PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/14, 13/20, H04L 12/64 — A1

(11) International Publication Number: **WO 99/45471**

(43) International Publication Date: 10 September 1999 (10.09.99)

(21) International Application Number: PCT/US99/04985

(22) International Filing Date: 5 March 1999 (05.03.99)

(30) Priority Data:
09/036,097 6 March 1998 (06.03.98) US

(71) Applicant: MOBILE INFORMATION SYSTEM, INC. [US/US]; 1333 Bordeaux Drive, Sunnyvale, CA 94089 (US).

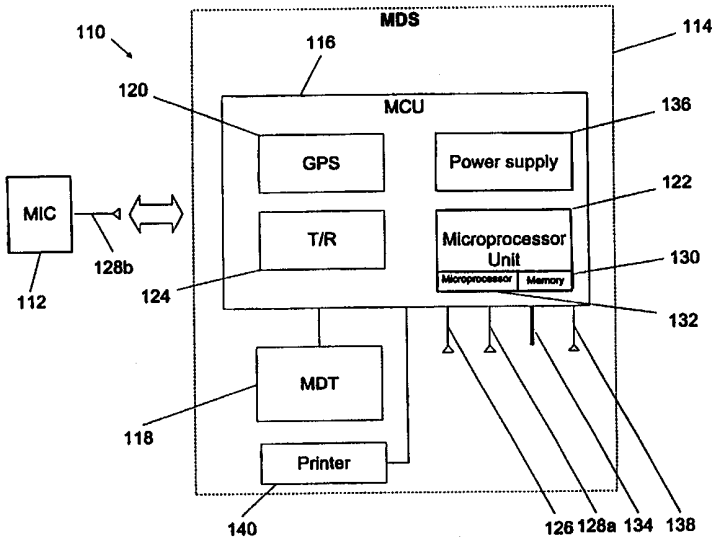
(72) Inventor: PRABHAKARAN, Sanjiv; 1484 Rojo Place, San Jose, CA 95128 (US).

(74) Agents: OGAWA, Richard, T. et al.; Townsend and Townsend and Crew LLP, 8th floor, Two Embarcadero Center, San Francisco, CA 94111-3834 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: MOBILE DATA SUITE AND METHOD



(57) Abstract

According to the present invention, a technique for processing data is provided. The invention provides a flexible way to process data such that a user may conveniently enter and transfer data. Also, the user may have ready access to powerful data processing. In an embodiment of the present invention, a novel system (110) for mobile data processing is provided. The system includes an interface unit (116) that includes a processor (122). A positioning system (120) couples to a first antenna (126) and to the processor. A remote data terminal (118) electrically couples to the interface unit during at least a first time period. The remote data terminal is capable of data transfers with the interface unit during the first time period and with a user.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

MOBILE DATA SUITE AND METHOD

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a nonprovisional application of, and claims priority from, commonly assigned U.S. provisional application Serial No. 60/040160, filed on March 7, 1997 (Attorney Docket No. 15517-001300), incorporated herein by reference. This application is also related to Application Serial Nos. _____ (Attorney Docket No. 15517-001320) and _____ (Attorney Docket No. 15517-001330) filed on the same date of this present application, all in the name to the present assignee. All these documents are hereby incorporated by reference for all purposes.

COPYRIGHT NOTICE

A portion of the disclosure of this patent contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all Copyright rights whatsoever.

BACKGROUND OF THE INVENTION

The present invention relates to an apparatus and system for data processing and, more specifically, data processing as it relates to transportation management. The present invention is illustrated by way of an example with regard to an apparatus and method capable of remotely transmitting and receiving information, especially information relevant to fleet management such as delivery locations and times. But it will be recognized that the invention has a wider range of applicability. Merely by way of example, the invention can be applied to other types of transportation, mapping, information management, and the like.

As the world becomes more industrialized and populated, transportation requirements also increase rapidly. In particular, the number of vehicles such as

automobiles, trucks, vans, and the like on typical city highways has increased to levels such that traffic jams are now a way of life for a typical driver using these highways as a means for travel. In fact, some of these highways are so constricted that anyone using them can experience significant delays often unexpectedly due to problems such as accidents, road construction, and others. These problems also exist on other transportation ways such as our city streets, airways, and waterways. Accordingly, it is often difficult to predict with any accuracy the location of a vehicle using these transportation ways.

Cities and governments have attempted to resolve some of these problems by adding more transportation infrastructure in highly populated areas. This infrastructure often comes in the form of improved roads or highways, train systems, and the like. Unfortunately, roads, highways, and train systems are often difficult to build in highly populated areas and are generally extremely expensive and time consuming to build. In most cases, construction used to provide this additional infrastructure often causes even more traffic congestion and other problems.

Based upon this state of the transportation infrastructure in most industrialized countries, it is often difficult for a company involved in the courier or delivery business to accurately track its vehicles and deliveries. The problems mentioned above severely limit the predictability for a fleet manager to track vehicles in its fleet for the pick-up and delivery of information, packages, and people. Moreover, it is desirable but difficult to keep the fleet manager up-to-date about the status of members of the fleet and to update the fleet members with information from the manager.

Industry also has attempted to resolve some of these problems. For instance, some companies are now providing their couriers with cellular phones and radios so that a dispatcher can communicate with them. Other companies retrofit their vehicles with navigational systems such as LORAN or a global positioning system (GPS) to determine vehicle location. Still, other companies are using maps and GPS to track vehicle location by dispatchers at a central office terminal.

One such company is Mobile Information Systems, Inc. ("Mobile Information Systems"), assignee of the present application, which pioneered a

technique for implementing easy-to-read maps for tracking vehicle location on a display or workstation at the central office terminal or any other terminal. In particular, Mobile Information Systems implemented one of the first techniques for using a raster-type map and vector data for referencing vehicle location. The raster-type map used on a display had features that were easy-to-read for a dispatcher or user. These features were generally geographical in nature and were easier to reference than the maps predominantly made using stick-type representations of geographical features. The techniques used by Mobile Information Systems have partly overcome some of the daily problems faced by a fleet manager or the like. It would, however, be desirable to develop other techniques for integrating further aspects of fleet management.

Based upon the above, it would be desirable to develop a device for improving a user's ability to create and receive data to help with the predictability, efficiency, and accuracy of task management such as fleet management or tracking any object that can be transported into our roadways, highways, waterways, airways, and the like.

SUMMARY OF THE INVENTION

According to the present invention, a technique for processing data is provided. In an exemplary embodiment, the invention provides a flexible way to process data at a remote location such that a user may conveniently enter and transfer data and also have ready access to powerful data processing. The present technique can be used in a variety of applications such as transportation and the like.

In an embodiment of the present invention a novel system for mobile data processing is provided. The system includes an interface unit (e.g., hand-held unit, mobile data terminal, personal information manager, commonly known as PIM or the like) that includes a processor, e.g., microprocessor, digital signal processor, microcomputer. A positioning system couples to a first antenna and to the processor. A remote data terminal electrically couples to the interface unit during at least a first time period. The remote data terminal is capable of data transfers with the interface unit during the first time period and with a user. This system allows a user to take the

remote data terminal on errands away from the interface unit, and transfer data to and from the interface unit.

In another embodiment, the remote data terminal is adapted to be hand-held. This allows the user to carry the remote data terminal on errands. Thus, the user can enter data conveniently in real-time when the user receives data. This, for example, allows the user to avoid writing the data onto paper only to be entered electronically later.

In still another embodiment, the present invention provides a method of data processing including receiving user data in a control unit, receiving positioning data from a first antenna of the control unit, and transmitting the user data and the positioning data, using a second antenna of the control unit, to a base station. This embodiment provides a method by which user data may be combined with positioning data thereby providing an indication of not only the substance but also the origin of the user data.

In yet an alternative embodiment, the present invention provides a microprocessor based system using a novel set of instructions or computer codes. The computer codes form a computer program to carry out the functionality and methods described herein. The functionality and methods are described throughout the present specification and more particularly below.

This invention provides myriad advantages. For example, quick access could be gained to valuable information such as the user's current location, speed, direction, destination, schedule, estimated time to destination, and required time to destination in some embodiments. The present invention can also store and transmit precise tracking information regarding the user's past, present, and future positions and locations at noteworthy times such as when the user reaches certain destinations including pickup and delivery points in other embodiments. When used in conjunction with scheduling techniques, the present invention can, for example, improve user efficiency by reducing the time and cost of travel between destinations while providing improved tracking of the user, packages, or the like. The present invention can provide accurate location and tracking information automatically, without requiring the user to enter this information which may require additional time or introduce human error.

Also, the present invention may provide these and other advantages in a convenient and portable package. Of course, the present invention provides other advantages. Hence, the description provided here is only exemplary and not an exhaustive list.

Some of the novel features of the invention are set forth in the appended claims. The invention, however, as well as other features and advantages thereof, will be best understood by reference to the detailed description which follows, when read in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a simplified block diagram of a data processing system according to an embodiment of the present invention;

Fig. 2A shows a simplified front view of an exemplary mobile data terminal according to an embodiment the present invention;

Fig. 2B is a simplified block diagram of an apparatus according to an embodiment of the present invention;

Fig. 3 is a simplified flow diagram of a method according to an embodiment of the present invention;

Fig. 4 illustrates a simplified sample startup activation screen on a display of the mobile data terminal according to an aspect of the present invention;

Fig. 5 shows a simplified sample warning screen on the mobile data terminal display in yet another aspect of the present invention;

Fig. 6 shows a simplified sample welcome screen on the mobile data terminal display according to an aspect of the present invention;

Fig. 7 shows a simplified sample mobile data terminal display primary menu according to an embodiment of the present invention;

Figs. 8A and 8B illustrate exemplary forms menus for the mobile data terminal display according to an embodiment of the present invention;

Figs. 9A and 9B show exemplary status menus for the mobile data terminal display according to an aspect of the present invention;

Fig. 10 shows a simplified sample secondary menu for the mobile data terminal according to an embodiment of the present invention; and

Fig. 11 illustrates a simplified sample shutdown screen for the mobile data terminal according to an embodiment of the present invention.

DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

The following descriptions use several acronyms and abbreviations. For the reader's convenience, the following is a list of the more frequently used acronyms and abbreviations:

GPS	Global Positioning System
MCU	Mobile Control Unit
MDS	Mobile Data Suite
MDT	Mobile Data Terminal
MIC	Mobile Information Center
T/R	Transmitter/Receiver

Hardware Description

Fig. 1 shows a simplified overall system 110 in an information management system, such as a fleet management system, according to an embodiment of the present invention. This diagram is merely an illustration and should not limit the scope of the claims herein. One of ordinary skill in the art would recognize other variations, alternatives, and modifications. System 110 includes a mobile information center (MIC), or base station, 112 that interacts with a mobile data suite (MDS) 114. The MIC may include a system, such as a software system, to manage several MDS users to efficiently control fleet use. As merely an example, U.S. Serial No.08/706,211, commonly assigned, describes a MIC according to the present invention. This application is incorporated by reference herein for all purposes. Of course, other types of systems which have similar functionality as the MIC described herein can also be used.

As shown, the MDS is an integrated module including a mobile control unit (MCU) 116 and a mobile data terminal (MDT) 118. In an embodiment, the MDS is an on-board module, adapted to fit conveniently in a fleet vehicle such as a car, van, or truck. Preferably, the MDS is easily portable, fitting in a carrying case no bigger

than approximately 18"×12"×10" and weighing approximately 10 pounds or less. This portability allows a user to remove the MDS from the vehicle for safe storage or for convenient maintenance away from the confines of the vehicle. In an embodiment, the MDT may act as an interface between the MCU and the user for receiving and displaying information.

Preferably, the MCU includes a global positioning system (GPS) 120, a microprocessor unit 122 (e.g., a microprocessor board, a microcomputer, a microcontroller, a programmable controller), and a transmitter/receiver (T/R) unit 124. The GPS allows monitoring of positioning-related information, e.g., latitude and longitude. Positioning data enters the GPS via a GPS antenna 126. The GPS antenna may be mounted on the exterior of the vehicle, but is not limited to this location. This may improve reception of the antenna while reducing space consumed by the system inside the vehicle. A magnetic base may be used to mount the antenna that will allow a user to quickly and easily remove the antenna for storage, to improve reception, or for other purposes. From the positioning data, the GPS can determine the current or past location of the MCU based on information received from global positioning satellites. The GPS, the microprocessor unit, or the MDT (including MDT software) either alone or in combination, may use the information received by the GPS to determine information related to system management such as the vehicle's speed and heading. Alternatively, equipment at the MIC or elsewhere in the MCU, such as in the microprocessor unit, may determine this system management information. In an embodiment, the GPS is a Trimble OEM GPS receiver card made by Trimble Navigation, although other GPS systems can be used.

Any form of positioning system capable of determining the location of the MCU could replace the GPS. For example, a system using information from local position detectors instead of global positioning satellites could be used.

The GPS is generally not required and may be omitted from the MCU. In particular, not using a GPS may reduce the size, weight, complexity, initial cost, and maintenance expense of the MCU. Eliminating the GPS may be desirable for applications when system size and weight are at a premium such as for a bicycle messenger.

In an embodiment, the MCU T/R receives data from, and transfers data to, the MIC via an MCU T/R antenna 128a. The MIC receives data from, and transmits data to, the MCU via a MIC antenna 128b (the data transfer represented by a double-ended arrow). Data from the microprocessor or GPS may be sent to the MIC or data from the MIC may be received and transferred to the GPS or microprocessor as needed by the MCU T/R. The MCU T/R may process received data as necessary to be in a form compatible with its destination. The MCU T/R is preferably a radio T/R, such as a radio frequency radio modem, due to cost and maintenance advantages. For example, the MCU T/R may be a RAM compatible Mobitek Modem made by Mobitek. Other forms of T/R units, however, may be used depending upon the application. The MCU T/R antenna, similar to the GPS antenna, may have a magnetic base, or be otherwise adapted, to assist mounting the antenna on the exterior of the fleet vehicle. To reduce interference, the GPS antenna and the MCU T/R antenna may be mounted at least about 12 inches apart.

In an embodiment, the microprocessor unit may act as an interface unit in the MDS. For example, the microprocessor may provide an interface between the MCU T/R and the MDT, the MCU T/R and the GPS, or the GPS and the MDT. Included in the microprocessor unit are a memory 130 and a microprocessor 132. The memory may, for example, store messages for the user of the MDS. These messages may come from, for example, the MIC, the GPS, or the user. Messages may be categorized into groups for convenience such as received but not yet read, received and previously read, and sent. The microprocessor can access desired portions of the memory for data insertion or retrieval. The microprocessor may also process data from the memory, the MIC, the GPS, or the MDT before transferring the data to the memory, the MIC, the GPS, or the MDT. Examples of such processing may include determining speed and heading information based upon positioning data.

In an embodiment of the present invention, the MDS architecture includes the MDT, MCU, GPS, microprocessor unit, and MCU T/R. The microprocessor unit may act as the central controller of the MDS, directing information flow between the components and storing information as necessary. The GPS receives information through its antenna and the microprocessor unit may store this data and/or

direct it to the MCU T/R for transmission to the MIC or to the MDT for user viewing. Information from the MDT may also pass to the MCU T/R and/or be stored in memory under the direction of the microprocessor unit. The MCU T/R may then transmit the data to the MIC. The MCU T/R may also receive information from the MIC which the microprocessor unit may then direct into memory and/or to the MDT.

A power cable 134 may connect the MCU to an external power supply. The external power supply may be the same power supply used by the vehicle. For example, the power cable may have a connector (not shown) that fits into a cigarette lighter socket of the vehicle. The MCU would then use the electrical energy from the vehicle such as a 12-volt battery. Using an external power supply could help reduce the size, weight, and cost of the MDS.

Alternatively, the MCU may have an internal power supply 136 in addition to, or in lieu of, the external power supply. Having an internal power supply would allow the MDS to operate independently of an external power supply. The user could, for example, use the MDS in places which would be inaccessible if the MCU was dependent upon a vehicle power supply. Also, having an internal power supply would allow the user to use the MDS as needed in case of a failure of the vehicle's power supply. For example, the user could relay information to the MIC if the user is in an accident that causes the battery to stop functioning, or during a malfunction which prevents energy from reaching the power cable.

The MDS may also include a printer 140. Having a printer would, for example, allow the user to make a hard copy of data received from the MIC, transmitted to the MIC, or entered into the MDT even if not transmitted to the MIC. The printer may also be utilized to provide a hard copy of other information such as the configuration of the MDS, GPS, MDT, or the MCU.

Fig. 2A illustrates an exemplary front view diagram of an MDT 200 according to an embodiment of the present invention. This diagram is merely an illustration and should not limit the scope of the claims herein. One of ordinary skill in the art would recognize other variations, alternatives, and modifications. Data from the GPS and MIC may be illustrated on a display 202 of the MDT. The display permits the user to visually inspect the displayed data and act accordingly. The display can be any

suitable output device such as a liquid crystal display or an active matrix liquid crystal display, as well as other types of displays, e.g., laser. diodes. The display should have a sufficient region for providing information (e.g., text, numbers) in an easy to read manner to a user. Additionally, the display can work under limited or low power conditions in some embodiments. The display also can be used as an input device such as a touch sensitive screen used in, for example, a product called PalmPilot™ made by 3Com™ Corporation of California. Preferably, the display is also resistant to extreme environmental temperature ranges (e.g., freezing) and is shock resistant. Furthermore, the display is sealed or isolated from moisture and particulates such as "dust" or contamination. These and other features will become more apparent throughout the present specification and more particularly below. Additionally, although the above display is configured in the MDT, the invention also has other applications. For example, the display can be detached from the MDT. The display can be loosely coupled to the MDT. Alternatively, the display can be integrated into another device such as a watch, a helmet, glasses, clipboard, and the like.

In an embodiment, information from the MIC such as changes in delivery or pick-up schedules, lunch break approvals, emergencies, traffic conditions, vehicle location, and personal messages may be displayed. Also, if the MDT is a pager, the user can receive business, personal, and other messages such as reminders, phone numbers, and emergencies. In an embodiment, the received data may include general information that may be broadcast to many users simultaneously, such as sports scores or other news. The display can also be coupled to another output device such as a beeper, a pager, electric shocker, or a vibrator to alert the user in a specific or selected situation such as an emergency.

In an embodiment, the MDT may also receive data from the user. In such an embodiment, the MDT may have a data entry portion 204. As shown in Fig. 2A, the data entry area may be a keypad. Other types of data entry tools may be utilized including a flat-panel keypad, a recorder that receives and stores sounds such as speech, a voice recognition unit which could recognize speech and convert the speech into data indicative of the speech such as text data, a touch-sensitive display, a display area allowing the user to write or draw characters or symbols such as a signature (e.g.,

pen computing), a bar-code reader, or a scanner (e.g., an optical character recognition device). If the data entry area is one of the last two examples, the data entry portion may coincide with the display 202. Alternatively, if the data entry portion receives data through writing or drawing on a display, the MDT or MCU may include circuitry to recognize hand-writings or signatures. This ability, for example, could assist the user in determining whether the person signing for a package has the authority to do so.

As shown in Fig. 2A, the MDT may include a housing 206 adapted to fit in the user's hand, for example. Such a hand-held design permits the user to hold the MDT with one hand and enter data with the other hand. Alternatively, if the data entry portion receives audio input, the user may conveniently hold the MDT with one hand, enter data, and still have one hand free. Such an arrangement also permits a user with physical handicap to use the MDT more conveniently. The housing is preferably made of a suitable material to withstand environmental variations such as temperature and weather. Accordingly, it is desirable to have a housing that is resistant to moisture and particulate contamination. This feature can be achieved by way of seals such as o-type rings and rubber gaskets, which seal one member of the housing with another member of the housing.

Furthermore, the housing can be sufficiently rigid to withstand mechanical shock, although other embodiments may require a flexible or "soft" housing for ergonomic purposes. In these embodiments, the housing can have a coating made of a soft or flexible synthetic material, which tends to be easier to handle with "hot" and "sweaty" palms, for example. Preferably, the housing is also made of an isolating or shielding material which can electrically isolate the internal electronics from external transmission lines that can lead to "noise" or multi-path influences. The housing is also chemically resistant and inert to isolate the internal components from chemical influences. Although some desirable features have been described, numerous other features can be implemented into the present housing design.

Additionally, the MDT is detachable from the MCU in some embodiments. A detachable design allows the user to use the MDT away from the MCU. Thus, the user may not need to carry the whole MCU in order to use the system. Furthermore, an internal MDT power supply 208 would permit the user to use the MDT

away from the MCU. The MDT power supply may also be mounted externally, for example, on a belt clip. Such a configuration may provide a lighter MDT which may be carried by hand for a longer period to distance. The MDT is generally less than 5 pounds, less than 2.5 pounds, or less than 1 pound to merely ounces in preferred embodiments.

If detachable, the MDT can be able to transfer data to, and receive data from, the MCU when connected to the MCU. In this embodiment, the user enters data while away from the MCU and later connects the MDT to the MCU. Once connected, the MDT and MCU can transfer data to each other. This configuration obviates driving or even having an antenna, and associated circuitry, in the MDT for receiving and transferring data while detached from the MCU.

Alternatively, the MDT may receive, transmit, or transmit and receive data while detached from the MCU. In such an embodiment, the MDT may have an MDT antenna 210 and associated circuitry, which is shown in reference to Fig. 2B. The MDT antenna may be internal, as shown, or external. The MDT can transfer data back and forth with the MCU, MDS, and MIC via the MDT antenna. To reduce the circuitry needed and/or power consumed, the MDT may be restricted to receiving data only (such as if the MDT is a pager), or transmitting data only. Depending upon the frequency chosen for remote data transfer between the MDT and MCU, the MCU may require an additional MCU/MDT antenna 138.

Given the various embodiments described above, the MDT may take several forms. In an embodiment, the MDT can be encased in a portable data terminal (PDT) 3100 made by Symbol Technologies, Inc. Alternatively, the MDT may be encased in an electronic personal organizer such as the PalmPilot™. The MDT may also include, or simply be, a printer.

Fig. 2B is a simplified block diagram of the MDT shown above. The simplified diagram includes, among other elements, a GPS 251, a transmitter/receiver unit 253, a microprocessor unit 254, a power supply 263, a printer driver 261, a display driver 259, which are coupled to each other by way of a common bus(es) 257. The GPS determines positioning information, which can be displayed by way of the display through the display driver or which can be sent to the MIC through the

transmitter/receiver unit. Additional output of information can be directed to an optional printer by way of the printer driver. The power supply provides energy in the form of electrical voltage and current to elements of the MDT. Preferably, the power supply is a rechargeable battery such as "NiCad" or the like. Alternatively, low voltage applications can be driven by a solar power supply unit, which can also be used to recharge the battery in some embodiments. Antenna 210 is coupled to the transmitter/ receiver. In an embodiment, the antenna is capable of communicating through radio frequency radio signals. The location and type of the antenna, however, is merely exemplary and one of ordinary skill in the art would recognize other variations, alternatives, and modifications. For example the antenna may be internal or external to the device, or may utilize other types of signals to communicate, such as infra red. Overall functionality of the MDT is often overseen using the microprocessor based unit in the present embodiment.

Fig. 3 shows an example of a simplified method 300 according to an embodiment of the present invention using a system such as that shown in Figs. 1 and 2. This method is exemplary only and does not limit the claims to this embodiment. The order in which the steps appear in the figure are largely arbitrary and may appear in many orders different from the specific order shown in Fig. 3.

The method starts at step 302 and proceeds to step 304, when user data is entered into the MDT. This data entry may take any of numerous forms depending on the particular MDT used. For example, the user may enter data by pressing keys on a keypad, touching portions of a touch screen, scanning a document, or reading a bar code.

At step 306, the user data is transferred from the MDT to the MCU. Again, this data transfer may take any of numerous forms. For example, the data transfer may be accomplished via the MDT antenna, or by coupling the MDT and MCU using a cable, or by positioning the MDT into a cradle in the MCU having MCU contacts adapted to couple to corresponding MDT contacts.

At steps 308 and 310, the MDS receives positioning data via the GPS antenna and MIC data via the T/R antenna. As described above, the positioning data could also come from local position detectors or other positioning systems. The GPS

and T/R antennas may be specifically designed or selected to reduce the power needed to provide adequate data transfer between the MCU and the positioning system or the MIC.

In step 312, the MCU processes MCU data (including, for example, the user data, positioning data, and MIC data) as needed. Such processing may include manipulating the user data, positioning data, and MIC data and converting them into formats compatible with the MIC or the MDT. For example, the positioning data may require manipulation in order to display this information on the MDT. Also, this step may include the processing needed to determine information such as vehicle speed and heading.

In step 314, the MCU data is transferred to the MIC. All, selected part, or none of the information available for transfer may be transferred. This data transfer may typically involve sending the data via the T/R antenna to a corresponding antenna at the MIC.

In step 316, the MCU transfers data to the MDT. All, selected part, or none of the information available for transfer may be transferred. This data transfer may occur in the same manner as described above with respect to transferring data from the MDT to the MCU. The MCU may be configured to transfer data when a communication link between the MCU and the MDT exists, or periodically. For example, the MCU may accumulate data for transfer to the MDT for a predetermined time period, then transfer some part of the accumulated data at the end of the time period. Alternatively, the MCU may transfer data to the MDT in real-time, as soon as data is ready for transfer.

In step 318, the MCU data is displayed on the MDT display. The displayed data may include the MCU data transferred from the MCU, the user data entered into the MDT, and/or predetermined data not entered by the user or received from the MCU. An example of the predetermined data would be prompts provided to the user to request data input, text or symbols displayed near the user data, or MCU data that may provide an indication of the significance, or meaning, of the displayed user or MCU data.

Finally, the process terminates at step 320. While the above method has been described using a specific order, many combinations and permutations of the order presented are possible. The steps may appear in almost any order within a few logical guidelines. These guidelines may merely ask that data exists before an attempt to transfer or process it. For example, step 306 of transferring user data from the MDT to the MCU may follow step 304 at a later time. Likewise, step 312 of processing MCU data may follow any combination of the steps that provide data to the MCU, such as steps 306 through 310. Similarly, steps 314 and 316 of transferring MCU data to the MIC and the MDT may follow any combination of the steps that provide data to the MCU. And, step 318 of displaying MCU data on the MDT display may follow the step of transferring MCU data to the MDT at a later time.

Additions and omissions to the steps of Fig. 3 is also acceptable. For example, the MCU need not receive MIC data from the MIC. As a further example, the MCU may not even have GPS (or other positioning system) capabilities and, hence, the method need not include receiving positioning data. Other deletions are also possible. Moreover, the method may include more steps or sub-steps including transferring data to a RAM network that may provide coverage nationwide, globally, and beyond.

Software Description

In an embodiment, the present invention includes software to control the user interface and data processing operations. The software may partly or completely reside in the MCU or MIC, but preferably resides in the MDT. Thus, while the software is sometimes referred to below as the MDT software, this label refers to an, but not the only, possible embodiment of the present invention. In some embodiments, the MDT may be encased in a Symbol terminal, such as Series 3100, that has its own software. The software described below may be an addition, a modification, or a replacement of software supplied with the Symbol terminal. The software of the present invention may control receipt, processing, and transmission of data between the MCU, MDT, MIC, user, etc.

Moreover, the software described below, in accordance with an embodiment of the present invention, may support several configurations of the MCU.

Moreover, one configuration does not necessarily include more or less hardware than another configuration. For example, an external modem configuration may support a Symbol Series 3100 terminal, an external RAM-compatible Mobitek Modem (sometimes referred to as a Mobidem), and a Motorola 505sd modem. This configuration may, for example, not support GPS functionality.

Similarly, a black box configuration may support a Symbol Series 3100 terminal and a black box including a RAM-compatible Mobitek Modem module, and a Trimble OEM GPS receiver card. This configuration, for example, could support, among others, modem communications and GPS information receipt, processing, and transmission.

The software preferably runs on the MDT, interfaces with the external Mobitek modem or a black box including the GPS and the MCU T/R. Also, the MDT software preferably includes a traffic manager and a report scheduler.

Preferably, the MDT software is organized in a modular fashion. This arrangement may provide compartmentalized functionality to assist with creation and modification of the software code and debugging of problems. For example, an engine module may communicate with the GPS. This module could prepare data for transmission to the GPS and process data received from the GPS. The data transmitted and received may or may not pass through or be stored in the microprocessor unit. In some embodiments, the engine module supports Magnavox 4200 and engine receivers used in the GPS.

In an embodiment, a hardware initialization module may provide initialization and hardware interface functions for the MDS. In this respect, the MDT may initialize the entire MDS by transmitting initialization parameters to the MCU to initialize the MCU T/R, the GPS, and the microprocessor unit. After initialization, this module can provide interface functions among the various MDS components. In a preferred embodiment, this module may also provide support for an event timer facility of the MDS. The event timer facility could, for example, allow users to have a number of timers based on the Symbol terminal system timer.

In another embodiment, an MDT module may provide MDT-specific functions. For example, the MDT module may support data entry forms (e.g., a

package delivery form or a hospital admittance form) that are larger than the physical size of the screen by allowing scrolling of the forms. In addition, a number of routines could support automatic list building by selecting and inserting an identification number in appropriate form field. The MDT module may provide a recall of information that is the same from job to job, such as customer-related information, by having the user provide the identification number. This would save the user time and effort in entering such information. For example, the customer's address, phone number, billing, and special instruction information may be recalled based upon entry of the customer's name or other identification tools such as the identification number.

Other features may include using a Menus module to support scrolling menus, time and distance position reporting, data compression, and blinking overlaid indicators for canceled, re-transmitted, and changed jobs. The user may select items in each menu by scrolling or entering an associated number. To exit from any menu, the user may press a clear key, etc. Some examples of the menu operations and options are discussed in more detail below.

In a preferred embodiment, an MDT utility module may provide utilities for an MDT interface. The MDT interface may include a display screen upon which data may be displayed prompting the user for information or providing the user with information. To assist the user with the MDT interface, the MDT utility module may provide processes for, among other things, creating menus, positioning a cursor in the menus, and checking for input from the user.

Fig. 4 shows a sample startup activation screen. This screen displays general information and, for example, prompts the user to press the acknowledge key to activate the system. In an embodiment, a timer requires the user to press the enter key within a predetermined time (e.g., five seconds) or the MDT would shut down, requiring reactivation by the user before use. Alternatively, the startup screen may ask for the user to enter a security code. This will, for example, only provide authorized personnel access to the MDT. Other methods of startup activation may include finger-print recognition, retina recognition, etc.

Fig. 5 shows a sample warning screen displayed by the MDT at least for safety purposes. This or other warning screens may be displayed for different purposes.

For example, the screen may display contact information in case the MDT is lost by the rightful owner.

Fig. 6 shows a welcome screen that the MDT may display. Information regarding matters including company name, version of the software, copyright, patent, trademark, or other intellectual property protection may be displayed on this screen. This screen may also display contact information in case the MDT is lost by the rightful owner.

Fig. 7 shows an primary menu 700 of the MDT with an upper portion 702 including status indicators, and the current date and time, and a lower portion 704 providing information about user selectable functions and the corresponding keys. The MDT status indicators shown in Fig. 7 provide information about the current operating status of the MDT. A NEW indicator 706 flashes when new messages are received. An indicator 707 displays the number of new messages. A LINK indicator 708 informs the user as to whether the network link is UP or DOWN. In an embodiment, the MDS communicates with the MIC only if the link is UP. A BATT indicator (not shown) may replace the LINK indicator when the MDT is disconnected from the MCU. The BATT indicator may provide general energy status of a battery in the MDT by, for example, indicating that the battery is either GOOD, LOW, or DEAD. Any of the MDT indicators may flash to indicate an abnormal status. A number 709 next to a pending indicator 710 displays the number of messages currently awaiting transmission. A number 711 next to a saved indicator 712 displays how many messages have been saved. A GPS indicator 714 provides information as to the age of GPS data and current GPS navigation capabilities. An "UNK" status may designate an unknown GPS status. Similarly, "N/A" may indicate that GPS information is unavailable or "OLD" may indicate that the most recent GPS data is older than 10 seconds. "NV2" may indicate that only 2-D navigation is available (i.e., only 3 satellites are visible) while "NV3" may indicate that 3-D navigation is available (i.e., more than 4 satellites are visible).

A signal-strength indicator 716 may indicate the signal strength of the modem. The signal-strength indicator may display up to six bars (e.g., right-facing arrowheads in Fig. 7). Three or more bars may indicate very strong communication signal, providing a very good coverage area.

The lower portion of the primary menu displays several options available to the user by pressing various keys. For example, pressing the F1 key may display a help description. Pressing the PREV or NEXT keys may then step through various help pages. To return to the primary menu, the user can press the MENU key.

Pressing F2 from the primary menu may allow the user to view sent messages. In an embodiment, the MDT retains the four most recently sent messages. Pressing F2 from the primary menu may cause the MDT to display a list of sent messages with corresponding current statuses such as delivered or pending and the time each was sent. Using the PREV/up-arrow and NEXT/down-arrow keys may step through these messages. An extended beep sounds may warn the user when no further scrolling is possible.

In an embodiment, to view saved messages the user may press F3 from the menu 700. The MDT then displays the latest saved message which could be scrollable as discussed above. Moreover, the user may delete sent and saved messages by navigating through the appropriate screens. Before deleting any messages, the MDT may confirm the deletion action.

Pressing F4 from the menu 700 may provide access to new messages. For example, if the NEW indicator in the primary menu is blinking, the MDT may have one or more new messages. Pressing F4 may cause the MDT to display the new messages, for example, with the newest message displayed first. Once the user reaches the new messages, the user may scroll through the various new messages.

Figs. 8A and 8B illustrate a simplified FORMS MENU 800. As shown, various options may be displayed on the FORMS MENU. This menu may also be scrollable to ensure all related information are displayed in the same menu. For example, Figs. 8A and 8B may be 2 of may available screens within the same menu 800. An area 805 may show an order of the present screen amongst the available screens within the same menu 800. The user may select a desired form by either highlighting the form's title using an up-arrow or a down-arrow key and pressing an enter key. Alternatively, numbers associated with each item may be entered.

Preferably, each form displayed in the FORMS MENU may be designed to ensure the user spends less time in preparing reoccurring information. For example,

form 810 may enable the user to enter orders by utilizing previously saved information in order to minimize data entry time and possible errors. Once a form is filled, the user may send the currently-selected form to, for example, the MIC.

Figs. 9A and 9B illustrates an exemplary STATUS MENU 900. Again, the user may select a desired form by using the up-arrow and down-arrow keys and the enter key, or by entering the appropriate code. The selected status screen is then displayed. If a different status is desired, the user may repeat the process, or may use the PREV and NEXT keys to cycle through the various status codes. Once the desired status is displayed, the user may send the status to, for example, the MIC. The user may also exit the status menu.

In an embodiment, the MDT keypad performs several generic operations according to the keys pressed. Table 1 lists some generic operations according to key stroke or combination of strokes that initiate corresponding functions.

TABLE 1

KEY	Function Performed
FORM	Displays the FORMS menu.
STATUS	Displays the STATUS menu.
F1	Executes function associated with F1 menu selection.
F2	Executes function associated with F2 menu selection.
F3	Executes function associated with F3 menu selection.
F4	Executes function associated with F4 menu selection.
MENU	Displays Primary menu.
SAVE	Saves currently displayed form to the "SAVED MESSAGES" list.
ACK	Sends an acknowledge message to the host when appropriate, and also may save the message.
NEXT	Performs the next logical operation in the current context.
PREV	Performs the previous logical operation in the current context.
ENTER	Moves the cursor to the next editable field in form editing mode. May also select a highlighted option.
CTL + SEND	Sends currently displayed FORM or STATUS.
DEL	Deletes a message.
CLR	Clears a field. In some menus, this key is used to go back to the previous or a primary screen.

KEY	Function Performed
CLR followed by CLR	Clears, when in edit mode, editable data from the form (by pressing CLR twice).
FNC	Selects the blue colored functions.
BSP	Deletes the character before to the cursor (e.g., BACKSPACE).
SPACE "FNC followed by BSP"	Inserts a space character.
LAMP	Toggles the back-light ON and OFF.
DARK and LIGHT	Cycles through the brightness level of the back-light.
UP ARROW (↑)	Selects next scrollable value for this field, scrolls up the displayed form, or produces error beep if at the top of the form.
DOWN ARROW (↓)	Selects previous scrollable value for this field, scrolls down the displayed form, or produces error beep if at the bottom of the form.
LEFT ARROW (←)	Moves to the previous character in an editable field of a FORM.
RIGHT ARROW (→)	Advances to the next character in an editable field of a FORM.
ALPHA KEYS	Input an alpha character.
NUMERIC KEYS	Input a numeric character.
SPECIAL CHARACTERS	The GREEN characters are special characters that can be inserted in any editable field.
FNC followed by SHF	Toggles the CAPS LOCK mode.

Fig. 10 shows a secondary menu of the MDT indicating various options for the user. In an embodiment, pressing a next key from the primary menu causes the MDT to display this screen. In the embodiment of Fig. 10, once at the secondary menu, the user may press any key (except F1 - F4) to go back to the primary menu.

Moreover, pressing F1 from the secondary menu may provide the user with time, speed, and direction information. This may be accomplished by displaying GPS information, if available. A time-speed-direction screen may show the current time, the speed of the vehicle, and the direction it is heading such as north, east, etc. It also may display how long the system has been turned on. In this embodiment, pressing any key returns the user to the secondary menu.

To turn OFF the MDT the user can select the F2 key from the secondary menu. If connected to the MCU, shutting down the MDT may also shut down the

MCU after the MCU sends a message to the MIC (which could take as long as five minutes or more if communication coverage is poor). It may take a few seconds to a minute before the MDT actually powers down.

Fig. 11 is an exemplary illustration of a shutdown screen according to an embodiment of the present invention. This screen may inform the user that the user will not receive messages while the MDT is shut down and prompts the user to proceed with shut down (ACK) or cancel the shut down request (CLR). Once the system has been shut down, the user may unplug the appropriate cables and remove the system from the vehicle for safe storage. The MDT may retain all data in the memory.

Returning to Fig. 10, pressing the F3 key may permit the user to enter a field-service screen. Access to this screen, however, may require a password and, preferably, only field service personnel have access to valid passwords or security codes.

In the embodiment of Fig. 10, the user may also view the MDT system application version number by pressing F4. From the system version screen, pressing any key may return the user to the secondary menu.

The MDS may be configured to save the last 40 job ID's (tag or identification numbers) to make it easier to enter them into an outgoing form rather than re-entering them using the alpha/numeric keys or other means such as a voice recognition, character recognition, etc. When the cursor is placed in the job ID field, the user may press the up-arrow or down-arrow keys to scroll through the list of the latest JOB IDs. Once the desired job ID is displayed in this field, the user can proceed to the next field using the ENTER key. Some IDs may be displayed in reverse order indicating that a "pickup form" was sent to the host.

Jobs in the MDT may be canceled by the dispatcher, retransmitted, or changed. In each of these instances, the MDT may display these conditions in a unique way. If a job is canceled, the MDT displays the job to be canceled with a big flashing "X" overlaying the job. The MDT automatically removes the job once the user acknowledges the message. If the job is retransmitted as is, the MDT displays the job with a big flashing "R." Finally, if the job is retransmitted with some changes in it, the MDT may detect that the job has changed and display the job with a big flashing "C."

The MDT may display these screens from either the primary or secondary menus. The retransmitted and changed jobs may replace the previous copies of the job. At any given time, the MDT may have only one copy of a job.

The MDT may also display various error messages to assist the user. Examples of error messages may include: invalid form number; invalid status; invalid queued message; could not translate incoming message; error saving data ... press any key to continue; error retrieving data ... press any key to continue; error setting default lat/long; RCV queue full ... incoming message was lost ... please delete SAVED/SENT messages or process NEW messages; out of memory ... incoming message was lost... please delete SAVED/SENT messages or process NEW messages; out of memory ... message was not sent ... please delete SAVED/SENT messages or process NEW messages; delete a message from the save queue before saving this message; etc.

Furthermore, the MDT may display various warning messages to assist the user. Examples of warning messages may include: Communications Out-Of-Range ... Press ACK to continue; GPS not available; Mobidem information not available; Running low on memory ... Please delete SAVED/SENT messages or Process NEW messages; Terminal cable out... Connect cable and press ACK to send shutdown message; etc.

In an embodiment, the MDT may be restarted by a warm boot. For example, if the MDT application appears to be frozen (e.g., text on the display does not change even after trying to go to a different menu) for any reason and nothing revives it, then the user may want to warm boot the MDT. To warm boot the MDT, the user may press the PWR key for about 30 seconds to shut off the MDT. Then the user may press the 4 and 5 keys simultaneously followed by pressing the PWR key twice. turn off in 15 seconds of no activity.

Trouble Shooting Guide

Table 2 indicates various possible trouble conditions and possible solutions to recover from them according to an embodiment of the present invention.

TABLE 2

	PROBLEM	CAUSE	SOLUTION
1.	MDT does not turn ON when disconnected from MCU	Battery may be weak.	Charge the battery by plugging in the MDT cable. If problem still persists, then replace the battery.
		Screen may have blanked out.	Hit any key to view data.
2.	PEND count increases.	Bad radio antenna or loose cable, LINK: DOWN	Re-position the radio antenna and tighten all cable connections.
		Out of coverage area or a dead spot area, LINK: DOWN "BATT" displayed in the Primary screen.	Move vehicle into an area of good coverage. Disconnect and reconnect the MDT cable so that BATT changes to "LINK: UP".
3.	Cannot SAVE any more messages	Memory buffers are full.	Delete messages from the various lists, such as SENT, SAVED or NEW message lists.
4.	MDT application is frozen	Unknown system error.	Try WARM BOOT first or else have a Field Service Personnel perform a COLD BOOT by resetting the unit.
5.	LINK says "DOWN" on the Primary screen	See item 2.	See item 2.
6.	"BATT: LOW" or "BATT: DEAD" displayed on the Primary screen	The MDT is disconnected from the MCU and the internal battery is weak.	Plug the MDT into the MCU so that the internal battery gets charged.
7.	MDT keeps even after NEW messages have been read	NEW messages have not been ACKNOWLEDGED	Use F4 to VIEW the new messages and acknowledge them by pressing the ACK (ENTER) key.

	PROBLEM	CAUSE	SOLUTION
8.	LCD Display screen on the MDT is UNREADABLE	MDT may have been exposed to direct sun-light for an extended period of time.	Move the MDT away from the sun. It may take several minutes to restore the display to normal. DO NOT EXPOSE the MDT to direct sun-light (The effect of sunlight on the face side is considerably worse than on the other side, so it is a good precaution to leave the MDT face-side down when it is on a car seat.)
9.	Date and time displayed is incorrect	The host (MIC) configuration for setting the clock may be incorrect or the MDT may not be communicating with the host.	Report problem to Mobile Information Systems, Inc.

Although the above description has been described in terms of hardware and software, the present invention is not limited to the specific hardware and software described. For example, the functionality described herein can be further combined in terms of hardware or further combined in terms of software. The hardware can also be separated or combined with other software. The software can also be separated from the hardware. Furthermore, the functionality can all be stored in the form of electronic data on an integrated circuit, for example. The integrated circuit can include, among others, DRAM, SRAM, FRAM, and Flash Memory Cells, as well as other integrated circuit devices in the form of "chips" or "cards." Accordingly, the present specification should not be construed as limiting the scope of the language of the claims herein.

Additionally, although an embodiment of the present invention is fully described above, implementation specific details and data structures are described in the

attached Detailed Design and Functional Specification in Appendix A. Various modifications, alternate constructions, and equivalents will be obvious to those with skill in the art. Thus, the scope of the present invention is limited solely by the appended claims and their full scope of equivalents.

APPENDIX A

Detailed Design and Functional Specification

Mobile Information Systems, Inc.

Table Of Contents

1. Introduction....5
2. Scope....5
3. Abbreviations and Definitions....5
4. Related Documents....5
5. Hardware Overview....5
6. Various Configurations....5
7. Software Overview....7
8. Module Level Description....9
8.1. MDS Architecture....9
8.2. MIS Module....9
8.3. ENGINE Module....11
8.4. HARDINIT Module....13
8.5. MDT Module....13
8.6. MDT_XLAT Module....18
8.7. MDTUTIL Module....18
8.8. MODEM Module....19
8.9. MODEM_XR Module....20
8.10. REPORTER Module....20
8.11. SYSUTIL Module....21
8.11.1. Queuing Functions....21
8.11.2. Circular Buffering Functions....21
8.11.3. Event Timer Functions....21
8.11.4. Semaphore Functions....22
8.11.5. Initialization Routines....22
8.12. COMPRESS Module....22
8.12.1. compress() function....22
8.12.2. un_compress() function....22
8.13. GPS_BIOS Module....22
8.13.1. GPSConnect() function....22
8.13.2. GPSInitialPosition() function....22
8.13.3. GPSPositionReport() function....23
8.13.4. GPSStatusReport() function....23
8.13.5. GPSSatelliteReport() function....23
8.13.6. GPSSetTime() function....23
8.13.7. GPSGetTime() function....23
8.14. MCU_INTF Module....23
8.15. MENUS Module....24
8.16. RAM_RFM Module....24
8.16.1. Function modem_xr_fsm()....25
8.16.2. Function SendAck()....25
8.16.3. Functions turn_modem_on() and turn_modem_off()....25

8.16.4. Function getRfmState()....25
8.16.5. Function modem_send_msg()....25
8.16.6. Function modem_sent_msg_status()....26
8.16.7. Function Modem_init()....26
8.16.8. Function Modem_shut_down()....26
9. Design Notes....26
9.1. User interactions....26
9.2. MDS_Init packet parameters definitions....27
9.3. Terminal ID in Demo Mode....29
9.4. Files on RAM Disk....29
9.5. Invalidation of Forms on MDS_Init message....29
9.6. Automatic update of sent messages list....29
9.7. Checksums maintained....29
9.8. JOBID Display Strategy....29
9.9. JOBID Maintenance Strategy....29
9.10. Reception of a new message....30
9.11. Out-of-cradle warning....30
9.12. Mobidem Status debug array....30
9.13. Setting of Initial GPS location....30
9.14. GPS polling rate....30
9.15. Time to first fix calculation....30
9.16. Distance traveled calculation....31
9.17. Accelerator Keys Support....31
9.18. Job Special Overlays....31
9.19. Position and Stats Reporting....31
9.20. Service Screens....32
9.20.1. System Debug Screen:....32
9.20.2. Mobidem Debug Screen:....32
9.20.3. Configuration Screen:....33
9.20.4. GPS Screen:....33
9.20.5. Raw GPS Screen:....34
9.21. GPS Status Report translation table (GPS to MIC)....34
9.22. Various System Parameters....34
9.23. Status Information Sent to Host....35
9.23.1. MDS_Stat Packet:....35
9.23.2. MDS_Pos Packet:....36
9.23.3. MDS_Alarm Packet:....36
9.23.4. MDS_Power_up Packet:....36
9.23.5. MDS_Power_down Packet:....36
10. System Data Definitions and Build Procedure Description....37
10.1. System Variables....37
10.2. Queues, Buffers, and Pools....38
10.3. MIS Packet Description....38
10.4. Makefile Description....38

10.5. Linkfile Description 38
10.6. Build Procedure Description 38
11. Development Tools 39

1. Introduction

The MDS is a mobile on board integrated module that consists of a Mobile Controller Unit (MCU), a GPS receiver, an RF radio modem, and Mobile Data Terminal (MDT). The unit collects its mobile information, namely, the location, speed, heading, and sends the information to Mobile Information Center (MIC). The unit also receives operator messages from the MIC for presentation to the MDT and sends messages from the MDT to the MIC. Setup and reporting commands are received from the MIC and implemented on the MCU. The MCU is constantly powered via a 12-volt power source.

2. Scope

This document describes the Design Specifications of the MDS (Mobile Data Suite) software release 5.1, based on Symbol Terminal Series 3100.

3. Abbreviations and Definitions

AVL	Automatic Vehicle Location
CMIS	Centralized Mobile Information System
GPS	Global Positioning System
MCU	Mobile Controller Unit
MIC	Mobile Information Center
MDS	Mobile Data Suite
MDT	Mobile Data Terminal
MTS	Mobile Tracking Station

4. Related Documents

4.1. Symbol System Administration Manual - Product Guide for Series 3100.
4.2. Functional Specifications for the SYMBOL Mobile Data Suite System Version 1.0.
4.3.Software Design Specifications for Mobile Data Suite (MDS) Release 3.0 Software, Version 1.0.
4.4. GPS Interface Library Requirements Specifications, Version 1.0.

5. Hardware Overview

The hardware for the MDS system consists of a Symbol Series 3100 terminal, a RAM compatible Mobitek Modem, and a Trimble OEM GPS receiver card. Please refer to the Hardware Design Specifications for further details.

6. Various Configurations

Three configurations are supported by this software. The first configuration consists of a Symbol Series 3100 terminal and an external RAM compatible Mobitek Modem. This configuration is known as **CFG_EXTM** and in this configuration, GPS functionality is not supported. The second configuration consists of a Symbol Series 3100 terminal and a black box consisting of a RAM compatible Mobitek Modem module, (a.k.a. MCU) and a

Trimble OEM GPS receiver card. This configuration is known as **CFG_BBOX**. A third configuration is provided consisting of Symbol Series 3100 terminal with RS-232 interface to the demo room CMIS system using old BDLC communications protocol. This configuration is known as **CFG_DEMO**.

This document describes functionality provided under these configurations.

7. Software Overview

The Symbol Series 3100 terminal is the central controller of the MDS system. The software running on this terminal, known as MDS software (MDS and MDS software are interchangeable in this document), interfaces to the external Mobitek modem or a black box consisting of the GPS receiver OEM module and the Mobitek modem OEM module Mobitek modem provides interface to the MIC. The MDS software consists of a traffic manager and a report scheduler.

mdt_process() is the main process to handle Mobile Data Terminal. mdt_process() services key strokes received and updates MDT display. It has four main modes of operation, menu mode, form entry mode, status send mode, and received message review mode. The MDS is initially downloaded by the MIC defining its forms and statuses. The MDS can also request a download from the MIC if so desired.

The modem interface is the key to communications between the mobile and the MIC. **modem_process()** is the main function to handle incoming and outgoing modem messages. It provides MIC message transfer interface for the rest of the modules within the MDS. It is also responsible for calling modem_xr_fsm() which executes modem transmit/receive functions.

The MDS communicates to the Trimble GPS receiver module via GPS Interface Library. Please see GPS Interface Library Requirements Specifications, Version 1.0 for more details.

The **reporter_process()** is responsible for dequeuing any messages received from the MIC, prepare responses, queue these up, and release the receive buffers.

The main module executes system start-up logic, restores system parameters, requests new forms load from MIC if so required, and then operates the unit in one of the following modes:

- POWER_UP_MODE
- NORMAL_RUN_MODE
- POWER_DOWN_MODE
- PGM_EXIT_MODE
- STAND_BY_MODE

The MDS enters **POWER_UP_MODE** after system initialization and from the STAND_BY_MODE. It is a transient/pass-through mode allowing a transition from system start up or STAND_BY_MODE to the NORMAL_RUN_MODE.

In **NORMAL_RUN_MODE** the following processes and state machine handlers are called until the MDS system needs to be shut-down and the system mode is changed to either POWER_DOWN_MODE or PGM_EXIT_MODE.

modem_process()
mdt_process()
reporter_process()
gps_fsm()
scan_timers()

In **POWER_DOWN_MODE**, the MCU or the external mobidem is turned off, the MDS system is shut down, and the system enters STAND_BY_MODE where the MDS waits for system to get re-started by the user by either Pwr key or side key depression.

In **PGM_EXIT_MODE,** the MCU or the external mobidem is turned off, the MDS system is shut down, and the control is returned back to the DOS.

In **STAND_BY_MODE,** the MDS waits for Pwr key or side key depression, at which event the MDS system is turned on, the MCU or the external mobidem is turned on, and the POWER_UP_MODE is entered.

The MDS system consists of a number of modules. These include:

COMPRESS module
ENGINE module
GPS_BIOS module
HARDINIT module
MCU_INTF module
MDT module
MDT_XLAT module
MDTUTIL module
MENUS module
MIS module
MODEM module
MODEM_XR module
RAM_RFM module
REPORTER Module
SYSUTIL Module

The main module for the MDS software is the MIS module.

8. Module Level Description

8.1. MDS Architecture

Please see Release 3.0 Design Document for MDS Architecture description.

8.2. MIS Module

This module provides the main() function for the MDS system.

main() is the main program for the MDS system. It performs system initialization, and then executes a state machine based on system_mode variable implementing the MDS functionality.

The control is transferred to this function after power-up, where it validates checksums by calling validate_checksums(). If checksums are invalid, the full system initialization takes place. During full system initialization position and status report frequencies are set to there default values, mds_version is invalidated, and the following queues, buffers, and pools are initialized:

MICRecvQue
MICRecvCCQue
MICXmitHPQue
MICXmitLPQue
GlobalPool

At this point the MDT, the MCU, and the MODEM processes are initialized to COLD_START state by calling init_mdt(), init_mcu(), and Modem_init().

If the checksums are valid, a warm start takes place, where current form and status definitions and all other system variables values are maintained.

After COLD_START initialization or WARM_START, the MDS enters **POWER_UP_MODE**. The MDS system has four modes of operation which are POWER_UP_MODE, NORMAL_RUN_MODE, POWER_DOWN_MODE, and STAND_BY_MODE.

In the **POWER_UP_MODE**, Symbol MDT backlight is turned on, the backlight timer is set, the Gps_init() function is called, latest GPS data is polled by calling gps_fsm(), and a power-up message is sent to the MIC. At this time position report, status report, background process, and GPS poll timers are started. A user warning message is then displayed and the system mode is then changed to NORMAL_RUN_MODE.

In **NORMAL_RUN_MODE** the following processes and state machine handlers are called until the system is shut down or the program is exited, in which case the system enters either POWER_DOWN_MODE or PGM_EXIT_MODE.

modem_process()
mdt_process()
reporter_process()
gps_fsm()
scan_timers()

Upon entering **POWER_DOWN_MODE**, a power-down message is prepared and sent to the MIC. The tracking number of this power down message is logged as power_down_msg_tracking_number and the system is not powered down until either this message has been sent or the maximum PowerDownDelayTime has expired. If the MDT is not connected to the black-box or the external Mobidem, the user is notified of the situation and the system is shutdown immediately with the power-down message waiting to be transmitted next time the communications is established. The checksum for forms and data are calculated and saved. The following functions are called while waiting for the Power Down message to go out:

modem_process()
process_power_down_mode()
gps_fsm()
scan_timers()

Once Power Down message has been sent, the MCU or external Mobidem is turned-off, and the system waits for a power or side-key depression for a transition to the STAND_BY_MODE.

In **STAND_BY_MODE.** the backlight is turned on, the MCU is turned on and the POWER_UP_MODE is entered.

In addition, the following functions are supported by this module:

GetSystemClock() returns system_clock in seconds.

BackgroundTimerFunction() is executed when BackgroundTimer expires. It calls mdt_background() function if the system is in NORMAL_RUN_MODE and it re-arms BackgroundTimer.

8.3. ENGINE Module

This module may only be linked-in for CFG_DEMO version. Though CFG_DEMO version does not use GPS information, it is linked-in to satisfy certain external definitions. The following section describes ENGINE module implementation.

This module is responsible for communicating with Magnavox GPS receiver. The SCB communicates to the Magnavox GPS receiver over a RS-232 port. The Magnavox receivers supported are the 4200 and the engine. The control port of the receiver is used for interface between the SCB and the GPS receiver. The communications parameters are:

Baudrate	4800
Databits	8
Data Parity	None
Stopbits	1

This port is used to initialize, monitor and control the receiver.

Startup initialization

The 007 message is used to control the output rate of the following messages.

000	Status	Every 5 seconds
011	Speed & Heading	Every second
012	Self-Test results	Every 9999 seconds
021	Position	Every second
023	Statistical Information	Every xxx seconds
030	Software version	Every 9999 seconds
533	Date	Every xxx seconds

Normal operations

The above messages are assimilated and stored in a structure defined as follows.

Latitude	long	Microdegrees
Longitude	long	Microdegrees
Altitude	short	Meters
Cog	short	degrees
Sog	short	MPH
Utc	long	Seconds since start of week
Utc_fix	long	Seconds since start of week
Receiver_status	char	0 => ACQ
		1 => ALT

		2 => IAC
		3 => IDL
		4 => NAV
		5 => STS
		6 => TRK
Current_mode	char	Navigating mode
Fix_mode	char	0 => no fix yet
		2 => 2 D
		3 => 3 D
Self_test	char	0 => RAM & ROM passed
		1 => RAM failed, ROM passed
		2 => RAM passed, ROM failed
		3 => RAM failed, ROM failed
Year	short	current year
Month	char	1 - 12
Day	char	1 - 31
Sat_vis	char	
Sat_trk	char	
Ttff	short	seconds
Dist_travelled	short	1/10th miles traveled since startup
Max_speed	short	tenths of MPH

Gps_init() performs COLD_START or normal initialization of the GPS receiver. Gps_time_offset() sets local time offset for the GPS receiver by sending a "$PMVXG,001" message to the receiver.

Gps_get() sets lat, long, altitude, and numerous other fields to the current values for the following packet types:

MDS_Stat
MDS_Pos
MDS_Alarm
MDS_Power_up
MDS_Power_down

Gps_time() returns current gps time.

txs() prepares an outgoing packet for the engine, including "$", checksum field, and terminating carriage-return and sends it to the appropriate port.

nmea_decode() checks for a completely received nmea message, and performs decoding if a complete message has been received. The following message number are currently decoded:

0
11
12
21
23
533

gps_fsm() function implements gps finite state machine. It scans printer port for a newly received character, extracts the character from the receive buffer, puts it in the nmea_buffer. If the character received is a carriage-return, it calls nmea_decode() function to check and process if a valid message has been received.

8.4. HARDINIT Module

This module provides hardware initialization and hardware interface functions for the MDS system. This module also provides support for the Event Timer facility of the MDS system. The event timer facility allows user to have a number of timers based on the Symbol terminal system timer. The event timer facility is executed by calling scan_timers() function.

scan_timers() is the event timers processing routine. The resolution of this service is 1 second. It services all the event timers in the system. This includes calling event timer handling routine if an event timer expires.

init_serial() function sets up extended serial interface for a standard 3-wire interface. It opens the serial port by calling BiosOpenPort() function.

shut_serial() function shuts down the serial interface by calling BiosClosePort() function.

8.5. MDT Module

The MDT module provides Mobile Data Terminal specific functions. This includes mdt_process() which is the main process for the MDT related functions. mdt_process() is called once every loop by the main() routine. The MDT module supports forms larger than the physical size of the screen by allowing scrolling of the forms. In addition, a number of routines are provided to support automatic JOBID list building and selection and insertion of the JOBID in appropriate form fields. Support for scrolling menus is provided MENUS module is utilized by this module.

A number of additional features are provided in this release. These include:

- Scrolling Menus

- Time and Distance Position Reporting
- Data Compression
- Blinking Overlaid indicators for Canceled, Re-transmitted, and Changed jobs

The following support routines are available for the mdt_process():

init_mdt() function initializes mdt_process() parameters, resets LEDs, and local variables. It is called by main() at cold start.

mdt_invalidate() invalidates MDS forms and status definitions. It also updates system version, GPS poll frequency, total number of forms, and total number of statuses in MDS. This function is called when invalidate message is received from MIC.

process_mds_init_msg() function checks new version and mobile_id against current, and total number of forms and statuses. If the current values do not match with new values, the MDS is invalidated and a request for forms and statuses download is initiated.

AddMicMessage(), **DelMicMessage()**, and **GetMicMessage()** functions manipulate MIC message queues. AddMicMessage() adds a mic_msg to given message array. DelMicMessage() deletes a mic_msg from given message array. GetMicMessage() gets a mic_msg from given message array.

DelMicMessage() and AddMicMessage() work with the given message number as the offset in the mic_msg array. AddMicMessage() adds messages in the mic_msg array in LIFO manner. The new added message is placed at index 0. If there are any other messages in the array, these messages are slid down, before new message is placed.

Two functions are provided to post warning messages on the user screen. These functions are **log_warning()** and **log_warning_and_wait().** The log_warning() function displays a warning message where as log_warning_and_wait() function is a special function to display a form, overlaid by a blinking "C", "X", or "R" to indicated a Changed, a Canceled, or a Re-transmitted form, respectively, and then wait for a user acknowledge. An "X" is displayed when a Cancel text message is received for a Job which still exists in the system, "C" is displayed when a received job already exists in the system and the data of this job have changed, and "R" is displayed when a received job already exists in the system but data of this job has not changed.

process_forms_desc() routine processes FORMS description received from MIC in ASCII format.

display_queued_msg() routine displays a queued FORMS or STATUS message (to MIC) in ASCII format.

process_status_desc() function processes STATUS description received from the MIC in ASCII format. This function updates status text information based on the status number and status text received in this message from the MIC. It also marks this status VALID. Any status description beyond mdt_total_statuses are rejected.

checksum_dynamic_data(), checksum_static_data(), checksum_forms_data(), checksum_forms_definition(), and **validate_checksum()** functions provide checksum and validation function for the MDS system. checksum_dynamic_data() checksums dynamic part of the FORMS data. checksum_static_data() checksums static part of the FORMS data. checksum_forms_data() checksums static and dynamic parts of the FORMS data. checksum_forms_definition() checksums static and dynamic parts of the FORMS definition. validate_checksum() checks the checksum.

mdt_display_form() function displays a FORM information (static and dynamic) from FORMS data. 4 (rows) * 40 (columns) are displayed.

mdt_view_sent() function lists sent messages to the MIC. Previous (up to 4) sent messages kept in mic_sent_msgs are displayed on the screen. One line is displayed per message. The messages are displayed in LIFO manner. This is accomplished by AddMicMessage() putting the newest message in offset zero.

mdt_view_saved() function lists saved messages from the MIC. Previous (up to 4) saved messages kept in mic_saved_msgs are displayed on the screen. One line is displayed per message. The messages are displayed in LIFO manner. This is accomplished by AddMicMessage() putting the newest message in offset zero.

mdt_view_new() function displays a new message. mic_msg_holding_ptr maintains a pointer a previously viewed message, which could not be added to the mdt_saved_msgs array. In that case, this message is displayed rather than a message on MDTRecvQue. This routine calls displaye_queued_msg() to display new message.

mdt_send_and_save() function sends currently selected STATUS or FORM message and saves it in the mic_sent_msg queue. A specific STATUS message can be configured to send a GPS Position report also. The first field of most FORMS is usually the JOBID and this field is mandatory in order to transmit the FORM.

PeriodicReportTimerFunction() is called when the PeriodicReportTimer expires. If the system is running in NORMAL_RUN_MODE, a position message is sent to the MIC and the PeriodicReportTimer is re-started with report frequency. Currently, time and distance report mode is supported. At the report rate (supplied by the host with the init packet), the distance traveled since last report is checked, and a new report is sent if the distance traveled is at-least the minimum distance specified by the init packet. If the scheduled report could not be transmitted, the timer is reset to a 10 second period so that the report can be transmitted as soon as communication is possible.

StatusReportTimerFunction() is called when the StatusReportTimer expires. If the system is running in NORMAL_RUN_MODE, and the RAM link is not down, a MDS_Stat packet is sent to the host.

mdt_background() function periodically scans for forms and statuses being invalid and if any invalid forms or statuses are found, it requests updated information from the MIC. One other function performed by this routine is to pop up threshold messages. The following rules are used to put these thresh-hold messages:

1. If number of sent, saved, and new messages altogether are greater than THRESH_CLEANUP, a clean-up warning messages is displayed on the screen.
2. If number of outgoing messages are greater than THRESS_OOR, out-of-range warning message is displayed on the screen.
3. On event detection, the following messages are displayed:
    - queue full message, when no more room in the receive queue.
    - out-of-buffers : could not sent, when no buffers for send
    - out-of-buffers : incoming lost, when no buffers for incoming message.

mdt_forms() function displays forms number menu.
mdt_status() function displays status number menu.

put_in_field() function places cursor in the given field of a given form. The cursor is always placed in the first character position of the given field. Variables cur_row, cur_col, left_in_field, and data_type are also updated. Variable mdt_forms_code points to current form number.

data_entry() function accepts data for current field and processes RIGHT_ARROW, LEFT_ARROW, NEXT, PREV and DEL keys as well. The forms scrolling is provided using UP_ARROW and DOWN_ARROW keys. This function is called by mdt_process() state machine.

prep_forms_data_msg() routine prepares a FORMS data message to be sent to the Mobile Information Center (MIC) in the given buffer. The message header (in buffer) already contains form number.

mdt_update_ack() function updates ACK led on the MDT if there are any messages requiring acknowledgment from the user.

mdt_update_message_stats() function displays the following information when in the main menu:
    - RAM Signal Strength BAR
    - Link status (UP, DOWN)
    - Battery Status (GOOD, LOW, DEAD)

- Number of NEW messages waiting
- Number of Pending (out going messages)
- Number of Saved messages
- GPS Status
- Date and Time

Each arrow in the RAM Signal Strength indicates 5 units of RSSI.

The following functions support accumulation, selection , and scrolling of JOBIDs:

init_jobids()
add_jobid()
del_jobid()
get_next_jobid()
get_prev_jobid()

The following functions support forms scrolling:

log2phy()
any_field_id_to_park()
phy_field_id_to_park()
last_field_id()
first_field_id()

system_backup() and **system_restore()** functions provide critical data save and restore vehicles between power down cycle.

mdt_process() is the main process to handle Mobile Data Terminal. The main loop calls this process every cycle. mdt_process() services key strokes received and updates MDT display. It has the following modes of operation:

MDT_MENU_MODE1
MDT_MENU_MODE2
MDT_FORM_ENTRY_MODE
MDT_STATUS_SEND_MODE
MDT_HELP_MODE
MDT_DIAG_MODE
MDT_FUTURE_MODE
MDT_MENU_TSD
MDT_VIEW_SENT_MODE
MDT_VIEW_SAVED_MODE
MDT_VIEW_RCVD_MODE
MDT_SERVICE_MODE
MDT_CFG_MODE

Configurable options in MDT_CFG are:

1. Time format (12 or 24 hour)
2. Backlight timer (0, 10, 30 and 60; 180 is special case)
3. Switch "Backspace" with "Space" key (0=No, 1=Yes)

In addition to mdt_process(), another process called **mdt_CC_process()** is provided to handle cancel and change job requests. When a forms packet is received, modem_received_msg() function is called. This function queues a forms message to the MICRecvCCQue, which in turn is processed by the mdt_CC_process(). The mdt_CC_process() processes the cancel job requests by calling process_cancel_job_msg() and it processes change job requests by calling process_change_job_msg(). Any other message type is queued to MICRecvQue which is processed during normal course of action.

8.6. MDT_XLAT Module

This module provides functions and tables to translate the MDS originated messages from low/high format to high/low format and the MIC originated message from high/low format to low/high format. A translation table for each message type is provided. The table msg_xlation_tbl provides message ID to message translation table cross-reference.

translate_msg() function provides translation of a message data from Intel format (low byte first) to Sun format (high byte first). msg_xlation_tbl provides translation type and offset information.

8.7. MDTUTIL Module

This module provides utilities for Mobile Data Terminal (MDT) interface. These utilities include:

sendt_mdt() and **send_mdt_count()** functions send a string or a given number of characters to Mobile Data Terminal interface.

mdt_goto() function positions MDT cursor at given position.

mdt_put_menu() and **mdt_put_line()** functions support old style menus for the display. mdt_put_menu() puts a menu on the MDT screen. A menu consists of 4 menu lines. mdt_put_menu() clears screen and then puts 4 menu lines. mdt_put_line() puts one menu line.

mdt_scan_key() function scans for the new keys input at the MDT keyboard. The mdt_process() calls this function once every cycle before executing main logic.

mdt_scan_key() polls mdt_receive_buffer for key codes, translates these codes into actual key strokes and places it into mdt_new_key.

8.8. MODEM Module

Modem module for the MDS system. This module provides interface between other modules in the MDS and the Mobile Radio Mode (MRM) or RAM. To send a message to the MRM or RAM, just call send_msg_to_mic(). Two queues are maintained by this module for outgoing messages, a high priority queue and a low priority queue. To send a message over the radio modem, this module calls modem_send_msg() function. When a message is received by radio modem interface handler (modem_xr_fsm()), it calls modem_received_msg() function. The status of the currently outgoing message is checked by calling modem_sent_msg_status(). The modem_process() is the main function for this module and it is called by the main() program every cycle. The modem_process() in turns calls modem_xr_fsm() every cycle to allow transmission and reception of the messages over Mobile Radio Modem or RAM interface.

modem_received_msg() function is called to deliver a message received over modem line. It copies the message into it's own buffer, leaving the responsibility of releasing the incoming buffer to caller. It translates the incoming message and processes it or queues it up to appropriate process. If the received message is of type MIP_Mts_form, it is queued to MICRecvCCQue to be processed by mdt_CC_process() at appropriate time, where as MIP_Mts_act type message is queued to the new message queue by calling add_mic_rcvd_msg().

send_msg_to_mic() function queues an out going message to the MIC transmit queue. Two priority queues are supported.

get_mic_message_status() function provides current status of a message outgoing to the MIC. It searches through MICXmitHPQue and MICXmitLPQue for the tracking number of the message.

send_new_msg() facilitates transmission of a previously queued MIC messages in FIFO and HIGH/LOW priority order.

modem_process() is the main function to handle incoming and outgoing modem messages. It provides MIC message transfer interface for the rest of the modules within the MDS. It is also responsible for calling modem_xr_fsm() which executes modem transmit/receive finite state machine. Special processing is provided to delete Job ID and the Job from saved list and possibly from new list in the following circumstances:

- when last transmitted form number is FORM 03, or
- when last transmitted form number is FORM 01 and it was a hand-off made

8.9. MODEM_XR Module

This module is responsible for sending and receiving messages to the modem in CFG_DEMO configuration.

modem_send_msg() function is called with message buffer pointer and message buffer length to transmit a message. A TRUE is returned if the message is accepted for transfer by the modem transfer process, a FALSE is returned otherwise.

modem_sent_msg_status() function returns current state of the last message given to the modem transfer process. The returned statuses are:

MSG_delivered
MSG_awaiting_delivery
MSG_local_nak
MSG_remote_nak
MSG_timeout
MSG_unknown_message

modem_xr_fsm() is the main loop for the modem transfer process. This function is called in the modem main loop.

modem_received_msg() is an externally supplied routine to the modem transfer process which it calls to deliver a newly received message. modem_received_msg() function is called with a pointer to a byte buffer and the length of the message. The message buffer may only be read and the data needs to be copied.

The following action function are provided:

action_init()
action_bput()
action_hput()
action_bcc()
action_bcc_check()
action_err0()

Modem_init() function initializes modem_xr_process() initial state.

modem_xr_fsm() is modem transfer/receive finite state machine. When a character is received, appropriate action routines are called based on the current state of the state machine, and new state is defined.

8.10. REPORTER Module

prepare_msg_to_mic() routine prepares a message to be sent to the mic. It allocates a buffer, fills it up with appropriate information and queues it to MicSendQueue. A new sequential tracking number is assigned to each new message. A tracking number of zero is illegal. If no buffers are available in the GlobalPool, an error condition is returned.

process_power_down_mode() function is responsible for extending power of the MDS system until the message with power_down_msg_tracking_number from the MicSendQueue is transmitted. This is done to make sure that the MDSPowerdown message is sent out before the MDS system is powered down. This routine calls get_mic_message_status() to see if the power down message has been sent.

reporter_process() function is called from the main loop. The reporter_process() is responsible for dequeuing any messages received from the MIC, prepare responses, queue it up, and release the receive buffer.

8.11. SYSUTIL Module

This module provides utility functions for the Mobile Data Suite(MDS). These functions include:

8.11.1. Queuing Functions

cdequeue :	de-queue an element from a circular queue.
reset_scan :	set the scan pointer to the get pointer.
cque_scan :	scan a queue
cenqueue :	enqueue an element to a circular queue.
dequeue :	de-queue an element from a linked-list queue.
enqueue :	enqueue an element to a linked-list queue.
getblk :	getblk gets a buffer from a pool.
retblk :	retblk returns given buffer to a pool.

8.11.2. Circular Buffering Functions

cgetch :	cgetch gets a character from a circular buffer.
cputch :	cputch puts a character in a circular buffer.

8.11.3. Event Timer Functions

StartTimer :	StartTimer starts an event timer for given period.
CancelTimer :	CancelTimer cancels given event timer.

PauseTimer : PauseTimer pauses given timer.
GetTimer : GetTimer gets remaining time period for the given timer

8.11.4. Semaphore Functions

PostEvent : PostEvent posts an event to a semaphore
ReceiveEvent : ReceiveEvent receives a posted event to a semaphore

8.11.5. Initialization Routines

InitCqueue : InitCqueue initializes given circular queue.
InitCbuf : InitCbuf initializes given circular buffer.
InitLLQue : InitLLQue initializes given linked-list queue.
InitBufPool : InitBufPool initializes given buffer pool.

8.12. COMPRESS Module

This module provides utility functions for data compression and data de-compression. These functions are:

8.12.1. compress() function

compress() function compresses given data using Huffman coding. If the compressed data size is larger than the original data, the data is left in the un-compressed.

8.12.2. un_compress() function

un_compress() function expands given data using Huffman coding. If the compressed data size is larger than the original data, the data is left in the un-compressed.

8.13. GPS_BIOS Module

This module provides functions which allows a Symbol terminal to communicate with the GPS sensor. The communications with the GPS sensor is in Trimble TAIP format. These functions are:

8.13.1. GPSConnect() function

The **GPSConnect()** function verifies the communications link with the GPS by retrieving sensor ID and version.

8.13.2. GPSInitialPosition() function

The **GPSInitialPosition()** function uses "SIP" TAIP command to send initial position data to the GPS sensor.

8.13.3. GPSPositionReport() function

The **GPSPositionReport()** function uses "QLN" TAIP command to retrieve position data from the GPS sensor.

8.13.4. GPSStatusReport() function

The **GPSStatusReport()** function uses "QST" TAIP command to retrieve status data from the GPS sensor.

8.13.5. GPSSatelliteReport() function

The **GPSSatelliteReport()** function uses "QLN" TAIP command to retrieve satellite data from the GPS sensor.

8.13.6. GPSSetTime() function

The **GPSSetTime()** function uses "STM" TAIP command to send date/time data to the GPS sensor.

8.13.7. GPSGetTime() function

The **GPSGetTime()** function uses "QTM" TAIP command to retrieve date/time data from the GPS sensor.

8.14. MCU_INTF Module

This module provides support for the MCU unit. As described above, the MCU units houses the GPS module and the RAM module. The interface to the MCU unit is via single serial port, and the MCU_INTF module is responsible for multiplexing the communications link as well as controlling the power to the MCU box. For more details, please see hardware specifications on the unit. The following functions are provided by the MCU_INTF module:

init_mcu() function initializes serial port and turns the power on for the RADIO module and the GPS module. It also selects and enables RADIO module. Support for all the functions in the ENGINE module in older BDLC system is provided in this module for MCU based system.

Gps_init() performs COLD_START or normal initialization of the gps data structure.

Gps_get() sets lat, long, altitude, and numerous other fields to the current values for the following packet types:

MDS_Stat
MDS_Pos
MDS_Alarm
MDS_Power_up
MDS_Power_down

Upto 3 attempts are made to get the latest GPS data before the above packets are filled. A value of -1 is returned for Mobidem related parameters if the Mobidem status can not be read with in three attempts.

Gps_time() returns current gps time.

gps_fsm() is called to get latest GPS data from the main loop. This function makes sure that this request is not attempted more often than once every MAX_GPS_POLL_RATE seconds. This function monitors the link status and if the link is on, it selects GPS (switches from the Mobidem module to the GPS module on multiplexed serial line), and then gathers GPS data by calling GPSPositionReport(), GPSStatusReport(), GPSSatelliteReport(), and GPSGetTime() function. At the end the Mobidem module is re-selected. The Symbol terminal clock is set, if previously not set, if the GPS is doing fixes

GetDistance() function computes distance (in meters) based on latitude and longitude.

8.15. MENUS Module

This module provides support for scrollable menu display and item selection functions.

Function **GetSelection()** is called with a Menu ID. This function displays menu by calling **DisplayMenu()** function. After displaying the menu, **ProcessKeys()** function is called which is responsible for selecting a menu item based on the user key depressions while calling all other processes in the background to make sure that communications with the base station and the gathering of the GPS data as well as sending of position reports is carried on without any interruption.

The Menu module is activated in the mdt_process() by calling **SelectFromMenu()** function. Currently this function may only be implemented for the FORM and STATUS message selections. The menu list is built dynamically at the time of the form loading. This provides base station complete control of the Menu List.

8.16. RAM_RFM Module

This module provides the same functionality as provided by the modem_xr module in the BDLC system.

8.16.1. Function **modem_xr_fsm()**

This function performs the same functionality as the function with the same name in the modem_xr module. If the power of the MCU box is detected to be OFF, it is turned ON. This provides protection against accidental power turn off and maximum availability. RfmPeekNumMsg() routine is called to find out if there are any messages waiting in the mobidem. RfmReadMsg() is called to read oldest pending message. The received message is then un-compressed and processed. The following type of messages are processed by this routine:

- remote acknowledge to previously sent message
- local acknowledge to a message just sent
- a new message from host

8.16.2. Function **SendAck()**

This function sends an acknowledgment for a received message with a particular sequence number. A function related to this function is **send_any_pending_ack()**. SendAck() is called by send_any_pending_ack(). A list of sequence numbers to be acknowledged is maintained by this module. This list could be manipulated by **initToBeAcked(), addToBeAcked(), and SeqNumToAck()** functions. This list is called **toAck[].**

8.16.3. Functions **turn_modem_on()** and **turn_modem_off()**

turn_modem_on() function turns external mobidem ON by toggling DTR (data terminal ready) line of the interface between the Symbol terminal and the external mobidem. turn_modem_off() function turns external mobidem OFF by sending RFM_POWER_OFF message to the mobidem.

8.16.4. Function **getRfmState()**

This function gathers the following RFM state information:

- network contact
- traffic area
- receiver signal strength level
- battery power level
- power mode
- modem power state
- line status
- network check
- inbound queue size
- mobidem software version

8.16.5. Function **modem_send_msg()**

This function sends a CMIS data message over the RAM network. A message can not be sent if the following conditions are true:

- the GPS interface with the MCU box is active
- the terminal is not in cradle (or connected)
- the network contact is lost
- the link is down
- acknowledgment to a previous transmission is awaited.

A timer is started, if the delivery of the data message to the mobidem is successful, waiting for an acknowledgment either from the mobidem or from the remote end depending on the confirmation required flag value.

8.16.6. Function **modem_sent_msg_status()**

This function returns status of the previously sent message. The statuses returned are similar to the modem_xr module. These are MSG_timeout, MSG_awaiting_delivery, MSG_delivered, or MSG_local_nak. If the link goes down (or terminal removed from the cradle), MSG_timeout status is returned immediately. If the transmission by the mobidem over the RAM network fails, MSG_local_nak is returned.

8.16.7. Function **Modem_init()**

This function is responsible for establishing initial network contact and bring the link up with the RAM network. It also initializes toAck list by calling initToBeAcked() function.

8.16.8. Function **Modem_shut_down()**

This function performs a dis-connect with the RAM network.

9. Design Notes

The MDS software has evolved over a long period of time and moved from one hardware platform to another numerous times. In addition, the basic components of the MDS system, such as GPS engine, Radio/Cellular communications platform, hardware platform Operating System, etc., have changed as well. The design and functional requirements changed along the way. Naturally, this lead to a number of special software features. An attempt is made to capture these special features in this section.

9.1. User interactions

9.1.1. When editing a form, the first CLEAR key depression clears current field, 2nd consecutive CLEAR key depression prompts user if he/she wants to clear entire form, and the third CLEAR key depression is taken as a confirmation that the user wants to clear entire form and the form is cleared. Any other key is accepted negative confirmation and the form is not cleared.

9.1.2. The up and down arrows in first field of forms 1, 2, 3, and 7 are used for scrolling through JOBID list. This JOBID is displayed in normal or reverse video depending on the status of the job execution.
9.1.3. The JOBIDs are scrolled from the latest to the oldest.

9.1.4. ENTER key is used to navigate between fields with in a form.

9.1.5. A form is sent using Control-SEND key combination rather than SEND key.

9.1.6. Control-E key allows forms editing in VIEW SENT and VIEW SAVED modes.

9.1.7. The ENTER key functionality is same as NEXT key.

9.1.8. The main screen display the following information
- Number of new messages with NEW blinking if there are new messages
- Link or Battery status, blinking when it is abnormal
- Number of pending and saved messages, and GPS status
- Current time
- RSSI signal strength indicator, 1 ▸ for each 5 units signal of strength, maximum 6.

clears A terminal ID in the demo mode may only be valid when all the characters in the ID are alphabetic and are uppercase. The parser function in modem_xr module assumes that first non-uppercase character terminates terminal ID.

9.2. MDS_Init packet parameters definitions

Parameter	Definition
system_version	MIC Version
froms_version	FORMS description version number
mobile_id	mobile ID
reporting frequency	position reporting rate in seconds
total_forms	total number of forms
total_activity	total number of activity statuses
local_time_offset	local time offset from GMT (HHMM) (e. g. 0700 is -7 hours east)
mdt_phys_rows	number of physical rows on MDT
mdt_phy_cols	number of physical columns on MDT
mdt_log_rows	number of logical rows on MDT
mdt_log_cols	number of logical columns on MDT
comm_sw_num_retry	number of retires
comm_retry_delay	delay (in seconds) before each retry for sending message

comm_xmit_tmout	timeout (in seconds) for message transmission
comm_recv_tmout	timeout (in seconds) for message reception
comm_xmit_srch_tmout	timeout(in seconds) for transmitting data after radio system switch has occurred (applicable to BDLC radio system implementation)
stats_rep_freq	stats reporting rate in seconds
gps_reporting_mode	GPS reporting mode 0 = Time based reporting 1 = Time and distance based reporting 2 = Distance based reporting. If gps_reporting_mode = 0, send position report after the reporting frequency time expires. If 1, send position report after the reporting frequency time expires, BUT ONLY if the distance moved EXCEEDS the gps_distance_thresh specified. If 2, send position report ONLY if the vehicle has moved a minimum of gps_distance_thresh specified.
gps_distance_thresh	minimum distance reporting threshold. See gps_reporting_mode description. If gps_reporting_mode = 1, the distance moved by the vehicle may EXCEED this threshold & the reporting timer may expire in order to send a position report. If mode = 2, a position report is sent out as soon as distance moved exceeds this value.
cmic_clock	CMIC clock time (in seconds from Jan 1, 1970). The MDT uses MDS_Powerup message to start a timer, and if it receives MDS_Init packet within 30 seconds of the MDS_Powerup message, it sets MDT's clock to this cmic_clock (after converting it into local time) time. Otherwise the MDT clock time is left unchanged.
init_reserve1	Used to indicate which STATUS message will trigger a POSITION message (0 = Disable). Valid STATUS values are 1 to 11.
init_reserve2	Used as a bit flag to indicate various options. 1 = Append Position message with all FORMS & STATUS messages. 2 = Delivery FORM may have a POD.

9.3. Terminal ID in Demo Mode

A terminal ID in the demo mode may only be valid when all the characters in the ID are alphabetic and are uppercase. The parser function in modem_xr module assumes that first non-uppercase character terminates terminal ID.

9.4. Files on RAM Disk

The following files are maintained on the RAM disk:

MISCFG	containing host ID/terminal ID and timezone information.
FORMDEF	containing FORMS/STATUS definitions and other information system variables.

9.5. Invalidation of Forms on MDS_Init message

The following conditions, upon reception of MDS_Init message, invalidate current form load:

- a change in software version
- a change in forms version
- a change in total number of forms
- a change in total number of statuses

9.6. Automatic update of sent messages list

The status of the sent messages is automatically updated once every second.

9.7. Checksums maintained

The following checksums are maintained by the system:

- forms data checksum
- forms definition checksum
- status definition checksum

9.8. JOBID Display Strategy

The job ID for a job which has not yet been picked-up is displayed in reverse-video in the job ID scroll list. Other job IDs in the job ID list are displayed in normal video mode.

9.9. JOBID Maintenance Strategy

A new job ID is added to the job ID scroll list when either FORM 0 or FORM 1 is received. If the job ID already exists in the job ID scroll list, it is not added again.
If the FORM 6 is received with "CANCEL" text or MIP_Msg_CANCEL status message is received, appropriate message is deleted from new and saved lists and the job ID is removed from job ID scroll list.

A job is deleted from SAVED and NEW list and from the job ID list a FORM 3 is received or a FORM 1 is received and a HAND-OFF is made.

A picked-up state is entered for a job ID when FORM 2 is received. A picked-up job ID is displayed in reverse video.

When a job is deleted from SAVED list, it's job ID from scroll list is also deleted.

9.10. Reception of a new message

Upon reception of a new low priority message a double beep is sounded. A triple beep is sounded upon reception of a new high priority message.

9.11. Out-of-cradle warning

A set of two triple beeps is sounded when the terminal is removed from the cradle.

9.12. Mobidem Status debug array

An array of mobidem function calls and return statuses is maintained and displayed on the debug screen. An entry may only be made when a new call type is invoked. Each entry consists of a call code ored with status return These codes are:

Function Call	Call Code
RfmNBWriteMsg	0x200
RfmState(from modem_xr_fsm)	0x400
RfmWriteMsg(data)	0x600
RfmPeekNumMsg	0x800
RfmReadMsg	0xa00
RfmState(from modem_send_msg)	0xc00
RfmWriteMsg(ACK)	0xe00

9.13. Setting of Initial GPS location

The setting of initial GPS location is not functional in Trimble GPS engine firmware and hence not supported by MDT software.

9.14. GPS polling rate

The GPS polling rate is controlled such that GPS engine is not polled any more frequently that once every MAX_GPS_POLL_RATE seconds. This is enforced in gps_fsm() function.

9.15. Time to first fix calculation

The time to first fix (ttff) is calculated after every power up. A fix is assumed accomplished if the gps fix status is 2 and gps current status is either 0, 1, 2, or 3. The ttff is number of seconds since power up.

9.16. Distance traveled calculation

The distance traveled is a running sum of distance traveled in meters. Every MAX_GPS_POLL_RATE seconds, the distance traveled is calculated and added to the running total. This value is distance traveled since last power up.

9.17. Accelerator Keys Support

Accelerator keys are supported by the MDS system wherever possible. The accelerator keys support is compiled-in by defining ACCELERATOR_KEYS macro in MCT.C module.

9.18. Job Special Overlays

Special reverse-video overlays on the displayed jobs indicate various conditions. These overlays are blinked once per seconds

9.18.1. A canceled job is indicated by:

X

9.18.2. A changed job is indicated by:

C

9.18.3. A re-transmitted job is indicated by:

R

The distance traveled is a running sum of distance traveled in meters. Every MAX_GPS_POLL_RATE seconds, the distance traveled is calculated and added to the running total. This value is distance traveled since last power up.

9.19. Position and Stats Reporting

The system maintains minimum and default report rates. The following are these rates:

	Minimum Frequency	Default Frequency
Position Report	15 seconds	300 seconds
Stats Report	20 seconds	1800 seconds

In current implementation, time(periodic) and distance traveled reporting is supported. The distance traveled supersedes periodic reporting. If periodic as well as distance traveled reporting is requested, possibly only distance traveled reporting is executed. At the time of reporting, if the link is down or the terminal is out of cradle, that particular report is skipped, but the reporting algorithm is executed normally.

Both position and stats reporting may only be executed if the system is in normal run mode.

9.20. Service Screens

The following service screens are supported by the system:

9.20.1. System Debug Screen:

CODE	DATA
DMC	duplicate message count
FDE	forms description errors
SDE	status description errors
MTN	mis-matched tracking numbers
IPI	illegal packet ID
BUF	number of free buffers available
LNK	terminal in-cradle status
NAK	number of NAKs received
SID	current ID for send packet
SEQ	next sequence number to use, waiting for ACK for this sequence number
TRK	MRM tracking number
MRM	MRM state
WRR	number of write errors
SND	modem_xr send state
TMO	number of time-outs in modem_xr
STS	last eight RAM calls and status returns

9.20.2. Mobidem Debug Screen:

CODE	DATA
TXPWR	transmitter power level
RXPWR	receiver power level
MDPWR	modem power level
PWMOD	power mode
BTLVL	battery level
NCNTC	network contact status

TAREA	traffic area
LNSTS	line status
NCHCK	network check
IBQUE	inbound queue
VERSN	software version

9.20.3. Configuration Screen:

CODE	DATA
TERM ID	terminal ID
HOST ID	host ID
MIC	MDS version
FORM	forms version
FORMS	total number of forms
STATUS	total number of statuses
RPT	position report mode
DIST	distance threshold in meters
TIME ZONE	time zone
POS	position report frequency
STS	stats report frequency

9.20.4. GPS Screen:

CODE	DATA
LN	longitude
FX	current status
LT	latitude
AG	fix status (age of fix)
TM	utc time
F	flags
SP	speed on ground (sog)
ALT	altitude
SV	number of visible satellite
HDG	course on ground (cog)
TTFF	time to first fix
TRK	number of satellite tracked
AL	stored almanac is not complete or current

BBK	battery backup failed
DSP	signal processor error
RF	excessive ref freq error
C1	alignment error channel or chip 1
C2	alignment error channel or chip 2
AN	antenna feedline fault
CL	battery powered timer/clock fault
SZ	synthesizer fault
AD	A-to-D converter fault

9.20.5. Raw GPS Screen:

In this mode, the raw data being received from the GPS engine is displayed.

9.21. GPS Status Report translation table (GPS to MIC)

GPS fix status	GPS current status	Report to MIC	Displayed on MDT
0	Don't care	59	N/A
1	Don't care	58	OLD
other than 0 or 1	0	2	NV2
other than 0 or 1	1	3	NV3
other than 0 or 1	2	4	DG2
other than 0 or 1	3	5	DG3
other than 0 or 1	4	10	UNK
other than 0 or 1	5	11	UNK
other than 0 or 1	6	6	DR
other than 0 or 1	7	12	UNK
other than 0 or 1	8	8	DDR
other than 0 or 1	9	9	UNK

9.22. Various System Parameters

This table defines various system parameters:

Parameter	Variable/Macro name	Min./Max.	Default
Start-up Delay	StartUpDalayTime		3 seconds
Power Down Delay	PowerDownDelayTime	90 seconds	
Background Task Timer	BackgroundRunTime		120 seconds
Brightness Timer	BrightnessControlTime		180 seconds
CMIC clock acceptance	CMIC_TIME_THRESH		60 seconds
Physical number of rows			8
Physical number of cols			20

Logical number of rows			22
Logical number of cols			20
Number of Forms		10	
Number of Statuses		20	
Static fields in a form		32	
Dynamic fields in a form		32	
Number of SAVED messages		40	
Number of SENT messages		4	
Number of RECEIVED messages		60	
Modem transmit timeout period			120 seconds
Modem Receive timeout period			20 seconds
position report mode			time based
GPS screen update rate			10 seconds
TSD screen update rate			5 seconds
Maximum JOBID length			8 characters

9.23. Status Information Sent to Host

The following status information is sent to host in MCU(CFG_BBOX) version in various messages:

9.23.1. MDS_Stat Packet:

Field ID	Data
current_time	utc
time_to_first_fix	ttff
distance	free buffer count
max_speed	NAK count
driving_time	ram sequence number
total messages	wait sequence number
sts_2	send TMO count
sts_3	mobidem rx_power
sts_4	mobidem power_mode
sts_5	mobidem battery_level
sts_6	mobidem network_contact
sts_7	mobidem traffic_area
sts_8	mobidem line_status

sts_9	mobidem network_check
sts_other	mobidem inbound_queue
sat_vis	number of visible satellites
sat_trk	number of satellites tracked
modem_retry	mrm_tracking_number
radio_switch	gps sts_2
dup_msg	gps sts_3
sbc_gps_test	gps sts_4
gps_mode	gps_status

9.23.2. MDS_Pos Packet:

Field ID	Data
time_to_last_fix	local PC time
latitude	latitude
longitude	longitude
speed	sog
heading	cog
current_status	current status as described above

9.23.3. MDS_Alarm Packet:

Field ID	Data
time_to_last_fix	local PC time
latitude	latitude
longitude	longitude
speed	sog
heading	cog
current_status	current status as described above

9.23.4. MDS_Power_up Packet:

Field ID	Data
time_to_last_fix	local PC time
latitude	latitude
longitude	longitude
speed	sog
heading	cog
current_status	current status as described above

9.23.5. MDS_Power_down Packet:

Field ID	Data
current_time	utc time
time_to_last_fix	local PC time
latitude	latitude
longitude	longitude
altitude	altitude
speed	sog
heading	cog
time_to_first_fix	ttff
distance	distance
max_speed	max_speed
driving_time	driving_time
total_messages	total_messages
sts_2	send TMO count
sts_3	mobidem rx_power
sts_4	mobidem power_mode
sts_5	mobidem battery_level
sts_6	mobidem network_contact
sts_7	mobidem traffic_area
sts_8	mobidem line_status
sts_9	mobidem network_check
sts_other	mobidem inbound_queue
sat_vis	number of visible satellites
sat_trk	number of satellites tracked
modem_retry	free buffer count
radio_switch	gps sts_2
dup_msg	gps sts_3
sbc_gps_test	gps sts_4
gps_mode	gps_status
current_status	current status as described above
status_of_last_fix	fix_sts

10. System Data Definitions and Build Procedure Description

Please see source code for detailed definitions of the system parameters.

10.1. System Variables

Please see source code for detail definitions of the system parameters.

10.2. Queues, Buffers, and Pools

Please see source code for detail definitions of the system parameters.

10.3. MIS Packet Description

This is the global packet description file used by all processes. Please refer to "Design Specification for System Handling of User Messaging" for the description of the packets. For details of the packet structure, please see source code.

10.4. Makefile Description

The following make file is used for this project:

makefile

Note: Modify CONFIG.H to configuration for DEMO, EXTERNAL, or BLACKBOX version and re-compile all the modules. May only set one of the following three flags to be 1.

- CFG_BBOX or
- CFG_EXTM or
- CFG_ DEMO

10.5. Linkfile Description

The following link files are used for this project:

MIS.LNK to build for mobidem(external and BLACKBOX) after re-compiling with CFG_BBOX or CFG_EXTM flag set.
DMIS.LNK to build demo version after re-compiling with CFG_DEMO flag set.

10.6. Build Procedure Description

10.6.1. Modify config.h to choose desired configuration.
10.6.2. Modify english.h to choose desired release version.
10.6.3. Compile and Link by executing make (MSDOS C 7.0, running under windows).
10.6.4. Run **usrcfg** program using one of the following RSP file depending on the configuration desired:

- BMIS.RSP for black-box version
- EMIS.RSP for external mobidem version

- DMIS.RSP for demo version.

10.6.5. Use sendhex program to download and burn EEPROM on the Symbol terminal.

11. Development Tools

The following development tools are used by the MDS Software:

11.1. Symbol 3000 Development Environment
11.2. Microsoft 7.0 C Compiler and Linker

WHAT IS CLAIMED IS:

1. A mobile data processing system comprising:

an interface unit, including a processor and a positioning system coupled to a first antenna and to the processor; and

a remote data terminal, electrically coupled to the interface unit during at least a first time period, capable of data transfers with the interface unit during the first time period and with a user.

2. The system of claim 1 wherein the processor is coupled to a second antenna and the remote data terminal includes a third antenna, and wherein the remote data terminal and the interface unit are capable of data transfer during the first time period using the second and third antennas.

3. The system of claim 1 wherein the interface unit and the remote data terminal are adapted to physically couple during the first time period.

4. The system of claim 1 further comprising a system power supply wherein the processing system is capable of operation independent of an external power supply.

5. The system of claim 1 wherein the interface unit is adapted to removably couple to an external power supply.

6. The system of claim 1 wherein the interface unit is electrically coupled to an external power supply of a vehicle.

7. The system of claims 1 wherein the remote data terminal includes a data terminal power supply.

8. The system of claims 1 wherein the remote data terminal includes a keypad.

9. The system of claims 2 wherein the first and second antennas are coupled to the interface unit with first and second cables and coupled to first and second magnetic bases respectively such that the first and second antennas may be removably mounted on a metallic surface.

10. The system of claim 1 wherein the positioning system is a global positioning system.

11. The system of claim 1 wherein the remote data terminal is adapted to be hand-held.

12. A method of processing data comprising steps of:

receiving user data from a user interface of a control unit;

receiving positioning data from a positioning system of the control unit; and

transmitting the user data and positioning data, using a second antenna of the control unit, to a base station.

13. The method of claim 12 wherein the step of receiving user data comprises:

entering the user data into a remote data entry terminal; and

transferring the user data to the control unit.

14. The method of claim 13 wherein the entering step comprises actuating keys on a keypad.

15. The method of claim 13 wherein the entering step comprises scanning a bar code.

16. The method of claim 13 wherein the entering step comprises scanning data.

17. The method of claim 12 wherein the transmitting step comprises transmitting data to a RAM network.

18. A method of processing data comprising steps of:

removably coupling a remote data terminal to a control unit that includes an antenna;

receiving positioning data in the control unit from the antenna;

transferring at least a portion of the received positioning data to the remote data terminal; and

displaying data indicative of at least a portion of the positioning data on a display of the remote data terminal.

19. A data processing system comprising:

a housing;

a user interface secured to the housing and adapted to receive user data from a user;

a positioning system including a first antenna;

a transmitter/receiver including a second antenna; and

a processor, coupled to the user interface, the positioning system, and the transmitter/receiver, capable of processing positioning data from the positioning system received by the first antenna, t/r data from the transmitter/receiver received by the second antenna, and the user data and further capable of transferring the processed data to the transmitter/receiver for transmission outside the system.

20. The system of claim 19 wherein the user interface is a plug adapted to receive a cable.

21. The system of claim 19 wherein the user interface is a cradle having electrical contacts.

22. The system of claim 19 wherein the user interface is a keypad.

23. A computer program product for use with a base station, a positioning system and a user interface, the computer program product comprising:

a computer-readable memory comprising:

code that processes user data entered into the user interface;

code that processes positioning data from the positioning system; and

code that directs transmission of the processed user data and processed positioning data to the base station.

24. The computer program product of claim 23 further comprising code that directs the user interface to prompt the user to enter the user data.

25. The computer program product of claim 23 further comprising code that directs the user interface to display the positioning data on a display screen of the user interface.

26. The computer program product of claim 23 wherein the code that processes the positioning data calculates a speed of the positioning system.

27. The computer program product of claim 23 wherein the code that processes the positioning data calculates a heading of the positioning system.

28. A mobile data terminal unit comprising:

a processor for processing user data;

a display for displaying output data;

a data entry tool for entering input data;

a power source for operating the mobile data terminal unit;

a mobile control unit removably coupled to the mobile data terminal and configured to transmit data between the mobile control unit and a base unit; and

a computer-readable memory including:

code that processes the input data;

code that processes the output data; and

code that directs transmission of the user data to a destination selected from a group consisting of the mobile control unit, the base unit and a mobile data suite.

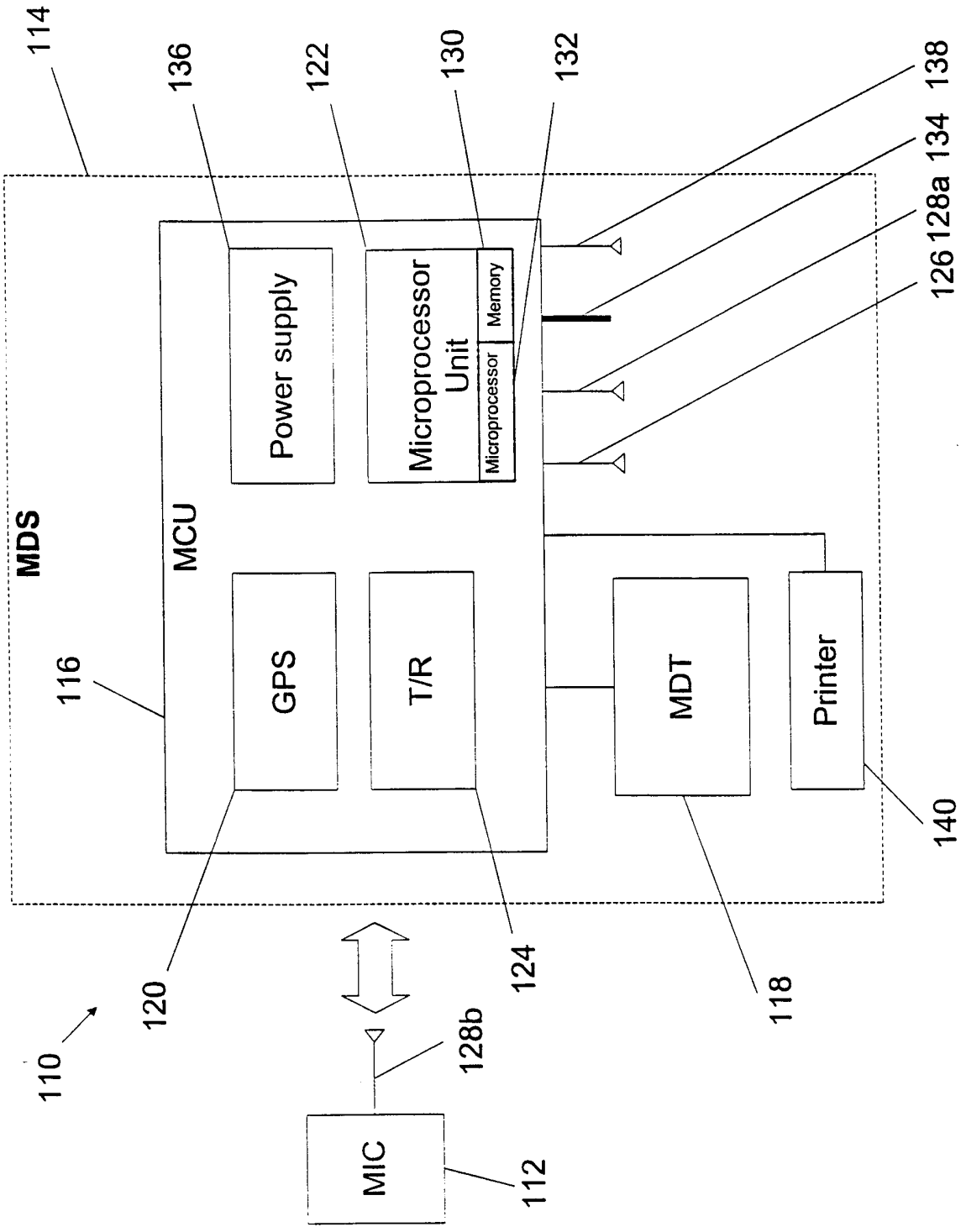

114
136
122
130
132
138
134
128a
126
MDS
MCU
Power supply
Microprocessor Unit
Memory
Microprocessor
GPS
T/R
MDT
Printer
116
120
124
118
140
110
128b
MIC
112

Fig. 1

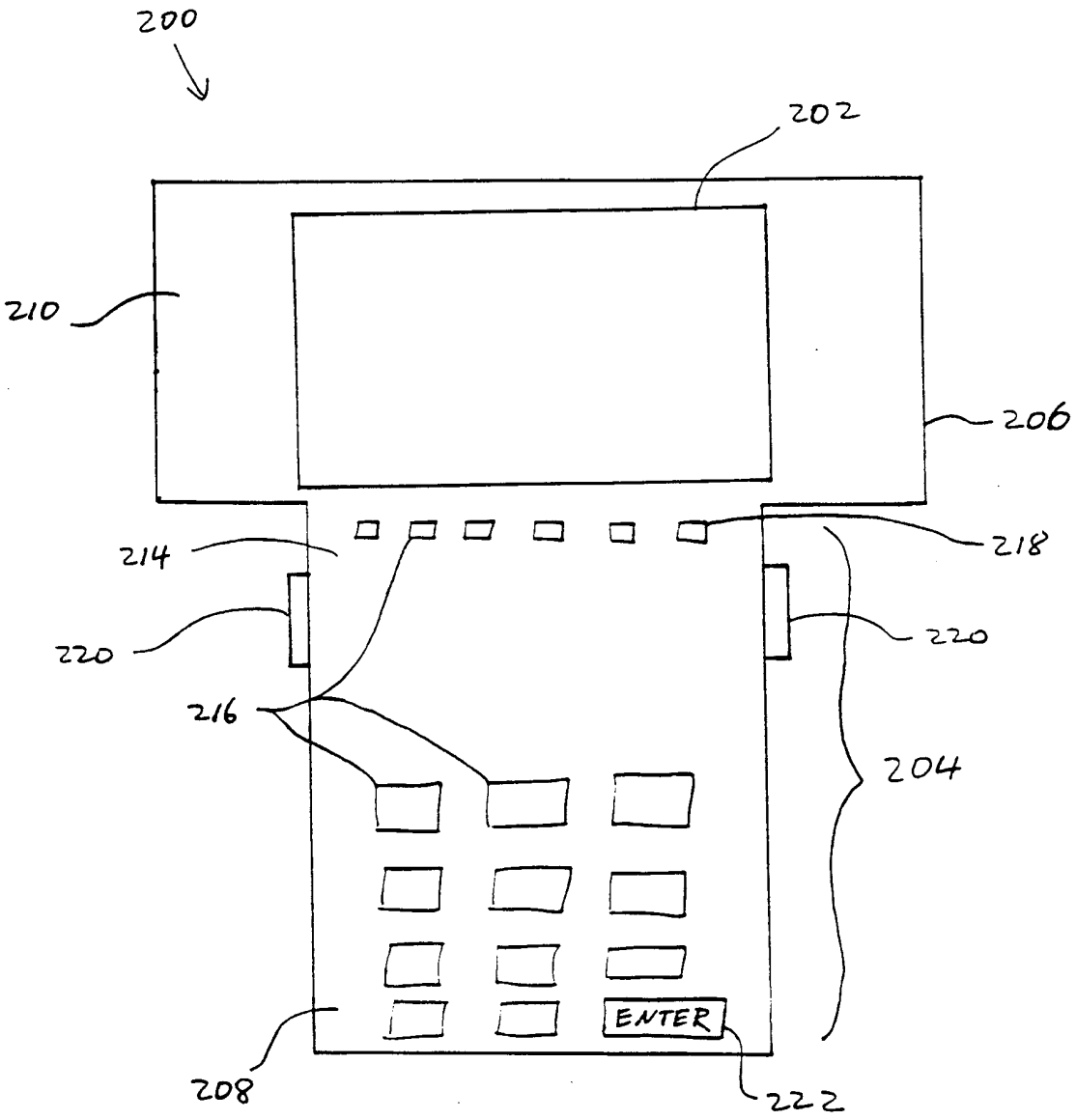


Fig. 2A

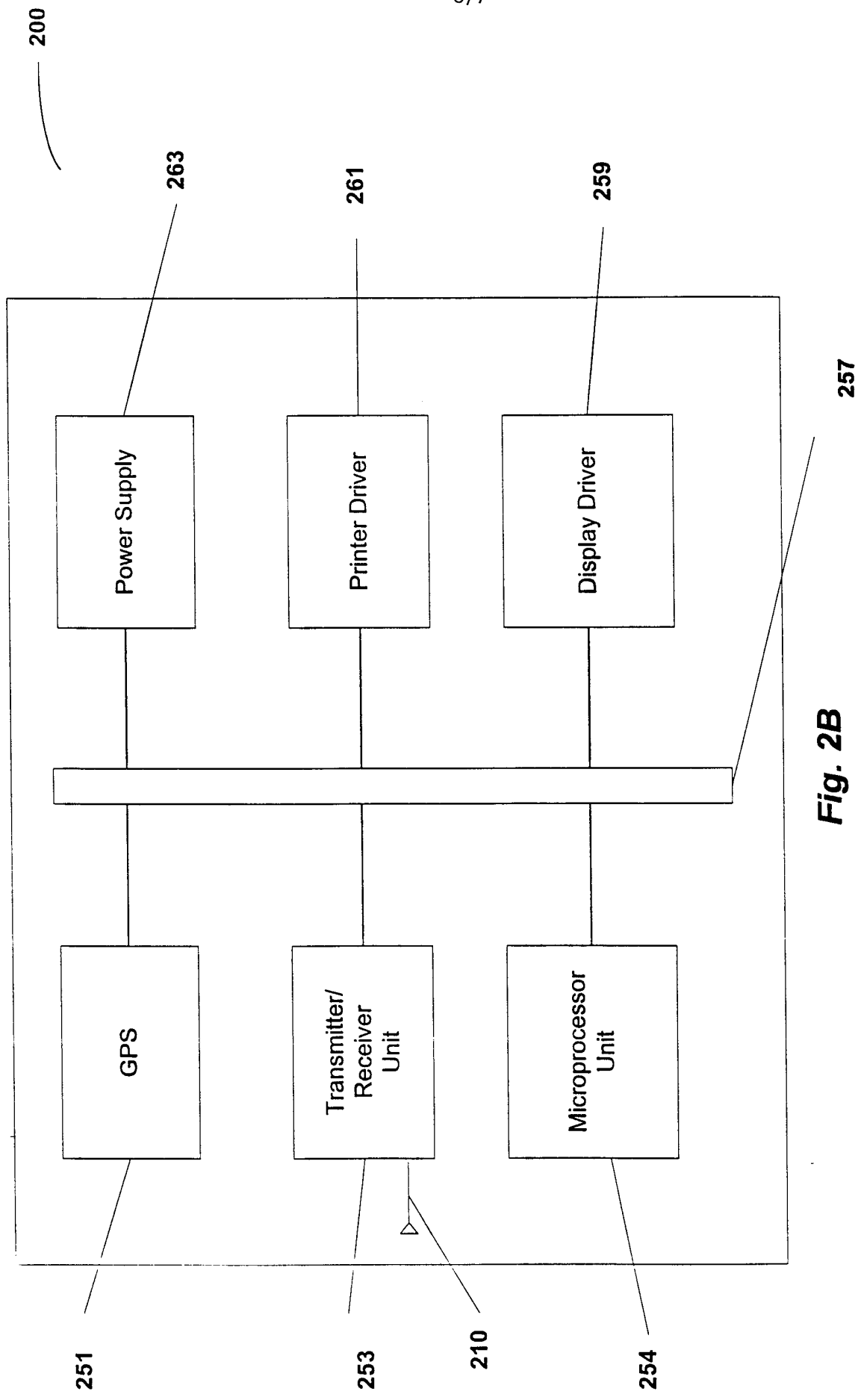

200
263
261
259
257
Power Supply
Printer Driver
Display Driver
GPS
Transmitter/
Receiver
Unit
Microprocessor
Unit
251
253
210
254

Fig. 2B

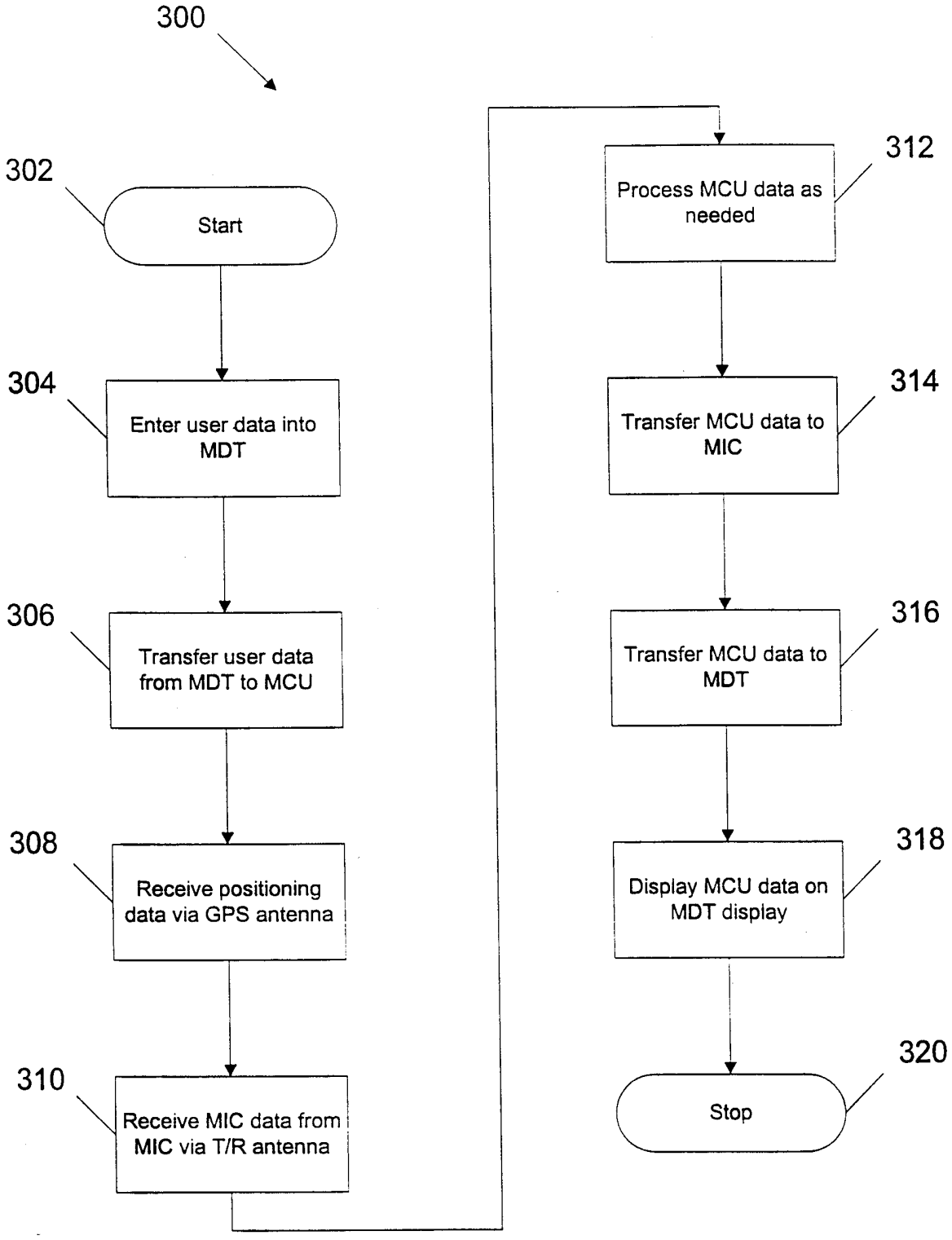

300
302
Start
304
Enter user data into MDT
306
Transfer user data from MDT to MCU
308
Receive positioning data via GPS antenna
310
Receive MIC data from MIC via T/R antenna
312
Process MCU data as needed
314
Transfer MCU data to MIC
316
Transfer MCU data to MDT
318
Display MCU data on MDT display
320
Stop

Fig. 3

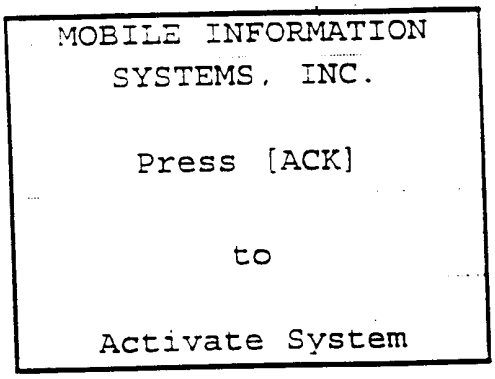


FIG. 4

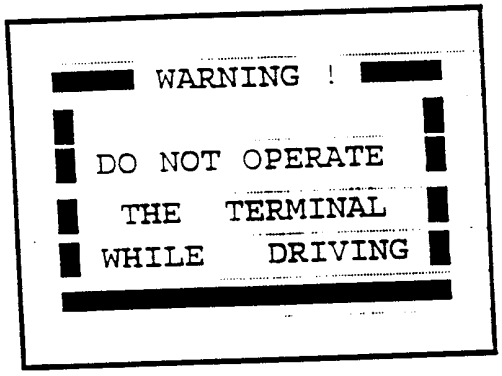


FIG. 5

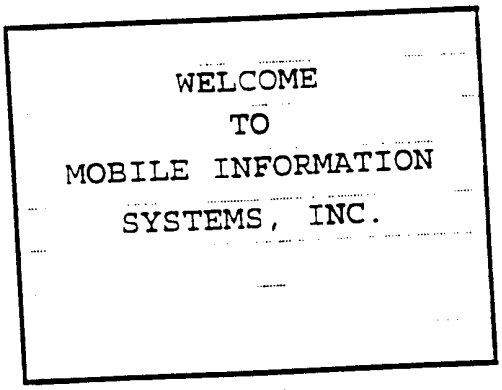


FIG. 6

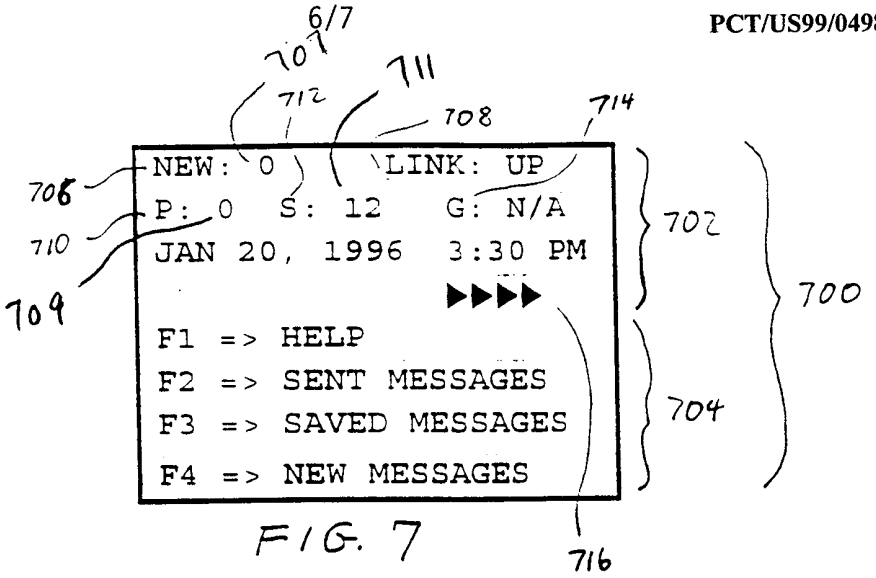


FIG. 7

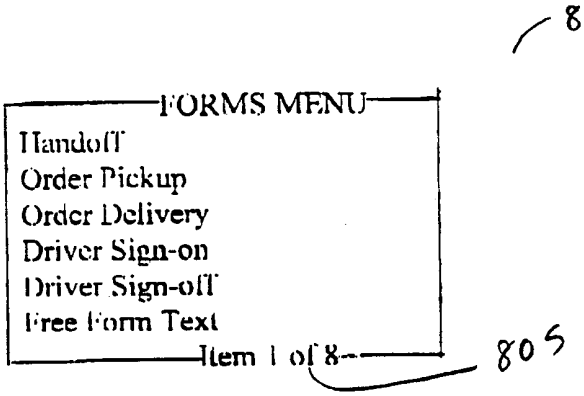


FIG. 8A

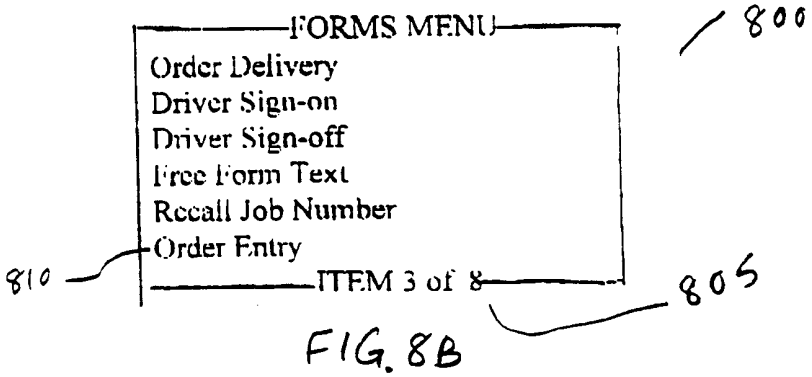


FIG. 8B

FIG. 8B

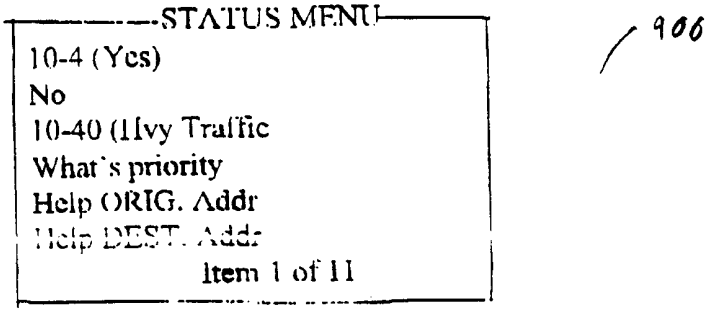


FIG. 9A

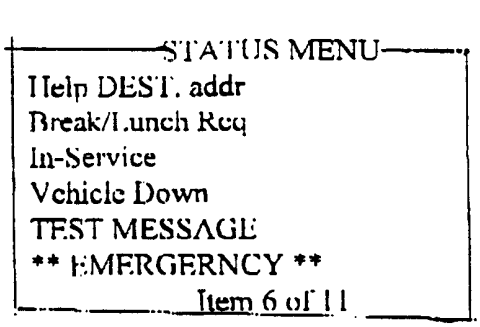


FIG. 9B

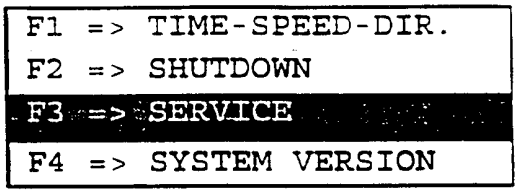


FIG. 10

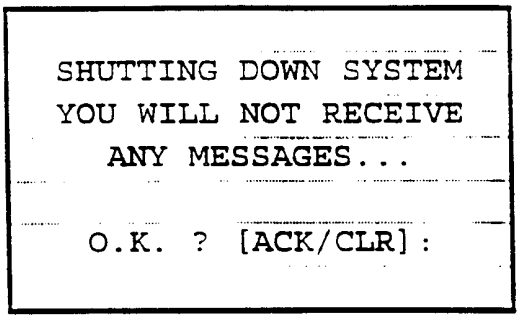


FIG. 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/04985

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 13/14, 13/20; H04L 12/64
US CL :370/349; 710/15, 33, 67

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 370/349; 710/15, 33, 67

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS
search terms: remote data transfer, mobile communication, antenna with user interface

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	us 5,638,374 A (HEATH) 10 June 1997, abstract, col. 6 line 1 - col. 7 line 64	1-28
Y	US 5,458,123 A (UNGER) 17 October 1995, abstract, col. 4 line 29 - col. 5 line 29	1-28
A	US 4,510,565 A (DUMMERMUTH) 09 April 1985, abstract and col. 25 line 21 - col. 26 line 59	1-28

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 02 JUNE 1999	Date of mailing of the international search report 14 JUN 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer THOMAS C LEE Telephone No. (703) 305-9717

Form PCT/ISA/210 (second sheet)(July 1992) ★