 150, "Press TAB to move from field to field");
    botlin(keylab2); /* display F9/F10 inside key labels on bottom */

    entry_form(1, NUMERIC, HP2, 60, 60, 8, "Enter GRID of your location: ");
    entry_form(2, NUMERIC, HP2, 60, 80, 6, "Enter range to target in ( ):");
    entry_form(3, NUMERIC, HP2, 60, 100, 4, "Enter mag azimuth in ( ): ");
    entry_dflt(1, saveGrid);
    entry_dflt(2, saveRange);
    entry_dflt(3, saveAzimuth);
    wrtext(320, 80, defaults.yardstick);
    wrtext(280, 100, defaults.compass);

    /* Now we process the input screen */
    badData = 0;
    istat=entry_proc(3, 500, 70, 500, 110); /* process 4 data entry fields */
    strcpy(saveGrid, get_entry(1));
    strcpy(saveRange, get_entry(2));
    strcpy(saveAzimuth, get_entry(3));

    if (istat && badGrid(get_entry(1))) {
        errwin("Invalid Input ");
        badData = 1;
        getksan();
    }

    close_win(1); /* close window */
    botlin(botmes); /* display default KEYLABELS */

    /* Now we check if the user accepted the input (F10 or F9) */
    /* input status OK=1 CANCEL=0 */
    if (istat==1 && !badData) {
        /* Here we display the input data */
        open_win(2, HP2, 30, 30, 600, 170,
            "Range / Azimuth Calculation Results");

        distance = atof(get_entry(2));
        // If we're not in KM mode
        if (strcmp(defaults.yardstick, "KM"))
            distance = convertNMtoKM(distance);

        azimuth = atoi(get_entry(3));
        azimuth = subtractGMangle(azimuth);
        if (azimuth < 0) {
            if (strcmp(defaults.compass, "Degs"))
                azimuth += 6400;
            else
                azimuth += 360;
        }

        // If we're not in Degs mode
        if (strcmp(defaults.compass, "Degs"))
            azimuth = convertMilstoDegs(azimuth);

        tgt_grid(get_entry(1), azimuth, distance, temp);

        wrtext(100, 60, "From your location at:");
        wrtext(430, 60, get_entry(1));

        wrtext(100, 75, "At a range of:");
        wrtext(430, 75, get_entry(2));
        wrtext(500, 75, defaults.yardstick);

        wrtext(100, 90, "Along a magnetic azimuth of:");
        wrtext(430, 90, get_entry(3));
        wrtext(500, 90, defaults.compass);

        wrtext(100, 120, "The target is located at Grid:");
    }
}

```

```

        wrtext(430, 120, temp);
        revblk(430,118,510,130);

        wrtext(200, 150, "Press any key to continue...");
        getkscan();          /* wait for a key */
        close_win(2);
    }
} while (istat);
}

void
intersection()
{
    int istat, badData;
    int azimuth1, azimuth2;
    char temp[10];
    char saveGrid1[9] = "";
    char saveGrid2[9] = "";
    char saveAzimuth1[4] = "";
    char saveAzimuth2[4] = "";

    do {
        open_win(1, HP2, 30, 30, 600, 170, "Intersection Calculation");
        wrtext(140, 150, "Press TAB to move from field to field");
        botlin(keylab2); /* display F9/F10 inside key labels on bottom */

        entry_form(1, NUMERIC, HP2, 60, 60, 8, "Enter GRID of your position #1:");
        entry_form(2, NUMERIC, HP2, 60, 80, 4, "Enter mag azimuth #1 in (    ):");
        entry_form(3, NUMERIC, HP2, 60, 110, 8, "Enter GRID of your position #2:");
        entry_form(4, NUMERIC, HP2, 60, 130, 4, "Enter mag azimuth #2 in (    ):");
        entry_dflt(1,saveGrid1);
        entry_dflt(2,saveAzimuth1);
        entry_dflt(3,saveGrid2);
        entry_dflt(4,saveAzimuth2);
        wrtext(310,80,defaults.compass);
        wrtext(310,130,defaults.compass);

        /* Now we process the input screen */
        badData = 0;
        istat=entry_proc(4, 500, 70, 500, 110); /* process 4 data entry fields */
        strcpy(saveGrid1,get_entry(1));
        strcpy(saveAzimuth1,get_entry(2));
        strcpy(saveGrid2,get_entry(3));
        strcpy(saveAzimuth2,get_entry(4));

        if (istat && (badGrid(get_entry(1)) || badGrid(get_entry(3)))) {
            errwin("Invalid Input ");
            badData = 1;
            getkscan();
        }

        close_win(1);          /* close window */
        botlin(botmes);        /* display default KEYLABELS */

        /* Now we check if the user accepted the input (F10 or F9) */
        /* input status OK=1 CANCEL=0 */
        if (istat==1 && !badData) {
            /* Here we display the input data */
            open_win(2, HP2, 30, 30, 600, 170,
                "Intersection Calculation Results");

            azimuth1 = atoi(get_entry(2));
            azimuth1 = subtractGMangle(azimuth1);
            azimuth2 = atoi(get_entry(4));
            azimuth2 = subtractGMangle(azimuth2);

            if (azimuth1 < 0) {
                if (strcmp(defaults.compass,"Degr"))
                    azimuth1 += 6400;
                else
                    azimuth1 += 360;
            }
        }
    }
}

```

```

    }
    if (azimuth2 < 0) {
        if (strcmp(defaults.compass,"Dega"))
            azimuth2 += 6400;
        else
            azimuth2 += 360;
    }

    // If we're not in Dega mode
    if (strcmp(defaults.compass,"Dega")) {
        azimuth1 = convertMilstoDega(azimuth1);
        azimuth2 = convertMilstoDega(azimuth2);
    }

    triang(get_entry(1), back_azimuth(azimuth1), get_entry(3),
        back_azimuth(azimuth2), temp);

    wrtext(100, 60, "From your 1st location at:");
    wrtext(430, 60, get_entry(1));

    wrtext(100, 75, "With a magnetic azimuth of:");
    wrtext(430, 75, get_entry(2));
    wrtext(500, 75, defaults.compass);

    wrtext(100, 90, "And your 2d location at:");
    wrtext(430, 90, get_entry(3));

    wrtext(100, 105, "With a magnetic azimuth of:");
    wrtext(430, 105, get_entry(4));
    wrtext(500, 105, defaults.compass);

    wrtext(100, 130, "The target is located at Grid:");
    wrtext(430, 130, temp);
    revblk(430,128,510,140);

    wrtext(200, 150, "Press any key to continue...");

    getkscan();          /* wait for a key */
    close_win(2);
    }
    } while (istat);
}

void
resection()
{
    int istat, badData;
    int azimuth1, azimuth2;
    char temp[10];
    char saveGrid1[9] = "";
    char saveGrid2[9] = "";
    char saveAzimuth1[4] = "";
    char saveAzimuth2[4] = "";

    do {
        open_win(1, HP2, 30, 30, 600, 170, "Resection Calculation");
        wrtext(140, 150, "Press TAB to move from field to field");
        botlin(keylab2); /* display F9/F10 inside key labels on bottom */

        entry_form(1, NUMERIC, HP2, 60, 60, 8, "Enter the GRID of object #1:  ");
        entry_form(2, NUMERIC, HP2, 60, 80, 4, "Enter mag azimuth #1 in (  ):");
        entry_form(3, NUMERIC, HP2, 60, 110, 8, "Enter the GRID of object #2:  ");
        entry_form(4, NUMERIC, HP2, 60, 130, 4, "Enter mag azimuth #2 in (  ):");
        entry_dflt(1,saveGrid1);
        entry_dflt(2,saveAzimuth1);
        entry_dflt(3,saveGrid2);
        entry_dflt(4,saveAzimuth2);
        wrtext(310,80,defaults.compass);
        wrtext(310,130,defaults.compass);

        /* Now we process the input screen */
        badData = 0;
    } while (istat);
}

```

```

    istat=entry_proc(4, 500, 70, 500, 110); /* process 4 data entry fields */
    strcpy(saveGrid1,get_entry(1));
    strcpy(saveAzimuth1,get_entry(2));
    strcpy(saveGrid2,get_entry(3));
    strcpy(saveAzimuth2,get_entry(4));

    if (istat && (badGrid(get_entry(1)) || badGrid(get_entry(3)))) {
        errwin("Invalid Input ");
        badData = 1;
        getksan();
    }

    close_win(1);      /* close window */
    botlin(botmes);    /* display default KEYLABELS */

    /* Now we check if the user accepted the input (F10 or F9) */
    /* input status OK=1 CANCEL=0 */
    if (istat==1 && !badData) {
        /* Here we display the input data */
        open_win(2, HP2, 30, 30, 600, 170,
            "Resection Calculation Results");

        azimuth1 = atoi(get_entry(2));
        azimuth1 = subtractGMangle(azimuth1);
        azimuth2 = atoi(get_entry(4));
        azimuth2 = subtractGMangle(azimuth2);

        if (azimuth1 < 0) {
            if (strcmp(defaults.compass,"Deps"))
                azimuth1 += 6400;
            else
                azimuth1 += 360;
        }
        if (azimuth2 < 0) {
            if (strcmp(defaults.compass,"Deps"))
                azimuth2 += 6400;
            else
                azimuth2 += 360;
        }

        // If we're not in Deps mode
        if (strcmp(defaults.compass,"Deps")) {
            azimuth1 = convertMilstoDeps(azimuth1);
            azimuth2 = convertMilstoDeps(azimuth2);
        }

        triang(get_entry(1), azimuth1, get_entry(3),
            azimuth2, temp);

        wrtext(100, 60, "With the 1st object at:");
        wrtext(430, 60, get_entry(1));

        wrtext(100, 75, "Along a magnetic azimuth of:");
        wrtext(430, 75, get_entry(2));
        wrtext(500, 75, defaults.compass);

        wrtext(100, 90, "And the 2nd object at:");
        wrtext(430, 90, get_entry(3));

        wrtext(100, 105, "Along a magnetic azimuth of:");
        wrtext(430, 105, get_entry(4));
        wrtext(500, 105, defaults.compass);

        wrtext(100, 130, "You are located at Grid:");
        wrtext(430, 130, temp);
        revblk(430,128,510,140);

        wrtext(200, 150, "Press any key to continue...");
        getksan();      /* wait for a key */
        close_win(2);
    }
} while (istat);

```

```

}

int
badGrid(char *grid)
{
    int length = strlen(grid);
    if (length == 4 || length == 6 || length == 8)
        return 0;
    else
        return 1;
}

int
badDefaults()
{
    if (get_mark(2) && get_mark(3))
        return 1;
    else if (get_mark(4) && get_mark(5))
        return 1;
    else if (!get_mark(2) && !get_mark(3))
        return 1;
    else if (!get_mark(4) && !get_mark(5))
        return 1;
    else
        return 0;
}

void
defaultSet()
{
    char temp[5];
    int istat, badData = 1;
    int NMStatus = 0, KMStatus = 0, DegStatus = 0, MilStatus = 0;
    FILE *defaultFile;

    do {
        open_win(1, HP2, 30, 30, 600, 170, "Default Program Settings");
        wrtext(100, 150, "Press <space> bar to mark/unmark check boxes");
        botlin(keylab2); /* display F9/F10 inside key labels on bottom */

        if (strcmp(defaults.yardstick, "KM"))
            ++NMStatus;
        else
            ++KMStatus;

        if (strcmp(defaults.compass, "Degs"))
            ++MilStatus;
        else
            ++DegStatus;

        entry_form(1, NUMERIC, HP2, 60, 60, 4, "Enter Grid Magnetic Angle:");
        entry_form(2, CHECKBOX, HP2, 60, 80, NMStatus, "Measure in Nautical Miles:");
        entry_form(3, CHECKBOX, HP2, 60, 95, KMStatus, "Measure in Kilometers: ");
        entry_form(4, CHECKBOX, HP2, 60, 115, DegStatus, "Compute angles in Degrees:");
        entry_form(5, CHECKBOX, HP2, 60, 130, MilStatus, "Compute angles in Mils: ");

        entry_dflt(1, itoa(defaults.gmAngle, temp, 10));

        /* Now we process the input screen */
        badData = 0;
        istat = entry_proc(5, 450, 70, 450, 110); /* process 4 data entry fields */

        if (istat && badDefaults()) {
            errwin("Checkboxes Invalid");
            badData = 1;
            getkscan();
        }

        close_win(1); /* close window */
        botlin(botmes); /* display default KEYLABELS */
    } while (badData);
}

```

```

    } while (badData);

    /* Now we check if the user accepted the input (F10 or F9) */
    /* input status OK=1 CANCEL=0 */
    if (istat==1) {
        /* Here we SAVE the input data in the defaults file */
        defaults.gmAngle = atoi(get_entry(1));

        if (get_mark(2))
            strcpy(defaults.yardstick,"NM");
        else
            strcpy(defaults.yardstick,"KM");
        if (get_mark(4))
            strcpy(defaults.compass,"Degs");
        else
            strcpy(defaults.compass,"Mils");
        if ((defaultFile = fopen(DEF_FILE, "w")) != NULL) {
            fputs(defaults.compass, defaultFile);
            fputs(defaults.yardstick, defaultFile);
            fputs(get_entry(1), defaultFile);
            fclose(defaultFile);
        }
    }
}

```

3. GRIDMATH.H Header File

This is the header file for the math calculations module:

```

#ifndef __GRIDMATH_H
#define __GRIDMATH_H

struct timegrid {
    char time[5];
    char grid[9];
};

void triang(char grid1[9], int azimuth1, char grid2[9], int azimuth2,
            char grid3[9]);
void tgt_grid(char grid1[9], int azimuth1, float range, char grid2[9]);
void trig_calc(int azimuth, float range, int *x_2, int *y_2);
void distazim(char grid1[9], char grid2[9], float *distance,
              int *azimuth);

int back_azimuth(int azimuth);
void time_space(char grid1[9], int azimuth1, float speed, char start_time[5],
               float time_interval, struct timegrid time_array[20]);
void add_time(char curr_time[5], int time_interval, char new_time[5]);
float convertKMtoNM (float KMvalue);
int convertDegstoMils (float Degsvalue);
float convertNMtoKM (float NMvalue);
int convertMilstoDegs (float Milsvalue);
int addGMangle(int azimuth);
int subtractGMangle(int azimuth);
#endif

```

4. GRIDMATH.C Source File

This is the source code file for the math calculations module:

```

#include <math.h>
#include <string.h>
#include <stdlib.h>
#include "gridmath.h"
#include "gridcalc.h"

```



```

extern struct Defaults defaults;
/*****
/** Function TIME_SPACE (uses function ADD_TIME). *
/** Accepts a 4-8 digit MGRS grid, azimuth, speed,*
/** start time, time interval, and returns a 20 *
/** unit array of type timegrid representing the *
/** forecasted position of the target travelling *
/** from the given grid, at the given azimuth and *
/** speed. Speed must be given in KPH. The time *
/** interval is in minutes. *
*****/

/*
void time_space(char grid1[9], int azimuth1, float speed, char start_time[5],
float time_interval, timegrid time_array[20])
{
    int i;
    float speed_dist;
    char time[5], new_time[5], curr_grid[9], new_grid[9];

    //Calculate initial values
    speed_dist = speed * (time_interval / 60);
    strcpy(curr_time, start_time);
    strcpy(curr_grid, grid1);
    tgt_grid(curr_grid, 0, 0, curr_grid); //convert to 8-digit

    //Load array with calculated values
    for (i = 0; i < 20 ; i++)
    {
        strcpy(time_array[i].time, curr_time);
        add_time(curr_time, time_interval, new_time);
        strcpy(curr_time, new_time);
        strcpy(time_array[i].grid, curr_grid);
        tgt_grid(curr_grid, azimuth1, speed_dist, new_grid);
        strcpy(curr_grid, new_grid);
    }
}

*/

/*****
/** Function ADD_TIME. Used by TIME_SPACE to *
/** calculate 24 hour military times based upon *
/** a current time and a time-interval in minutes.*
/** Minute intervals > one day (1440) minutes will *
/** be modulus truncated. *
*****/

void add_time(char curr_time[5], int time_interval, char new_time[5])
{
    char h_string[3], m_string[3], *zero_string = "0000";
    int i, j, hours, minutes, total_minutes;

    //Extract substrings and convert to integers
    for (i = 0; i < 2 ; i++)
        h_string[i] = curr_time[i];
    for (i = 0, j = 2; j < 4; i++, j++)
        m_string[i] = curr_time[j];
    hours = atoi(h_string);
    minutes = atoi(m_string);

    //Calculate total number of minutes and convert back to hours/mins
    total_minutes = (60 * hours) + minutes + time_interval;
    if (total_minutes >= 1440)
        total_minutes = total_minutes % 1440;
    hours = (int) total_minutes / 60;
    minutes = total_minutes % 60;

    //Convert back to strings; Add zeros as required
    itoa(hours, h_string, 10);

```

```

        itoa(minutes, m_string, 10);
        strcpy(new_time, zero_string);
        for (i = (2-strlen(h_string)), j = 0; i < 2; i++, j++)
            new_time[i] = h_string[j];
        for (i = (4-strlen(m_string)), j = 0; i < 4; i++, j++)
            new_time[i] = m_string[j];
    }

    /*******
    /** Function TRIANG (uses function TGT_GRID.
    /** Accepts 2 4-8 digit MGRS grids with azimuths
    /** from/to an unknown location and calculates
    /** that grid. Returned grid is an 8-digit grid.
    /** All azimuths must be in degs.
    /*******
void triang(char grid1[9], int azimuth1, char grid2[9], int azimuth2,
            char grid3[9])
{
    int a_angle, c_angle, azimuth3;
    float a_dist, c_dist;

    distazim(grid2, grid1, &a_dist, &azimuth3);
    a_angle = abs(azimuth1 - azimuth2);
    if (a_angle > 180)
        a_angle = 360 - a_angle;
    c_angle = abs(back_azimuth(azimuth2) - azimuth3);
    if (c_angle > 180)
        c_angle = 360 - c_angle;

    c_dist = (sin(c_angle * (M_PI / 180)) * a_dist) /
              sin(a_angle * (M_PI / 180));
    tgt_grid(grid1, back_azimuth(azimuth1), c_dist, grid3);
}

    /*******
    /** Function TGT_GRID (uses function TRIG_CALC).
    /** Accepts a 4-8 digit MGRS grid, azimuth, and
    /** range, and computes the target grid from that
    /** location. Returned grid is an 8-digit grid.
    /** Range must be less than 100km from the origin
    /** grid coordinate. Azimuth of 360 should be
    /** called as "0." All azimuths must be in degs.
    /*******
void tgt_grid(char grid1[9], int azimuth1, float range, char grid2[9])
{
    char grid1x[5] = "0000", grid1y[5] = "0000";
    char grid2x[5], grid2y[5], *zero_string = "00000000";
    int x_1, x_2, y_1, y_2, i, j, halfway, raw_x, raw_y;

    /*******
    /** Extract substrings from input grid.
    /** Convert input grid into 8-digits.
    /*******
    halfway = strlen(grid1) / 2;
    for (i = 0; i < halfway; i++)
        grid1x[i] = grid1[i];
    for (i = halfway; i < strlen(grid1); i++)
        grid1y[i-halfway] = grid1[i];

    /*******
    /** Convert to integer and perform trig
    /** calculations to determine abscissa
    /** and ordinate.
    /*******
    x_1 = atoi(grid1x);
    y_1 = atoi(grid1y);
    if ((azimuth1 >= 0) && (azimuth1 <= 90))
    {
        trig_calc(90-azimuth1, range, &x_2, &y_2);
        raw_x = x_1 + x_2;
        raw_y = y_1 + y_2;
    }
}

```

```

    }
    else if ((azimuth1 > 90) && (azimuth1 <= 180))
    {
        trig_calc(azimuth1-90, range, &x_2, &y_2);
        raw_x = x_1 + x_2;
        raw_y = y_1 - y_2;
    }
    else if ((azimuth1 > 180) && (azimuth1 <= 270))
    {
        trig_calc(270-azimuth1, range, &x_2, &y_2);
        raw_x = x_1 - x_2;
        raw_y = y_1 - y_2;
    }
    else if ((azimuth1 > 270) && (azimuth1 < 360))
    {
        trig_calc(azimuth1-270, range, &x_2, &y_2);
        raw_x = x_1 - x_2;
        raw_y = y_1 + y_2;
    }

    /*******
    /* Ensure we havn't crossed into the
    /* adjacent 100,000m grid square.
    /*******
    if (raw_x >= 10000)
        raw_x -= 10000;
    else if (raw_x < 0)
        raw_x += 10000;
    if (raw_y >= 10000)
        raw_y -= 10000;
    else if (raw_y < 0)
        raw_y += 10000;

    /*******
    /* Convert back to string and load into *
    /* 8-digit coordinate result
    /*******
    itoa(raw_x, grid2x, 10);
    itoa(raw_y, grid2y, 10);
    strcpy(grid2, zero_string);
    for (i = (4-strlen(grid2x)), j = 0; i < 4; i++, j++)
        grid2[i] = grid2x[j];
    for (i = (8-strlen(grid2y)), j = 0; i < 8; i++, j++)
        grid2[i] = grid2y[j];
    }

    /*******
    /* Function TRIG_CALC. Used by TGT_GRID*
    /* to compute abscissa and ordinate of *
    /* target grid coordiante.
    /*******
    void trig_calc(int azimuth, float range, int *x_2, int *y_2)
    {
        *x_2 = (int) 0.5 + (100 * range * cos(azimuth * (M_PI / 180)));
        *y_2 = (int) 0.5 + (100 * range * sin(azimuth * (M_PI / 180)));
    }

    /*******
    /* Function BACK_AZIMUTH. Used by various*
    /* functions to return back azimuth of a *
    /* given azimuth.
    /*******
    int back_azimuth(int azimuth)
    {
        if (strcmp(defaults.compass, "Degr")
            return ((azimuth + 3200) % 6400);
        else
            return ((azimuth + 180) % 360);
    }

    /*******

```

```

/** Function DistAzim - computes the distance and *
/** azimuth between two grid coordinates in the *
/** Cartesian plane. Grids must be either 4, 6, *
/** or 8 digits in UTM/MGRS format and located *
/** within 50 LINEAR KM of each other in order to *
/** provide for crossing into adjacent 100,000 *
/** grid-square boundaries. *
/** *****

void
distazim(char grid1[9], char grid2[9], float *distance, int *azimuth3)
{
    char grid1x[5] = "0000", grid1y[5] = "0000",
    grid2x[5] = "0000", grid2y[5] =
"0000";

    int x_1, x_2, y_1, y_2, i, halfway;
    float abscissa, ordinate, raw_azimuth;

    /** *****
    /** Extract substrings from input grids.*/
    /** Coordinates will be converted into */
    /** 8-digits for use in this function */
    /** *****
    halfway = strlen(grid1) / 2;
    for (i = 0; i < halfway; i++)
        grid1x[i] = grid1[i];
    for (i = halfway; i < strlen(grid1); i++)
        grid1y[i-halfway] = grid1[i];
    halfway = strlen(grid2) / 2;
    for (i = 0; i < halfway; i++)
        grid2x[i] = grid2[i];
    for (i = halfway; i < strlen(grid2); i++)
        grid2y[i-halfway] = grid2[i];

    /** *****
    /** Convert to integer and perform calcs */
    /** If crossing into adjacent 100,000 */
    /** grid square, adjust accordingly. */
    /** *****
    x_1 = atoi(grid1x);
    y_1 = atoi(grid1y);
    x_2 = atoi(grid2x);
    y_2 = atoi(grid2y);
    if (abs(x_2 - x_1) > 5000)
        if (x_2 < x_1)
            abscissa = (10000 + x_2) - x_1;
        else
            abscissa = x_2 - (10000 + x_1);
    else
        abscissa = x_2 - x_1;
    if (abs(y_2 - y_1) > 5000)
        if (y_2 < y_1)
            ordinate = (10000 + y_2) - y_1;
        else
            ordinate = y_2 - (10000 + y_1);
    else
        ordinate = y_2 - y_1;

    /** *****
    /** Calculate distance and azimuth as pointed to */
    /** in main program by aptly named pointers. */
    /** *****
    *distance = sqrt((abscissa*abscissa)+(ordinate*ordinate)) / 100;
    if ((abscissa == 0) && (ordinate == 0))
        raw_azimuth = 0;
    else
        raw_azimuth = 450-(atan2(ordinate,abscissa) * (180 / M_PI));
    *azimuth3 = (int) (raw_azimuth + 0.5) % 360;
}

float

```

```

convertKMtoNM (float KMvalue)
{
    return (KMvalue * 0.54);
}
int
convertDegstoMils (float Degsvalue)
{
    return (Degsvalue * 17.778);
}

float
convertNMtoKM (float NMvalue)
{
    return (NMvalue * 1.852);
}
int
convertMilstoDegs (float Milsvalue)
{
    return (Milsvalue * 0.0563);
}

int
addGMangle(int azimuth)
{
    return (azimuth + defaults.gmAngle);
}

int
subtractGMangle(int azimuth)
{
    return (azimuth - defaults.gmAngle);
}

```


APPENDIX E. SOURCE CODE FOR THE REDMAN APPLICATION

A. REDMAN APPLICATION SOURCE CODE

The following code (in alphabetical order by module) represents the Delphi language source code for the DACT version of the REDMAN Order Processing application. This code was developed in conjunction with Borland's Delphi 1.0 integrated development environment. The author created all code that follows (however, source code annotated with "public domain" in the introductory comments has been adapted from public sources). Delphi forms (DFM files) are not included due to space constraints, however, these will be available online (as well as all other source code developed in this thesis) at <http://www.stl.nps.navy.mil/~iirg/>

1. ABOUT.PAS Source Code

This is the source code file for the "About Box" module:

```
unit About;

interface

uses WinTypes, WinProcs, Classes, Graphics, Forms, Controls, StdCtrls,
    Buttons, ExtCtrls;

type
    TfrmAbout = class(TForm)
        Panel1: TPanel;
        OKButton: TBitBtn;
        ProgramIcon: TImage;
        ProductName: TLabel;
        Version: TLabel;
        Copyright: TLabel;
        Comments: TLabel;
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    frmAbout: TfrmAbout;

implementation

{$R *.DFM}

end.
```

2. ADDUNIT.PAS Source Code

This is the source code file for the “Add a T/O Unit” module:

```
unit Addunit;

interface

uses WinTypes, WinProcs, Classes, Graphics, Forms, Controls, Buttons,
    StdCtrls, Mask, DBCtrls, DB, Wwdatsrc, ExtCtrls, DBTables, Wwtable;

type
    TfrmAddUnit = class(TForm)
        OKBtn: TBitBtn;
        CancelBtn: TBitBtn;
        HelpBtn: TBitBtn;
        Bevel1: TBevel;
        dtasrcTOAdd: TwwDataSource;
        bxeddbUnitName: TDBEdit;
        bxeddbUnitCmdr: TDBEdit;
        txtUnitCmdr: TLabel;
        txtUnitName: TLabel;
        procedure FormActivate(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    frmAddUnit: TfrmAddUnit;

implementation
Uses Order, Mainform;

{$R *.DFM}

procedure TfrmAddUnit.FormActivate(Sender: TObject);
begin
    bxeddbUnitName.SetFocus;
end;

end.
```

3. CHKLIST.PAS Source Code

This is the source code file for the “Checklist Item” module:

```
unit Chklist;

interface

uses WinTypes, WinProcs, Classes, Graphics, Forms, Controls, Buttons,
    StdCtrls, ExtCtrls, DBCtrls, Mask, DB, Wwdatsrc, DBTables, Wwtable;

type
    TfrmCheckList = class(TForm)
        OKBtn: TBitBtn;
        CancelBtn: TBitBtn;
        HelpBtn: TBitBtn;
        Bevel1: TBevel;
        tblCheckLst: TwwTable;
        dtsCheckList: TwwDataSource;
    end;
```



```

        dbnCheckLst: TDBNavigator;
        DBMemo1: TDBMemo;
        procedure FormShow(Sender: TObject);
        private
            { Private declarations }
        public
            { Public declarations }
        end;

var
    frmCheckList: TfrmCheckList;

implementation

uses MainForm, Order;

{$R *.DFM}

procedure TfrmCheckList.FormShow(Sender: TObject);
var
    ChildPointer : TfrmOrderView;
begin
    ChildPointer := frmMain.GetCurrentChildPointer(Self);
    with tblCheckLst do
        begin
            filter.clear;
            filter.add('MISSION_AREA = ' + ChildPointer.tblMissions.FieldByName('AREA').AsString) ;
            filter.Activate;
        end;
    end;
end.

```

4. CLIPSTUF.PAS Source Code

This is the public domain source code file for the "Clipboard Operations" module:

```

unit Clipstuf;

interface

uses SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
    Forms, Dialogs, Buttons, ExtCtrls, StdCtrls, Menus;

procedure DoEditPaste;
procedure DoEditCut;
procedure DoEditCopy;
procedure DoEditUndo;
function IsEditControl(Handle : HWnd): Boolean;
function HasText(Control : TWinControl): Boolean;
function CanUndo(Control : TWinControl): Boolean;
function CanPaste(Control : TWinControl): Boolean;

implementation

uses Clipbrd;

procedure DoEditPaste;
begin
    SendMessage(GetFocus, WM_PASTE, 0, 0);
end;

procedure DoEditCut;
begin
    SendMessage(GetFocus, WM_CUT, 0, 0);
end;

procedure DoEditCopy;
begin

```

```

    SendMessage(GetFocus, WM_COPY, 0, 0);
end;

procedure DoEditUndo;
begin
    SendMessage(GetFocus, EM_UNDO, 0, 0);
end;

function IsEditControl(Handle : HWnd): Boolean;
var
    szName : array[0..4] of Char;
begin
    if (Handle = 0) then
        Result := False
    else begin
        GetClassName(Handle, szName, SizeOf(szName));
        Result := (StrIComp(szName, 'edit') = 0);
    end;
end;

function HasText(Control : TWinControl): Boolean;
var
    Handle : HWnd;
    Select : LongInt;
    First : Word;
    Last : Word;
begin
    if (Control is TCustomEdit) then
        Result := (TEdit(Control).SelLength > 0)
    { this is here basically to handle combo box edit fields }
    else begin
        Handle := GetFocus;

        if IsEditControl(Handle) then begin
            Select := SendMessage(Handle, EM_GETSEL, 0, 0);
            First := LOWORD(Select);
            Last := HIWORD(Select);
            Result := (First <> Last);
        end
        else
            Result := False;
    end
end;

function CanUndo(Control : TWinControl): Boolean;
begin
    if (Control is TCustomEdit) or IsEditControl(GetFocus) then
        Result := (SendMessage(GetFocus, EM_CANUNDO, 0, 0) > 0)
    else
        Result := False;
end;

function CanPaste(Control : TWinControl): Boolean;
begin
    if (Control is TCustomEdit) or IsEditControl(GetFocus) then
        Result := Clipboard.HasFormat(CF_TEXT)
    else
        Result := False;
end;

end.

```

5. DDGUTILS.PAS Source Code

This is the public-domain source code file for the "DOS Operations" module:

```
unit Ddgutils;

interface
uses SysUtils, Dialogs, Forms, controls, LZExpand, WinProcs, WinTypes;

type
  ELZCopyError = class(Exception); { Create an LZCopy error exception class }
  EWinExecError = class(Exception); { Create a WinExec error exception class }

procedure CopyFile(FromFileName, ToFileName: string);
procedure MoveFile(FromFileName, ToFileName: string);
function GetDirectoryName(Dir: string): string;
function DDGExec(RunFile: string): word;

implementation

procedure CopyFile(FromFileName, ToFileName: string);
var
  FromFile, ToFile: File;
begin
  AssignFile(FromFile, FromFileName); { Assign FromFile to FromFileName }
  AssignFile(ToFile, ToFileName);     { Assign ToFile to ToFileName }
  ShowMessage('From File is ' + FromFileName + ' and To File is ' + ToFileName);

  Reset(FromFile);                    { Open file for input }
  try
    Rewrite(ToFile);                  { Create file for output }
    try
      { copy the file and if a negative value is returned raise an exception }
      if LZCopy(TFileRec(FromFile).Handle, TFileRec(ToFile).Handle) < 0 then
        raise ELZCopyError.Create('Error using LZCopy')
      finally
        CloseFile(ToFile); { Close ToFile }
      end;
    finally
      CloseFile(FromFile); { Close FromFile }
    end;
  end;
end;

procedure MoveFile(FromFileName, ToFileName: string);
begin
  RenameFile(FromFileName, ToFileName); { Rename the file }
end;

function GetDirectoryName(Dir: string): string;
begin
  if Dir[length(Dir)] <> '\' then { If the directory doesn't contain a '\' }
    Result := Dir + '\'          { character add one. }
  else
    Result := Dir;               { Result is the Dir }
  end;
end;

function DDGExec(RunFile: string): Word;
var
  RunFilePC: PChar;
begin
  GetMem(RunFilePC, length(RunFile)+1); { Allocate memory for RunFilePC }
  try
    Result := WinExec(StrPCopy(RunFilePC, RunFile), { Run the program pass back }
      sw_ShowNormal); { the returned result }
    if Result < 32 then { An error was returned }
      raise EWinExecError.Create('Error using WinExec'); { Raise an exception }
    finally
      FreeMem(RunFilePC, length(RunFile)+1); { Free RunFilePC's memory }
    end;
  end;
end;
```

```
end;
end.
```

6. DEFAULTS.PAS Source Code

This is the source code file for the “Defaults” module:

```
unit Defaults;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, TabNotBk, StdCtrls, Buttons, EIniFile, Grids, Outline,
  DBOutln, DB, Wwdatsrc, DBTables, Wwtable, Menus;

type
  TfrmDefaults = class(TForm)
    TabbedNotebook1: TTabbedNotebook;
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    HelpBtn: TBitBtn;
    tblTOBase: TwwTable;
    dtasrcTOBase: TwwDataSource;
    outlineTOBase: TDBOutline;
    pmnuTOBase: TPopupMenu;
    pmnuitemAddChild: TMenuItem;
    pmnuitemAddSibling: TMenuItem;
    pmnuitemSeparator1: TMenuItem;
    pmnuitemEdit: TMenuItem;
    pmnuitemSeparator2: TMenuItem;
    pmnuitemDelete: TMenuItem;
    edtWhoAmI: TEdit;
    lblWhoAmI: TLabel;
    procedure FormActivate(Sender: TObject);
    procedure dtasrcTOBaseDataChange(Sender: TObject; Field: TField);
    procedure pmnuitemAddChildClick(Sender: TObject);
    procedure pmnuitemEditClick(Sender: TObject);
    procedure outlineTOBaseAutoDragDrop(Sender: TObject;
      var Accept: Boolean; var FromNode, ToNode: OpenString);
    procedure outlineTOBaseDragDrop(Sender, Source: TObject; X,
      Y: Integer);
    procedure outlineTOBaseDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure pmnuitemDeleteClick(Sender: TObject);
    procedure DeleteTOTree(Sender: TObject; NodeNum : Integer);
    function AnyNodesThatReportToThisNode(Sender: TObject; NodeNum : Integer): Boolean;
    procedure OKBtnClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmDefaults: TfrmDefaults;

implementation
Uses MainForm, AddUnit, EditUnit;

{$R *.DFM}

Const IniFileName = 'REDMAN.INI';

Var iniFile : TEIniFile;
    currentDirectory : String;
    StartSynching: Boolean;
```

```

procedure TfrmDefaults.FormActivate(Sender: TObject);
begin
    StartSynching := false;
    tblTOBase.Open;
    outlineTOBase.LoadFromDataSet;
    StartSynching := true;

end;

procedure TfrmDefaults.dtasrcTOBaseDataChange(Sender: TObject;
    Field: TField);
begin
    if StartSynching then
        outlineTOBase.SynchOutline(Self);
end;

procedure TfrmDefaults.pmmuitemAddChildClick(Sender: TObject);
var
    intParentUnitNo, intNewUnitNo: integer;
begin
    StartSynching := false;

    if Sender = pmmuitemAddChild then
    begin
        intParentUnitNo := tblTOBase.FieldByName('UNIT_CODE').AsInteger;
        frmAddUnit.Caption := 'Add a Unit UNDER ' +
tblTOBase.FieldByName('UNIT_TEXT').AsString;
        end
        { else if popup menu option 'Add Sibling' was selected }
        else
        begin
            intParentUnitNo := tblTOBase.FieldByName('REPORTS_TO').AsInteger;
            frmAddUnit.Caption := 'Add a Unit AT THE SAME LEVEL as ' +
tblTOBase.FieldByName('UNIT_TEXT').AsString;;
            end;

            If tblTOBase.RecordCount = 0 Then
                intNewUnitNo := 1
            Else
            Begin
                tblTOBase.Last;
                intNewUnitNo := tblTOBase.FieldByName('UNIT_CODE').AsInteger + 1;
            End;
            tblTOBase.Append;
            tblTOBase.FieldByName('RESERVE').AsBoolean := False;

            { if parent employee number is not 0, i.e. if you are not adding a peer
              to a top-level parent }
            if intParentUnitNo <> 0 then
                tblTOBase.FieldByName('REPORTS_TO').AsInteger := intParentUnitNo;
            tblTOBase.FieldByName('UNIT_CODE').AsInteger := intNewUnitNo;
            frmAddUnit.dtasrcTOAdd.DataSet := frmDefaults.tblTOBase;
            frmAddUnit.ShowModal;

            { if OK button pressed to add to employee }
            if frmAddUnit.ModalResult=mrOK then
                begin
                    if (tblTOBase.State = dsEdit) or (tblTOBase.State = dsInsert) then
                        begin
                            { Add new employee to database }
                            tblTOBase.Post;
                            { TDBOutline method to add current record (in this case, the new employee)
                              to the existing outline }
                            with tblTOBase do
                                outlineTOBase.AddDBRecord(FieldByName('UNIT_CODE').AsString,
                                    FieldByName('UNIT_TEXT').AsString, FieldByName('REPORTS_TO').AsString);
                            end;
                        end
                    else tblTOBase.Cancel;
                end;
            end;

```

```

        outlineTOBase.SynchOutline(Self);
        StartSynching := true;
    end;

    procedure TfrmDefaults.pmnuitemEditClick(Sender: TObject);
    begin
        StartSynching := false;
        frmEditUnit.dtasrcTOEdit.DataSet := frmDefaults.tblTOBase;
        frmEditUnit.ShowModal;

        With tblTOBase Do
        Begin
            { if OK button pressed to add to employee }
            if frmEditUnit.ModalResult=mrOK then
            begin
                if (State = dsEdit) or (State = dsInsert) then
                begin
                    Post;
                    outlineTOBase.ChangeDBRecord(FieldByName('UNIT_CODE').AsString,
                    FieldByName('UNIT_CODE').AsString,
                    FieldByName('UNIT_TEXT').AsString);
                end;
            end
            else Cancel;
        End;

        StartSynching := true;
    end;

    procedure TfrmDefaults.outlineTOBaseAutoDragDrop(Sender: TObject;
    var Accept: Boolean; var FromNode, ToNode: OpenString);
    begin
        { this code overrides the default confirmation message presented when
        a node is dragged-and-dropped, replacing it with the custom message.
        This ability gives the programmer control over the confirmation
        message presented to the user.}
        if MessageDlg(' Have unit [' + FromNode + '] be task organized under [' + ToNode + ']?'
        ,mtConfirmation, mbOkCancel, 0) = mrOk
        then Accept := true
        else Accept := false
    end;

    procedure TfrmDefaults.outlineTOBaseDragDrop(Sender, Source: TObject; X,
    Y: Integer);
    begin
        If Source = outlineTOBase then
        { AutoDrop is the single method call required to perform the move
        of the node and update the database }
        outlineTOBase.AutoDrop(X, Y);
    end;

    procedure TfrmDefaults.outlineTOBaseDragOver(Sender, Source: TObject; X,
    Y: Integer; State: TDragState; var Accept: Boolean);
    begin
        if Source = outlineTOBase then Accept := true;
    end;

    procedure TfrmDefaults.pmnuitemDeleteClick(Sender: TObject);
    var
        rtnMsg: String;
    begin
        StartSynching := false;

        if messagedlg('Remove the Unit "' + tblTOBase.FieldByName('UNIT_TEXT').AsString +
        '" from the T/O (and ALL of its subordinates)?', mtconfirmation, mbOkCancel, 0) = mrOk
        then
        begin
            { !! be sure to delete the Table record first!! If you delete the Outline node
            first, then DataAutoSynch will move the datapointer to match the new

```

```

        DBoutline.SelectedItem, and you'll end up deleting the wrong record! }
        DeleteTOTree(Self, tblTOBase.FieldName('UNIT_CODE').AsInteger);
        outlineTOBase.Delete(outlineTOBase.SelectedItem);
        tblTOBase.Pack (rtnMsg);
    end;

    outlineTOBase.SetFocus;
    StartSynching := true;
end;

procedure TfrmDefaults.DeleteTOTree(Sender: TObject; NodeNum : Integer);
Begin
    With tblTOBase Do
    Begin
        DisableControls;
        If (NOT AnyNodesThatReportToThisNode(Self, NodeNum))Then
            Delete
        Else
            Begin
                First;
                While NOT EOF DO
                Begin
                    If (FieldName('REPORTS_TO').AsInteger = NodeNum) Then
                        DeleteTOTree(Self, FieldByName('UNIT_CODE').AsInteger)
                    Else
                        Next;
                End;
                FindKey([NodeNum]);
                Delete;
            End;
        EnableControls;
    End;
End;

function TfrmDefaults.AnyNodesThatReportToThisNode(Sender: TObject; NodeNum : Integer):
Boolean;
var
    Found : Boolean;
Begin
    {This can be replaced by a FindKey if we can dynamically create secondary indices}
    With tblTOBase do
    Begin
        Found := False;
        First;
        While (Not EOF AND Not Found) Do
        Begin
            If (FieldName('REPORTS_TO').AsInteger = NodeNum) Then
                Found := True
            Else
                Next;
        End;
        FindKey([NodeNum]); {Go back to Node we started at}
    End;
    AnyNodesThatReportToThisNode:= Found;
End;

procedure TfrmDefaults.OKBtnClick(Sender: TObject);
begin
    GetDir(0,currentDirectory);
    iniFile := TIniFile.Create(currentDirectory + '\ ' + IniFileName);
    IniFile.WriteString('Task Organization', 'Baseline WhoAmI', edtWhoAmI.Text);
    iniFile.Free;
end;

procedure TfrmDefaults.FormCreate(Sender: TObject);
begin
    GetDir(0,currentDirectory);
    iniFile := TIniFile.Create(currentDirectory + '\ ' + IniFileName);
    edtWhoAmI.Text := IniFile.ReadString('Task Organization', 'Baseline WhoAmI', '');
    frmMain.WhichUnitAmIText := edtWhoAmI.Text;
    iniFile.Free;
end;

```

```
end;

End.
```

7. DLGTOADD.PAS Source Code

This is the source code file for the “T/O Add Unit Dialogue” module:

```
unit Dlgtoadd;

interface

uses WinTypes, WinProcs, Classes, Graphics, Forms, Controls, Buttons,
    StdCtrls, DB, Mask, DBCtrls, ExtCtrls, DBTables, Wwdatsrc, Wwtable;

type
    TdlgAddUnit = class(TForm)
        OKBtn: TBitBtn;
        CancelBtn: TBitBtn;
        HelpBtn: TBitBtn;
        Bevel1: TBevel;
        bxeddbUnitName: TDBEdit;
        bxeddbUnitCmdr: TDBEdit;
        lblUnitName: TLabel;
        lblUnitCmdr: TLabel;
        dtasrcTOAdd: TwwDataSource;
        procedure FormActivate(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    dlgAddUnit: TdlgAddUnit;

implementation
Uses Order, MainForm;

{$R *.DFM}

procedure TdlgAddUnit.FormActivate(Sender: TObject);
begin
    dtasrcTOAdd.DataSet := OrderView.tblTO;
    bxeddbUnitName.SetFocus;
end;

end.
```

8. EDITUNIT.PAS Source Code

This is the source code file for the “T/O Edit Unit” module:

```
unit Editunit;

interface

uses WinTypes, WinProcs, Classes, Graphics, Forms, Controls, Buttons,
    StdCtrls, ExtCtrls, DBCtrls, Mask, DB, Wwdatsrc, DBTables, Wwtable, Dialogs;

type
    TfrmEditUnit = class(TForm)
        OKBtn: TBitBtn;
```



```

CancelBtn: TBitBtn;
HelpBtn: TBitBtn;
Bevel1: TBevel;
dtasrcTOEdit: TwwDataSource;
bxeddbUnitName: TDBEdit;
bxeddbUnitCmdr: TDBEdit;
memUnitTask: TDBMemo;
txtUnitName: TLabel;
txtUnitCmdr: TLabel;
txtUnitTask: TLabel;
cboRESERVE: TDBCheckBox;
procedure FormActivate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  frmEditUnit: TfrmEditUnit;

implementation
  Uses Order, MainForm;

  {$R *.DFM}

  procedure TfrmEditUnit.FormActivate(Sender: TObject);
  begin
    bxeddbUnitName.SetFocus;
  end;
end.

```

9. EINIFILE.PAS Source Code

This is the public domain source code file for the "Enhanced INI File" module:

```

unit EIniFile;

interface

uses WinTypes, Classes, IniFiles;

type

  (*
   * Derived version of TIniFile
   *)
  TEIniFile = class(TIniFile)
  private
    (*
     * Obtain full pathname to the current .INI file
     *)
    function GetIniPathName : string;

    (*
     * Obtain a backup .INI pathname
     *)
    function GetBackupIniPathName( iniName : string ) : string;

  public

    (*
     * Searches entire .INI file for a specified item.
     *
     * The item to search for can be a substring.
    *)

```

```

*
* If found the function returns true and the Section
* and Line parameters are filled to hold what was found.
*)
function ItemExists( const Item
: string;

var Section : string;
var Line : string ) :
boolean;

(*
* Add a new item to a section of a .INI
*
* This function is useful for that wierd section of SYSTEM.INI
* where there are multiple lines like: device=...
*
* This function will add a new item. It first renames the
* .INI to .0 or something that works. Then it copies the entire
* file line by line looking for the specified section. Once found
* it adds the new Line as the first item of the specified section.
*
* Pass in the section that you expect to find Item in WITHOUT the
* braces. IE for [386Enh] pass in 386Enh.
*
* Finally pass in the ENTIRE line to be added. This means
* you must pass in the WHOLE line as in: device=c:\windows\test.386
*)
function AddItem( const
Section : string;

const Line : string ) : boolean;

(*
* Replace a specific item in a .INI file.
*
* This function is useful for that wierd section of SYSTEM.INI
* where there are multiple lines like: device=...
*
* This function will replace an item. It first renames the
* .INI to .0 or something that works. Then it copies the entire
* file line by line looking for the specified line. Once found
* it substitutes Line for the found one.
*
* To make this function work correctly pass in the item you want
* to replace in 'Item'. This can be a substring but be careful to
* make it unique.
*
* Pass in the section that you expect to find Item in WITHOUT the
* braces. IE for [386Enh] pass in 386Enh.
*
* Finally pass in the ENTIRE line to replace it with. This means
* you must pass in the WHOLE line as in: device=c:\windows\test.386
*)
function ReplaceItem( const Item
: string;

const Section : string;
const Line : string ) : boolean;

(*
* Delete a specific item from a .INI file.
*
* This function is useful for that wierd section of SYSTEM.INI
* where there are multiple lines like: device=...
*
* This function will delete an item. It first renames the
* .INI to .0 or something that works. Then it copies the entire
* file line by line looking for the specified line. Once found
* it simply does not copy it to the new .INI.

```

```

*
* To make this function work correctly pass in the item you want
* to delete in 'Item'. This can be a substring but be careful to
* make it unique.
*
* Pass in the section that you expect to find Item in WITHOUT the
* braces. IE for [386Enh] pass in 386Enh.
*)
      function                      DeleteItem(          const          Item
      : string;

                                const          Section : string ) : boolean;

end;

implementation

uses WinProcs, SysUtils;

(*
* This function determines if a path needs
* to be appended to the .INI filename.
*
* By definition, if there is no path info in the .INI file
* name, the GetProfile functions assume it is in the Windows
* directory.
*)
function TIniFile.GetIniPathName : string;
var
    winDir  : array[0..127] of char;
begin
    { Use TIniFile property for filename }
    Result := FileName;

    { Check to see if it has a full path. If not then assume windows directory }
    if Pos('\',Result) = 0 then
    begin
        GetWindowsDirectory( winDir, 127 );
        Result := StrPas( winDir );
        Result := Result + '\' + FileName;
    end;
end;

(*
* This function takes the passed in .INI filename
* and determines a backup filename by adding a simple
* numerical suffix.
*)
function TIniFile.GetBackupIniPathName( iniName : string ) : string;
var
    c          : integer;
    i          : integer;
begin
    { We need to rename it to a backup name }
    c := -1;
    repeat
        { Get our name }
        Result := iniName;

        { Increment the extension counter }
        inc(c);

        { Locate the .INI part and delete it }
        i := Pos('.INI',Result);
        Delete(Result,i,4);

        { Add a new extention }
        Result := Result + '.' + IntToStr(c);
    until

```

```

    until not FileExists(Result);
end;

(*
* Searches entire .INI file for a specified item.
*
* The item to search for can be a substring.
*
* If found the function returns true and the Section
* and Line parameters are filled to hold what was found.
*)
function TEIniFile.ItemExists( const      Item           : string;
                               var         Section       :
string;
                               var         Line         : string
) : boolean;
var
    iniFile : text;
    iniName  : string;
    curItem  : string;
    theItem  : string;
    i        : integer;
begin
    try
        { Initialize return }
        Result      := false;
        Section     := '';
        Line        := '';

        { Convert searched for item to upper case }
        theItem     := UpperCase( Item );

        { Get ini file path and name }
        iniName     := GetIniPathName;

        { Point to the file and open it }
        AssignFile( iniFile, iniName );
        Reset( iniFile );

    repeat
        { Read a line }
        readln( iniFile, curItem );

        { See if this is a section - Something like: [386Enh] }
        if (Pos('[',curItem)>0) and (Pos(']',curItem)>0) then
            begin
                { Save it for our caller }
                Section := curItem;

                { Remove left brace }
                i       := Pos('[',Section);
                Delete( Section, i, 1 );

                { Remove right brace }
                i       := Pos(']',Section);
                Delete( Section, i, 1 );

                end
            else
                { Search for our item in this line }
                Result  := (Pos(theItem,UpperCase(curItem)) > 0);
                until Result or eof(iniFile);

            { If we found it then return the line }
            if Result then
                Line    := curItem
            else
                Section := '';

            finally
                CloseFile( iniFile );

```

```

        end;
end;

(*
* Add a new item to a section of a .INI
*
* This function is useful for that wierd section of SYSTEM.INI
* where there are multiple lines like: device=...
*
* This function will add a new item. It first renames the
* .INI to .0 or something that works. Then it copies the entire
* file line by line looking for the specified section. Once found
* it adds the new Line as the first item of the specified section.
*
* Pass in the section that you expect to find Item in WITHOUT the
* braces. IE for [386Enh] pass in 386Enh.
*
* Finally pass in the ENTIRE line to be added. This means
* you must pass in the WHOLE line as in: device=c:\windows\test.386
*)
function TEIniFile.AddItem(
string;
const
Section
:
const
Line
: string
) : boolean;
var
    iniFile : text;
    outFile : text;
    iniName : string;
    outName : string;
    savName : string;
    iniItem : string;
    curItem : string;
    curSect : string;
    theSect : string;
    i : integer;
    AddCnt : integer;
begin
    try
        { Assume no additions }
        Result := false;
        AddCnt := 0;

        { Convert searched for items to upper case }
        theSect := UpperCase( Section );

        { Get ini file path and name }
        iniName := GetIniPathName;
        outName := iniName;
        savName := GetBackupIniPathName( iniName );

        { Rename the input file }
        RenameFile( iniName, savName );

        { Open input file }
        AssignFile( iniFile, savName );
        Reset( iniFile );

        { Open the output file }
        AssignFile( outFile, outName );
        Rewrite( outFile );

        repeat
            { Read a line }
            readln( iniFile, iniItem );

            { Upper case it }
            curItem := UpperCase(iniItem);

            { See if this is a section }
            if (Pos('[',iniItem)>0) and (Pos(']',iniItem)>0) then

```

```

begin
  { Save it - Uppercase for later compare }
  curSect := curItem;

  { Remove left brace }
  i := Pos('[',curSect);
  Delete( curSect, i, 1 );

  { Remove right brace }
  i := Pos(']',curSect);
  Delete( curSect, i, 1 );

  { Write it to the output file }
  writeln( outFile, iniItem );
end
else
begin
  { See if this is the section they wanted }
  if (curSect=theSect) and (AddCnt=0) then
  begin
    { Ok output our newline }
    writeln( outFile, Line );

    { Count it }
    inc(AddCnt);
  end;

  { Output the line we read }
  writeln( outFile, iniItem );
end;
until eof(iniFile);

finally
  CloseFile( outFile );
  CloseFile( iniFile );
end;

{ Return status }
Result := (AddCnt > 0);
end;

(*
* Replace a specific item in a .INI file.
*
* This function is useful for that wierd section of SYSTEM.INI
* where there are multiple lines like: device=...
*
* The standard WriteString member of this class will ADD a new
* line but it will not allow a replace.
*
* This function will allow a replace. It first renames the
* .INI to .0 or something that works. Then it copies the entire
* file line by line looking for the specified line. Once found
* it substitutes Line for the found one.
*
* To make this function work correctly pass in the item you want
* to replace in 'Item'. This can be a substring but be careful to
* make it unique.
*
* Pass in the section that you expect to find Item in WITHOUT the
* braces. IE for [386Enh] pass in 386Enh.
*
* Finally pass in the ENTIRE line to replace it with. This means
* you must pass in the WHOLE line as in: device=c:\windows\test.386
*)
function TIniFile.ReplaceItem( const Item : string;
                               const Section :
string;

```

```

                                const           Line           : string
) : boolean;
var
    iniFile : text;
    outFile : text;

    iniName  : string;
    outName  : string;
    savName  : string;

    iniItem  : string;
    curItem  : string;
    theItem  : string;
    curSect  : string;
    theSect  : string;

    1                : integer;
    RepCnt : integer;

begin
    try
        { Assume no replacements }
        Result      := false;
        RepCnt      := 0;

        { Convert searched for items to upper case }
        theItem      := UpperCase( Item );
        theSect      := UpperCase( Section );

        { Get ini file path and name }
        iniName      := GetIniPathName;
        outName      := iniName;
        savName      := GetBackupIniPathName( iniName );

        { Rename the input file }
        RenameFile( iniName, savName );

        { Open input file }
        AssignFile( iniFile, savName );
        Reset( iniFile );

        { Open the output file }
        AssignFile( outFile, outName );
        Rewrite( outFile );

        repeat
            { Read a line }
            readln( iniFile, iniItem );

            { Upper case it }
            curItem  := UpperCase(iniItem);

            { See if this is a section }
            if (Pos('[',iniItem)>0) and (Pos(']',iniItem)>0) then
                begin
                    { Save it - Uppercase for later compare }
                    curSect := curItem;

                    { Remove left brace }
                    i      := Pos('[',curSect);
                    Delete( curSect, i, 1 );

                    { Remove right brace }
                    i      := Pos(']',curSect);
                    Delete( curSect, i, 1 );

                    { Write it to the output file }
                    writeln( outFile, iniItem );
                    end
                else
                    begin
                        { Search for our item in this line }

```

```

        if Pos(theItem,curItem) > 0 then
        begin
            { See if this is the section they
wanted }
            if (curSect=theSect) and (RepCnt=0) then
            begin
                { Ok output our newline if anything there }
                if length(Line) > 0 then
                    writeln( outFile, Line );

                { Count it }
                inc(RepCnt);
            end
            else
                writeln( outFile, iniItem );
            end
            else
                writeln( outFile, iniItem );
            end;

            until eof(iniFile);

            { Set return }
            Result := (RepCnt > 0);

            finally
                CloseFile( outFile );
                CloseFile( iniFile );
            end;
        end;

(*
* Delete a specific item from a .INI file.
*
* This function is useful for that wierd section of SYSTEM.INI
* where there are multiple lines like: device=...
*
* This function will delete an item. It first renames the
* .INI to .0 or something that works. Then it copies the entire
* file line by line looking for the specified line. Once found
* it simply does not copy it to the new .INI.
*
* To make this function work correctly pass in the item you want
* to delete in 'Item'. This can be a substring but be careful to
* make it unique.
*
* Pass in the section that you expect to find Item in WITHOUT the
* braces. IE for [386Enh] pass in 386Enh.
*)
function TEIniFile.DeleteItem(      const      Item      : string;
                                   const      Section : string ) : boolean;
begin
    { Call ReplaceItem with a zero length string to do the work }
    Result := ReplaceItem( Item, Section, '' );
end;

end.

```

10. ELEMENT.PAS Source Code

This is the source code file for the "Order Element" module:

```

unit Element;

interface

```



```

uses WinTypes, WinProcs, Classes, Graphics, Forms, Controls, Buttons,
    StdCtrls, ExtCtrls, DBCtrls, Mask, DB, Wwdatsrc, DBTables, Wwtable;

type
    TfrmElement = class(TForm)
        OKBtn: TBitBtn;
        CancelBtn: TBitBtn;
        HelpBtn: TBitBtn;
        Bevel1: TBevel;
        tblElement: TwwTable;
        dtsElement: TwwDataSource;
        dbnElement: TDBNavigator;
        dbmElement: TDBMemo;
        procedure FormShow(Sender: TObject);
        procedure OKBtnClick(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    frmElement: TfrmElement;

implementation

uses MainForm, Order;

{$R *.DFM}

procedure TfrmElement.FormShow(Sender: TObject);
var
    ChildPointer : TfrmOrderView;
begin
    ChildPointer := frmMain.GetCurrentChildPointer(Self);
    with tblElement do
        begin
            filter.clear;
            filter.add('MISSION_AREA = ' + ChildPointer.tblMissions.FieldByName('AREA').AsString) ;
            filter.Activate;
        end;
    end;

    procedure TfrmElement.OKBtnClick(Sender: TObject);
    var
        ChildPointer : TfrmOrderView;
    begin
        ChildPointer := frmMain.GetCurrentChildPointer(Self);
        ChildPointer.AddTaskElement(Self, tblElement.FieldByName('ELEM_TEXT').AsString);
    end;

end.

```

11. MAINFORM.PAS Source Code

This is the source code file for the “Main Framing Form” main program:

```

unit MainForm;

interface

uses WinTypes, WinProcs, Classes, Graphics, Controls, Printers,
    Menus, Dialogs, Forms, ExtCtrls, Buttons, Order, Clipbrd, StdCtrls,
    DBTables, About, EditUnit, AddUnit, SysUtils, DB, Wwtable;

type
    TfrmMain = class(TForm)

```

```

SpeedBar: TPanel;
SpeedButton1: TSpeedButton;
SpeedButton2: TSpeedButton;
SpeedButton3: TSpeedButton;
spdCut: TSpeedButton;
spdCopy: TSpeedButton;
spdPaste: TSpeedButton;
spdUndo: TSpeedButton;
SpeedButton8: TSpeedButton;
MainMenu1: TMainMenu;
File1: TMenuItem;
Exit1: TMenuItem;
N1: TMenuItem;
Open1: TMenuItem;
New1: TMenuItem;
Help1: TMenuItem;
About1: TMenuItem;
HowtoUseHelp1: TMenuItem;
SearchforHelpOn1: TMenuItem;
Contents1: TMenuItem;
N7: TMenuItem;
Window1: TMenuItem;
Cascade1: TMenuItem;
Tile1: TMenuItem;
GenerateanOutgoingOrder1: TMenuItem;
PrintSetup1: TMenuItem;
Print1: TMenuItem;
N2: TMenuItem;
batmovInOutTO: TBatchMove;
txtOrderCount: TLabel;
tblTOBase: TWinTable;
tblTOBaseUNIT_CODE: TSmallintField;
tblTOBaseUNIT_TEXT: TStringField;
tblTOBaseUNIT_CMDR: TStringField;
tblTOBaseREPORTS TO: TSmallintField;
tblTOBaseRESERVE: TBooleanField;
tblTOBaseTASK: TMemoField;
procedure NewChild(Sender: TObject);
procedure OpenChild(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure Tile1Click(Sender: TObject);
procedure Cascade1Click(Sender: TObject);
procedure About1Click(Sender: TObject);
procedure New1Click(Sender: TObject);
procedure Open1Click(Sender: TObject);
procedure Cut1Click(Sender: TObject);
procedure GenerateanOutgoingOrder1Click(Sender: TObject);
procedure GenerateTO(Sender: TObject);
procedure LoadTOEditdtasrc(Sender: TObject);
procedure LoadTOAdddtasrc(Sender: TObject);
procedure spdCutClick(Sender: TObject);
procedure spdCopyClick(Sender: TObject);
procedure spdPasteClick(Sender: TObject);
procedure spdUndoClick(Sender: TObject);
function GetCurrentChildPointer(Sender: TObject) : TfrmOrderView;
public
    WhichUnitAmIText, WhichUnitAmICode : String;
end;

var
    frmMain: TfrmMain;
    NewFlag: Boolean;
    temp : Tform;

implementation
Uses Defaults;

{$R *.DFM}

var
    IncomingOrder, OutgoingOrder : TfrmOrderView;

```

```

procedure TfrmMain.NewChild(Sender: TObject);
var
  OrderForm: TfrmOrderView;
begin
  NewFlag := True;
  OrderForm := TfrmOrderView.Create(Application);
  OrderForm.Show;
end;

procedure TfrmMain.OpenChild(Sender: TObject);
var
  OrderForm: TfrmOrderView;
begin
  NewFlag := False;
  OrderForm := TfrmOrderView.Create(Application);
  OrderForm.Caption := OrderForm.tblOrders.FieldByName('ORDER_NAME').AsString;
  OrderForm.Show;
end;

procedure TfrmMain.Exit1Click(Sender: TObject);
begin
  Close;
end;

procedure TfrmMain.Tile1Click(Sender: TObject);
begin
  Tile;
end;

procedure TfrmMain.Cascade1Click(Sender: TObject);
begin
  Cascade;
end;

procedure TfrmMain.About1Click(Sender: TObject);
begin
  frmAbout.ShowModal;
end;

procedure TfrmMain.New1Click(Sender: TObject);
begin
  NewChild(Self);
end;

procedure TfrmMain.Open1Click(Sender: TObject);
begin
  OpenChild(Self);
end;

procedure TfrmMain.Cut1Click(Sender: TObject);
begin
  {ActiveControl.CutToClipboard;}
end;

procedure TfrmMain.GenerateanOutgoingOrder1Click(Sender: TObject);
var
  rtnMsg,DTGString : String;
  memoStream : TStream;
begin
  TForm(IncomingOrder) := ActiveMDIChild;

  If Not (IncomingOrder = Nil) then
  Begin
    If Not IncomingOrder.SearchForWhichUnitIAM(Self) then
    Begin
      IncomingOrder.csNotebook1.ActivePage := 'Task Organization';
      ShowMessage('The Unit you have identified yourself as is not in this T/O');
    End
    Else
    Begin

```

```

        NewChild(Self);
        TForm(OutgoingOrder) := ActiveMDIChild;
        GenerateTO(Self);

        OutgoingOrder.tblOrders.FieldByName('ORDER_ID').AsString :=
IntToStr(IncomingOrder.tblOrders.RecordCount + 1);

        OutgoingOrder.tblOrders.FieldByName('MISSION_ID').Assign(IncomingOrder.tblOrders.FieldByName(
e('MISSION_ID')));
        OutgoingOrder.tblOrders.FieldByName('INIT_UNIT').AsString := WhichUnitAmIText;

        OutgoingOrder.tblOrders.FieldByName('LOCATION').Assign(IncomingOrder.tblOrders.FieldByName(
'LOCATION'));

        {Clean up DTG Text--maybe add a Calendar/Time Pick Box and Zulu Time Option in
Defaults}
        DTGString := UpperCase(FormatDateTime('ddhhnnmm' 'yy',Now));
        Insert(IncomingOrder.tblOrders.FieldByName('TIMEZONE').AsString, DTGString, 7);

        OutgoingOrder.tblOrders.FieldByName('DTG').AsString := DTGString;

        OutgoingOrder.tblOrders.FieldByName('SERIAL_NO').Assign(IncomingOrder.tblOrders.FieldByName(
('SERIAL_NO')));

        OutgoingOrder.tblOrders.FieldByName('ORDER_NAME').Assign(IncomingOrder.tblOrders.FieldByName(
e('ORDER_NAME')));

        OutgoingOrder.tblOrders.FieldByName('REFERENCE').Assign(IncomingOrder.tblOrders.FieldByName(
('REFERENCE')));

        OutgoingOrder.tblOrders.FieldByName('TIMEZONE').Assign(IncomingOrder.tblOrders.FieldByName(
'TIMEZONE'));

        OutgoingOrder.tblOrders.FieldByName('ORIENT').Assign(IncomingOrder.tblOrders.FieldByName('O
RIENT'));

        OutgoingOrder.tblOrders.FieldByName('PARA1A').Assign(IncomingOrder.tblOrders.FieldByName('P
ARA1A'));

        OutgoingOrder.tblOrders.FieldByName('PARA1B').Assign(IncomingOrder.tblOrders.FieldByName('P
ARA1B'));

        OutgoingOrder.tblOrders.FieldByName('PARA1C1B').Assign(IncomingOrder.tblOrders.FieldByName(
'PARA3A1'));

        OutgoingOrder.tblOrders.FieldByName('PARA1C1C').Assign(IncomingOrder.tblOrders.FieldByName(
'PARA3A2A'));

        OutgoingOrder.tblOrders.FieldByName('PARA1C3').Assign(IncomingOrder.tblOrders.FieldByName('
PARA1C3'));
        OutgoingOrder.memoPARA1C3.Lines.Add(' ');
        OutgoingOrder.memoPARA1C3.Lines.AddStrings(IncomingOrder.memoPARA3A2B.Lines);

        OutgoingOrder.tblOrders.FieldByName('PARA1E').Assign(IncomingOrder.tblOrders.FieldByName('P
ARA1E'));

        OutgoingOrder.tblOrders.FieldByName('PARA3D').Assign(IncomingOrder.tblOrders.FieldByName('P
ARA3D'));

        OutgoingOrder.tblOrders.FieldByName('PARA4').Assign(IncomingOrder.tblOrders.FieldByName('PA
RA4'));

        OutgoingOrder.tblOrders.FieldByName('PARA5A').Assign(IncomingOrder.tblOrders.FieldByName('P
ARA5A'));

        OutgoingOrder.tblOrders.FieldByName('PARA5B').Assign(IncomingOrder.tblOrders.FieldByName('P
ARA5B'));

        OutgoingOrder.tblOrders.FieldByName('PARA5C').Assign(IncomingOrder.tblOrders.FieldByName('P
ARA5C'));

```

```

        OutgoingOrder.DBOutline1.LoadFromDataSet;
    {
        if OutgoingOrder.DBOutline1.ItemCount > 1 then
            OutgoingOrder.DBOutline1.Items[1].Expand; }
    End;
End
Else
    ShowMessage('Cannot generate without an incoming order being open');
end;

procedure TfrmMain.GenerateTO(Sender: TObject);
Var
    rtnMsg : String;
    intMergedTOUnitOffset : Integer;
begin
    {Copy incoming TO to the outgoing TO in its entirety}
    batmovInOutTO.Source := IncomingOrder.tblTO;
    batmovInOutTO.Destination := OutgoingOrder.tblTO;
    batmovInOutTO.Mode := batAppend;
    batmovInOutTO.Execute;

    {Delete all Nodes which aren't my unit}
    With OutgoingOrder.tblTO Do
    Begin
        FindKey([WhichUnitAmICode]);
        Edit;
        FieldByName('REPORTS_TO').AsString := '';
        Post;
        First;
        While NOT EOF Do
        Begin
            If Length(FieldByName('REPORTS_TO').AsString) = 0 Then
                If NOT (FieldByName('UNIT_CODE').AsString = WhichUnitAmICode) Then
                Begin
                    OutgoingOrder.DeleteTOTree(Self, FieldByName('UNIT_CODE').AsInteger);
                    First;
                End
            Else
                Next
            Else
                Next;
        End;
        Pack(rtnMsg);
        Last;
        intMergedTOUnitOffset := FieldByName('UNIT_CODE').AsInteger;
    End;

    { Renumber keyfields of baseline TO to ensure no collisions with outgoing TO}
    With tblTOBase Do
    Begin
        Open;
        Last;
        While NOT BOF Do
        Begin
            Edit;
            FieldByName('UNIT_CODE').AsInteger := FieldByName('UNIT_CODE').AsInteger +
intMergedTOUnitOffset;
            If FieldByName('REPORTS_TO').AsInteger <> 0 Then
                FieldByName('REPORTS_TO').AsInteger := FieldByName('REPORTS_TO').AsInteger +
intMergedTOUnitOffset;
            Post;
            Prior;
        End;
        Close;
    End;

    {Append all baseline TO nodes to the outgoing TO}
    batmovInOutTO.Source := tblTOBase;
    batmovInOutTO.Destination := OutgoingOrder.tblTO;
    batmovInOutTO.Mode := batAppend;
    batmovInOutTO.Execute;

```

```

{Renummer baseline TO to minimal Unit Code key fields}
With tblTOBase Do
Begin
  Open;
  First;
  While NOT EOF Do
  Begin
    Edit;
    FieldByName('UNIT_CODE').AsInteger := FieldByName('UNIT_CODE').AsInteger -
intMergedTOUnitOffset;
    If FieldByName('REPORTS_TO').AsInteger <> 0 Then
      FieldByName('REPORTS_TO').AsInteger := FieldByName('REPORTS_TO').AsInteger -
intMergedTOUnitOffset;
    Post;
    Next;
  End;
  Close;
End;

{Load Outline component from outgoing TO dataset}
OutgoingOrder.DBOutline1.LoadFromDataSet;
{ if OutgoingOrder.DBOutline1.ItemCount > 1 then
  OutgoingOrder.DBOutline1.Items[1].Expand;}

end;

procedure TfrmMain.LoadTOEditdtasrc(Sender: TObject);
var
  CurrentOrder : TfrmOrderView;
begin
  TForm(CurrentOrder) := ActiveMDIChild;
  frmEditUnit.dtasrcTOEdit.DataSet := CurrentOrder.tblTO;
end;

procedure TfrmMain.LoadTOAdddtasrc(Sender: TObject);
var
  CurrentOrder : TfrmOrderView;
begin
  TForm(CurrentOrder) := ActiveMDIChild;
  frmAddUnit.dtasrcTOAdd.DataSet := CurrentOrder.tblTO;
end;

procedure TfrmMain.spdCutClick(Sender: TObject);
var
  CurrentOrder : TfrmOrderView;
begin
  TForm(CurrentOrder) := ActiveMDIChild;
  CurrentOrder.Cut1Click(Self);
end;

procedure TfrmMain.spdCopyClick(Sender: TObject);
var
  CurrentOrder : TfrmOrderView;
begin
  TForm(CurrentOrder) := ActiveMDIChild;
  CurrentOrder.Copy1Click(Self);
end;

procedure TfrmMain.spdPasteClick(Sender: TObject);
var
  CurrentOrder : TfrmOrderView;
begin
  TForm(CurrentOrder) := ActiveMDIChild;
  CurrentOrder.Paste1Click(Self);
end;

procedure TfrmMain.spdUndoClick(Sender: TObject);
var
  CurrentOrder : TfrmOrderView;
begin

```

```

    TForm(CurrentOrder) := ActiveMDIChild;
    CurrentOrder.Undo1Click(Self);
end;

function TfrmMain.GetCurrentChildPointer(Sender: TObject) : TfrmOrderView;
var
    tempPointer : TfrmOrderView;
begin
    TForm(tempPointer) := ActiveMDIChild;
    GetCurrentChildPointer := tempPointer;
end;

end.

```

12. ORDER.PAS Source Code

This is the source code file for the “Order Viewing” module:

```

unit Order;

interface

uses
    SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
    Forms, Dialogs, StdCtrls, DBCtrls, Mask, CSNoteBk, ExtCtrls, DB, DBTables,
    DBLookup, Grids, Outline, Menus, DBOutln, Wwdatsrc, Wwtable, wwdblook,
    Wwquery, XceedZip;

type
    TfrmOrderView = class(TForm)
        Panel1: TPanel;
        DBNavigator: TDBNavigator;
        csNotebook1: TcsNotebook;
        Label1: TLabel;
        EditORDER_ID: TDBEdit;
        Label2: TLabel;
        Label4: TLabel;
        edtINIT_UNIT: TDBEdit;
        Label5: TLabel;
        EditLOCATION: TDBEdit;
        EditDTG: TDBEdit;
        Label7: TLabel;
        EditSERIAL_NO: TDBEdit;
        Label8: TLabel;
        Label9: TLabel;
        MemoREFERENCE: TDBMemo;
        Label10: TLabel;
        EditTIMEZONE: TDBEdit;
        MemoORIENT: TDBMemo;
        ScrollBox1: TScrollBox;
        MemoPARAE: TDBMemo;
        ScrollBox2: TScrollBox;
        MemoPARA2: TDBMemo;
        ScrollBox3: TScrollBox;
        Label30: TLabel;
        MemoPARAA3: TDBMemo;
        MemoPARAB3: TDBMemo;
        MemoPARAC4: TDBMemo;
        DBEditOrderName: TDBEdit;
        DBLookupCombo1: TDBLookupCombo;
        Label3: TLabel;
        Label25: TLabel;
        Label31: TLabel;
        Label32: TLabel;
        MainMenu1: TMainMenu;
        Edit1: TMenuItem;
        Undo1: TMenuItem;
    end;

```

```

Repeatcommand1: TMenuItem;
N5: TMenuItem;
Cut1: TMenuItem;
Copy1: TMenuItem;
Paste1: TMenuItem;
N4: TMenuItem;
Find1: TMenuItem;
Replace1: TMenuItem;
GoTo1: TMenuItem;
N3: TMenuItem;
Object1: TMenuItem;
XmitRcv1: TMenuItem;
ReviewOrderStatus1: TMenuItem;
ReceiveOrders1: TMenuItem;
AutoSelect1: TMenuItem;
RS232Serial1: TMenuItem;
InfraRed1: TMenuItem;
EthernetLANwired1: TMenuItem;
EthernetLANwireless1: TMenuItem;
SINGARSRadiol: TMenuItem;
N2WayPaging1: TMenuItem;
CellularPhone1: TMenuItem;
SatelliteLEO1: TMenuItem;
TransmitOrders1: TMenuItem;
AutoSelect2: TMenuItem;
RS232Serial2: TMenuItem;
InfraRed2: TMenuItem;
EthernetLANwired2: TMenuItem;
EthernetLANwireless2: TMenuItem;
RadioTCIM1: TMenuItem;
N2WayPaging2: TMenuItem;
CellularPhone2: TMenuItem;
SatelliteLEO2: TMenuItem;
DBOutline1: TDBOutline;
pmnuUnits: TPopupMenu;
pmnuitemAddChild: TMenuItem;
pmnuitemAddSibling: TMenuItem;
pmnuitemSeparator1: TMenuItem;
pmnuitemEdit: TMenuItem;
pmnuitemSeparator2: TMenuItem;
pmnuitemDelete: TMenuItem;
batmovCopyTO: TBatchMove;
tblEmptyTO: TTable;
tblOrders: TwwTable;
dtasrcOrders: TwwDataSource;
tblMissions: TwwTable;
dtasrcMissions: TwwDataSource;
tblTO: TwwTable;
dtasrcTO: TwwDataSource;
Label33: TLabel;
Label6: TLabel;
dbmemoTask: TDBMemo;
memoMission: TMemo;
pnlTOWhoAmI: TPanel;
txtF10Warning: TLabel;
dmmTask: TDBMemo;
lblTask: TLabel;
lblTO: TLabel;
mnuAssistants: TMenuItem;
mnuCheckList: TMenuItem;
mnuOrderElements: TMenuItem;
zipOutgoingOrder: TXceedZip;
qryOutgoingOrder: TwwQuery;
bmVOutgoingOrder: TBatchMove;
tblOutgoingOrder: TwwTable;
csnSituation: TcsNotebook;
MemoPARAB: TDBMemo;
MemoPARAA: TDBMemo;
Label12: TLabel;
pnlSituation: TPanel;
lblSituation: TLabel;

```



```

DBMemo1: TDBMemo;
MemoPARACA: TDBMemo;
MemoPARACB: TDBMemo;
MemoPARACC: TDBMemo;
MemoPARAC: TDBMemo;
memoPARA1C3: TDBMemo;
MemoPARAD: TDBMemo;
MemoPARAD2: TDBMemo;
csnExecution: TcsNotebook;
MemoPARAA2: TDBMemo;
MemoPARAAA: TDBMemo;
memoPARAA2B: TDBMemo;
mmoTasks: TMemo;
mmoReserve: TMemo;
MemoPARAD3: TDBMemo;
Panel2: TPanel;
lblExecution: TLabel;

procedure FormCreate(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure New1Click(Sender: TObject);
procedure Tile1Click(Sender: TObject);
procedure Cascade1Click(Sender: TObject);
procedure Open1Click(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure DBNavigatorClick(Sender: TObject; Button: TNavigateBtn);
procedure DBOutline1DragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
procedure DBOutline1DragDrop(Sender, Source: TObject; X, Y: Integer);
procedure DBOutline1AutoDragDrop(Sender: TObject; var Accept: Boolean;
    var FromNode, ToNode: OpenString);
procedure DataSource3DataChange(Sender: TObject; Field: TField);
procedure DBOutline1KeyDown(Sender: TObject; var Key: Word;
    Shift: TShiftState);
procedure pmnuitemAddChildClick(Sender: TObject);
procedure pmnuitemDeleteClick(Sender: TObject);
procedure pmnuitemEditClick(Sender: TObject);
procedure DeleteTOTree(Sender: TObject; NodeNum : Integer);
function AnyNodesThatReportToThisNode(Sender: TObject; NodeNum : Integer): Boolean;
procedure Object1Click(Sender: TObject);
procedure csNotebook1Click(Sender: TObject);
procedure Cut1Click(Sender: TObject);
procedure Copy1Click(Sender: TObject);
procedure Paste1Click(Sender: TObject);
procedure Edit1Click(Sender: TObject);
procedure Undo1Click(Sender: TObject);
function SearchForWhichUnitIAM(Sender: TObject) : Boolean;
procedure mnuCheckListClick(Sender: TObject);
procedure mnuOrderElementsClick(Sender: TObject);
procedure AutoSelect2Click(Sender: TObject);
procedure ZiptheOrder(Sender: TObject);
procedure AddTaskElement(Sender: TObject; strElement: String);
procedure csnSituationPageChanged(Sender: TObject);
procedure memoMissionClick(Sender: TObject);
procedure csnExecutionPageChanged(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    frmOrderView: TfrmOrderView;
    StartSynching: Boolean;

implementation
uses MainForm, AddUnit, EditUnit, Defaults, ClipStuf, ChkList, Element;

{$R *.DFM}

procedure TfrmOrderView.FormCreate(Sender: TObject);

```

```

var
  newOrder_ID : Integer;
  DTGString : String;
begin
{ Application.OnIdle := AppIdle; Move this to Frame Form for buttons enabling for cut/paste
}
  StartSynching := false;
  tblOrders.Open;
  tblOrders.Last;
  tblMissions.Open;

  If NewFlag then
  begin
    tblOrders.Last;
    newOrder_ID := tblOrders.FieldByName('ORDER_ID').AsInteger + 1;
    tblOrders.Insert;
    tblOrders.FieldByName('ORDER_ID').AsInteger := newOrder_ID;

    { make a new TO file}
    tblTO.TableName := 'TO' + IntToStr(newOrder_ID) + '.DB';
    batmovCopyTO.Source := tblEmptyTO;
    batmovCopyTO.Destination := tblTO;
    batmovCopyTO.Mode := batCopy;
    batmovCopyTO.Execute;
    tblTO.AddIndex('UNIT_CODE', 'UNIT_CODE', [ixPrimary]);

    frmMain.txtOrderCount.Caption := 'NEW ORDER';
    DTGString := UpperCase(FormatDateTime('ddhhnnmm' 'yy', Now));
    { add read TimeZone from defaults }
    Insert(tblOrders.FieldByName('TIMEZONE').AsString, DTGString, 7);
    tblOrders.FieldByName('DTG').AsString := DTGString;
    tblOrders.FieldByName('INIT_UNIT').AsString := frmDefaults.edtWhoAmI.Text;
  end
  else
  begin
    tblTO.TableName := 'TO' + tblOrders.FieldByName('ORDER_ID').AsString + '.DB';
    frmMain.txtOrderCount.Caption := 'ORDER #' + tblOrders.FieldByName('ORDER_ID').AsString
+ ' of ' +
    IntToStr(tblOrders.RecordCount);
  end;

  tblTO.Open;
  csNotebook1.ActivePage := 'General Info';
  WindowState := wsMaximized;
  DBOutline1.LoadFromDataSet;
  StartSynching := true;

  { if DBOutline1.ItemCount > 1 then
    DBOutline1.Items[1].Expand; }

  {Search the TO to get a WhoAmI Match}
  If NOT NewFlag Then
    if SearchForWhichUnitIAM(Self) then
      txtFlOWarning.Caption := 'You have identified yourself as ' +
frmMain.WhichUnitAmIText;
      tblTO.First;
    end;

  procedure TfrmOrderView.FormClose(Sender: TObject; var Action: TCloseAction);
  begin
    Action := caFree;
  end;

  procedure TfrmOrderView.NewClick(Sender: TObject);
  begin
    frmMain.NewChild (Self);
  end;

  procedure TfrmOrderView.TileClick(Sender: TObject);
  begin

```

```

    frmMain.Tile1Click (Self);
end;

procedure TfrmOrderView.Cascade1Click(Sender: TObject);
begin
    frmMain.Cascade1Click (Self);
end;

procedure TfrmOrderView.Open1Click(Sender: TObject);
begin
    frmMain.OpenChild (Self);
end;

procedure TfrmOrderView.Exit1Click(Sender: TObject);
begin
    Close;
end;

procedure TfrmOrderView.DBNavigatorClick(Sender: TObject;
    Button: TNavigateBtn);
    var rtnMsg: String;
begin
    { Add Case logic for efficiency
      case Button of
        nbFirst : BtnName := 'nbFirst';
        nbPrior : BtnName := 'nbPrior';
        nbNext  : BtnName := 'nbNext';
        nbLast  : BtnName := 'nbLast';
        nbInsert: BtnName := 'nbInsert';
        nbDelete: BtnName := 'nbDelete';
        nbEdit  : BtnName := 'nbEdit';
        nbPost  : BtnName := 'nbPost';
        nbCancel: BtnName := 'nbCancel';
        nbRefresh: BtnName := 'nbRefresh'; }

    { Change delete message to "Okay to Delete Order"}

    Caption := tblOrders.FieldByName('ORDER_NAME').AsString;
    frmMain.txtOrderCount.Caption := 'ORDER #' + tblOrders.FieldByName('ORDER_ID').AsString
+ ' of ' +
    IntToStr(tblOrders.RecordCount);
    tblTO.Close;
    tblTO.TableName := 'TO' + tblOrders.FieldByName('ORDER_ID').AsString + '.DB';
    tblTO.Open;
    DBOutline1.LoadFromDataSet;

    {if DBOutline1.ItemCount > 1 then
      DBOutline1.Items[1].Expand; }
    if SearchForWhichUnitIAM(Self) then
        txtF10Warning.Caption := 'You have identified yourself as ' + frmMain.WhichUnitAmIText
    Else
        txtF10Warning.Caption := 'Click on YOUR Unit and Press <F10>';
    tblTO.First;
end;

procedure TfrmOrderView.DBOutline1DragOver(Sender, Source: TObject; X,
    Y: Integer; State: TDragState; var Accept: Boolean);
begin
    if Source = DBOutline1 then Accept := true;
end;

procedure TfrmOrderView.DBOutline1DragDrop(Sender, Source: TObject; X,
    Y: Integer);
begin
    If Source = DBOutline1 then
        { AutoDrop is the single method call required to perform the move
          of the node and update the database }
        DBOutline1.AutoDrop(x, y);
end;

procedure TfrmOrderView.DBOutline1AutoDragDrop(Sender: TObject;

```

```

    var Accept: Boolean; var FromNode, ToNode: OpenString);
begin
    { this code overrides the default confirmation message presented when
      a node is dragged-and-dropped, replacing it with the custom message.
      This ability gives the programmer control over the confirmation
      message presented to the user.}
    if MessageDlg(' Have unit ['+FromNode+'] be task organized under ['+ToNode+ ']?'
      ,mtConfirmation, mbOkCancel,0) = mrOk
    then Accept := true
    else Accept := false
end;

procedure TfrmOrderView.DataSource3DataChange(Sender: TObject; Field: TField);
begin
    if StartSynching then
        DBOutline1.SynchOutline(Self);
end;

procedure TfrmOrderView.DBOutline1KeyDown(Sender: TObject; var Key: Word;
    Shift: TShiftState);
var
    txtUnitName, txtUnitNumber : String;
begin
    if Key = VK_F10 then
    begin
        txtUnitName := tblTO.FieldByName('UNIT_TEXT').AsString;
        txtUnitNumber := tblTO.FieldByName('UNIT_CODE').AsString;
        txtF10Warning.Caption := 'You have identified yourself as ' + txtUnitName;
        frmMain.WhichUnitAmIText := txtUnitName;
        frmMain.WhichUnitAmIcode := txtUnitNumber;
        frmDefaults.edtWhoAmI.Text := txtUnitName;
        frmDefaults.OKBtnClick(Self);
    end;
end;

procedure TfrmOrderView.pmnuitemAddChildClick(Sender: TObject);
var
    intParentUnitNo, intNewUnitNo: integer;
begin
    StartSynching := false;

    { if popup menu option 'Add Child' was selected}
    if Sender = pmnuitemAddChild then
    begin
        intParentUnitNo := tblTO.FieldByName('UNIT_CODE').AsInteger;
        frmAddUnit.Caption := 'Add a Unit UNDER ' + tblTO.FieldByName('UNIT_TEXT').AsString;
    end
    { else if popup menu option 'Add Sibling' was selected}
    else
    begin
        intParentUnitNo := tblTO.FieldByName('REPORTS_TO').AsInteger;
        frmAddUnit.Caption := 'Add a Unit AT THE SAME LEVEL as ' +
tblTO.FieldByName('UNIT_TEXT').AsString;;
    end;

    {move to end of table (indexed on EmpNo) and increment last EmpNo by one.}
    tblTO.Last;
    intNewUnitNo := tblTO.FieldByName('UNIT_CODE').AsInteger + 1;
    tblTO.Append;
    tblTO.FieldByName('RESERVE').AsBoolean := False;

    { if parent employee number is not 0, i.e. if you are not adding a peer
      to a top-level parent }
    if intParentUnitNo <> 0 then
        tblTO.FieldByName('REPORTS_TO').AsInteger := intParentUnitNo;

    tblTO.FieldByName('UNIT_CODE').AsInteger := intNewUnitNo;

    { ActiveMDIChild must be called from the MDI Form}
    frmMain.LoadTOAdddtasrc(Self);
    frmAddUnit.ShowModal;

```

```

{ if OK button pressed to add to employee }
if frmAddUnit.ModalResult=mrOK then
begin
  if (tblTO.State = dsEdit) or (tblTO.State = dsInsert) then
  begin
    { Add new employee to database }
    tblTO.Post;
    { TDBOutline method to add current record (in this case, the new employee)
    to the existing outline }
    with tblTO do
      DBOutline1.AddDBRecord(FieldByName('UNIT_CODE').AsString,
        FieldByName('UNIT_TEXT').AsString, FieldByName('REPORTS_TO').AsString);
    end;
  end
  else tblTO.Cancel;

  StartSynching := true;
end;

procedure TfrmOrderView.pmnuitemDeleteClick(Sender: TObject);
var
  rtnMsg: String;
begin
  StartSynching := false;

  if messagedlg('Remove the Unit "' + tblTO.FieldByName('UNIT_TEXT').AsString +
    '" from the T/O (and ALL of its subordinates)?',mtconfirmation,mbOKCancel,0) = mrOK
  then
  begin
    { !! be sure to delete the Table record first!! If you delete the Outline node
    first, then DataAutoSynch will move the datapointer to match the new
    DBOutline.SelectedItem, and you'll end up deleting the wrong record! }
    DeleteTOTree(Self, tblTO.FieldByName('UNIT_CODE').AsInteger);
    DBOutline1.Delete(DBOutline1.SelectedItem);
    tblTO.Pack (rtnMsg);
  end;
  DBOutline1.SetFocus;
  StartSynching := true;
end;

procedure TfrmOrderView.pmnuitemEditClick(Sender: TObject);
begin
  StartSynching := false;
  { ActiveMDIChild must be called from the MDI Form}
  frmMain.LoadTOEditdtasrc(Self);
  frmEditUnit.ShowModal;
  With tblTO Do
  Begin
    { if OK button pressed to add to employee }
    if frmEditUnit.ModalResult=mrOK then
    begin
      if (State = dsEdit) or (State = dsInsert) then
      begin
        Post;
        DBOutline1.ChangeDBRecord(FieldByName('UNIT_CODE').AsString,
FieldByName('UNIT_CODE').AsString,
        FieldByName('UNIT_TEXT').AsString);
      end;
    end
    else Cancel;
  End;
  DBOutline1.SetFocus;
  StartSynching := true;
end;

procedure TfrmOrderView.DeleteTOTree(Sender: TObject; NodeNum : Integer);
Begin
  With tblTO Do
  Begin

```

```

DisableControls;
If (NOT AnyNodesThatReportToThisNode(Self, NodeNum))Then
  Delete
Else
Begin
  First;
  While NOT EOF DO
  Begin
    If (FieldByName('REPORTS_TO').AsInteger = NodeNum) Then
      DeleteTOTree(Self, FieldByName('UNIT_CODE').AsInteger)
    Else
      Next;
  End;
  FindKey([NodeNum]);
  Delete;

End;
EnableControls;
End;
End;

function TfrmOrderView.AnyNodesThatReportToThisNode(Sender: TObject; NodeNum : Integer):
Boolean;
var
  Found : Boolean;
Begin
  {This can be replaced by a FindKey if we can dynamically create secondary indices}
  With tblTO do
  Begin
    Found := False;
    First;
    While (Not EOF AND Not Found) Do
    Begin
      If (FieldByName('REPORTS_TO').AsInteger = NodeNum) Then
        Found := True
      Else
        Next;
    End;
    FindKey([NodeNum]); {Go back to Node we started at}
  End;
  AnyNodesThatReportToThisNode:= Found;
End;

procedure TfrmOrderView.Object1Click(Sender: TObject);
begin
  frmDefaults.ShowModal;
end;

procedure TfrmOrderView.csNotebook1Click(Sender: TObject);
var
  blnFound : Boolean;
  intCurrentRecord : Integer;
begin
  blnFound := False;
  intCurrentRecord := tblTO.FieldByName('UNIT_CODE').AsInteger;

  If csNotebook1.ActivePage = 'Mission' Then
  Begin
    With tblTO Do
    Begin
      First;
      While Not EOF And Not blnFound Do
      Begin
        If (tblOrders.FieldByName('INIT_UNIT').AsString =
FieldByName('UNIT_TEXT').AsString) then
          Begin
            blnFound := True;
            memoMission.Lines := dbmemoTask.Lines;
          End;
        Next;
      End;
    End;
  End;

```

```

End;
If Not blnFound Then
Begin
    memoMission.Lines.Clear;
    csNotebook1.ActivePage := 'General Info';
    ShowMessage('Error--Unit in INITIATING UNIT Field is not in the T/O');
End;
End;
End;

If csNotebook1.ActivePage = 'Execution' Then
Begin
    mmoTasks.Clear;
    mmoReserve.Clear;
    If (Length(tblOrders.FieldByName('INIT_UNIT').AsString) <> 0) then
    Begin
        With tblTO Do
        Begin
            First;
            While Not EOF Do
            Begin
                If (FieldByName('REPORTS_TO').AsString = '') AND NOT
FieldByName('RESERVE').AsBoolean
                AND NOT (FieldByName('UNIT_TEXT').AsString =
tblOrders.FieldByName('INIT_UNIT').AsString) Then
                Begin
                    mmoTasks.Lines.Add(FieldByName('UNIT_TEXT').AsString);
                    mmoTasks.Lines.Add('=====');
                    mmoTasks.Lines.AddStrings(dbmemoTask.Lines);
                    mmoTasks.Lines.Add('');
                End
            Else
                If (FieldByName('REPORTS_TO').AsString = '') AND
FieldByName('RESERVE').AsBoolean
                AND NOT (FieldByName('UNIT_TEXT').AsString =
tblOrders.FieldByName('INIT_UNIT').AsString) Then
                Begin
                    mmoReserve.Lines.Add(FieldByName('UNIT_TEXT').AsString);
                    mmoReserve.Lines.Add('=====');
                    mmoReserve.Lines.AddStrings(dbmemoTask.Lines);
                    mmoReserve.Lines.Add('');
                End;
            Next;
        End;
    End;
    End;
    End
Else
    Begin
        csNotebook1.ActivePage := 'General Info';
        ShowMessage('Error--Unit in INITIATING UNIT Field must be filled out');
    End;
End;

tblTO.FindKey([intCurrentRecord]);
End;
(*
procedure TfrmOrderView.AppIdle(Sender: TObject; var Done: Boolean);
begin
    { use idle processing to enable disable/speed buttons }
    btnCut.Enabled := HasText(Screen.ActiveControl);
    btnCopy.Enabled := HasText(Screen.ActiveControl);
    btnPaste.Enabled:= CanPaste(Screen.ActiveControl);
    btnUndo.Enabled := CanUndo(Screen.ActiveControl);
end;
*)

procedure TfrmOrderView.Cut1Click(Sender: TObject);
begin
    DoEditCut;
end;

```

```

procedure TfrmOrderView.Copy1Click(Sender: TObject);
begin
  DoEditCopy;
end;

procedure TfrmOrderView.Paste1Click(Sender: TObject);
begin
  DoEditPaste;
end;

procedure TfrmOrderView.Edit1Click(Sender: TObject);
begin
  (*
  { enable/disable submenu items when main Edit menu clicked }
  Edit1.Enabled := HasText(Screen.ActiveControl);
  Copy1.Enabled := HasText(Screen.ActiveControl);
  Paste1.Enabled:= CanPaste(Screen.ActiveControl);
  Undo1.Enabled:= CanUndo(Screen.ActiveControl);
  *)
end;

procedure TfrmOrderView.Undo1Click(Sender: TObject);
begin
  DoEditUndo;
end;

function TfrmOrderView.SearchForWhichUnitIAM(Sender: TObject) : Boolean;
Var
  blnFound : Boolean;
  intCurrentRecord : Integer;
begin
  blnFound := False;
  With tblTO do
    Begin
      intCurrentRecord := FieldByName('UNIT_CODE').AsInteger;
      First;
      While (NOT EOF AND NOT blnFound) Do
        Begin
          If FieldByName('UNIT_TEXT').AsString = frmDefaults.edtWhoAmI.Text Then
            Begin
              frmMain.WhichUnitAmIText := FieldByName('UNIT_TEXT').AsString;
              frmMain.WhichUnitAmICode := FieldByName('UNIT_CODE').AsString;
              blnFound := True;
            End;
          Next;
        End;
      If NOT blnFound Then
        Begin
          frmMain.WhichUnitAmIText := '';
          frmMain.WhichUnitAmICode := '';
        End;
      FindKey([intCurrentRecord]);
    End;

    SearchForWhichUnitIAM := blnFound;
  end;

procedure TfrmOrderView.mnuCheckListClick(Sender: TObject);
begin
  frmCheckList.ShowModal;
end;

procedure TfrmOrderView.mnuOrderElementsClick(Sender: TObject);
begin
  frmElement.ShowModal;
end;

procedure TfrmOrderView.AutoSelect2Click(Sender: TObject);
begin
  ZiptheOrder(Self);
  ShowMessage('Zipping Complete--now transmitting via AUTO-SELECT');

```



```

end;

procedure TfrmOrderView.ZiptheOrder(Sender: TObject);
var
  strOutgoingOrderPathName : String;
  strTOFileNumber : String;
begin
  qryOutgoingOrder.ParamByName('Current_Order').AsInteger :=
tblOrders.FieldByName('ORDER_ID').AsInteger;
  qryOutgoingOrder.Open;
  bmvOutgoingOrder.Source := qryOutgoingOrder;
  bmvOutgoingOrder.Destination := tblOutgoingOrder;
  bmvOutgoingOrder.Mode := batCopy;
  bmvOutgoingOrder.Execute;
  tblOutgoingOrder.AddIndex('ORDER_ID', 'ORDER_ID', [ixPrimary]);

  { add outgoing orders pathname from defaults here }
  strOutgoingOrderPathName := 'f:\delphi\redman3\';

  {Set up zip component and zip outgoing order and matching TO}
  strTOFileNumber := tblOrders.FieldByName('ORDER_ID').AsString;
  zipOutgoingOrder.ZipFilename := strOutgoingOrderPathName + strTOFileNumber + '.ZIP';
  zipOutgoingOrder.FilesToProcess.Add(strOutgoingOrderPathName + 'OUT_ORD.*');
  zipOutgoingOrder.FilesToProcess.Add(strOutgoingOrderPathName + 'TO' + strTOFileNumber +
'.*');
  tblTO.Close;
  zipOutgoingOrder.Add(xecAll);
  tblTO.Open;
end;

procedure TfrmOrderView.AddTaskElement(Sender: TObject; strElement: String);
begin
  tblTO.Edit;
  dmmTask.Lines.Add('');
  dmmTask.Lines.Add(strElement);
  tblTO.Post;
end;

procedure TfrmOrderView.csnSituationPageChanged(Sender: TObject);
begin
  if csnSituation.ActivePage = 'General' then
    lblSituation.Caption := 'PARAGRAPH 1A (Situation; General)'
  else if csnSituation.ActivePage = 'Enemy' then
    lblSituation.Caption := 'PARAGRAPH 1B (Situation; Enemy)'
  else if csnSituation.ActivePage = 'Mission' then
    lblSituation.Caption := 'PARAGRAPH 1C1A (Situation; Friendly Higher Mission)'
  else if csnSituation.ActivePage = 'Intent' then
    lblSituation.Caption := 'PARAGRAPH 1C1B (Situation; Friendly Higher Intent)'
  else if csnSituation.ActivePage = 'Scheme' then
    lblSituation.Caption := 'PARAGRAPH 1C1C (Situation; Friendly Higher Scheme of
Maneuver)'
  else if csnSituation.ActivePage = 'Adjacent' then
    lblSituation.Caption := 'PARAGRAPH 1C2 (Situation; Friendly Adjacent)'
  else if csnSituation.ActivePage = 'Support' then
    lblSituation.Caption := 'PARAGRAPH 1C3 (Situation; Friendly Supporting)'
  else if csnSituation.ActivePage = 'Attach' then
    lblSituation.Caption := 'PARAGRAPH 1D1 (Situation; Attachments)'
  else if csnSituation.ActivePage = 'Detach' then
    lblSituation.Caption := 'PARAGRAPH 1D2 (Situation; Detachments)';
end;

procedure TfrmOrderView.memoMissionClick(Sender: TObject);
begin
  MessageDlg('Read Only Field! To edit this field, please go ' +
'to the Task Organization Page and edit the Initiating Unit Mission',
  mtInformation, [mbOK], 0);
  csNoteBook1.ActivePage := 'Task Organization';
end;

procedure TfrmOrderView.csnExecutionPageChanged(Sender: TObject);

```

```

begin
  if csnExecution.ActivePage = 'Intent' then
    lblExecution.Caption := 'PARAGRAPH 3A1 (Execution; Commander'#39's Intent)'
  else if csnExecution.ActivePage = 'Scheme' then
    lblExecution.Caption := 'PARAGRAPH 3A2A (Execution; Scheme of Maneuver)'
  else if csnExecution.ActivePage = 'Support' then
    lblExecution.Caption := 'PARAGRAPH 3A2B (Execution; Fire Support Plan)'
  else if csnExecution.ActivePage = 'Tasks' then
    lblExecution.Caption := 'PARAGRAPH 3B (Execution; Tasks)'
  else if csnExecution.ActivePage = 'Reserve' then
    lblExecution.Caption := 'PARAGRAPH 3C (Execution; Reserve)'
  else if csnExecution.ActivePage = 'Coord' then
    lblExecution.Caption := 'PARAGRAPH 3D (Execution; Coordinating Instructions)'
  end;
end.

```

APPENDIX F. REDWEB HTML AND SOURCE FOR THE SAMPLE ORDER

A. REDWEB HTML SOURCE OUTPUT

The following HTML ASCII source represents the output from the RedWeb HTML Processing application developed in this thesis.

```
<! ***** >
<! * This chunk calls the CHECKPASSWORD MACRO to set the value * >
<! * of the passwordStatus literal appropriately * >
<! ***** >

<! ***** >
<! * This is the DISPLAYORD Page for displaying the Operation * >
<! * Order for authorized users * >
<! ***** >
<! Load all picklists from database values >

<H1><A NAME="Top">Frag Order 5-92 (OPERATION DEEP BANDINI)</A></H1>
<UL>
<LI><A HREF="#General"><B>General Information</B></A>
<LI><A HREF="#Task"><B>Task Organization</B></A>
<LI><A HREF="#Orientation"><B>Orientation</B></A>
<LI><A HREF="#Situation"><B>Situation</B></A>
<LI><A HREF="#Mission"><B>Mission</B></A>
<LI><A HREF="#Execution"><B>Execution</B></A>
<LI><A HREF="#Administration"><B>Administration and Logistics</B></A>
<LI><A HREF="#Command"><B>Command and Signal</B></A>
</UL>
<HR>

<H2><A NAME="General"><font size=5>General Information</font></A></H2>
<UL>
<LI><B>Initiating Unit: </B>2d Bn (-) (Rein), 7th Mar
<LI><B>Mission Type: </B>Movement to Contact
<LI><B>Location: </B>MCAGCC, 29 PALMS
<LI><B>DTG: </B>141800Z OCT 91
<LI><B>Time Zone: </B>T
<LI><B>Serial No: </B>MTR-5
<LI><B>Order Name: </B>Frag Order 5-92 (OPERATION DEEP BANDINI)
<LI><B>Reference: </B><UL><LI>a. Twentynine Palms East/West, Edition 3-DMA, Series V7953
1:50,000</UL>
</UL>
<P>Return to <A HREF="#Top">Top</A></P>
<HR>

<H2><A NAME="Task">Task Organization</A></H2>
<UL><P><LI>2d Bn (-) (Rein), 7th Mar<UL><LI> Det, Comm Plt, HQ Co, RCT-7<LI> Arty Ln Tm, G Btry,
Tm, Det 9th Comm Bn<LI> H&S Co (-)<LI> Wpns Co
3d Bn, 11th Mar<LI> Det, Co D, 3d AAV Bn<LI> Det, H&S Co<UL><LI> Det, Comm<LI>
(-)</UL><P><LI>CAAT<UL><LI> HMG Plt<LI> Det, Wpns Co<UL><LI> 81 FO Tm #1</UL><LI>
Det, Med<LI> FAC Party #1</UL><LI> Det, Wpns Co<UL><LI> OCD #1</UL><P><LI>Team Mech
Arty FO Tm #1<LI> 1st Sqd, 2d Sec, TOW Plt, HQ Co, RCT-7<LI> Det, Comm<LI> Det,
(Co F)<UL><LI> Co F (-)<LI> Det, H&S Co<UL><LI> Det, Comm 2<LI>
Med</UL><LI> Det, Wpns Co<UL><LI> 81 FO Tm #2<LI> 1st/2d Sec, AA
Plt</UL><LI> Arty FO Tm #2<LI> 1st Plt, Co D, 3d Tk Bn<LI> 1st Plt (-), Co C, 3d AAV
Bn<LI> 2d Sqd, 2d Sec, TOW Plt, HQ Co, RCT-7<LI> Tm Tank</UL><P><LI>Team Tank<UL><LI>
Co D (-), 3d Tk Bn<LI> 1st Plt, Co F, 2/7<LI> Det, H&S Co<UL><LI> Det, Comm 2<LI>
Det, Med<LI> FAC Party #2</UL><LI> Det, Wpns Co<UL><LI> 81 FO Tm #3</UL><LI>
Arty FO Tm #3<LI> OCD #2<LI> 1st Sec, Co C, 3d AAV Bn</UL><P><LI>81mm Mortar Plt
(Rein)<UL><LI> Det, H&S Co<UL><LI> Det, Med</UL><LI> 1st Sec, 3d Plt, Co C, 3d
AAV Bn</UL><P><LI>2d Plt (-), Co C, 1st CEB<UL><LI> det, Engr Spt Co, 1st CEB</UL><P><LI>Det, 1st
Sec, 3d LAAD Bn<P><LI>Co G (Rein)<UL><LI> Det, H&S Co<UL><LI> Det, Comm<LI>
```

Det, Med Det, Wpns Co 81 FO Tm #3 3d Sec, AA
Plt Arty FO #3 3d Plt (-), Co C, 3d AAV Bn
<P>Return to Top</P>
<HR>

<H2>Orientation</H2>
This direction is generally North. . .
<P>Return to Top</P>
<HR>

<H2>Situation</H2>

General
Omitted.</P>
Enemy
Wahabian forces in the vicinity of Los Angeles have temporarily halted the advance of allied forces though they have suffered up to 35% casualties. The allied advance will not continue for at least 24 more hours while allied forces wait for their supply lines to catch up with their forward units. The 11th Brigade has taken advantage of the lull in fighting to the West to attempt a push south from Sunshine Peak to Maumee Mine and Emerson Lake in an attempt to disrupt allied east-west supply lines. Though the 11th attack was unsuccessful due largely to allied air superiority, and the enemy has withdrawn the majority of its forces, the 11th Bde has left behind up to one Mech battalion in Maumee Mine and Emerson Lake. These remain-behind forces are likely to establish platoon and company sized defensive positions, with minimal engineer support in an effort to delay any allied counterattack.</P>
Higher Mission
At 0900 on D-Day, 7th MEB conducts a movement to contact to destroy enemy forces within the GCE's zone of action in order to assume a forward defensive posture and deny enemy forces into the GCE's TAOR.</P>
Higher Commander's Intent
Argos Pass is the key to the overall effort in order to ensure the northern movement of GCE forces, and deny enemy access to the high speed avenue of approach of Quackenbush Lake. In order to isolate the battle area, I want to prevent enemy forces from escaping and divulging friendly disposition of forces. Overall, I am buying time and distance against the potential for a subsequent enemy attack. Our efforts will provide an eyes forward advantage and prevent enemy indirect firing platforms from reaching the port of 29 Palms.</P>
Higher Scheme of Maneuver
1/7 attacks to our east in Quackenbush corridor. 3d Tank Bn (-) follows in trace of 1/7 as RCT reserve.</P>
Adjacent Units
1/7 attacks to our east in Quackenbush corridor. 3d Tank Bn (-) follows in trace of 1/7 as RCT reserve.</P>
Supporting Units
a. Artillery. 2/7 has priority of artillery fires in Phase I & II.</P>b. Air. 2/7 has priority of CIFS (10) in Phase I & II. 1/7 has priority of CAS.</P>
Attachments
See T/O.</P>
Detachments
See T/O.</P>

<P>Return to Top</P>
<HR>

<H2>Mission</H2>
2/7 conducts a movement to contact along Axis Jane IOT clear enemy forces in zone within the Emerson Corridor, control Gay's and Maumee Pass, deceive the enemy as to the location of the RCT main effort, and guard the MEB's left flank.
<P>Return to Top</P>
<HR>

<H2>Execution</H2>

Commander's Intent
We are a supporting attack for RCT-7's deliberate attack. It is believed that the enemy is employing a battalion (-) sized mechanized force in the area which we will be conducting our battalion movement. He is expected to delay initially, and then transition into a deliberate defense. Unfortunately, the enemy situation is somewhat vague. The enemy's critical vulnerability, however, is that he has failed to provide adequate logistical sustainability for his forward deployed forces. RCT-7's pre-planned deep and our own close air sorties will significantly deteriorate his will to fight. I want to maximize our reconnaissance and security efforts forward to ensure we commit our main force under the most favorable circumstances possible. We will remain in

tactical column as long as possible to enhance the speed and control for which we can deploy for combat. Unit commanders must be prepared to act boldly to seize the initiative, keep the enemy off balance, and to exploit opportunities as they arise on the battlefield. Our first priority is to protect the column against surprise. Our second is to be prepared to bring our superior combat power to bear against the enemy if he should challenge us. Hence, we will employ our air defense in bounds to ensure the column is protected at critical passage points. Engineers will be employed well forward to facilitate our movement by ensuring they are prepared to rapidly breach any obstacles we may encounter. Armor and anti-armor assets will be distributed throughout the column to provide all-around protection, and facilitate our integrated entry into combat. I intend to rapidly suppress and eliminate any forward enemy forces, develop the situation with my lead maneuver units, and deploy my main body to destroy enemy resistance IOT control Gays and Maumee Pass.

Scheme of Maneuver
2/7 conducts a single-axis movement to contact in tactical column with one CAAT Team and one Team Mech forward, and one Team Tank and one mechanized company back.

Fire Support Plan

The Fire Support Plan is . . .

Tasks

CAAT - Advance guard of tactical column. You are March Serial #1. March Serial Commander. Provide forward and flank security during movement. - BPT clear hastily emplaced obstacles with attached OGD. - BPT transition into screening element of approach march formation. Screen forward of column 2 km to provide forward security for column. - BPT suppress enemy armor and mechanized units and allow Team Mech to maneuver to destroy enemy forward units. - At RP, screen forward to PL BARREL. BPT continue the attack east. - Team Mech (Co F) - FIT of CAAT as lead march unit of March Serial #2. March Serial Commander. - BPT transition into advance guard element of approach march formation. Provide forward security during movement. - BPT maneuver to destroy enemy forward units supported by CAAT. - At RP, occupy BP 25 in hasty defensive positions. BPT continue the attack east towards MEB Obj A. - Team Tank - FIT 81s Plt as fifth march unit of March Serial #2. - BPT transition into lead element of the main body in an approach march formation. Provide forward security of the main body during movement. - BPT deploy for combat and maneuver rapidly to exploit the success of Team Mech forward. - At RP, occupy BP 35 in hasty defensive positions. BPT continue the attack east towards MEB Obj A. - 81mm Mortar Plt (Rein) - FIT of Engineers as fourth march unit of March Serial #2. - BPT establish hasty firing positions ISO march column. POF to CAAT initially. - At RP, establish firing positions at FP 55 in GS of the battalion. POF to Tm Mech. BPT continue the attack east. - 2d Plt (-), Co C, 1st CEB - FIT of the Alpha Command Group as the third march unit of March Serial #2. - BPT to displace additional OGDs forward to breach deliberate obstacle belts. - At RP, occupy an AA at GC-234572 in GS of the battalion. BPT construct hasty obstacles vicinity battalion defensive area. BPT continue the attack east towards MEB Obj A. - Det, 1st Sect, 3d LAAD Bn - Establish firing positions along Axis Jane in support of battalion scheme of maneuver. - Displace forward in bounds to ensure continuous umbrella of anti-air support for the march column.

Reserve

Co G (Rein) - FIT Bravo Command Group as second march unit of March Serial #3. March Serial Commander. Provide rear security march column. - BPT transition into lead element of the rear guard in an approach march formation. - BPT deploy for combat and maneuver rapidly to assume the mission of either maneuver unit forward. - At RP, occupy BP 55 in hasty defensive positions. BPT continue the attack east.

Coordinating Instructions

(1) MOPP level 0 initially. (2) Order of Movement in Battalion Zone: March Serial #1: CAAT. March Serial #2: Team Mech, Alpha Command, Engineers, 81 Plt, Team Tank March Serial #3: Bravo Command, Co G (Rein). (3) Units start engines for movement from assembly areas to SP at 150835Z OCT 91. (4) LAAD priority of protection: - Tm 1 to Team Mech - Tm 2 to Team Tank. (5) LAAD Weapons Control- Yellow (Weapons Tight). (6) SP, RP, Route, Pls, March Objectives, and BPs per Operations Overlay. (7) Designate Air Sentries for movement. (8) All units conduct immediate action drills for ambush situations; disabled vehicle troop and cargo transfer; dismounted close security; and establishment of outpost security during halts. (9) Dispersion between march units: Tactical Column: 250m. Approach March: 500m. Dispersion between march serials: Tactical Column: 500m. Approach March: 1000m. (10) Establish STA/Recon Locations and BPT to control supporting arms to engage enemy forces when sighted at: - STA 1- GC 276439 - STA 2- GC 297548 - Recon 1- GC 325983 - Recon 2- GC 376004

Return to [Top](#)

Administration and Logistics

(1) Ensure accurate crew-served weapon ammunition counts and resupply requests are delivered to the S-4 NLT H-6. (2) Submit casualty replacement requests to S-1 NLT 2100 this evening.

```

<P>Return to <A HREF="#Top">Top</A></P>
<HR>

<H2><A NAME="Command">Command and Signal</A></H2>
<UL>
<LI><B>Command Relationships</B>
<UL><LI>Omitted.</UL></P>
<LI><B>Command</B>
<UL><LI>(1) Alpha command initially second march unit in March Serial #2.</P><LI>(2) Bravo Command
initially lead march unit in March Serial #3.</UL></P>
<LI><B>Signal</B>
<UL><LI>(1) Pass SNOWSTORM on Arty COF and Bn TAC 1 upon receiving indirect fires.</P><LI>(2)
Codewords:<UL><LI> -- STORMY WEATHER: Transition to Approach March Formation</P><LI> -- CLEAR
SKIES: Resume Tactical Column Formation</UL></P>
</UL>
<P>Return to <A HREF="#Top">Top</A></P>

```

B. REDWEB/NETSCAPE RENDERED HTML

The following Netscape (Version 3.0b7) created output shows the rendered representation of the above HTML source.

Frag Order 5-92 (OPERATION DEEP BANDINI)

- General Information
 - Task Organization
 - Orientation
 - Situation
 - Mission
 - Execution
 - Administration and Logistics
 - Command and Signal
-

General Information

- **Initiating Unit:** 2d Bn (-) (Rein), 7th Mar
- **Mission Type:** Movement to Contact
- **Location:** MCAGCC, 29 PALMS
- **DTG:** 141800Z OCT 91
- **Time Zone:** T
- **Serial No:** MTR-5
- **Order Name:** Frag Order 5-92 (OPERATION DEEP BANDINI)
- **Reference:**
 - a. Twentynine Palms East/West, Edition 3-DMA, Series V7953 1:50,000

Return to [Top](#)

Task Organization

- 2d Bn (-) (Rein), 7th Mar
 - Det, Comm Plt, HQ Co, RCT-7
 - Arty Ln Tm, G Btry, 3d Bn, 11th Mar
 - Det, Co D, 3d AAV Bn
 - Tm, Det 9th Comm Bn
 - H&S Co (-)
 - Wpns Co (-)
- CAAT
 - HMG Plt
 - Det, H&S Co
 - Det, Comm
 - Det, Med
 - FAC Party #1
 - Det, Wpns Co

- 81 FO Tm #1
 - Arty FO Tm #1
 - 1st Sqd, 2d Sec, TOW Plt, HQ Co, RCT-7
 - OCD #1
- Team Mech (Co F)
 - Co F (-)
 - Det, H&S Co
 - Det, Comm
 - Det, Med
 - Det, Wpns Co
 - 81 FO Tm #2
 - 1st/2d Sec, AA Plt
 - Arty FO Tm #2
 - 1st Plt, Co D, 3d Tk Bn
 - 1st Plt (-), Co C, 3d AAV Bn
 - 2d Sqd, 2d Sec, TOW Plt, HQ Co, RCT-7
 - Tm Tank
- Team Tank
 - Co D (-), 3d Tk Bn
 - 1st Plt, Co F, 2/7
 - Det, H&S Co
 - Det, Comm 2
 - Det, Med
 - FAC Party #2
 - Det, Wpns Co
 - 81 FO Tm #3
 - Arty FO Tm #3
 - OCD #2
 - 1st Sec, 1st Plt, Co C, 3d AAV Bn
- 81mm Mortar Plt (Rein)
 - Det, H&S Co
 - Det, Med
 - 1st Sec, 3d Plt, Co C, 3d AAV Bn
- 2d Plt (-), Co C, 1st CEB
 - det, Engr Spt Co, 1st CEB
- Det, 1st Sect, 3d LAAD Bn
- Co G (Rein)
 - Det, H&S Co
 - Det, Comm
 - Det, Med
 - Det, Wpns Co
 - 81 FO Tm #3
 - 3d Sec, AA Plt
 - Arty FO #3

- 3d Plt (-), Co C, 3d AAV Bn

Return to [Top](#)

Orientation

- This direction is generally North. . . .

Return to [Top](#)

Situation

- **General**
 - Omitted.
- **Enemy**
 - Wahabian forces in the vicinity of Los Angeles have temporarily halted the advance of allied forces though they have suffered up to 35% casualties. The allied advance will not continue for at least 24 more hours while allied forces wait for their supply lines to catch up with their forward units. The 111th Brigade has taken advantage of the lull in fighting to the West to attempt a push south from Sunshine Peak to Maumee Mine and Emerson Lake in an attempt to disrupt allied east-west supply lines. Though the 111th attack was unsuccessful due largely to allied air superiority, and the enemy has withdrawn the majority of its forces, the 111th Bde has left behind up to one Mech battalion in Maumee Mine and Emerson Lake. These remain-behind forces are likely to establish platoon and company sized defensive positions, with minimal engineer support in an effort to delay any allied counterattack.
- **Higher Mission**
 - At 0900 on D-Day, 7th MEB conducts a movement to contact to destroy enemy forces within the GCE's zone of action in order to assume a forward defensive posture and deny enemy forces into the GCE's TAOR.
- **Higher Commander's Intent**
 - Argos Pass is the key to the overall effort in order to ensure the northern movement of GCE forces, and deny enemy access to the high speed avenue of approach of Quackenbush Lake. In order to isolate the battle area, I want to prevent enemy forces from escaping and divulging friendly disposition of forces. Overall, I am buying time and distance against the potential for a subsequent enemy attack. Our efforts will provide an eyes forward advantage and prevent enemy indirect firing platforms from reaching the port of 29 Palms.
- **Higher Scheme of Maneuver**
 - 1/7 attacks to our east in Quackenbush corridor. 3d Tank Bn (-) follows in trace of 1/7 as RCT reserve.
- **Adjacent Units**

- 1/7 attacks to our east in Quackenbush corridor. 3d Tank Bn (-) follows in trace of 1/7 as RCT reserve.
- **Supporting Units**
 - a. Artillery. 2/7 has priority of artillery fires in Phase I & II.
 - b. Air. 2/7 has priority of CIFS (10) in Phase I & II. 1/7 has priority of CAS.
- **Attachments**
 - See T/O.
- **Detachments**
 - See T/O.

Return to [Top](#)

Mission

- 2/7 conducts a movement to contact along Axis Jane IOT clear enemy forces in zone within the Emerson Corridor, control Gay's and Maumee Pass, deceive the enemy as to the location of the RCT main effort, and guard the MEB's left flank.

Return to [Top](#)

Execution

- **Commander's Intent**
 - We are a supporting attack for RCT-7's deliberate attack. It is believed that the enemy is employing a battalion (-) sized mechanized force in the areawhich we will be conducting our battalion movement. He is expected to delay initially, and then transition into a deliberate defense. Unfortunately, the enemy situation is somewhat vague. The enemy's critical vulnerability, however, is that he has failed to provide adequate logistical sustainability for his forward deployed forces. RCT-7's pre-planned deep and our own close air sorties will significantly deteriorate his will to fight. I want to maximize our reconnaissance and security efforts forward to ensure we commit our main force under the most favorable circumstances possible. We will remain in tactical column as long as possible to enhance the speed and control for which we can deploy for combat. Unit commanders must be prepared to act boldly to seize the initiative, keep the enemy off balance, and to exploit opportunities as they arise on the battlefield. Our first priority is to protect the column against surprise. Our second is to be prepared to bring our superior combat power to bear against the enemy if he should challenge us. Hence, we will employ our air defense in bounds to ensure the column is protected at critical passage points. Engineers will be employed well forward to facilitate our movement by ensuring they are prepared to rapidly breach any obstacles we may encounter. Armor and anti-armor assets will be distributedthroughout the column to provide all-around protection, and facilitate our integrated entry into combat. I intend to rapidly suppress and

eliminate any forward enemy forces, develop the situation with my lead maneuver units, and deploy my main body to destroy enemy resistance IOT control Gays and Maumee Pass.

- **Scheme of Maneuver**

- 2/7 conducts a single-axis movement to contact in tactical column with one CAAT Team and one Team Mech forward, and one Team Tank and one mechanized company back.

- **Fire Support Plan**

- The Fire Support Plan is

- **Tasks**

- CAAT

- - Advance guard of tactical column. You are March Serial #1. March Serial Commander. Provide forward and flank security during movement.
- - BPT clear hastily emplaced obstacles with attached OCD.
- - BPT transition into screening element of approach march formation. Screen forward of column 2 km to provide forward security for column.
- - BPT suppress enemy armor and mechanized units and allow Team Mech to maneuver to destroy enemy forward units.
- - At RP, screen forward to PL BARREL. BPT continue the attack east.

- Team Mech (Co F)

- - FIT of CAAT as lead march unit of March Serial #2. March Serial Commander.
- - BPT transition into advance guard element of approach march formation. Provide forward security during movement.
- - BPT maneuver to destroy enemy forward units supported by CAAT.
- - At RP, occupy BP 25 in hasty defensive positions. BPT continue the attack east towards MEB Obj A.

- Team Tank

- - FIT 81s Plt as fifth march unit of March Serial #2.
- - BPT transition into lead element of the main body in an approach march formation. Provide forward security of the main body during movement.
- - BPT deploy for combat and maneuver rapidly to exploit the success of Team Mech forward.
- - At RP, occupy BP 35 in hasty defensive positions. BPT continue the attack east towards MEB Obj A.

- 81mm Mortar Plt (Rein)

- - FIT of Engineers as fourth march unit of March Serial #2.

- - BPT establish hasty firing positions ISO march column. POF to CAAT initially.
- - At RP, establish firing positions at FP 55 in GS of the battalion. POF to Tm Mech. BPT continue the attack east.
- 2d Plt (-), Co C, 1st CEB
 - - FIT of the Alpha Command Group as the third march unit of March Serial #2.
 - - BPT to displace additional OCDs forward to breach deliberate obstacle belts.
 - - At RP, occupy an AA at GC-234572 in GS of the battalion. BPT construct hasty obstacles vicinity battalion defensive area. BPT continue the attack east
- towards MEB Obj A.
- Det, 1st Sect, 3d LAAD Bn
 - - Establish firing positions along Axis Jane in support of battalion scheme of maneuver.
 - - Displace forward in bounds to ensure continuous umbrella of anti-air support for the march column.
- **Reserve**
 - Co G (Rein)
 - - FIT Bravo Command Group as second march unit of March Serial #3. March Serial Commander. Provide rear security march column.
 - - BPT transition into lead element of the rear guard in an approach march formation.
 - - BPT deploy for combat and maneuver rapidly to assume the mission of either maneuver unit forward.
 - - At RP, occupy BP 55 in hasty defensive positions. BPT continue the attack east.
- **Coordinating Instructions**
 - (1) MOPP level 0 initially.
 - (2) Order of Movement in Battalion Zone: March Serial #1: CAAT. March Serial #2: Team Mech, Alpha Command, Engineers, 81 Plt, Team Tank March Serial #3: Bravo Command, Co G (Rein).
 - (3) Units start engines for movement from assembly areas to SP at 150835Z OCT 91.
 - (4) LAAD priority of protection:
 - - Tm 1 to Team Mech
 - - Tm 2 to Team Tank
 - (5) LAAD Weapons Control- Yellow (Weapons Tight)
 - (6) SP, RP, Route, PLs, March Objectives, and BPs per Operations Overlay.

- (7) Designate Air Sentries for movement.
- (8) All units conduct immediate action drills for ambush situations; disabled vehicle troop and cargo transfer; dismounted close security; and establishment of outpost security during halts.
- (9) Dispersion between march units: Tactical Column: 250m. Approach March: 500m. Dispersion between march serials: Tactical Column: 500m. Approach March: 1000m.
- (10) Establish STA/Recon Locations and BPT to control supporting arms to engage enemy forces when sighted at:
 - - STA 1- GC 276439
 - - STA 2- GC 297548
 - - Recon 1- GC 325983
 - - Recon 2- GC 376004

Return to [Top](#)

Administration and Logistics

- (1) Ensure accurate crew-served weapon ammunition counts and resupply requests are delivered to the S-4 NLT H-6.
- (2) Submit casualty replacement requests to S-1 NLT 2100 this evening.

Return to [Top](#)

Command and Signal

- **Command Relationships**
 - Omitted.
- **Command**
 - (1) Alpha command initially second march unit in March Serial #2.
 - (2) Bravo Command initially lead march unit in March Serial #3.
- **Signal**
 - (1) Pass SNOWSTORM on Arty COF and Bn TAC 1 upon receiving indirect fires.
 - (2) Codewords:
 - -- STORMY WEATHER: Transition to Approach March Formation
 - -- CLEAR SKIES: Resume Tactical Column Formation

Return to Top

APPENDIX G. SOURCE CODE AND MISCELLANEOUS INI AND HTML FILES FOR THE REDWEB APPLICATION

A. REDWEB.INI INITIALIZATION FILE

The following INI file controls the macros, defaults and page definitions with their corresponding HTML chunk components for the RedWeb system:

```
[TWebApp.Macros]
PageBegin=<h3>ChunkAsPage:
ChunkBegin=<h3>Chunk:
EndInfo=</h3>
EndBuffer=<!-- *****
SiteTitle=REDMAN HTML OpOrd Retrieval System
ReturnToTop=<P>Return to <A HREF="#Top">Top</A></P>

[TWebApp.Defaults]
ChunkDir=c:\redweb\
AppRoot=/htdemo/
MacroPageBegin=PageBegin
MacroChunkBegin=ChunkBegin
MacroEndMarker=EndInfo
MacroEndBuffer=EndBuffer
ImportPageHeader=pageheader
ImportPageFooter=pagefooter

[HOME PAGE]
1=pageheader
2=HOME PAGE
3=pagefooter

[ENTERPWD]
1=pageheader
2=ENTERPWD
3=pagefooter

[DISPLAYORD]
1=PWDCHECK
2=PWDREJECT,PasswordStatus=No
3=DISPLAYORD,PasswordStatus=Yes

[TWebApp.Events]
OrderPick=g1
UnitPick=g1
CheckPassword=g1
LoadOrderValuesToLiterals=g1
WriteReference=g1
WriteTaskOrg=g1
WriteOrientation=g1
WriteGeneral=g1
WriteEnemy=g1
WriteHigherMission=g1
WriteHigherIntent=g1
WriteHigherScheme=g1
WriteAdjacentUnits=g1
WriteSupportingUnits=g1
WriteAttachments=g1
WriteDetachments=g1
WriteMission=g1
WriteIntent=g1
WriteScheme=g1
WriteFireSupport=g1
```

```

WriteTasks=g1
WriteReserve=g1
WriteCoordInst=g1
WriteAdminLogistics=g1
WriteCommandRelationships=g1
WriteCommand=g1
WriteSignal=g1

[TWebApp.Files]
start=Redweb,

[TWebApp.Pages]
HOMEPAGE=,,,REDMAN HTML OpOrd Retrieval System (Page #1)
ENTERPWD=,,,REDMAN HTML OpOrd Retrieval System (Page #2)
DISPLAYORD=,,,REDMAN HTML OpOrd Retrieval System (Page #3)

```

B. REDWEB HTML CHUNK FILE

The following HTML ASCII text file defines the HTML chunks which comprise the RedWeb system:

```

%=chunkBegin=%pageheader%=endInfo=%
<! ***** >
<! * This is the chunk used as the header for all pages. * >
<! ***** >
<HTML><HEAD>
<TITLE>%=SiteTitle=%</TITLE>
</HEAD><BODY>
<IMG SRC='/images/redman.jpg'>
<H1>%=SiteTitle=%</H1><HR>

%=PageBegin=%homepage=,,,%=SiteTitle=%%=where=%
<! ***** >
<! * This is the Site's HOMEPAGE. It includes the form for * >
<! * selecting the particular Operations Order in the database * >
<! * the user wants to retrieve. Note all images should be * >
<! * stored in the c:\website\htdocs\images directory. * >
<! ***** >
<H2>Please select the Operations Order you wish to retrieve below:</H2>

<FORM METHOD=POST ACTION=%=action|enterPwd,=%>
%=OrderPick=%</P>
<INPUT TYPE=SUBMIT VALUE="Select Order"></P>
</FORM>

%=PageBegin=%enterPwd=,,,%=SiteTitle=%%=endInfo=%
<! ***** >
<! * This is the ENTERPWD page which allows the user to enter * >
<! * his password to compare against that stored in the database.* >
<! ***** >
<H2>Please select your unit and enter your password below:</H2>

<FORM METHOD=POST ACTION=%=action|displayOrd,=%>
%=UnitPick=%</P>
<B>Password:</B> <INPUT TYPE="PASSWORD" NAME="Password" SIZE = 8></P>
<INPUT TYPE=SUBMIT VALUE="Enter Password"></P>
</FORM>

%=ChunkBegin=%pwdCheck=,,,%=SiteTitle=%%=endInfo=%
<! ***** >
<! * This chunk calls the CHECKPASSWORD MACRO to set the value * >
<! * of the passwordStatus literal appropriately * >
<! ***** >
%=CheckPassword=%

```



```

%=ChunkBegin=%pwdReject=,,,%=SiteTitle=%%=endinfo=%
<! ***** >
<! * This chunk is displayed if the passwordStatus literal is      * >
<! * set to "No"                                                  * >
<! ***** >
<H1>%=SiteTitle=%</H1>
<H2>Security Violation Detected!</H2>
<HR>
The password you entered was invalid. %=jump|enterPwd|Return=% to
the password entry screen to retype.<P>

```

```

%=PageBegin=%displayOrd=,,,%=SiteTitle=%%=endinfo=%
<! ***** >
<! * This is the DISPLAYORD Page for displaying the Operation      * >
<! * Order for authorized users                                    * >
<! ***** >
<! Load all picklists from database values >
%=LoadOrderValuesToLiterals=%

```

```

<H1><A NAME="Top">%=OrderName=%</A></H1>
<UL>
<LI><A HREF="#General"><B>General Information</B></A>
<LI><A HREF="#Task"><B>Task Organization</B></A>
<LI><A HREF="#Orientation"><B>Orientation</B></A>
<LI><A HREF="#Situation"><B>Situation</B></A>
<LI><A HREF="#Mission"><B>Mission</B></A>
<LI><A HREF="#Execution"><B>Execution</B></A>
<LI><A HREF="#Administration"><B>Administration and Logistics</B></A>
<LI><A HREF="#Command"><B>Command and Signal</B></A>
</UL>
<HR>

```

```

<H2><A NAME="General"><font size=5>General Information</font></A></H2>
<UL>
<LI><B>Initiating Unit: </B>%=InitUnit=%
<LI><B>Mission Type: </B>%=MissionType=%
<LI><B>Location: </B>%=Location=%
<LI><B>DTG: </B>%=DTG=%
<LI><B>Time Zone: </B>%=TimeZone=%
<LI><B>Serial No: </B>%=SerialNumber=%
<LI><B>Order Name: </B>%=OrderName=%
<LI><B>Reference: </B>%=WriteReference=%
</UL>
%=ReturnToTop=%
<HR>

```

```

<H2><A NAME="Task">Task Organization</A></H2>
%=WriteTaskOrg=%
%=ReturnToTop=%
<HR>

```

```

<H2><A NAME="Orientation">Orientation</A></H2>
%=WriteOrientation=%
%=ReturnToTop=%
<HR>

```

```

<H2><A NAME="Situation">Situation</A></H2>
<UL>
<LI><B>General</B>
%=WriteGeneral=%</P>
<LI><B>Enemy</B>
%=WriteEnemy=%</P>
<LI><B>Higher Mission</B>
%=WriteHigherMission=%</P>
<LI><B>Higher Commander's Intent</B>
%=WriteHigherIntent=%</P>

```

```

<LI><B>Higher Scheme of Maneuver</B>
%=WriteHigherScheme=%</P>
<LI><B>Adjacent Units</B>
%=WriteAdjacentUnits=%</P>
<LI><B>Supporting Units</B>
%=WriteSupportingUnits=%</P>
<LI><B>Attachments</B>
%=WriteAttachments=%</P>
<LI><B>Detachments</B>
%=WriteDetachments=%</P>
</UL>
%=ReturnToTop=%
<HR>

<H2><A NAME="Mission">Mission</A></H2>
%=WriteMission=%
%=ReturnToTop=%
<HR>

<H2><A NAME="Execution">Execution</A></H2>
<UL>
<LI><B>Commander's Intent</B>
%=WriteIntent=%</P>
<LI><B>Scheme of Maneuver</B>
%=WriteScheme=%</P>
<LI><B>Fire Support Plan</B>
%=WriteFireSupport=%</P>
<LI><B>Tasks</B>
%=WriteTasks=%</P>
<LI><B>Reserve</B>
%=WriteReserve=%</P>
<LI><B>Coordinating Instructions</B>
%=WriteCoordInst=%</P>
%=ReturnToTop=%
</UL>
<HR>

<H2><A NAME="Administration">Administration and Logistics</A></H2>
%=WriteAdminLogistics=%</P>
%=ReturnToTop=%
<HR>

<H2><A NAME="Command">Command and Signal</A></H2>
<UL>
<LI><B>Command Relationships</B>
%=WriteCommandRelationships=%</P>
<LI><B>Command</B>
%=WriteCommand=%</P>
<LI><B>Signal</B>
%=WriteSignal=%</P>
</UL>
%=ReturnToTop=%

%=chunkBegin=%pagefooter%=endInfo=%
<!-- ***** >
<!-- * This is the footer for all pages. * >
<!-- ***** >
<HR>
%=GO|HomePage|Return to Main Page=%
<P>For information on this system, send email to Major J. C. Cummiskey, USMC at
<A HREF="mailto:jccummis@nps.navy.mil">jccummis@nps.navy.mil</A>.</P>
</BODY></HTML>

```

C. REDWEB APPLICATION SOURCE CODE

The following code (in alphabetical order by module) represents the Delphi language source code for the DACT version of the RedWeb HTML Processing application. This code was developed in conjunction with Borland's Delphi 1.0 integrated development environment. The author created all code that follows (however, source code annotated with "public domain" in the introductory comments has been adapted from public sources). Delphi forms (DFM files) are not included due to space constraints, however, these will be available online at <http://www.mbay.net/jccummis>.

1. REDWEBF.DPR Source Code

This is the source code file for the "Main Program" module:

```
unit Redwebf;

interface

uses
  SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs
  , Buttons, StdCtrls, Menus, ExtCtrls
{$IFDEF WIN32}
  , Windows
{$ELSE}
  , WinProcs, WinTypes
{$ENDIF}
  , Toolbar
  , Tpmenu, Tpmemo, WebMemo, Combobox
  , WebTypes, WebVars, WebBase, WebCore, WebApp, WebCall
  , Webmenu, HtmlBase, HtmlCore, HtmlSend, WebInfo, WebBrows,
  UpdateOk, WebIniFl, WebHttp, WebServ, IniLink
  , WebForm, Restorer, TpApplic, TpLabel, WebLink, WebPage, WebPHub, DB,
  DBTables, DBCtrls, Grids, Outline, DBOutln;

type
  TfrmRedWeb = class(TWebAppForm)
    StatusBar1: TtpStatusBar;
    StatusPanel1: TtpStatusPanel;
    WebAppOutput: TWebAppOutput;
    WebInfo: TWebInfo;
    WebIniFileLink: TWebIniFileLink;
    WebServer: TWebServer;
    WebBrowser: TWebBrowser;
    WebCommandLine: TWebCommandLine;
    WebAppRedman: TWebApp;
    WebSession0: TWebSession;
    tpComponentPanel1: TtpComponentPanel;
    tpAppRole1: TtpAppRole;
    FormRestorer1: TFormRestorer;
    DoMenuBarFile: TtpDfmMenuItem;
    DoExit: TtpDfmMenuItem;
    DoMenuBarHelp: TtpDfmMenuItem;
    DoContents: TtpDfmMenuItem;
    DoTopicSearch: TtpDfmMenuItem;
    DoHowtouseHelp: TtpDfmMenuItem;
    DoTtpDfmMenuItem: TtpDfmMenuItem;
    DoAbout: TtpDfmMenuItem;
    DoActionComponents: TtpMenuItem;
    DoWebPages: TtpMenuItem;
```

```

WebMenu1: TWebMenu;
tpComboBar1: TtpComboBar;
ToolBar2: TtpToolBar;
DoMenuItemView: TtpMenuItem;
DoMenuItemTool: TtpMenuItem;
DoMenuItemVerb: TtpMenuItem;
DoMenuItemIdle: TtpMenuItem;
tpToolButtonBrowser: TtpToolButton;
tpToolButtonMemo: TtpToolButton;
tpToolButtonActivated: TtpToolButton;
tbBlankPage: TtpToolButton;
tbUserPage: TtpToolButton;
tbMemoPage: TtpToolButton;
Notebook: TNotebook;
WebHtmlMemo1: TWebHtmlMemo;
tpLabel1: TtpLabel;
tpLabel2: TtpLabel;
tblOrders: TTable;
tblTO: TTable;
tblMissions: TTable;
dbmReference: TDBMemo;
dtsOrders: TDataSource;
dbmOrientation: TDBMemo;
dbmGeneral: TDBMemo;
dbmEnemy: TDBMemo;
dbmHigherMission: TDBMemo;
dbmHigherIntent: TDBMemo;
dbmHigherScheme: TDBMemo;
dbmAdjacentUnits: TDBMemo;
dbmSupportingUnits: TDBMemo;
dbmAttachments: TDBMemo;
dbmDetachments: TDBMemo;
dbmMission: TDBMemo;
dbmIntent: TDBMemo;
dbmScheme: TDBMemo;
dbmFireSupport: TDBMemo;
dbmCoordInst: TDBMemo;
dbmAdminLogistics: TDBMemo;
dbmCommandRelationships: TDBMemo;
dbmCommand: TDBMemo;
dbmSignal: TDBMemo;
dtsTO: TDataSource;
mmoReserve: TMemor;
mmoTasks: TMemor;
dboTaskOrg: TDBOutline;
mmoTaskOrg: TMemor;
procedure SelectPage(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure WebAppRedmanEventMacro(Sender: TWebOutputApp; const aMacro,
    aParams, aID: String);
procedure WriteMemoToHTMLUnorderedList(Sender: TObject; const fieldName : TMemor);

private
{ Private declarations }
public
{ Public declarations }
    stlOrderNames : TStringList;
    stlUnitNames : TStringList;
end;

var
    frmRedWeb: TfrmRedWeb;

implementation

{$R *.DFM}

Uses weblist;

{-----}

```

```

procedure TfrmRedWeb.FormCreate(Sender: TObject);
begin
    SelectPage(tbBlankPage);
end;

{-----}

procedure TfrmRedWeb.SelectPage(Sender: TObject);
begin
    with TtpToolButton(Sender) do begin
        NoteBook.PageIndex:=Tag;
        Down:=True;
    end;
end;

procedure TfrmRedWeb.FormShow(Sender: TObject);
var
    strTemp : String;
begin
    stlOrderNames := TStringlist.create;
    With tblOrders do
    begin
        Open;
        while not EOF do
        begin
            strTemp := FieldByName('ORDER_NAME').AsString + '=' +
FieldByName('ORDER_ID').AsString;
            stlOrderNames.Add(strTemp);
            Next;
        end;
    end;
end;

end;

procedure TfrmRedWeb.WebAppRedmanEventMacro(Sender: TWebOutputApp;
const aMacro, aParams, aID: String);
var
    temp : Tweblist;
    strTemp : String;
    i : integer;
    level1Flag, level2Flag : Boolean;
begin
    if comparetext (amacro, 'OrderPick')=0 then
    begin
        temp:=Tweblist.create(self);
        with webappoutput do
        begin
            filldropdown(temp,stlOrderNames,'OrderName:','OrderSelected','',ddmvalueasvalue);
            sendstringlist(temp,false);
        end;
        temp.free;
    end

    else if Comparetext (amacro, 'UnitPick')=0 then
    begin
        stlUnitNames := TStringlist.create;
        With tblTO do
        begin
            { check if user didn't select an order--if so select '1' for him (or
            better yet, display an error page and allow him to return to OrderPick}

            WebAppRedman.literal['Password'] := '';
            if WebAppRedman.literal['OrderSelected'] = '' then
                WebAppRedman.literal['OrderSelected'] := '1';
            TableName := 'TO' + WebAppRedman.literal['OrderSelected'] + '.DB';
            Open;
            while not EOF do
            begin

```

```

        strTemp := FieldByName('UNIT_TEXT').AsString + '=' +
FieldByName('UNIT_CODE').AsString;
        stlUnitNames.Add(strTemp);
        Next;
    end;
    Close;
end;

temp:=Tweblist.create(self);
with webappoutput do
begin
    filldropdown(temp,stlUnitNames,'UnitName:', 'UnitSelected','',ddmvalueasvalue);
    sendstringlist(temp,false);
end;
temp.free;
end

else if Comparetext (amacro, 'CheckPassword')=0 then
begin
    {check password}
    With tblTO do
    begin
        TableName := 'TO' + WebAppRedman.literal['OrderSelected'] + '.DB';
        Open;
        FindKey([StrToInt(WebAppRedman.literal['UnitSelected'])]);
        If FieldByName('PASSWORD').AsString = WebAppRedman.literal['Password'] then
            WebAppRedman.literal['PasswordStatus'] := 'Yes'
        else
            WebAppRedman.literal['PasswordStatus'] := 'No';
        Close;
    end;
end

else if Comparetext (amacro, 'LoadOrderValuesToLiterals')=0 then
begin
    With tblOrders do
    begin
        FindKey([StrToInt(WebAppRedman.literal['OrderSelected'])]);

        WebAppRedman.literal['OrderName'] := FieldByName('ORDER_NAME').AsString;
        WebAppRedman.literal['InitUnit'] := FieldByName('INIT_UNIT').AsString;

        {look up Mission Type}
        With tblMissions do
        begin
            Open;
            if FindKey([tblOrders.FieldByName('MISSION_ID').AsInteger]) then
                WebAppRedman.literal['MissionType'] := FieldByName('TYPE').AsString
            else
                WebAppRedman.literal['MissionType'] := 'Not Defined';
            Close;
        end;

        WebAppRedman.literal['Location'] := FieldByName('LOCATION').AsString;
        WebAppRedman.literal['DTG'] := FieldByName('DTG').AsString;
        WebAppRedman.literal['TimeZone'] := FieldByName('TIMEZONE').AsString;
        WebAppRedman.literal['SerialNumber'] := FieldByName('SERIAL_NO').AsString;
    end;
end

else if Comparetext (amacro, 'WriteReference')=0 then
begin
    WriteMemoToHTMLUnorderedList(Self, TMemodbmReference));
end

else if Comparetext (amacro, 'WriteTaskOrg')=0 then
begin
    With tblTO do
    begin
        TableName := 'TO' + WebAppRedman.literal['OrderSelected'] + '.DB';
        Open;
    end;
end

```

```

        {Load up TDEOutline StringList from TO data structure}
        dboTaskOrg.LoadFromDataSet;
        mmoTaskOrg.Clear;
        mmoTaskOrg.Lines.AddStrings(dboTaskOrg.Lines);
        Close;
    end;

    WebAppRedman.SendString('<UL>');
    level1Flag := False; {Flag to indicate we are inside the first level of the list}
    level2Flag := False; {Flag to indicate we are inside the second level of the list}

    {Continue until we have processed every unit in the TO list; handles three embedded
    levels
    of lists--one of which is hard coded at entry; we can hardcode <CR><LF> combinations
    after each SendString with '+ #13#10' if we want to make the HTML more legible,
    however,
    this will increase packet size for transmission}
    for i := 0 to mmoTaskOrg.Lines.Count - 1 do
    begin
        {we are at the top most level, so simply write out the unit}
        if (Copy(mmoTaskOrg.Lines[i], 1, 1) <> #9) AND NOT level1Flag then
            WebAppRedman.SendString('<P><LI>' + mmoTaskOrg.Lines[i])

        {we are at the top most level, but need to close out an embedded list first}
        else if (Copy(mmoTaskOrg.Lines[i], 1, 1) <> #9) AND level1Flag then
            begin
                if level2Flag then
                    WebAppRedman.SendString('</UL></UL><P><LI>' + mmoTaskOrg.Lines[i])
                else
                    WebAppRedman.SendString('</UL><P><LI>' + mmoTaskOrg.Lines[i]);
                level1Flag := False;
                level2Flag := False;
            end

        {this is the first unit in an embedded list, so start the list}
        else if (Copy(mmoTaskOrg.Lines[i], 1, 1) = #9) AND NOT level1Flag then
            begin
                level1Flag := True;
                WebAppRedman.SendString('<UL><LI>' + mmoTaskOrg.Lines[i]);
            end

        {this is NOT the first unit in an embedded list, so check to see if we are at level
    2}
        else if (Copy(mmoTaskOrg.Lines[i], 1, 1) = #9) AND level1Flag then
            begin
                {this is the first unit in a sub-embedded list, so start the list}
                if (Copy(mmoTaskOrg.Lines[i], 2, 1) = #9) AND NOT level2Flag then
                    begin
                        level2Flag := True;
                        WebAppRedman.SendString('<UL><LI>' + mmoTaskOrg.Lines[i]);
                    end

                {this is another unit in the same list, so just write it out}
                else if (Copy(mmoTaskOrg.Lines[i], 2, 1) = #9) AND level2Flag then
                    WebAppRedman.SendString('<LI>' + mmoTaskOrg.Lines[i])

                {this is the first unit in ANOTHER sub-embedded list at the same level,
                so end the previous list first}
                else if (Copy(mmoTaskOrg.Lines[i], 2, 1) <> #9) AND level2Flag then
                    begin
                        level2Flag := False;
                        WebAppRedman.SendString('</UL><LI>' + mmoTaskOrg.Lines[i]);
                    end

                {this is another unit at level one, so simply write it out}
                else if (Copy(mmoTaskOrg.Lines[i], 2, 1) <> #9) AND NOT level2Flag then
                    WebAppRedman.SendString('<LI>' + mmoTaskOrg.Lines[i]);

            end;
        end;
    end;

```

```

        {Close out lists appropriately based on current level we stopped at}
        WebAppRedman.SendString('</UL>');
        if level1flag then
            WebAppRedman.SendString('</UL>');
        if level2flag then
            WebAppRedman.SendString('</UL>');
        end

    else if Comparetext (amacro, 'WriteOrientation')=0 then
        begin
            WriteMemoToHTMLUnorderedList(Self, TMemo(dbmOrientation));
        end

    else if Comparetext (amacro, 'WriteGeneral')=0 then
        begin
            WriteMemoToHTMLUnorderedList(Self, TMemo(dbmGeneral));
        end

    else if Comparetext (amacro, 'WriteEnemy')=0 then
        begin
            WriteMemoToHTMLUnorderedList(Self, TMemo(dbmEnemy));
        end

    else if Comparetext (amacro, 'WriteHigherMission')=0 then
        begin
            WriteMemoToHTMLUnorderedList(Self, TMemo(dbmHigherMission));
        end

    else if Comparetext (amacro, 'WriteHigherIntent')=0 then
        begin
            WriteMemoToHTMLUnorderedList(Self, TMemo(dbmHigherIntent));
        end

    else if Comparetext (amacro, 'WriteHigherScheme')=0 then
        begin
            WriteMemoToHTMLUnorderedList(Self, TMemo(dbmHigherScheme));
        end

    else if Comparetext (amacro, 'WriteAdjacentUnits')=0 then
        begin
            WriteMemoToHTMLUnorderedList(Self, TMemo(dbmAdjacentUnits));
        end

    else if Comparetext (amacro, 'WriteSupportingUnits')=0 then
        begin
            WriteMemoToHTMLUnorderedList(Self, TMemo(dbmSupportingUnits));
        end

    else if Comparetext (amacro, 'WriteAttachments')=0 then
        begin
            WriteMemoToHTMLUnorderedList(Self, TMemo(dbmAttachments));
        end

    else if Comparetext (amacro, 'WriteDetachments')=0 then
        begin
            WriteMemoToHTMLUnorderedList(Self, TMemo(dbmDetachments));
        end

    else if Comparetext (amacro, 'WriteMission')=0 then
        begin
            With tblTO do
                begin
                    TableName := 'TO' + WebAppRedman.literal['OrderSelected'] + '.DB';
                    Open;
                    First; {First Record is always the Initiating Unit in TO structure}
                    WriteMemoToHTMLUnorderedList(Self, TMemo(dbmMission));
                    Close;
                end;
            end

    else if Comparetext (amacro, 'WriteIntent')=0 then
        begin

```



```

        WriteMemoToHTMLUnorderedList(Self, TMemo(dbmIntent));
    end

    else if Comparetext (amacro, 'WriteScheme')=0 then
    begin
        WriteMemoToHTMLUnorderedList(Self, TMemo(dbmScheme));
    end

    else if Comparetext (amacro, 'WriteFireSupport')=0 then
    begin
        WriteMemoToHTMLUnorderedList(Self, TMemo(dbmFireSupport));
    end

    else if Comparetext (amacro, 'WriteTasks')=0 then
    begin
        With tblTO do
        begin
            mmoTasks.Clear;
            TableName := 'TO' + WebAppRedman.literal['OrderSelected'] + '.DB';
            Open;
            First;
            While Not EOF Do
            Begin
                If (FieldByName('REPORTS_TO').AsString = '') AND NOT
FieldByName('RESERVE').AsBoolean
AND NOT (FieldByName('UNIT_TEXT').AsString =
tblOrders.FieldByName('INIT_UNIT').AsString) Then
                Begin
                    mmoTasks.Lines.Add(FieldByName('UNIT_TEXT').AsString);
                    mmoTasks.Lines.AddStrings(dbmMission.Lines);
                    mmoTasks.Lines.Add('');
                End;
            Next;
            End;
            WriteMemoToHTMLUnorderedList(Self, mmoTasks);
            Close;
        end;
        WebAppRedman.SendString('</UL>');
    end

    else if Comparetext (amacro, 'WriteReserve')=0 then
    begin
        With tblTO do
        begin
            mmoReserve.Clear;
            TableName := 'TO' + WebAppRedman.literal['OrderSelected'] + '.DB';
            Open;
            First;
            While Not EOF Do
            Begin
                If (FieldByName('REPORTS_TO').AsString = '') AND FieldByName('RESERVE').AsBoolean
AND NOT (FieldByName('UNIT_TEXT').AsString =
tblOrders.FieldByName('INIT_UNIT').AsString) Then
                Begin
                    mmoReserve.Lines.Add(FieldByName('UNIT_TEXT').AsString);
                    mmoReserve.Lines.AddStrings(dbmMission.Lines);
                End;
            Next;
            End;
            WriteMemoToHTMLUnorderedList(Self, mmoReserve);
            Close;
        end;
        WebAppRedman.SendString('</UL>');
    end

    else if Comparetext (amacro, 'WriteCoordInst')=0 then
    begin
        WriteMemoToHTMLUnorderedList(Self, TMemo(dbmCoordInst));
    end

    else if Comparetext (amacro, 'WriteAdminLogistics')=0 then

```

```

begin
  WriteMemoToHTMLUnorderedList(Self, TMemo(dbmAdminLogistics));
end

else if Comparetext (amacro, 'WriteCommandRelationships')=0 then
begin
  WriteMemoToHTMLUnorderedList(Self, TMemo(dbmCommandRelationships));
end

else if Comparetext (amacro, 'WriteCommand')=0 then
begin
  WriteMemoToHTMLUnorderedList(Self, TMemo(dbmCommand));
end

else if Comparetext (amacro, 'WriteSignal')=0 then
begin
  WriteMemoToHTMLUnorderedList(Self, TMemo(dbmSignal));
end

end;

procedure TfrmRedWeb.WriteMemoToHTMLUnorderedList(Sender: TObject; const fieldName :
TMemo);
var
  i : integer;
  blnListFlag : boolean;
begin
  blnListFlag := false;
  With tblOrders do
  begin
    WebAppRedman.SendString('<UL><LI>');
    for i := 0 to fieldName.Lines.Count - 1 do
    begin
      {handles two levels of indentation only (one of which is hard coded upon entry)}
      if fieldName.Lines[i] <> '' then
      begin
        if (Copy(fieldName.Lines[i], 1, 1) = ' ') and not blnListFlag then
        begin
          WebAppRedman.SendString('<UL>');
          WebAppRedman.SendString('<LI>' + fieldName.Lines[i]);
          blnListFlag := true;
        end
        else if (Copy(fieldName.Lines[i], 1, 1) <> ' ') and blnListFlag then
        begin
          WebAppRedman.SendString('</UL>');
          WebAppRedman.SendString('<LI>' + fieldName.Lines[i]);
          blnListFlag := false;
        end
        else if (Copy(fieldName.Lines[i], 1, 1) = ' ') and blnListFlag then
          WebAppRedman.SendString('<LI>' + fieldName.Lines[i])
        else
          WebAppRedman.SendString(fieldName.Lines[i]);
        end
      end
    begin
      WebAppRedman.SendString('</P>');
      if not blnListFlag then
        WebAppRedman.SendString('<LI>');
    end;
  end;

  WebAppRedman.SendString('</UL>');
end;
end;
end.

```

2. REDWEBP.DPR Source Code

This is the source code file for the RedWeb Project File:

```
program Redwebp;

uses
  Forms,
  Redwebf in 'REDWEBF.PAS' {frmRedWeb};

{$R *.RES}
{$IFDEF WIN32}
{$R tpack32.res}
{$R HREF32.res}
{$ELSE}
{$R tpack16.res}
{$R HREF16.res}
{$ENDIF}

begin
  Application.Title := 'RedWeb Prototype 1.0';
  Application.CreateForm(TfrmRedWeb, frmRedWeb);
  Application.Run;
end.
```


REFERENCES

- Ace Technologies, Inc., *HP 200LX*, advertisement, *The HP Palmtop Paper*, vol. 5, no. 4, 1996.
- Accurite Corporation, *Accurite Home Page*, Web Page, 1996. Available at <http://www accurite.com/>
- Allied Signal Aerospace, *Fortezza Plus/LP Crypto Card Marketing Brochure*, Allied Signal, 1996.
- American Mobile Satellite Corporation (AMSC), *Getting Mobile Satellite Communications Off the Ground*, AMSC, 1996.
- April, C., "Microsoft Gets Pegasus OS Ready for New PDAs," *Infoworld*, p. 14, 6 May 1996.
- Ardis Corporation, *Introduction to Ardis*, Web Page, 1996. Available at http://www.ardis.com/ardis_hp/website2.htm
- Armbruster, H., "The Flexibility of ATM: Supporting Future Multimedia and Mobile Communications," *IEEE Communications Magazine*, 1995.
- Bajarin, T., "New Chip Paves the Way for Palm-Size Supercomputers," *Mobile Office*, pp. 30-31, May 1996.
- Boardman, B., "20 V.34 PC (Politically Correct) Card Modems Can't Change the Weather," *Network Computing*, 15 January 1996.
- Borland International, *Borland Delphi for Windows 95 and Windows NT Manual*, 1996.
- Brodsky, I., *Totally Unplugged: Wireless Technologies and Business Applications*, Datacomm Research Company, San Jose, California, 1996.
- Brown, B., *Megahertz AllPoints Makes the Wireless Connection*, *PC Magazine*, p. 66, 28 May 1996.
- Brown, E., *Still Waiting for USB; Peripheral makers ready Universal Serial Bus devices for '96*, *PC Magazine*, p. 37, 12 March 1996.
- Bryant, J., "Paging and Messaging Enter a New Era," *Wireless Business Technology*, p. 17, March 1996.

Brutzman, D., Assistant Professor, NPS, email to jccummis@nps.navy.mil, Naval Postgraduate School, Monterey, California, 1996.

Buddenberg, R., Assistant Professor, NPS, email to iirg@stl.nps.navy.mil, Naval Postgraduate School, Monterey, California, 1996

C41 Division, Headquarters, U. S. Marine Corps, *Command and Control; A U.S. Marine Corps Concept Paper*, USMC, Washington DC, 1995.

Cellular Digital Packet Data (CDPD) Forum, *What is the CDPD Forum?*, Web Page, 1996. Available at <http://www.cdpd.org>

Classic Software, *TcsNotebook Component Help File*, filename CSNOTEBK.HLP, Classic, 1996.

Cochran, R., *HP-100/200LX Frequently Asked Questions List*, USENET comp.sys.palmtops newsgroup, 1996. Available at http://www.netmar.com/users/doc/technote_hp200lx.html

Colston, D., *Net-Tamer Home Page*, Web Page, 1996. Available at <http://people.delphi.com/davidcolston/>

Condon, J., Duff, T., Jukl, M., Kalmanek, C., Locanthi, B, Savicki, J, Venutol, J., Rednet: *A Wireless ATM Local Area Network using Infrared Links*, AT&T Murray Hill, 1996.

Cummiskey, J., "A Few Good Men—One Good Palmtop," *The HP Palmtop Paper*, Iowa, Fairfield, Iowa, September/October 1993. Available at <http://www.mbay.net/~jccummis>

Cummiskey, J., "Using a Cellular Phone and a Fax/Modem PC Card," *The HP Palmtop Paper*, vol. 4, Bonus Issue, Fairfield, Iowa, 1995. Available at <http://www.mbay.net/~jccummis>

Data Critical Corporation, *Data Critical Corporation*, Web Page, 1996. Available at <http://www.datacrit.com>

Defense Information Systems Agency (DISA), Joint Interoperability and Engineering Organization, *Variable Message Format Technical Interface Design Plan (Test Edition)*, vols. I, II, III, and IV, 1995.

Defense Information Systems Agency (DISA), *MIL-STD-188-220A, Interoperability Standard for Digital Message Transfer Device Subsystems*, DISA, 27 July 1995.

Drucker, F., *Commo User Guide and Reference Manual*, filename COMMO.DOC, Columbus, Ohio, 1996. Available at <http://www.cris.com/~Jmeddaug/commo.shtml>

Duff, David A. *Wireless Applications for Marine Air Ground Task Forces*, Masters thesis, Naval Postgraduate School, Monterey, California, June 1996.

EduCalc Corporation, *EduCALC ON-LINE*, Web Page, 1996. Available at <http://www.educalc.com/>

Enbloc Incorporated, *R/F PalmStation Plus Marketing Brochure*, Enbloc, 1996.

Finger, S. et.al., "Rapid Design and Manufacture of Wearable Computers," *Communications of the ACM*, pp. 63-67, February 1996.

Frezza, B., *Wireless LANs: The Search For Indoor Plumbing*, Wireless Computing Associates, 1995.

Garzotto, A., *HV HTML Viewer Documentation*, 1995. Available at <http://www.ics.uci.edu/~garzotto/palmtop.html>

Geoworks, Inc., *Corporate Background flier*, Geoworks, Alameda, California, 1995. Available at <http://www.geoworks.com>

Gibbons, P., Commandant's Warfighting Laboratory, MCCDC, telephone interview, Quantico, Virginia, 21 November 1995.

Globalstar Limited Partnership, *Globalstar*, Web Page, 1996. Available at http://www.wp.com/mcintosh_page_o_stuff/globals.html

Greystone Peripherals, *Greystone Peripherals*, Web Page, 1996. Available at <http://www.grystone.com./>

Griffith, S., *Sun Tzu's The Art of War*, Oxford University Press, New York, 1978.

Groz, M., "FireWire Plays Starring Role in Simple, Fast PCs," *PC Magazine*, p. 36, July 1996.

Gundavaram, S., *CGI Programming on the World Wide Web*, O'Reilly & Associates, Inc., 1996.

Hamming, R., interview, Naval Postgraduate School, Monterey, California, 13 April 1996.

Hewlett-Packard (HP) Corporation, *HP 95LX User's Guide*, Part Number F1000-90003, HP, 1992.

Hewlett-Packard (HP) Corporation, *HP 200LX User's Guide*, Part Number F1060-90001, Edition 2, HP, 1994.

- Hewlett-Packard (HP) Corporation, *HP OmniGo 700*, advertisement, HP, 1995.
- HRef Corporation, *WebHub User's Manual*, HRef Corporation, New York, 1996.
- Imielinski, T., *Mobile Computing*, Rutgers University, Camden, New Jersey, 1995.
- INMARSAT Corporation, *Inmarsat at a Glance*, Web Page, 1996. Available at <http://www.inmarsat.org/>
- Johnson, R., Director, General Services Administration (GSA), Lecture to the Naval Postgraduate School, Monterey, California, 23 Aug 1994.
- Joint Interoperability Test Command (JITC), *Joint Warrior Interoperability Demonstration 1995 (JWID 95) Guide Book*, MCTSSA, Camp Pendleton, California 1995.
- Kay, A., Keynote Address at Nomadic '96--The Nomadic Computing and Communications Conference, Technology Transfer Institute, San Jose, California, 15 March 1996.
- Keenan, J., Col, USMC, TEECG OIC, telephone interview, MCAGCC, Twentynine Palms, California, 8 December 1995.
- Kelly, B., *From Stone to Silicon: A Revolution in Information Technology and Implications for Military Command and Control*, AFCEA, Washington, DC, 1993.
- Kirvan, P., "Wondering about Wandering Connectivity," *Communications Management*, p. 38, May 1996.
- Kistler, J. and Satyanarayanan, M., "Disconnected Operation in the Coda File System," *ACM Transactions on Computer Systems*, vol. 10, February 1992.
- Kleinrock, L., Keynote Presentation, MobiCom 95--The first Annual International Conference on Mobile Computing and Networking, Berkeley, California, 14 Nov 1995.
- Kohl, G., *Palmtop Application Library (PAL) Version 1.6 Documentation*, The PAL Group, 1996. Available at <http://ourworld.compuserve.com:80/homepages/gilles/>
- Leach, N. and Moore, M., "Pegasus' OS Taking Flight," *PC Week*, p. 116, 6 May 1996.
- Le-Ngoc, S., *Math Fix said to Boost Communications Capacity*, Electronic Engineering Times, p37, January 8 1996.

Levin, C., "Space Race; Competition Intensifies for Satellite Networks," PC Magazine, p. 31, 12 March 1996.

Marine Corps Combat Development Command (MCCDC), Requirements Division, *Draft Concept of Employment for DACT*, MCCDC, Quantico, Virginia, 1996.

Marine Corps Tactical Systems Support Activity (MCTSSA), *DACT Technical Characteristics*, brochure, MCTSSA, San Diego, California, 1995.

Maelstrom Software, *TDBOutline Component User's Manual*, filename TDBOUTLI.HLP, 1996.

Mathias, C., *Emerging Mobile Technologies and Systems*, Presentation at the Digital Consulting Inc. Field and Sale Force Automation seminar, San Jose, California, 22 February 1996.

McCarthy, E., Mitre Corporation, interview, Berkeley, California, 14 November 1995.

McCracken, E., Silicon Graphics Corporation, Lecture to the Naval Postgraduate School, Monterey, California, 27 August 1996.

Megahertz, *AllPoints White Paper*, U.S. Robotics, Skokie, Illinois, 1995.

Merwick, K., Windows CE Group, Microsoft Corporation, telephone interview, Redmond, Washington, 6 September, 1996.

Metricom Corporation, *1996 Limited Higher Education Program for the Ricochet Wireless Network Service*, Metricom Corporation, Los Gatos, California, 1996.

Microsoft Corporation, *Microsoft Visual C++ User's Guide*, vol. 1, Microsoft Press, 1995.

Microsoft Corporation, *Microsoft Pegasus Developer Conference Agenda*, Microsoft Press, 1996.

Microsoft Corporation, *Microsoft Pegasus Developer Conference SDK Reference Programmer's Guide*, vols. I and II, Microsoft Press, 1996.

Mitre Corporation, *MITRE/ESC Year 2000 Homepage*, Web Page, 1996. Available at <http://www.mitre.org/research/y2k/>

Motorola Corporation, *Personal Messenger 100D Marketing Brochure*, Motorola, Phoenix, Arizona, 1995.

Motorola Corporation, *Motorola Corporation*, Web Page, 1996. Available at <http://www.motorola.com>

Naval Air Warfare Center (NAWC), Aircraft Division, "DACT Announcement," *Commerce Business Daily*, , Issue No. PSA-1473, 15 November, 1995.

Negroponte, N., *Being Digital*, Alfred A. Knopf, New York, 1995.

Nemzow, M., *Implementing Wireless Networks*, McGraw-Hill, 1995.
Networks User's Guide, Solectek Corporation, 1994.

Nierle, J., *Internetworking: Technical Strategy for Implementing the Next Generation Internet Protocol (IPv6) in the Marine Corps Tactical Data Network*, Masters thesis, Naval Postgraduate School, Monterey, California, June 1996. Available at <http://www.stl.nps.navy.mil/~jenierle/thesis.html>.

Pause, G. and Defibaugh, R., Torrey Science Corporation Managers, interview, San Diego, California, 8 February 1996.

PCSI Organization, *CDPD Primer*, Web Page, 1996. Available at <http://www.pcsi.com/html/cdpdfaq.shtml>

Perkins, C., *Mobile IP: Adding Mobility to the Internet*, IBM, T.J. Watson Research Center, 1996.

Peters, T., Lecture to the Naval Postgraduate School, Monterey, California, 13 September 1994.

Qualcomm Corporation, *QUALCOMM Incorporated*, Web Page, 1996. Available at <http://www.qualcomm.com/>

Quatech, Inc., *Quatech, Inc.*, Web Page, 1996. Available at http://www.isa.org/isa/directory/manufact/man_159040.html

Ram Mobile Data Corporation, *Ram Mobile Data*, Web Page, 1996. Available at <http://www.ram-wireless.com/>

RDC Communications, *Market Position of PortLAN as a Wireless LAN Solution*, RDC Communications Ltd, 1995.

Richter, J., *Advanced Windows; A Developer's Guide to the Win32 API for Windows NT 3.5 and Windows 95*, Microsoft Press, 1995.

Rigney, S., "JetEye Net Plus Beams Your Notebook to the Network," *PC Magazine*, p. 48, July 1996.

Romkey, J., *RFC 1055; A Non-standard for Transmission of IP Datagrams over Serial Lines: SLIP*, Web Page, 1988. Available at <http://www.cis.ohio-state.edu/htbin/rfc/rfc1055.html>

RySavy, P. and Mathias, C., "On the Road to PCS," *Network Computing*, pp. 74-80, 15 February 1996.

Sandisk Corporation, *Sandisk Home Page*, Web Page, 1996. Available at <http://www.sandisk.com/>

Schmoll, J., *Introduction to Defense Acquisition Management*, DSMC Press, 1993.

Science Applications International Corporation (SAIC), *Science Applications International Corporation*, Web Page, 1996. Available at <http://www.saic.com/>

Siig, J., "Access Internet with Your HP Palmtop," *The HP Palmtop Paper*, Fairfield, Iowa, 1996.

Silicom Connectivity Solutions, Inc., *Silicom Connectivity Solutions, Inc.*, Web Page, 1996. Available at <http://www.halcyon.com/wsa/directories/membership/SilicomConn/info.html>

Simone, L., "Data Transfer at the Speed of (Infrared) Light," *PC Magazine*, p. 36, 25 June 1996.

Simpson, W., *RFC 1661; The Point-to-Point Protocol (PPP)*, Web Page, 1994. Available at <http://www.cis.ohio-state.edu/htbin/rfc/rfc1661.html>

Solectek Corporation, *AIRLAN/Bridge Plus Wireless Campus Area Networks*, San Diego, California, 1994.

Somers, A., Blimp Based Bandwidth, *Mobile Office Magazine*, p 18, August 1996.

Tennenhouse, D., Bose, V., "The SpectrumWare Approach to Wireless Signal Processing," *Wireless Network Journal*, vol. 1, issue 4, 1996.

Trimble Navigation, *Mobile GPS Marketing Brochure*, Trimble, 1995.

Trimble Navigation, *Trimble Navigation Limited: The GPS Solution*, Web Page, 1996. Available at <http://www.trimble.com/>

United States Marine Corps (USMC), *FMFM 3-1; Command and Staff Action*, USMC, MCCDC, Quantico, Virginia, 1982.

United States Marine Corps (USMC), OH 6-1; *Ground Combat Operations*, USMC, MCCDC, Quantico, Virginia, 1986.

United States Marine Corps (USMC), *FMFM 1; Warfighting*, USMC, MCCDC, Quantico, Virginia, 1989. Available at <http://www.hqmc.usmc.mil/warfight/contents.htm>

United States Special Operations Command, Command, Control, Communications (USSOC), *Computers, and Intelligence (C4I) Architecture*, USSOC, Fort McDill, Florida, 1994.

University of Kansas, *DosLynx 0.8 Alpha Release Information*, Web Page, 1996. Available at <http://kuhttp.cc.ukans.edu/ftp/pub/DosLynx/readme.htm>

University of Minnesota, *Minuet--Text based WWW client*, FTP Site, 1996. Available at <ftp://minuet.micro.umn.edu/pub/minuet/latest/minuarc.exe>

Van Riper, P. *The Challenge of Command and Control*, video, Marine Corps Combat Development Center, Quantico, Virginia, 1995.

Velasquez, S., "Is 802.11 Just Around the Corner?", *Internetwork*, p. 18, July 1996.

Wexler, J., "Vendors Ally on Wireless LAN front," *Network World*, p. 25, 3 June 1996.

Woll2Woll Software, *Infopower Developer's Guide*, filename INFOPOWR.HLP, Livermore, California, 1995.

Xceed Software Inc., *Xceed Zip Compression Library Help File*, filename XCEEDZIP.HLP, Longueuil, Quebec, 1996.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
8725 John J. Kingman Road, Ste 0944
Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library.....2
Naval Postgraduate School
411 Dyer Rd.
Monterey, California 93943-5101

3. Director, Training and Education.....1
MCCDC, Code C46
1019 Elliot Rd.
Quantico, Virginia 22134-5027

4. Director, Marine Corps Research Center 2
MCCDC, Code: C40RC
2040 Broadway Street
Quantico, Virginia 22134-5107

5. Director, Studies and Analysis Division 1
MCCDC, Code C45
3300 Russell Road
Quantico, Virginia 22134-5130

6. Don Brutzman, Code UW/Br.....4
Undersea Warfare Department
Naval Postgraduate School
Monterey, California 93943-5000

7. Maj James Cummiskey.....10
Marine Corps Tactical Systems Support Activity
Camp Pendleton, California

8. Marine Corps Systems Command, Code C4ICIS-DSS.....1
203 Barnett Avenue, Suite 315
Quantico, Virginia 22134-5010

9. Kathy Houshmand, CSD 1
Marine Corps Tactical Systems Support Activity
Camp Pendleton, California 92055-5171

10. Rick Konrad, CSD 1
Marine Corps Tactical Systems Support Activity
Camp Pendleton, California 92055-5171

11. David Bartell.....1
Naval Air Warfare Center, Code 11XJ10C/MS-37
6000 E. 21st Street
Indianapolis, Indiana 46219-2189

12. Marine Corps Operational Test and Evaluation Activity 1
3035 Barnett Avenue
Quantico, Virginia 22134-5014

13. Ted Lewis 1
Chairman, Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5000

14. Lou Stevens, Code CS/S_ 1
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5000

15. Bert Lundy, Code CS/L_ 1
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5000

16. Geoffrey Xie, CS/X_ 1
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5000

17. Rex Buddenberg, Code SM/B_ 1
Systems Management Department
Naval Postgraduate School
Monterey, California 93943-5000

18. Capt James Nierle, USMC 1
Commander, Joint Interoperability Test Command
CODE: JTAC
Fort Huachuca, Arizona 85613-7020

19. Tawnya Wood 1
Intel Corporation
Fab 6 Materials Group, CH2-88
Chandler, Arizona 85226

20. LtCol William Card, USMC 1
Head, Ground Requirements
Marine Corps Combat Development Command
3250 Catlin Avenue
Quantico, Virginia, 22134-5001

21. Kheng Joo Khaw 1
General Manager
Asia Pacific Personal Computer Division
Hewlett-Packard Singapore (Pte) Ltd
1150 Depot Road
Singapore 109673

22. Christopher Koh.....1
Developer Program Manager
Asia Pacific Personal. Computer Division
Hewlett-Packard Singapore (Pte) Ltd
72 Bendemeer Road #01-01/07-01
Singapore 339941

23. Scott McNinch 1
Business Development Manager
Hewlett-Packard Company
20 Perimeter Summit Blvd.
Atlanta, Georgia 30319-1417

24. J. W. Gowens 1
Chief, Telecommunications Division
Army Research Laboratory
Attn: AMSRL-IS-TG
115 O'Keefe Building
Atlanta, Georgia 30332-0862

25. Edward J. McCarthy.....1
C2 & Systems Integration
Mitre Corporation
Washington C³ Center
145 Wyckoff Road
Eatontown, New Jersey 07724

26. Dan Boger 1
C4I Academic Group Chair
Naval Postgraduate School
Monterey, California 93943-5000

27. Katie Merwick.....3
Microsoft Corporation
Code 10S-1
One Microsoft Way
Redmond, WA 98052-6399

28. Capt George Zolla, USN (Ret).....1
Library, Code 52
Naval Postgraduate School
Monterey, California 93943-5101

29. Office of the Under Secretary of Defense for Acquisition and Technology 1
DDDR&E, Office of Laboratory Management and Technology Transition
Pentagon, Room 3D129
Washington, DC

30. Senate Armed Services Committee.....1
SR-228 Russell Senate Office Building
Washington, DC 20510-6050

31.	House National Security Committee	1
	Room 2120 Rayburn House Office Building	
	Washington, DC 20515	
32.	Defense Advanced Research Projects Agency	1
	Attn: John Schull	
	3701 North Fairfax Drive	
	Arlington, VA 22203-1714	
33.	Michelle Votava	1
	Microsoft Corporation, Code 10s/1	
	One Microsoft Way	
	Redmond, WA 98052	
34.	Captain David Duff	1
	6 New Brunswick Circle	
	Stafford, Virginia 22554	