Vital Signs Monitoring for a Patient Data Management System in an ICU

Chris Tak Ming Yien

B.Eng., (McGill University), 1990

Department of Electrical Engineering
McGill University, Montréal
July 1993

A Thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements of the degree of M.Eng.

Abstract

This thesis presents the design and implementation of a Vital Signs Monitoring System for a Patient Data Management System in an intensive care unit. The Vital Signs Monitoring System provides graphical display of patient data to assist medical decision making. It performs real-time patient data acquisition, and supports data management. Visual coding of information has been investigated to ensure effective graphical representation of patient data, to reduce screen clutter, and to enhance interpretability of graphical displays. A survey of existing patient monitoring systems, and patient data management systems is presented to give an overview of the recent advancements in these medical systems. Emphasis is placed on the design of the user interface. Important interface design considerations are discussed, and a survey of interactive hardware, interaction tasks, and dialog style is presented.

The Vital Signs Monitoring System was developed in C language under the Presentation Manager window environment, and the operating system environment is OS/2 version 2.0.

Sommaire

Cette thèse présente la conception et l'implémentation d'un système de suivi des signes vitaux intégré à un système de gestion de données médicales concernant des patients au sein d'une unité de soins intensifs. Le système de suivi des signes vitaux procure sous forme graphique les données concernant le patient dans le bût d'aider le personnel infirmier dans sa prise de décision. Les données acquises en temps réel peuvent être ensuite gérées par le système. Une revue des systèmes de suivi ainsi que de gestion de données médicales existants fait part des récents développements dans ce domaine. Dans le bût de faciliter la lecture et l'interprétation des données médicales, des techniques de codage visuel sont abordées. De plus, une attention particulière a été portée à la conception de l'interface usager. Des considérations importantes concernant l'interface usager sont traitées. Les techniques courantes d'interaction tant sur le plan matériel, des tâches, que du style de dialogues sont également présentées.

Le Système de Suivi des Signes Vitaux a été développé en language C sous la version 2.0 de l'environnement Presentation Manager du système d'exploitation OS/2.

Acknowledgements

I would like to first give thanks to Professor Alfred Malowany for his supervision and his numerous helpful suggestions during my research, as well as to all my colleagues in the PDMS project team for their cooperation and suggestions during the development of PDMS.

Thanks also to Mr. Franco Carnevale and Dr. Ron Gottesman of the Montreal Children's Hospital for their help and evaluation during the development of our PDMS project. The financial support of NSERC is gratefully appreciated.

Finally, my deepest thanks to my parents and family members for their loving kindness during the past few years of my study here in Montreal. My study would have been impossible without their unfailing support and encouragement.

Table of Contents

List of Figures

List of Tables

1. Introduction

In the past two decades, computers have been widely applied to health care problems, and medical information processing. Systems have also been developed within the last four or five years to help in the planning of medical and nursing care, to provide tools for supporting decision making, and to optimize the effective use of computers in the management of nursing resources.

Medical personnel are increasingly demanded to deliver quality care in less time, with fewer resources, and in a much more complex environment. With the number of bedside monitoring devices available today, the quantity of data collected on a single patient in the ICU could be overwhelming. There are pressures to measure and control staffing, provide thorough and effective care plans, improve productivity, and provide increasingly extensive documentation of the care given. The advances in computer technology and the reduction in cost of hardware have encouraged the use of patient data management systems (PDMS) to meet these demands especially in the intensive care unit (ICU).

One of the daily activities performed by a nurse in the ICU is to collect patient data for trend graphs from the bedside monitors. Studies have shown that conventional manual record keeping is less reliable compared to a computerized patient data management system, and physicians make significantly less mistakes in recalling patient data when using a patient data management system. Graphical presentation of physiological data provided by a computer system is a fast, and reliable tool for medical decision making. Automatic data acquisition and computer graphics combined with a window user interface environment have the potential to decrease the time that nurses spent on paperwork, and improve the quality of patient care.

Researchers have found that a successful patient data management system not only has to meet the requirements in functionality, but it must also have a user-friendly interface, and it should be modified and upgraded easily to accommodate the rapid changes in computer technology and the ICU environment. A new patient data management system that requires drastic changes in the way medical personnel are working in the ICU may easily be rejected due to the lack of consideration of the human factors of the system design.

This thesis presents a Vital Signs Monitoring System (VSMS) of a PDMS developed for a pediatric ICU in collaboration with the Montreal's Children's Hospital. The next section will develop the issues just described here. The needs of a modern ICU are illustrated and some of the existing systems are reviewed. Section 1.2 discusses the human factors in the development of medical systems. A survey of existing user-computer interaction hardware, and various interaction styles follows. The last section of this chapter gives an overview of this thesis, which presents the design and implementation of the VSMS.

1.1. Computers in Nursing

The benefits of using computers in nursing have been documented in numerous reports and journal papers [Ford, 1990] [Hendrickson & Kovner, 1990]. Although computerised systems offer the advantage of more accurate and reliable data management, effective use of nursing resources, and improved quality in patient care, studies have suggested that the success of a computer system maybe related to the user acceptance and development of positive attitudes [Chang, 1984] [Burkes, 1991]. A survey of studies conducted in the late 1960s and 1970s found that the average staff had a negative attitude toward computers [Bongartz, 1988]. These findings studied the attitudes of all employees in a given institution and therefore did not specifically identify the attitudes and perceptions held by professional nurses. More

recent surveys have shown that nurses' attitudes were generally favorable to the use of computers in nursing practice [Jacobson *et al*, 1989] [Scarpa *et al*, 1992]. A study in 1989 showed that more positive attitudes were found in nursing students with more computer experience [Schwirian *et al*, 1989], and a similar result was also found in practicing nurses [Scarpa *et al*, 1992]. However, nurses who had experiences with unreliable computer systems were less favorable toward the use of computers in nursing [Jacobson *et al*, 1989]. This suggested that a good understanding of the operations carried out by the nurses is required, that the system has to be user-friendly, as well as functionally robust.

A number of computer-based data management systems have been developed and installed in hospitals in the past decade, and many of them are installed in the intensive care unit (ICU). Gardner [Gardner *et al*, 1989] identifies four functions of computers in an ICU setting: data communication, medical records management, physiological monitoring, and expert systems that support decision making. The following subsections introduce the various elements of a patient data management system, and a sampling of the existing systems are presented.

1.1.1. Medical Data

The gathering of data and their interpretation are central to the health-care process. The types of medical data include narrative, textual data, numerical measurements, recorded signals, and pictures. A considerable amount of information is gathered in the form of narrative data which is composed of the description of a patient's illness, his social and family history, and the communications among medical personnel. Many data used in medicine take on discrete numeric values. These include parameters such as laboratory-test results, vital signs (pulse rate, respiratory rate, and arterial blood pressure, etc.), and certain measurements taken during the physical examination. In some fields of medicine, analog data, such as ECG, in the form of continuous signals are particularly important. Images, or pictures, can be

acquired from machines or sketched by the physicians.

Physicians play a key role in collecting medical data during interviews with patients [McDonald & Barnett, 1990], while nurses play another important role in data acquisition from their observations of patients [Ozbolt *et al*, 1990]. Generation of nursing care plans, and assessment of patients by physicians are greatly influenced by the data gathered by the nurses. Other health-care workers such as office staff, admissions personnel, therapists, laboratory personnel, radiologists, and pharmacists contribute to the data-collection process also. Finally, data are gathered from devices such as laboratory instruments [Smith & Svirbely, 1990], imaging machines [Greenes & Brinkley, 1990], and physiological monitors [Gardner, 1990].

An important function of medical data is to help in providing coordinated care to a patient over time, and to provide communication among health-care workers. Medical data form the basis for the historical record, and provide the information for anticipating future health problems for individual patients. Clinical research can profit from the medical data collected from populations of patients through the aggregation and statistical analysis of observations [Shortliffe & Barnett, 1990].

There are disadvantages in the traditional information storage and retrieval of paper records. In an ICU, patients are usually connected to a number of bedside monitors. Data are periodically collected from the instruments by nurses, and are presented as graphs or tables over a period of time. Because of the stressful environment in the ICU, this procedure is error prone [Hammond *et al.*, 1991]. The traditional record system also has the problem of redundant recording in order to match alternate modes of access, and is inefficient in meeting the goals of different medical personnel. Furthermore, retrospective chart review for clinical research is a laborious and tedious process, and people performing it are prone to make transcription errors and to overlook key data [Shortliffe & Barnett, 1990]. Because of these disadvantages, more and more bedside monitors feed their results directly into a computer [Dasta, 1990] so that the data can be analyzed or formatted for electronic

storage as well as reported on paper.

Many improvements to the acquisition, usage, and storage of medical data are possible through the use of computer systems [Bishop, 1990]. First of all, medical records, stored in electronic form, are always available on demand, and can be assessed and viewed from many locations within or without the hospital by using local area networks [Price & Chandrasekhar, 1989], and wide area networks [Scherrer, 1990]. Secondly, computers can assimilate, process, and display large amounts of data in easy-to-understand formats for the users [Dasta, 1990]. Printed reports produced by computers can provide quick and legible patient summaries. Graphical on-screen displays can better highlight trends in patient data [James *et al*, 1990]. Hypertext links provide customized browsing features [Chaney *et al*, 1989], while multimedia technology permit integration of data of various nature such as numerical data from monitors, taped voice reports, image data, and motion video of surgical procedures [Goldberg *et al*, 1989] [Brown & Krishnamurthy, 1990] [Chnadrasekhar & Price, 1989]. Finally, patient data management systems allow medical staff to perform complex queries on the patient data [Safran *et al*, 1990].

1.1.2. Patient Monitoring Systems

Normally associated with operation of life support equipment, patient monitoring is the continuous observation of a patient's physiological function for the purpose of making and assessing therapeutic interventions [Hudson, 1985]. Patient monitors are used to collect and display physiological data for efficient critical care decision making where speed and accuracy are important factors.

In 1903, Cushing stated that vital-sign measurement should be done routinely, and that accuracy was important [Cushing, 1903]. Since the 1920s, vital signs (there were four at that time: temperature, respiratory rate, heart rate, and arterial blood pressure) have been recorded in all patient charts, and measurement of physiological parameters has become a central feature of critical care nursing. As new therapeutic

interventions were developed, prompt quantitative evaluation of physiological and biochemical data became essential in the decision-making of physicians. ICUs were established in hospitals in the 1950s to meet the increasing needs of critically ill patients for more acute and intensive care. The advances in electronic instrumentation has increased the number of physiological variables that could be monitored, and monitoring devices have found their way rapidly to the bedside in the ICU. Nurses and physicians in ICUs are now confronted with a bewildering number of instruments and an overwhelming amount of physiological data.

Although the monitoring devices are continuous, the parameters monitored by the physiological monitors are not automatically recorded in the patient's chart, and the nurse has to take a reading at half-hour or hour intervals and enter it into the paper chart [Collet *et al.*, 1989]. Transcription errors occur frequently as described in the last section, and papers are often missing or misplaced. Another problem with the manually recorded data is the loss of information in sampling. As shown in Figure 1.1, a continuous signal sampled at different sampling rates may result in different degrees of accuracy, and important features may be missing between samples.

In the late 1960s, computers were introduced into the ICU for patient monitoring to increase the speed and accuracy of physiological data gathering, and to increase the efficacy of intensive care. However, the hardware and operating costs at that time were so expensive that very few hospitals were able to buy these computers. With the advances in computer technology, and the decrease in hardware costs in recent years, computer-based monitoring has become feasible and affordable.

Many systems have been developed in recent years to attack different problem areas in monitoring. A microcomputer based obstetrics information management system was developed by Subramanian to monitor an eight bed unit continuously [Subramanian, 1989]. This system can display waveform information on a central monitor for all eight beds simultaneously. Strickland Jr. [Strickland Jr., 1991] de-

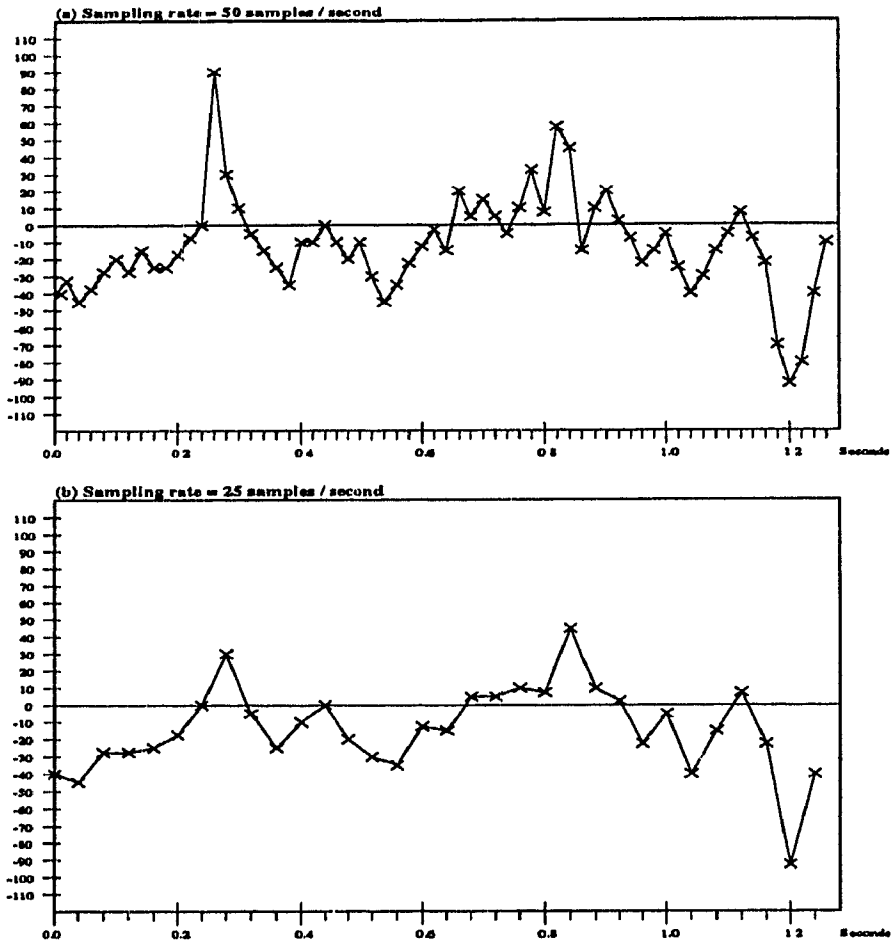

Figure 1.1 Analog to Digital Conversions at Two Different Sampling Rates

scribes an information management system developed at the University of Louisville Medical School to assist in the monitoring of patients with severe head injuries which concentrates on the recording of brain function. Smith describes some application programs running on bedside and central station monitors, which provide not only information on vital signs, but also information about drug therapy and pacemaker data [Smith, 1992].

A trend analysis module in a PDMS has been developed by Collet *et al.* for the Montreal Children's Hospital which analyzes cardio-vascular data from physiological monitors to generate an early warning alarm [Collet *et al.*, 1990]. Relationships

between different parameters are analyzed to recognize trend patterns.

Krieger [Krieger *et al*, 1991] describes a distributed real-time system for monitoring neurophysiologic function that provides immediate access to real-time life-critical data being acquired at multiple sites across the health center and allows one neurophysiologist to simultaneously monitor multiple surgical procedures. Collura [Collura *et al.*, 1992] describes a continuous EEG monitoring system for Epilepsy integrated with audio/video monitoring and seizure alarms.

1.1.3. Patient Data Management Systems

Some medical systems can handle only a specific physiologic monitoring, and their databases are usually limited. A comprehensive database is required in the ICU environment to integrated different physiologic data, and administrative data related to patients. A Patient Data Management System (PDMS) is a system that is capable of performing operations on such a comprehensive database. A PDMS should be able to collect, organize, store, retrieve, manipulate, and analyze the high volume of physiologic and administrative data required for accurate data acquisition, reliable patient monitoring, efficient medical record management, and critical care decision making [Milholland, 1988], [Gardner *et al.*, 1989].

PDMSs were introduced into the ICU over two decades ago, and their basic function at that time was to perform patient monitoring. Physiological monitoring systems can monitor physiological and biochemical variables, and the monitoring can be performed either continuously or intermittently. A decade after PDMSs were brought into the ICU, the focus on PDMS had begun to shift to the aspects of database management [Milholland, 1988] [Malowany *et al*, 1989] [Schroeder & Carter, 1989]. PDMSs are now able to integrate data from different sources, such as various bedside monitors [Fumai *et al*, 1991], laboratory results, manually entered data, and nursing care plans [Roger *et al*, 1991]. There are systems that perform real-time trend analysis [Collet *et al*, 1990], initiate therapeutic interventions [Tolbert &

Partuz, 1977], and plan nursing care [Bailey, 1988] [Probst & Rush, 1990].

The data management functions that a PDMS should be capable of performing are data entry, data storage, data retrieval, multiple access, data integration, and data security [Milholland, 1986]. Data storage in a PDMS avoids redundant duplication of patient data, and provides fast access to a patient's past medical history. Data retrieval includes the representation of data which can be displayed on a computer screen, or printed as hardcopies. Studies have shown that patient charts generated by a PDMS have improved the quality, accuracy, and accessibility of nursing documentation [Hammond *et al*, 1991] [Hendrickson & Kovner, 1990] [Staggers, 1988]. Data retrieval using PDMS is also more suitable for research analysis than traditional paper record system [Hammond *et al*, 1991]. Multiple access on a PDMS improves the consistency, and accessibility of a patient's data to different medical personnel. Data from different sources are integrated into the same PDMS database so that relationships among different parameters can be analyzed to provide more efficient and effective health care. In order to provide reliable critical care, data security of a PDMS is important to assure data validity and consistency, as well as the confidentiality of a patient's data.

There have been a number of successful commercial, and customized hospital-developed PDMSs implemented in the past few years. Some commercial systems are: the Hewlett Packard PDMS system [Hewlett, 1988], the EMTEK system [EMTEK, 1988], the MedTake system [Pesce, 1988], and the CliniCare [Hughes, 1988]. The hospital-developed systems include: the Health Evaluation through Logical Processing (HELP) system [Bradshaw *et al*, 1989], the workstation-based system for use in critical care environments [Prakash *et al*, 1991], the medical workstation with knowledge-based user support and multimedia documents [Kuhn *et al*, 1990], and the PDMS for an ICU in Montreal Children's Hospital [Panisset *et al*, 1989]. Factors influencing the choice between hospital-developed and commercial systems are: ability to customize, functionality, cost, user interface, and support for equipment

from different vendors [Paganelli, 1989].

Positive results have been reported from the studies on the effects of using PDMS. Time spent on the clerical activities, such as writing or telephoning for services and supplies, and assembling charts, has been reduced [Hendrickson & Kovner, 1990] [Staggers, 1988], and thus resulted in gains of efficiency and productivity [Hammond *et al*, 1991]. Errors in carrying out orders have been reduced [Hendrickson & Kovner, 1990].

Although time spent on direct patient care was expected to increase as a result of saving time in clerical activities, different results have been found in the literature. Some studies found that time spent on direct patient care had been increased [Roger, 1992], while other studies have shown that time spent in direct patient care remained the same [Hendrickson & Kovner, 1990] [Staggers, 1988]. Many benefits have been documented in health care literature, but few empirical studies have been completed to verify these perceived benefits. Further research with improved methodology is required to address variables not yet researched: the effect of computer on patient care quality, nurse administrators' attitudes toward computers, the channeling of saved time into cost-effective activities, and the expansion of programs to help make nurses more efficient [Staggers, 1988].

1.2. User-Computer Interaction

The human-factors of a program, such as its interaction style and its ease of learning and of use, are as important as its functional completeness and correctness. Program efficiency continued to be the main concern of computer scientists until the early 1980s. As the costs of hardware and software have dropped drastically in the last decade, large number of computer facilities now exist in our modern offices and homes. With the increase in computing power and the decrease in costs, we can

afford to optimize user efficiency rather than computer efficiency because a poorly designed user interface usually results in the rejection by the user despite its sound functionality. A well designed user interface is an important factor to lower the hidden costs to nurses [Staggers, 1991]. These hidden costs include the training time of how to use a system, memory burdens of recalling different functions of a system, and the errors made in an interaction with the system. In such critical application as ICU monitoring, a poor user interface can contribute to and even cause fatal accidents.

An important property of graphics packages, called device independence, is used to protect programs from obsolescence, and to enhance the portability of applications to different platforms. Device-independent programs are shielded from the details of the physical input devices by using a set of logical input devices. Special code modules, called device managers, are used to map application programs' graphics requirements into the specific graphics vocabulary of the target device [Warner, 1981].

An approach to application design, called dialog independence, isolates the human-computer dialog from the system structure, and the computational software [Hartson & Hix, 1989]. Without dialog independence, dialog and computational modules are tightly interspersed in the system, which makes modification difficult to perform, and human factors hard to provide. Dialog independence allows the computational and dialog components of an application to evolve independently, and the dialog may even be shared among different applications [Coutaz, 1985].

The evaluation of interaction devices can be considered on three levels: device, task, and dialogue. The hardware characteristic such as device resolution, operator fatigue, and device footprint (the area that a device occupies), is determined at the device level. At the task level, interaction techniques are compared by using different devices for the same task. A device is evaluated at the dialogue level if a sequence of interaction tasks is performed. The following sections discuss the three

basic components of user interfaces: input devices, interaction tasks, and interaction style.

1.2.1. Interaction Hardware

There are four types of input devices: locator, keyboard, valuator, and choice devices. A locator is a device for specifying coordinates or orientations, while a valuator is used for entering a single real number. A logical keyboard device is used for specifying character string input. A choice device is used to select from a set of objects of actions [Foley *et al*, 1990].

Locators are classified according to three independent characteristics: absolute or relative, direct or indirect, discrete or continuous. Absolute devices, such as data tablet or touch panel, have a frame of reference and report position with respect to that origin, while relative devices, such as mice, track ball, and joysticks, report only changes from their former position. The advantage of a relative device is that an arbitrarily large change in position can be specified. Direct devices, such as light pen or touch screen, are used to point directly at the screen with a finger or surrogate finger, whereas indirect devices, such as a tablet, mouse, or joystick, move a cursor on the screen. Precise positioning is difficult with direct devices, and direct pointing can cause arm fatigue, if the arm is not supported. Continuous devices (e.g. mice) can create a smooth cursor motion on the screen, and thus allow more natural, easier, and faster cursor movement than do discrete ones.

Valuator devices can be either bounded or unbounded. A bounded valuator, like the dial on a radio, inputs an absolute quantity. On the other hand, a continuous-turn potentiometer can be turned an unbounded number of times in either direction.

Although different types of keyboard have been designed to replaced the traditional Qwerty design, Qwerty keyboards remain the most widely accepted keyboard

organization.

Choice devices include function keys on a keyboard, and keys mounted on the CRT bezel. The position of the mounted keys affects the efficiency of a choice device. Keys mounted in the keyboard is the most common form of choice device.

Other experimental interaction devices, such as a voice recognizer which frees the user's hands for other tasks, is a natural way of communication [Bailey, 1988] [Waterworth, 1984]. Voice recognition systems can be classified as either speaker-dependent or speaker independent, and being capable of single word recognition or continuous speech recognition. On a speaker-dependent system, the accuracy of the system would be highly impeded by the variation of the voice of the speaker. For example, if the user has a cold, the system may not be able to recognize his voice at all. Speaker-independent systems are not without limitations. They usually have a much smaller vocabulary; typically 50 to 100 words. Several speech recognition interface systems have been successfully developed [Kuhn *et al*, 1990] [Petroni *et al*, 1991].

1.2.2. Dialog as an Element of Interaction

A basic interaction task (BIT) is the smallest unit of task that a user can interact with the system. Compositions, or sequences, of BITs forms dialog interactions, which will be discussed in detail later in this section. A complete set of BITs for interactive graphics includes positioning, selecting objects, entering text, and entering numeric quantities [Foley *et al*, 1990].

A positioning task enters a coordinate in the form of (x, y) or (x, y, z) to the application program. Positioning can be achieved by moving a cursor on the screen, or typing the desired coordinates. The feedback from the application program during positioning must be given in the screen-coordinate system, in order to follow the principle of stimulus-response compatibility; the system response to the user action

must be in the same direction or orientation, and the magnitude of the response should be proportional to the action. Superimposing a grid on the work area at low intensity is an useful visual aid in aligning positions [Swezey & Davis, 1983].

A selection task chooses an element from a choice set such as commands, attribute values, object classes, and object instances. There are two types of choice sets: fixed-sized, and varying-sized choice sets. Selecting an object is possible by using different kinds of logical devices. For example, a keyboard device can be used to type the name of the object, or a locator can be used to point at an object which can then be followed by pushing a button on the locator indicating a selection.

A text input task enters character strings to an application, and it is usually performed by typing at the keyboard.

A quantify task specifies a numeric value within some valid range. Typical interaction techniques are typing the value, setting a dial to the value, and using an up-down counter to select the value. This task may be either linguistic or spatial. When it is linguistic, the user enters a specific value. When it is spatial, the user seeks to increase or decrease a value by a certain amount.

A composite of BITs forms the dialog between the user and the computer. In the user-computer dialog, dialog boxes are used to specify multiple units of information [Foley *et al*, 1990]. Dialog boxes can allow multiple items to be selected in the same selection set, provide multiple selection sets, and include areas for text and value entry.

1.2.3. Dialog Styles

This section discusses four of the most common dialog styles for user interface design: menu selection systems, command languages, direct manipulation, and iconic user interfaces. A good user interface should focus on human factors such as ease of use, speed of learning, retention rate, and error handling.

Menu selection systems reduce the learning time for a novice user, and they provide a structural sequencing of complex commands [Shneiderman, 1987]. Another advantage of using menu selection systems is that they eliminate the memorization of command sequences by recognition of menu items [Foley *et al*, 1990] [Shneiderman, 1987].

Direct Manipulation systems create a visual representation of objects, attributes, or task concepts that can be operated on by manipulating objects on the screen [Shneiderman, 1982]. They are easy to learn for novice users, and tasks can be carried out extremely fast by experts [Shneiderman, 1987]. Error messages are rarely needed, since the user can immediately see the result of his actions on the screen.

Command languages are used traditionally to interact with a computer. Systems using command languages are flexible, and easy to extend. Possibility of typing and recall errors is high, and the learning time for command language systems is long [Foley *et al*, 1990]. Therefore, they are most appropriate for experienced, frequent users, rather than for casual novice users [Shneiderman, 1987].

Iconic interfaces use icons (pictures) to represent objects, properties, or user commands instead of textual description. Iconic user interfaces reduce the learning curve, and facilitate user performance [Lodding, 1983]. If carefully designed, icons can be language independent, and systems using them can be delivered to the international market [Foley *et al*, 1990] [Lodding, 1983].

1.2.4. Visual Codings

Since there is a large amount of data to be displayed on a limited area of the screen, visual coding is very useful in distinguishing different types of data, and in grouping relevant objects together. Many different coding types are available: color, shape, size, length, typeface, angular orientation, brightness, texture, line width, and line

style are the common ones in designing user interfaces. It is obvious that the degree of confusion experienced by the user is directly proportional to the number of data types encoded [Swezey & Davis, 1983]. Redundant coding can be used effectively to help the user to distinguish among different types of information. It uses a combination of two or more different codes to represent the same information at the same time. Figure 1.2 illustrates a redundant code using three codes: shape, fill pattern, line width, and line style, to differentiate different objects.

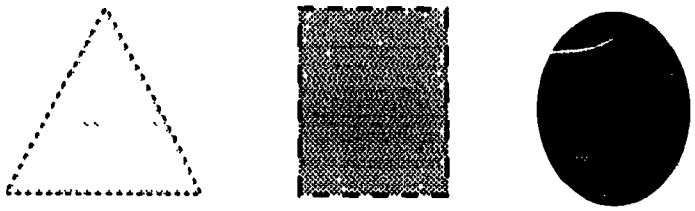

Figure 1.2 An Example of Redundant coding

For a 95% error-free code recognition, a maximum number of 10 colors, 6 area sizes, 6 lengths, 4 intensities, 24 angles, or 15 geometric shapes can be used for coding [Swezey & Davis, 1983] [Foley *et al*, 1990]. The error rate can be further decreased by the use of a legend indicating the meaning of each code value.

It is important to understand the type of information being coded when selecting a code, because some codes are more efficient in recognition for a particular type of information than others. There are mainly three types of information: nominative, ordinal, and ratio. Nominative information simply designates different things which have no notion of order. Ordinal information is ordered and has a greater than and less than relation, while ratio information has also a metric defined. For a fixed number of nominative-code values, color is the most accurate in recognizing objects when the code values are appropriately spaced [Rice, 1991] [van Cott & Kinkade, 1972]. Ordinal information must use codes that have an obvious ordering

such as fill-pattern density, or text size. Ratio information must be presented with a code that can be vary continuously. Graphs positioned along a common scale is found to be the most accurately recognized coding for ratio information [Foley *et al*, 1990].

A study by Hoadley [Hoadley, 1990] has shown that the use of color in tables, pie charts, and bar graphs improves the time performance of a decision maker's ability to extract information, while it improves the accuracy performance when used in pie charts and line graphs. The effective use of visual codes, and a number of presentation schemes are illustrated extensively in two books written by Bertin [Bertin, 1981] [Bertin, 1983].

1.2.5. User Interface Design Considerations

This section presents a number of important interface design guidelines which should be followed closely, but not blindly. An effective application of these guidelines can produce good human factors in a design.

1. **Principle of Least Astonishment** - design a system, or its features, to surprise the user as little as possible [Thimbleby, 1990].

2. **Consistency** - allow the user to generalize knowledge about one aspect of the system to other aspects by following a few simple rules, and maintaining uniformity in the conceptual model, functionality, sequencing, and hardware bindings [Foley *et al*, 1990].

3. **Minimize Error Possibilities** - disable unavailable items, provide only valid commands in the current context and mode, and avoid side effects which the user does not expect [Foley *et al*, 1990].

4. **Feedback** - providing normal prompts, advisory messages, and system responses to commands may influence user perceptions, but the phrasing of

error messages or diagnostic warning is critical [Shneiderman, 1987].

5. **Error Recovery** - provide some forms of error recovery such as *Undo*, *Cancel*, or *Abort*, when an unexpected result is encountered [Thimbleby, 1990].

6. **Human Diversity** - accommodating the different skill levels, as well as the physical, intellectual, and personality differences among users is vital [Shneiderman, 1987].

7. **Minimize Memorization** - never force unnecessary memorization [Thomson, 1984] [Foley *et al*, 1990].

1.3. Thesis Overview

This thesis presents the design and implementation of a Vital Signs Monitoring System (VSMS) for a Patient Data Management System (PDMS) which is being developed in conjunction between McGill Research Center for Intelligent Machines (McRCIM) and the Montreal Children's Hospital. The PDMS is intended to be used in the Pediatric Intensive Care Unit of the Montreal Children's Hospital. Being an integral part of the PDMS, the VSMS is responsible for real time data acquisition, and graphical display of physiological data for patient monitoring. Emphasis is placed on the consideration of human factors in medical systems. The user interface of VSMS is developed under the OS/2 Presentation Manager, and utilizes the multitasking capability of the OS/2 operating system.

An overview of the PDMS system is presented in Chapter 2. The hardware and software environment of the system is described in detail, and the integration of all the modules in PDMS is also formally discussed here. The functionality and structure of each individual module is presented, and the design decisions and tradeoffs are mentioned as well. Low-level implementation details of all the modules is

not the objective of this chapter, and thus will not be discussed here.

Chapter 3 presents an overview of the VSMS, and details the design issues of the system. Functional models of different components in the VSMS system are given, and the design decisions of each module are justified. Topics discussed include real-time data acquisition, real-time data management, graphical representation of vital signs data, and graphical user interface environment.

Chapter 4 develops with the implementation of the modules outlined in Chapter 3, and the performance of different aspects of the VSMS system is evaluated for future improvements. Possible extensions to the PDMS system are also suggested here. Finally, the material presented in this thesis is summarized in the conclusion of Chapter 5.

2. PDMS Overview

This chapter describes the Patient Data Management System (PDMS) developed for the Pediatric Intensive Care Unit (PICU) of the Montreal Children's Hospital. The PDMS is capable of collecting, storing, retrieving, and manipulating both physiological data and administrative data available from bedside monitors, health care staff, and laboratories. The functions that are supported by the PDMS are patient registration, vital signs monitoring, fluid balance monitoring, generation of nursing care plans, nursing workload calculation, request of laboratory tests and entry of laboratory results, requests of pharmaceuticals, and tracking of medications. The current implementation of the PDMS will be outlined.

The hardware configuration of the PDMS in the PICU is shown in Figure 2.1. The HP CareNet System consists of fourteen HP 78534A physiological monitors linked in a star configuration local area network to a HP 78581A Network Systems Communication Controller. Each bedside monitor is capable of automatic smoothing of measured parameters, real-time display of selected data, and alarm generation. A HP 78588A CarePort Network Interface is used to send data gathered from the bedside monitors to the PDMS host computer through a standard RS-232C serial line, and to translate the commands of the host computer into the proper format for the Network Systems Communication Controller.

The PDMS host computer system consists of an IBM PS/2 model 50 computer, and an IBM PS/2 model 80 computer connected to an IBM Token Ring network using the IBM OS/2 LAN server/requester network software. The IBM PS/2 model 80 computer functions as the PDMS server, which is dedicated to the acquisition and storage of patient data. It has an 80386 processor, 8 Mbytes of RAM, a 150 Mbyte hard disk, and an 8514 high resolution colour display. The model 50 computer, which has an 80286 processor, 5 Mbytes of RAM, a 30 Mbyte hard disk, and an

8514 colour display, is used mainly as a front-end system for the interactive PDMS modules such as vital signs monitoring, and fluid balance monitoring.

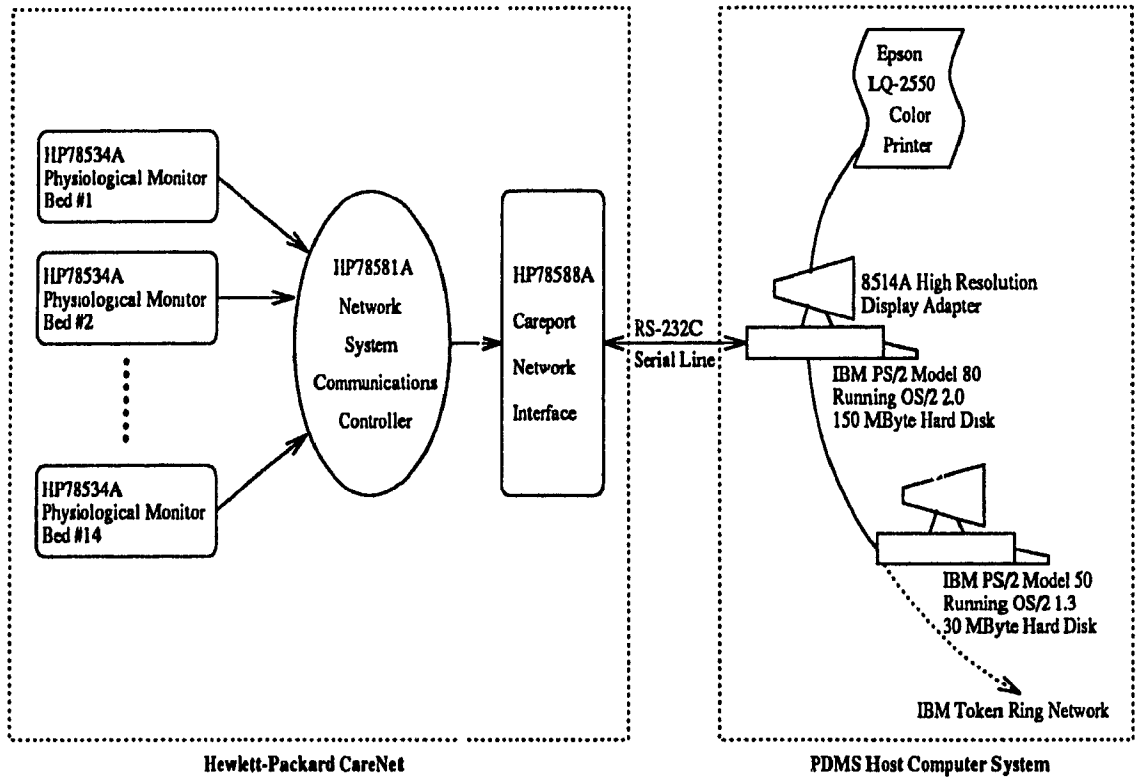

Figure 2.1 Hardware Configuration of PDMS

The model 80 and model 50 computer systems currently run OS/2 versions 2.0 and 1.3 respectively. Figure 2.2 presents an overview of the PDMS software structure. The system is first started up by running PDMS Manager, which is not shown in the diagram. The PDMS Manager does the initialization for the system, and starts up the Data Link Controller (DLC) module for the communication between the HP CareNet system and the PDMS host system. After the initialization has been done, the PDMS Manager is used either to start up the other PDMS modules, or to bring a module to the foreground if it is already running in the background. The other PDMS modules are: Registration module which manages

the administrative patient information and initializes the real-time data acquisition from the CareNet system, Vital Signs Monitoring System (VSMS) which provides a graphical interface for patient monitoring, Fluid Balance module which manages the measurement of substances taken in (ingesta) and put out (excreta) by the patient, Trend Analysis module which generate early warning alarms, and Nursing Workload Manager which aids the planning and documentation of the nurse's workload.

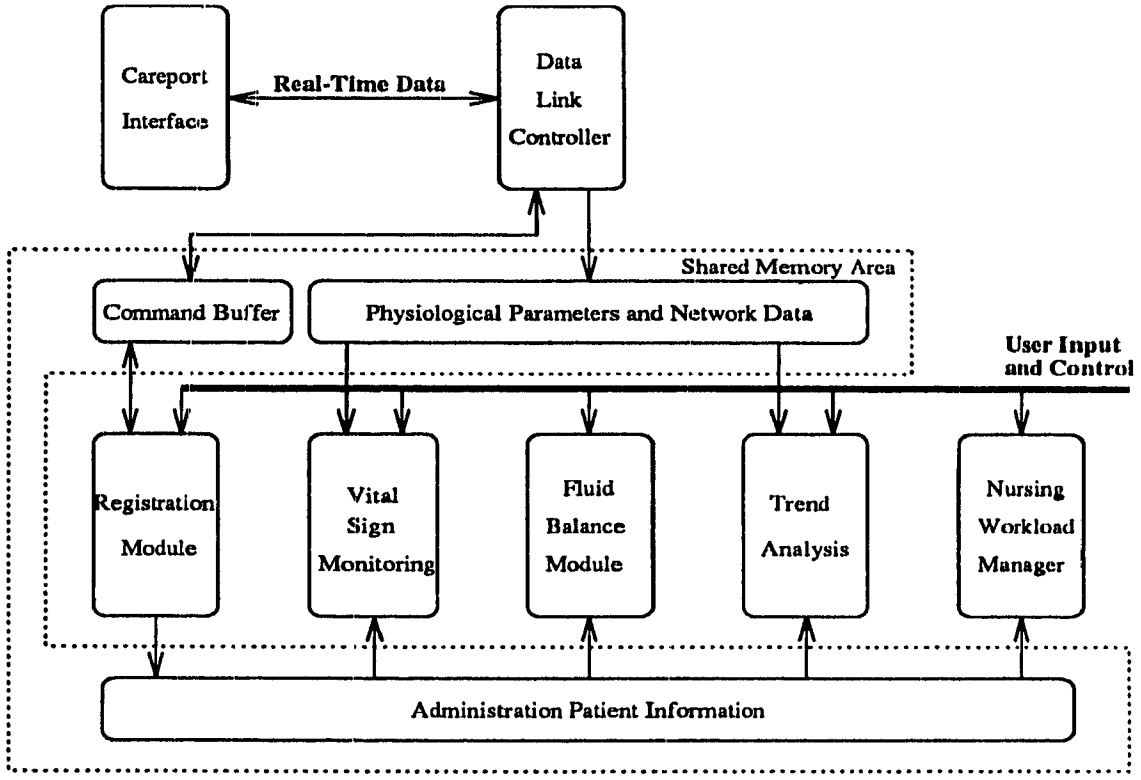

Figure 2.2 Software Configuration of PDMS

The current version of PDMS is implemented under OS/2 2.0 which provides the capability of multitasking, addressing a large memory space, managing inter-process communication, device independent program interface, and graphics user interface. Multitasking of application programs is fully supported to provide fast switching among processes, while preventing one application from corrupting the

memory and other applications' resources. One program is brought to the foreground at a time, while all the other programs continue to run at the background. The large memory space and virtual memory of OS/2 make it more suitable for running large applications than DOS. OS/2 provides a large set of functions for interprocess communication, such as shared memory, semaphores, pipes, queues, and signals, allowing processes to access shared resources without mutual interference, and providing channels to synchronize activities among cooperating processes.

2.1. Data Link Controller

The Data Link Controller (DLC) is responsible for interfacing the PDMS processes with the CarePort unit. As far as the host computer is concerned, only the CarePort unit exists; it has no knowledge of the functioning or even existence of the System Configuration Controller (SCC). DLC manages collected data for other modules' access by placing the data into circular queues in the shared memory. It is also in charge of storing the physiological data in the PDMS database for future consultation and archive purposes.

There are four types of data available through the CarePort unit: parameter data, wave data, alarm data, and status data. Parameter and wave data is marked as valid when the source has detected no problem with the data. Invalid data possess some detectable problem and is unreliable. All this data is transmitted by the CarePort unit in eight different kinds of messages.

Parameter data are discrete, numeric, monitored vital signs data, such as pulse rate, respiratory rate, and arterial blood pressure, identified through a Medical Function Code (MFC). There are two messages related to parameter data: Parameter Value (PV) message which contains the binary value for the parameter from an active selected data source coming from a bedside monitor, and Parameter Support

Data (PSD) message which contains the ASCII text displayed on the monitor for an active source. Parameter data, which can be single valued or triple valued, is updated every 1024 ms. Triple valued parameters are associated with MFC's greater than or equal to 128.

Wave Data are discrete, sampled data of continuous signals, such as ECG, which are also identified by the MFC. As for parameter data, values and support data are transmitted in two separate messages. The Wave Value (WV) messages contain digitized wave data samples as a function of time, and the Wave Support Data (WSD) messages contain wave calibration and scale information, as well as the ASCII wave label. Wave data can be updated every 32 ms.

Alarm Data are messages describing the alarm or inoperative states of a monitor. It is identified by the couplet of MFC and Qualifier Code. The Alert Status (AS) message contains the current status of alerts for a bedside monitor. The Alarm (AT) and Inop Text (IT) messages contain the text being displayed on the monitor. Because alerts usually remain in action for more than a second, a change count is associated with each alarm. It indicates by remaining identical or changing its value if the alarm is being repeated or occurred for the first time. There may be multiple simultaneous alarms or inops for a bedside monitor. Alert data is updated every 1024 ms.

Status Data corresponds to the Instrument Status (IS) message broadcasted on CareNet by each bedside monitor every 1024 ms. The status data contains the internal states of the instruments as well as a list of the waves and parameters which it is transmitting onto CareNet.

Logical sources are created to make virtual connection to different data sources so that the host-network interface can be easier to program. Since CareNet supports different data acquisition methods, data origins, medical function codes, and support data for each active bed, the definition of all these options for each data source is

time consuming. DLC defines and manipulates the desired options for a group of data sources by referring to a single logical source number (LSN). Logical sources can be in one of three states: not connected, connected but suspended, and connected and active. Connecting a logical source is equivalent to defining it with its options to the CarePort. Activation signifies that CarePort will send the available data according to the logical source definition. Data will not be transmitted if a logical source is suspended. More details are presented in section 3.2.

Semaphore handshaking is implemented for circular queues and network data access to avoid read-write conflicts. Commands from the Registration module are placed in the command buffer in shared memory to be passed to DLC. Access control and even signalling are also done by using semaphores. DLC uses multiple threads to receive real-time data, periodically store data in memory and in the database, receive command messages, modify logical source definitions, and average parameter values.

2.2. Registration Module

This module is used to enter the demographic and administrative data of patients. It manages the admission, activation, suspension, and discharge of patients in the ICU. It also controls the connection, activation, and suspension of a bed from the network. The Registration module can be activated from the PDMS Manager, and the main menu will appear in the foreground as shown in Figure 2.3.

The *Admit* option is used to register a patient into the PDMS system. The admission menu shown in Figure 2.4 consists of various text-entry fields for the administrative data of a patient to be entered. This function communicates with the DLC via the command buffer. If a patient is admitted, DLC will modify the logical source definition, and start collecting physiological data from the CarePort.

Patient's data is saved both to the PDMS database, and in the shared memory such that other modules in PDMS can access the patient information. Not all the data is required at the time of admission so that emergencies can be dealt with immediately. The only necessary information is the hospital ID, and the bed number which can be assigned automatically. Missing data can be supplied later on, and wrong data modified using the *Edit* option. Before the data is saved, an error checking routine is used to minimize the possibility of incorrect data entry due to fatigue or stress.

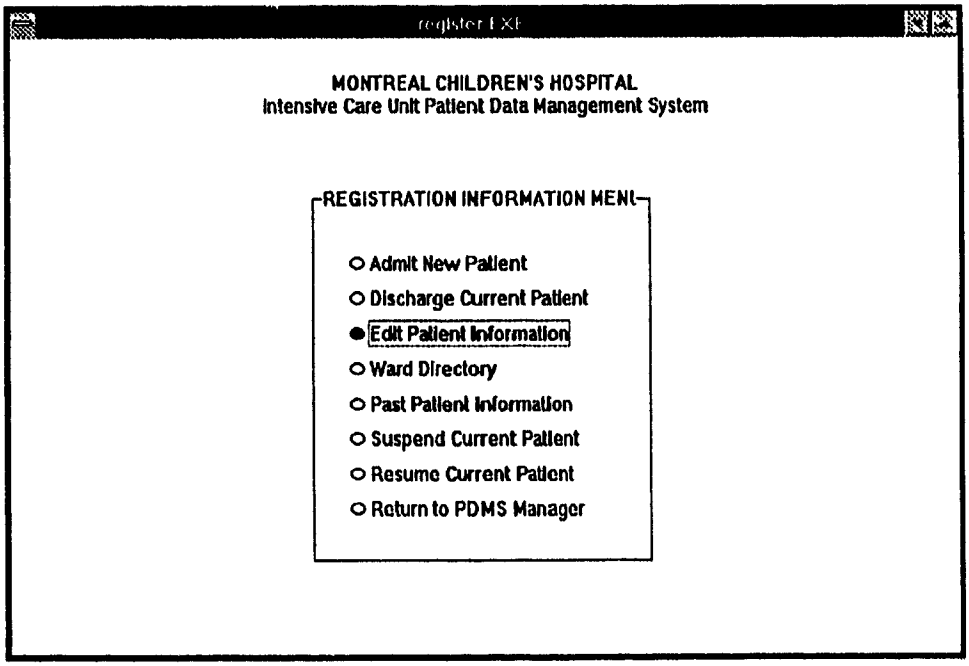

Figure 2.3 Registration Module

The *Edit* option is useful for editing patient information. This function uses the same dialog box as the *Admit* option so that the user can recognize the positions of different fields faster than using a different layout. In fact, all the options that enter and view patient information are also done using the same dialog box for the purpose of fast recognition. If the bed number has been changed, DLC will

modify the logical source to reflect the new situation, otherwise, the network is left untouched. When editing patient information such as hospital ID and admission date the original information is still kept for legal purposes, and a cross reference is made to the new information. In order present a consistent display, the original data is marked with a special code.

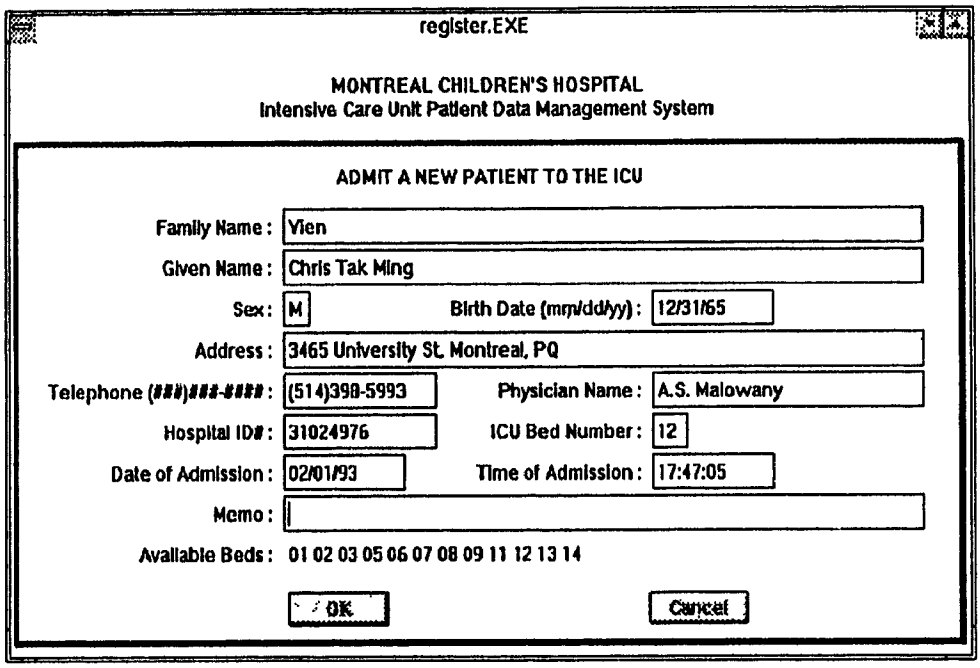

Figure 2.4 Dialog Box for entering patient information

The *Ward Directory* option allows fast access to the patients' information, and provides a summary of bed occupation in the ICU. The dialog box shown in Figure 2.5 is used to select a patient at a time from the list of ICU beds with their status (unoccupied or in use), and the patients' name.

If a patient is discharged from the ICU, the *Discharge* option will bring up a dialog box to confirm the selection. The discharge date and time can either be entered by the user or provided by the system with the current date and time. Once

discharged, DLC will stop collecting network data for the patient, and the memory resident data will be flushed to the PDMS database. The ICU bed number is freed for a new admission.

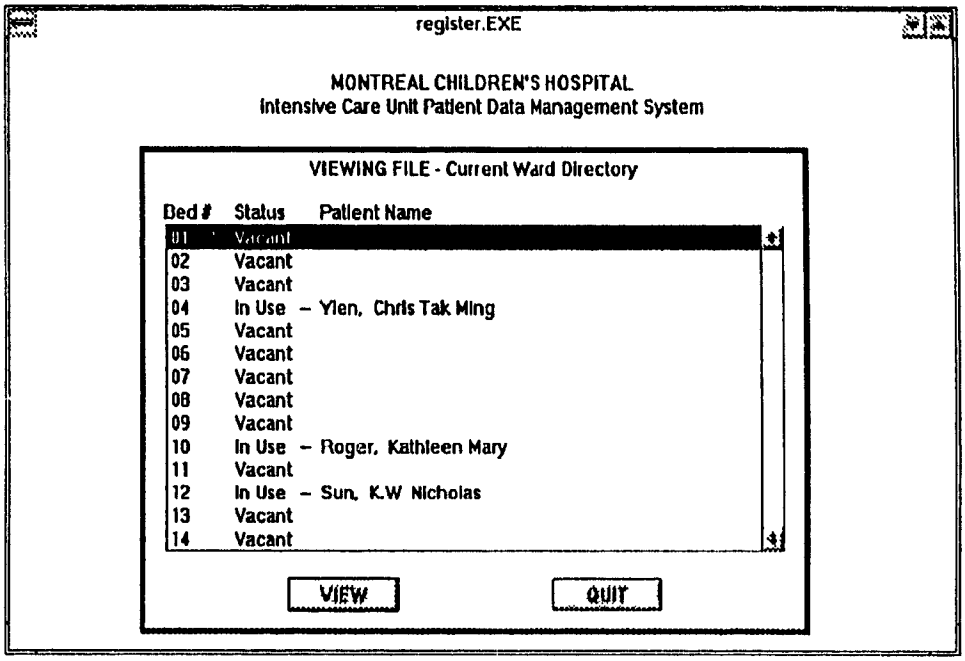

Figure 2.5 Dialog Box for selecting a patient from the ICU

The *Suspend* option temporarily suspends the DLC from network data acquisition for a specific patient. This is mainly used when the patient is transferred to another department for examinations (e.g. laboratory tests, XRays). Suspending the data acquisition in such situation will prevent invalid data from being stored to the database, and avoid the interference to the patient's trends by such data. The *Resume* option is used to resume data acquisition when a patient is ready for physiological monitoring after being suspended from the system.

If required, the *Past Patient Information* option is used to query the patient's information from the database by providing the hospital ID as a key to the query.

2.3. Vital Signs Monitoring

Vital Signs Monitoring is one of the most important functions of PDMS. Graphical representation of vital signs has been used to provide visual correlation of selected parameter trends, and much effort has been devoted to the design of graphical user interface. Graphs can be used to display large amount of data at the same time on the screen so that critical elements can be inspected in a minimal amount of time. Therefore, this module is a very useful tool for supporting decision making in the ICU. It was decided to maintained a resemblance between the screen display and the current manual charts for consistency and user acceptance.

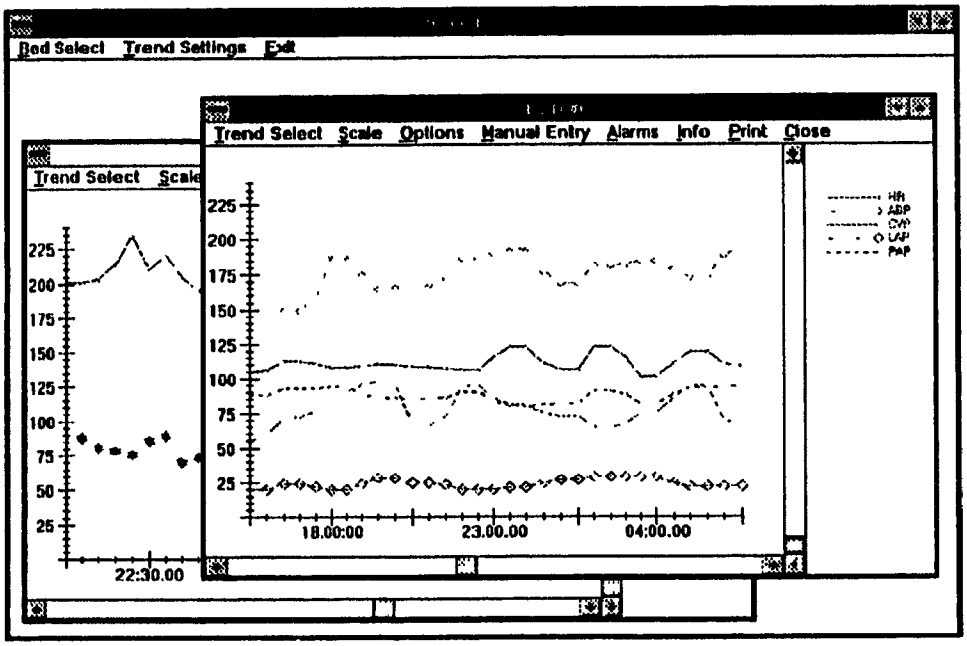

Figure 2.6 Vital Signs Monitoring

This module allows the monitoring of multiple beds and the display of multiple

vital signs at the same time as shown in Figure 2.6. Parameter data, wave data, as well as alarm data can be displayed in real-time for a remote location. The source of the data, which can be either real-time data collected by DLC or data queried from the database, is transparent to the user. Therefore, formal knowledge of database query language is not required for the user to retrieve data from the database.

Since the computer screen is a relatively small area to display the large amount of data, horizontal and vertical scroll bars can be used by the user to navigate through the graph in an intuitive way. Both the vertical scale and time range is adjustable by the user, and a maximum of fifty consecutive data points for each vital sign can be displayed. Careful design of visual coding (a combination of line color, marker color, line style, and marker type) is used to encode as much information as possible on the display, and to ensure visual clarity, as well as accurate recognition at the same time. The user is given extensive control over the settings and appearance of the display. It is also designed to accommodate multiple skill levels, and to accept different input devices such as mouse, function keys, arrow keys, and keyboard input. The next two chapters will discuss the details of the design and implementation of this module.

2.4. Fluid Balance Module

The Fluid Balance module enables the entry, calculation and correction of volumes of all the fluid intake and output of a given patient. The spreadsheet format of the form given in Figure 2.7 emulates as closely as possible the actual paper forms used in the ICU by the nurses. Fluid balance is defined as the measurement of all substances (usually liquid) going in (ingesta), and all substances coming out of the body (excreta), to determine the overall balance of fluids. This has to be monitored because of the effect the balance has on blood pressure, dehydration, pooling, drowning, or thrombosis.

Keyboard data entry for this module has presented a bottleneck, since there are fourteen beds in the ICU. An ideal solution would be having infusion pumps linked electronically to the PDMS host computer, much in the same manner as the bedside monitors are linked through CarePort. The ICU currently has a variety of infusion pumps. Each one of these infusion pumps has a different interface and uses different data formats. Therefore, it is impractical to attach the growing number of devices to the host computer by building special software and hardware interfaces. A speech interface was developed by Petroni [Petroni *et al*, 1989] to facilitate data entry at the bedside using a voice recognition system. If any field of the Fluid Balance data sheet is edited, the previous value is still stored for legal purposes as with the Registration module, and the field is marked with a special code to indicate that it has been edited.

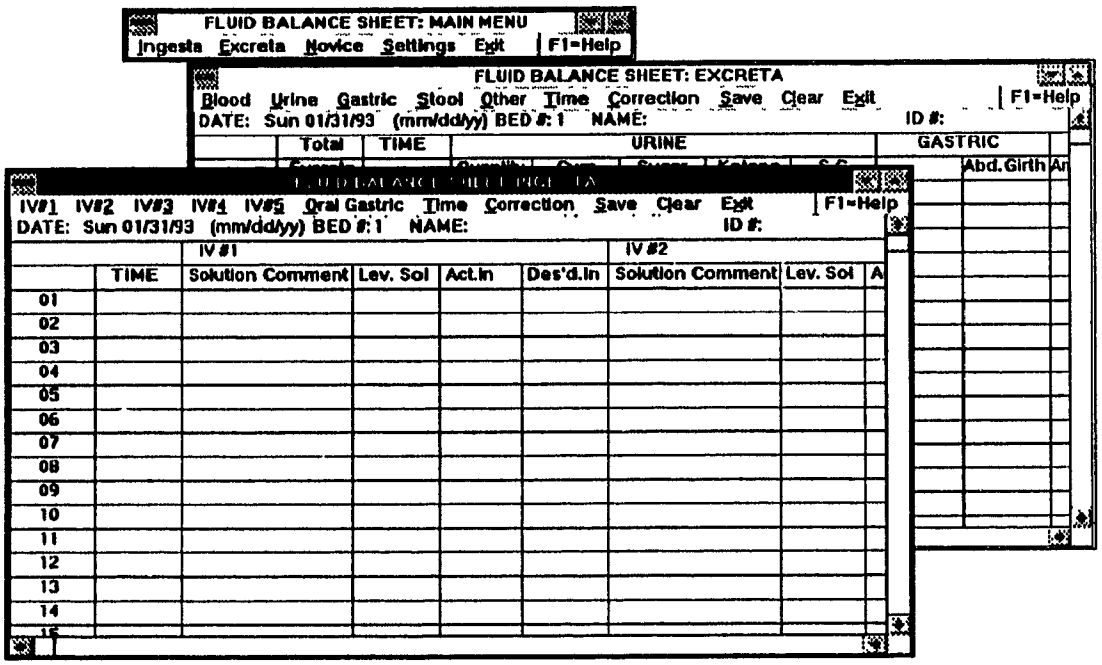

Figure 2.7 Fluid Balance Module

2.5. Expert System for Trend Analysis

Although threshold alarms are an efficient way of attracting the user's attention, only real-time analysis of the trends can provide a warning about slowly varying trends or short interval disturbances before the underlying condition results in a critical situation. A three level expert system is being developed to help in the detection and diagnosis of critical patient conditions.

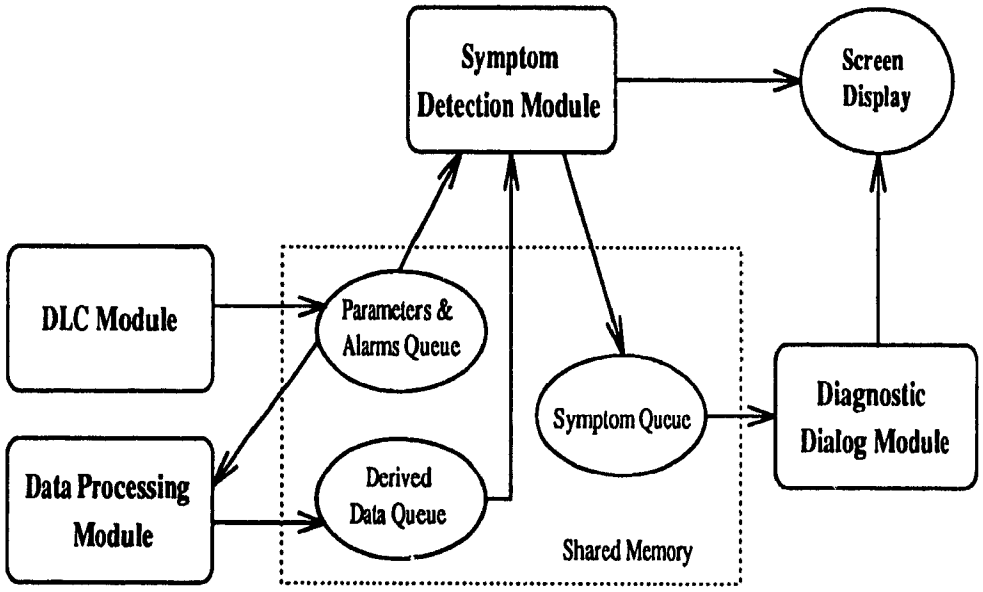

Figure 2.8 Block Diagram of Trend Analysis

The organization of the expert system is composed of three modules as shown in Figure 2.8., namely the Data Processing, Symptom Detection, and Diagnostic Dialogue modules. The first level of the expert system generates derived information from the incoming data and finds special trend patterns. The second module detects and displays symptomatic patient conditions by using a decision tree of rules on the

available data from the first module and from the DLC. The third module uses the symptoms generated by the second module and presents this information as an ordered list for the user to arrive at an advisory diagnosis.

2.6. Nursing Workload Manager

For each patient in the ICU, a nurse prepares a Nursing Care Plan Worksheet which lists the admittance diagnosis, medical particulars on the patient, and all the tasks required to care for the patient. Tasks are categorized under eight headings: respiration, nutrition and hydration, elimination, personal care, ambulation, communication, treatments, and diagnostic procedures.

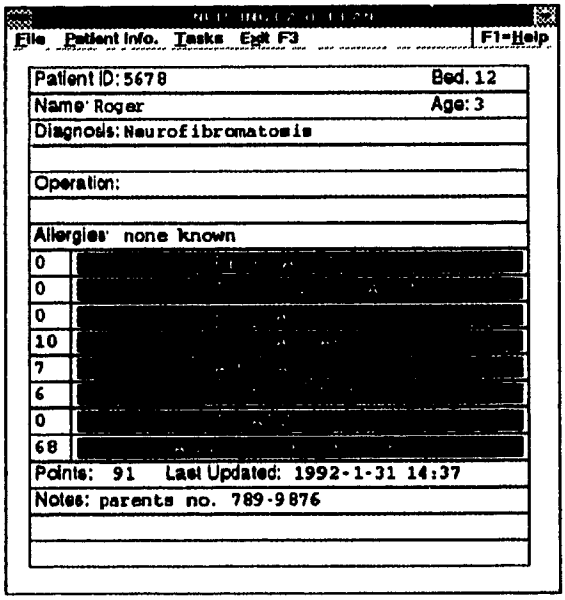

Figure 2.9 Nursing Workload Manager Main Menu

The nurse's activity must be documented and score values are assigned from

the Progressive Research in Nursing (PRN) system table which lists all the possible tasks with a value that is derived from the time required to complete the task and the difficulty of the task. In effect, they measure the level of nursing care required.

The Nursing Workload Manager module is used to list the standard care tasks required based on the admittance diagnostics, allot the PRN points and then, after verification and possibly modifications by the nurse, plan the shift according to the list of tasks to be done and output a schedule using an expert system scheduler. This will offer the administration a more efficient method of measuring the nurse's workload and that of the ICU.

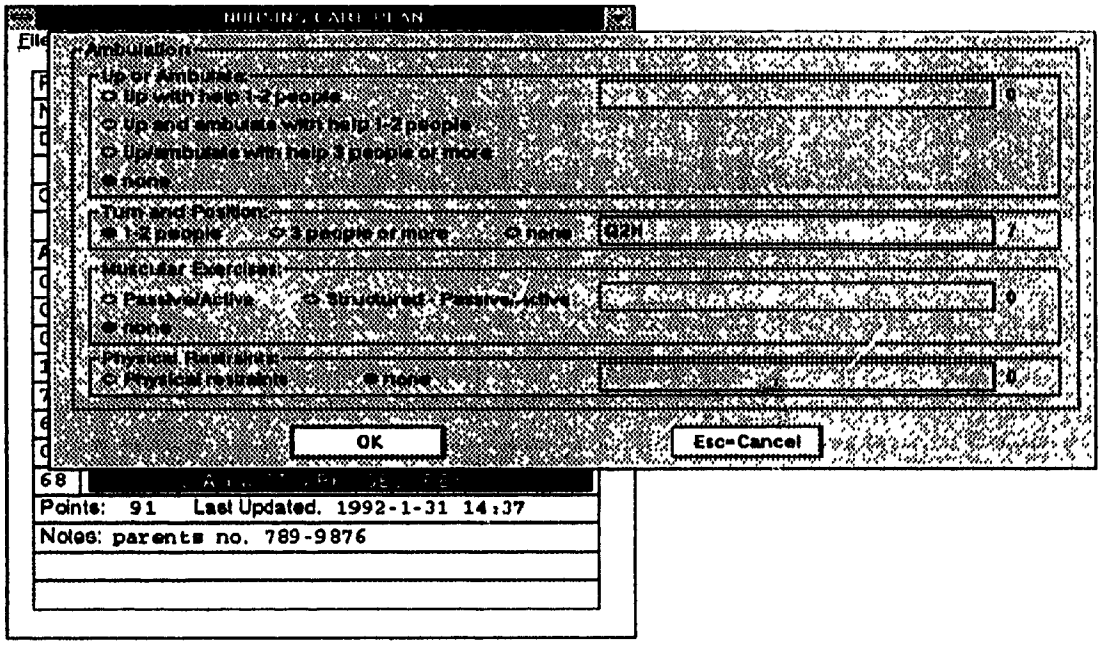

Figure 2.10 Nursing Workload Manager

Figure 2.9 shows the main menu of the Nursing Workload Manager. The task category headings are sensitized. The user creates a new care plan or opens an existing one by pulling down the *File* menu, and selecting the *New* or *Open* options.

Once a care plan has been created or opened, the user can enter or edit information on any text-field of the main menu. The medical diagnosis of the patient can be entered or selected from a list box which displays the most common diagnoses. At this point, the user can elect to load a standard care plan, or a diagnosis-specific care plan. Each category uses a dialog box to show all the tasks available in that category with frequency intervals and PRN points as shown in Figure 2.10. PRN points are printed in the column to the left of the heading. The care plan worksheet can be printed out along with a separate sheet listing the medications and intravenous solutions entered onto the care plan.

This chapter has outlined the current implementation of the PDMS system. In the next chapter we shall present the Vital Sign Monitoring System in greater detail.

3. Vital Signs Monitoring System

The Vital Signs Monitoring System (VSMS) described here is a subsystem of the PDMS developed for the Montreal Children's Hospital. VSMS is designed to provide real-time vital signs data acquisition, storage, display, and printing. It is also an interactive system for viewing data stored in the PDMS database.

3.1. Overview of VSMS

Various modules are involved in the design of VSMS as shown in Figure 3.1. The only visible interfaces to the user are the Registration module, and the VSMS display. Therefore, the user controls all the activities in VSMS using menus and dialog boxes of these modules. The Registration module is used to release commands such as: adding or deleting a bed number to the list of active beds, suspending or resuming data acquisition for a bed, as well as reseting and testing the CarePort unit. The commands from Registration module are actually executed by the DLC, and sent to the CarePort. The Registration module is also responsible for providing patient administrative information for the VSMS display.

Once a patient is registered, DLC is responsible for modifying logical source definitions, acquiring data from CarePort unit, storing data periodically in the PDMS database, and calculating data averages. DLC is composed mainly of five threads: *command_line*, *read_port*, *accum_params*, *save_params*, and *alarm_handler*, which are used for data acquisition and data management.

There are three sources of parameter data for VSMS: real-time data collected by DLC, data stored in the database, and manual data entered by nurses using dialog boxes provided by VSMS. Although automated data acquisition has advantages over

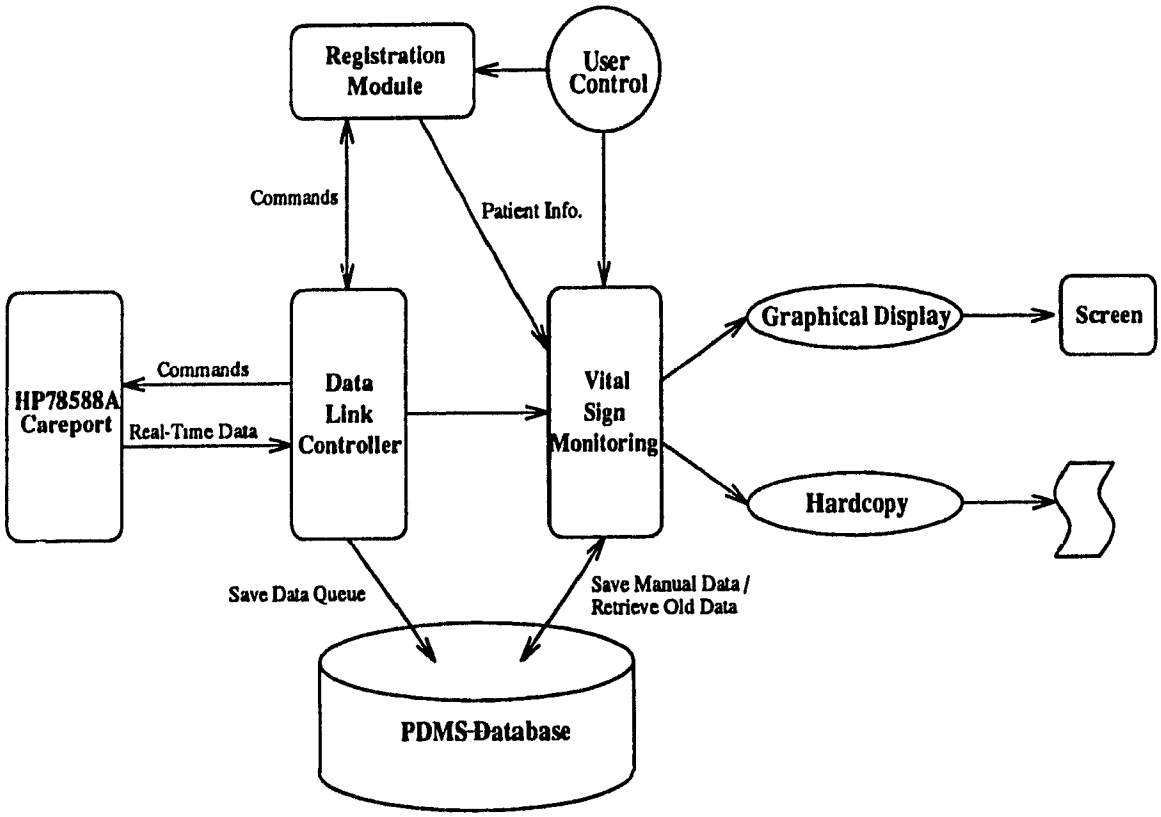

Figure 3.1 System Overview of Vital Signs Monitoring

manual data acquisition in areas such as accuracy and speed, manual data entry is supported by VSMS because it is necessary in case of equipment failure, or when a patient's condition is not appropriate for using certain monitoring equipment (either invasive, or noninvasive). The system can also handle a mixture of data from all three sources for screen display and hardcopy. The display uses two scroll bars to move horizontally, and vertically among the data points. When the specified time range of a graph is outside the circular queue data, it will try to query the data from database, and then display a continuous graph in the window.

Shown in Figure 3.2 is the data-flow diagram of the VSMS. All the commands to be carried out by the DLC are first placed in the command buffer of the shared memory. After a command is executed, the error code is returned to the shared

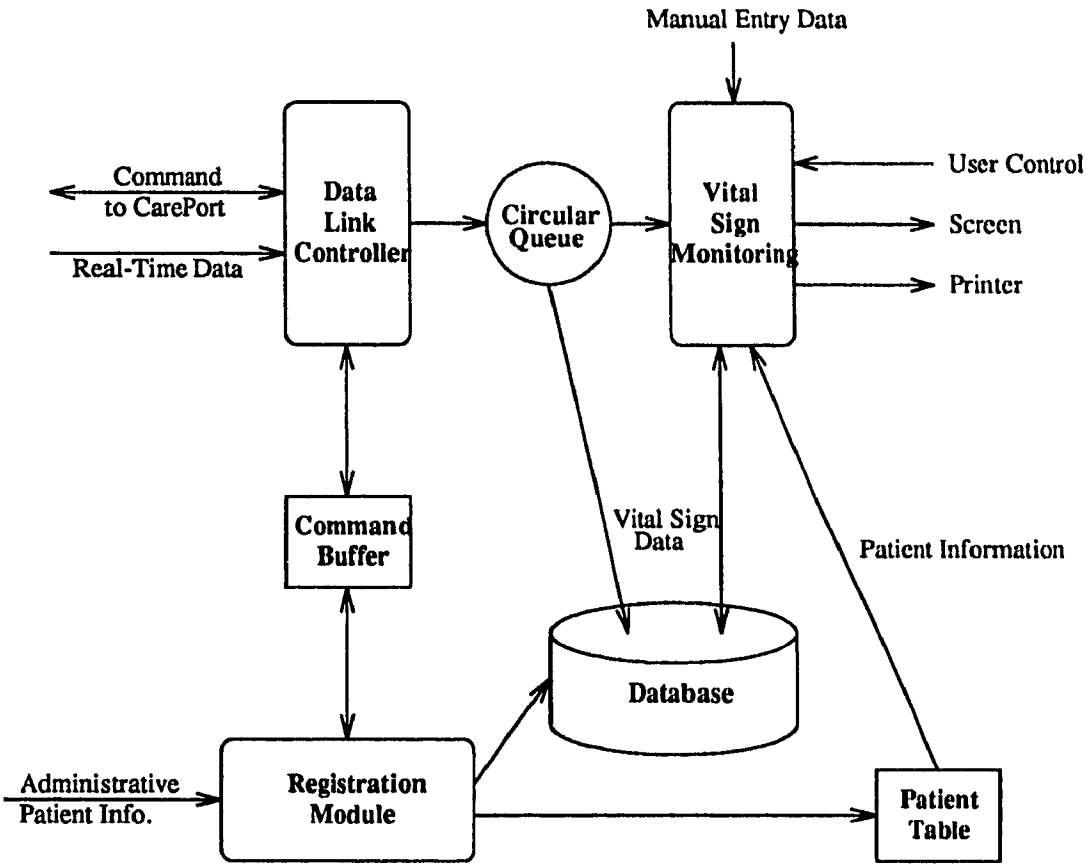

Figure 3.2 Data-flow Diagram of VSMS

memory. Administrative data of a patient currently in the ICU and bed status are stored in a memory resident patient table. The patient table is created in the shared memory segment to allow all PDMS modules access to this data. When the patient is discharged from the ward, the data is flushed from memory and stored in the relational database on disk.

All the real-time data collected from the CarePort by DLC is stored in three circular data queues, which are created in the shared memory segment to allow data access from other modules. Data from the circular queues is stored in the PDMS database on a periodic basis.

Manual data entry is performed by using a dialog box in the VSMS. VSMS

receives control settings from the user to manipulate the data on the graphical output, and to produce hardcopies.

3.2. Real-Time Data Acquisition

Three modes of data acquisition are supported by the CarePort: polling mode, scheduled mode, and continuous mode. In polling mode, CarePort sends the most recent data only if it is requested by the host computer. The highest frequency at which polling mode can be operated is one sample per 1024 ms. With the scheduled automatic mode, the host specifies how often and for how long CarePort should automatically transmit the most recent data. Continuous automatic mode allows the user to obtain all data which is acquired for the logical source, from the moment of activation to suspension. With this data acquisition method, CarePort provides buffering of the data.

Table 3.1 Communication Port Initialization

Serial Communication Port Settings	
Option	**Setting**
Baud Rate	9600
Data Bits	8
Parity	none
Stop Bit	1
Write Timeout	None
Read Timeout	0.01 second
Handshaking	DTR, CTS, DSR, DCD disabled
Flow Control	Automatic XON/XOFF disabled
Timeout Mode	'Write Infinite' timeout 'Wait for Something' Read timeout

DLC opens the serial communication port and defines its options according to the settings shown in Table 3.1. Eight logical sources are defined but not connected. Although the actual definitions of these logical sources show that all beds are disabled, the bits corresponding to the beds to be activated will be set in the logical source definition before the source is connected and activated. Table 3.2 gives a

summary of the options selected in the logical source definitions. Message types were discussed earlier in section 2.1. Because of the baud rate setting and of the possible use by DLC of parameters from all fourteen beds, CarePort can correctly send wave values for only one bed waveform at a time. LSN 6 and 7 will then be activated only with one bed active.

Table 3.2 Logical Source Definitions

LSN	*Acquisition Mode*	*Bed Numbers*	*Messages*	*Activation*
0	Infinite Scheduled	1-7	PV	Immediate
1	Infinite Scheduled	8-14	PV	Immediate
2	Polled	1-14	AT & IT	Immediate
3	Polled	1-14	IS	Upon Request
4	Polled	1-14	PSD	Upon Request
5	Polled	1-14	AS	Upon Request
6	Polled	Any One	WSD	Upon Request
7	Infinite Scheduled	Any One	WV	Immediate

There are two categories of messages: request, and response. A request is always sent by the host computer to the CarePort which answers back with a response message. The CarePort does not do anything unless specifically requested by the host computer. The response to a request might not be sent immediately after the request, and the response is not restricted to one message. For the *schedule mode* and *continuous mode*, the CarePort will respond with an acknowledge message, and the requested data will be sent periodically, upon an initial request by the host computer. Additional control characters are added to the logical message, which consists of the command and its options as defined in the logical source, to ensure that the communication transmitted to CarePort is successful. The actual message being sent to CarePort is called physical message. Details of the commands supported by CarePort can be found in the *CareNet: Programmers Reference Guide* [Hewlett, 1988].

3.2.1. Construction of Physical Messages

Physical messages are constructed according to the ANSI x3.28-1976 protocol. Each

physical message is composed of a logical message placed between two series of control characters; beginning with the ASCII DLE and STX characters and ending with the DLE and ETX characters. A complete list of the control characters is given in Table 3.3. An additional byte of Block Check Character (BCC) terminates the physical message. BCC is used for error detection by calculating the exclusive-or of all the characters in the physical message, starting with the first character following the DLE-STX control characters. To distinguish the DLE-STX and DLE-ETX control character pairs from chance occurrence inside the logical message, a second DLE character is added after every DLE character found inside the logical message, before the BCC is computed.

Table 3.3 ASCII Equivalents of DLC Symbols

Symbol	Hex Value	Meaning
DLE	0X10	Data Link Escape
ACK	0X06	Acknowledge
WACK	0X3B	Wait before transmit Acknowledge (preceded by a DLE)
NAK	0X15	Negative Acknowledge
ENQ	0X05	Enquiry
STX	0X02	Start of Text
ETB	0X17	End of Transmission Block
ETX	0X03	End of Text
EOT	0X04	End of Transmission
RINT	0X3C	Reverse Interrupt (preceded by a DLE)

3.2.2. Transmitting Commands to CarePort

Both the host computer and the CarePort can be in one of three states: control, text-in, and text-out modes. In control mode, the transmission link is disabled and each station, the host computer and CarePort, can bid for master status in order to establish the direction of the next transmission. This is done by ENQ-ACK handshaking. If the station sending ENQ receives an ACK from the other station, it becomes the master of the transmission and the other becomes the slave. As a master, the station is in text-out mode which has the control of the communication link and sends physical messages. As a slave, the other station is in text-in mode,

and receives the transmitted messages. If a physical message is successfully sent, as indicated by the reception of the DLE-ETX pair and the correct BCC character, the slave will send another ACK to the master to confirm good reception. If this is the last message being sent, the master will return an EOT to the slave, and both stations will be back in control mode again. DLC converts the logical message into a physical message and then sends it on the serial link in the manner shown in Figure 3.3. In the *Get Master* and *Get Answer* states, DLC enters the loops for obtaining serial link mastership and message reception acknowledgement from CarePort respectively.

3.2.3. Receiving Messages from CarePort

The thread in DLC responsible for receiving messages from CarePort passes from state to state within a loop as shown in Figure 3.4. The loop follows the states found in the handshaking and reception protocol of a physical message from CarePort. The ASCII codes associated with the signals are listed in Table 3.3. The length of the message is calculated by incrementation upon every reception of a character. BCC is calculated by performing successive exclusive-or operations between the received character and a running value. A semaphore is defined to ensure unique access to the serial port during the reception of a message and is relinquished as soon as possible.

3.3. Data Management

Patient data collected from the CarePort unit is stored in memory and in the PDMS database. A patient's parameter values are sent approximately every two seconds by CarePort. Keeping all these values would be too space consuming as well as slowing down vital signs display and trend analysis. Furthermore, as more and more bedside monitors are connected to the CarePort, degradation of CarePort

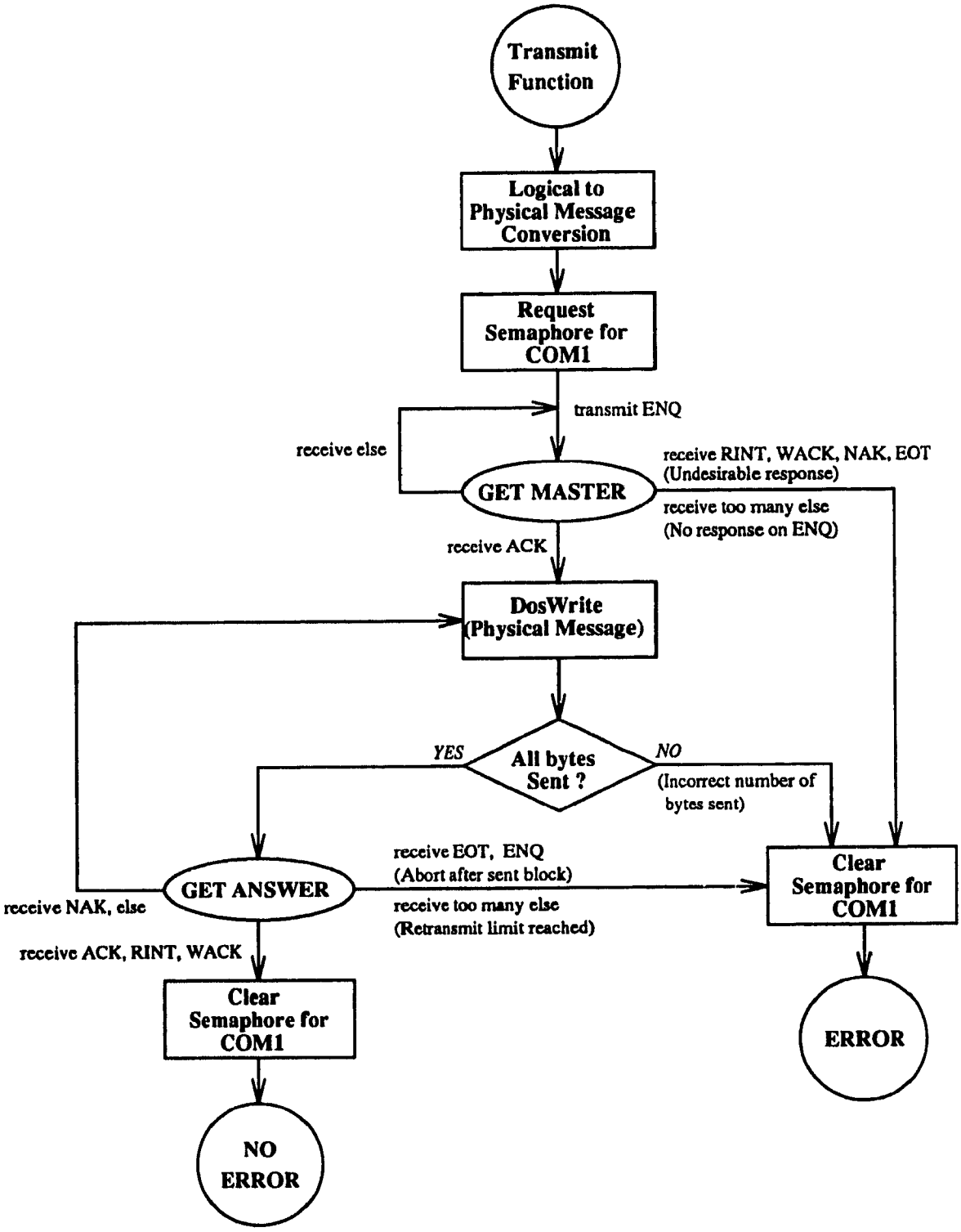

Figure 3.3 State Diagram for Message Transmission

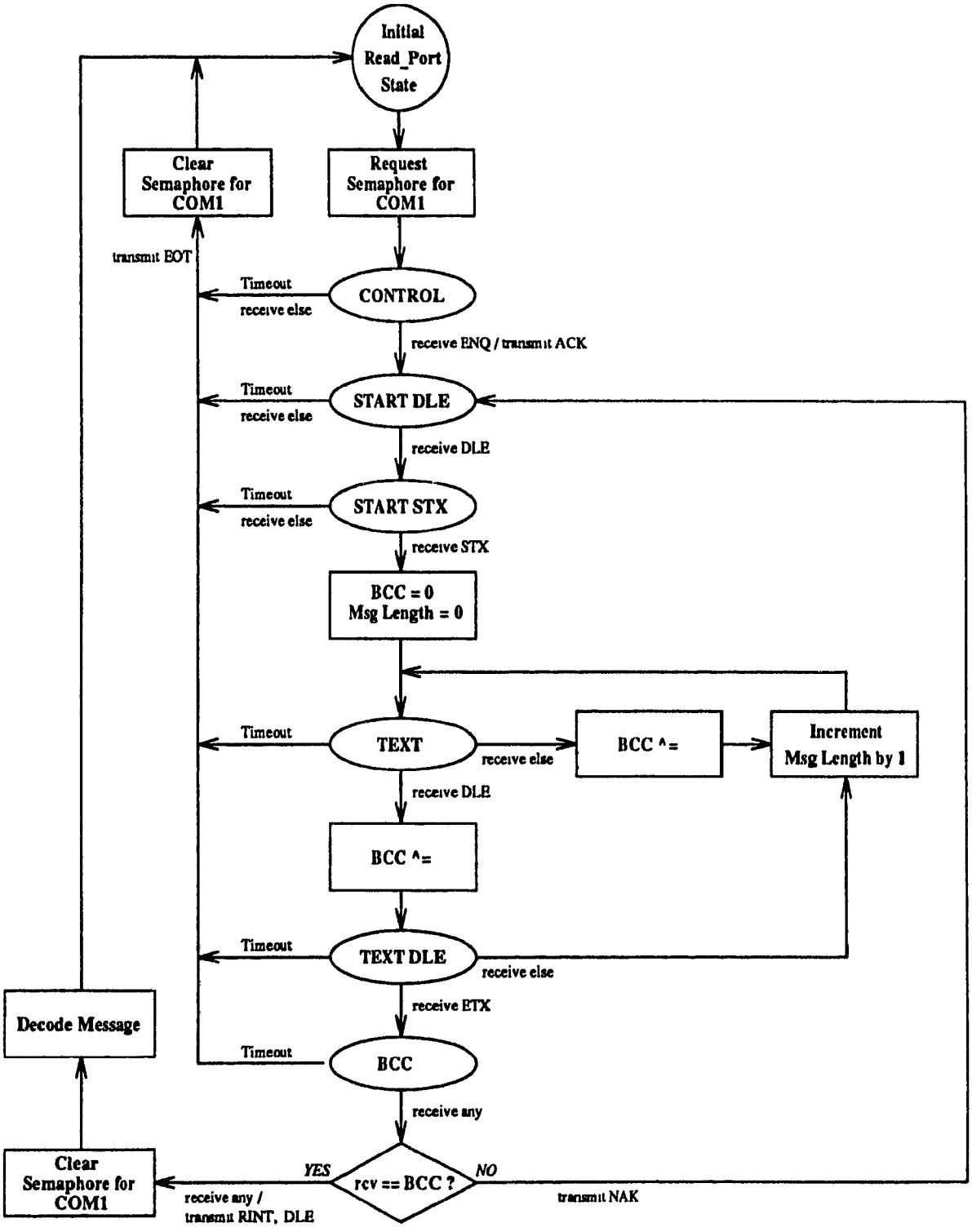

Figure 3.4 State Diagram for Message Reception

service can be expected, and the periodicity of parametric entries would become less reliable.

3.3.1. Averaging Parameters

To avoid all the shortcomings described above, parametric entries are averaged for each minute and for each half-hour. To generate these averages, second and minute accumulators with count numbers are stored for each bed number-MFC (Medical Function Code) combination. Every minute and every half hour, these averages are calculated for each MFC of all the active beds. The averaged parameters are kept in two circular queues; one for the minute averages and the other for the half hour averages. The minute queue can hold minute data up to a maximum of two hours, and the half-hour queue can store a maximum of two days of half-hour data. Since these averages have to be accessed by other PDMS processes, such as the expert system for trend analysis, the circular queues containing them are located in shared memory.

3.3.2. Data Structures

Real-time data acquired by DLC for patients currently in the ICU is stored temporarily in RAM, and is saved to the PDMS database on a periodic basis. Figure 3.5 shows two data structures, the Patient Table and the Data Circular Queues, which are used for storing administrative data and parameter data respectively. The Patient Table is an array of Bed Status Records (BSR), indexed by bed number. Each BSR corresponds to a patient in the ICU, and the fields of BSR contain information such as patient name, hospital ID, attending doctor, bed status, and parameter data queue pointers. When a new patient is admitted using the Registration module, DLC first initiates network connection with the CarePort interface. If the network connection is successful, patient information in the BSR fields will be entered by the Registration module, the bed status indicator will be set to 'in-use', and the data

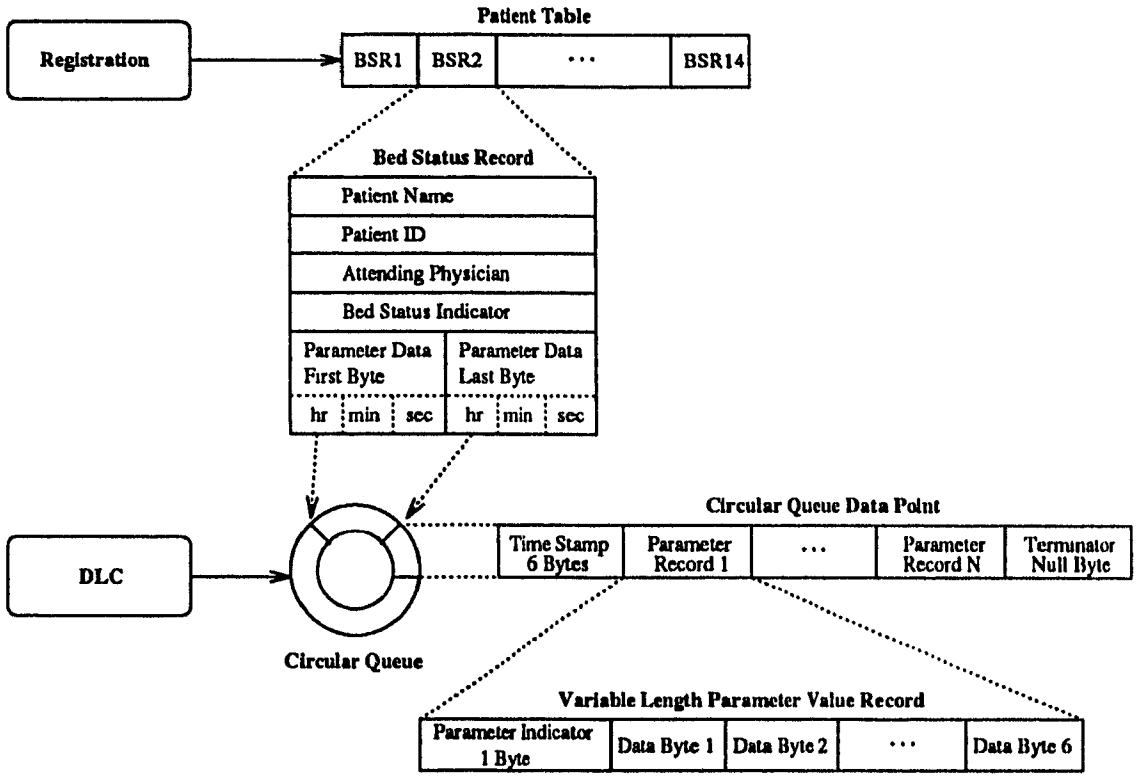

Figure 3.5 Patient Data Queue Structure

queues will be initialized for storing patient parameter data.

There are three circular queues for each patient parameter data: one second data queue, one minute data queue, and one half-hour data queue. The parameter data pointer fields in the Patient Table are used to access the first and the last data points in the circular queues. A data point in each circular queue is an array of bytes occupying a maximum of 39 bytes. Each data point in a circular queue stores multiple parameter values for one patient corresponding to a certain time interval. The first six bytes of a circular queue data point records the time stamp for the data, while the remaining bytes contain a number of parameter value records. The length of the parameter value record depends on the type of data (single valued or triple valued) indicated by the first byte of the parameter value record. If the parameter data is single valued, then two bytes are used to stored the data. If it

is triple valued, then six bytes are required. The last parameter value record is followed by a zero byte indicating the end of data point. All the structures are placed in the shared memory so that other processes can have direct access to this data.

3.3.3. Saving Parameter Averages

The *save_params* thread is locked in an infinite loop. This thread periodically saves the latest parameter values in the circular queues for all the beds indicated as active in the patient table. The thread goes circularly backward, from the next position index, until it finds a valid string. It then calculates the length of the string from the MFC code. Each MFC value represents an additional length of three or seven bytes in the string depending on whether the parameter is single valued or triple valued. DLC flushes the values in the minute circular data queue to the PDMS database every five minutes, and a new half-hour data point is saved every half-hour. To perform these updates requires the patient hospital ID number. Since this is stored in the memory resident patient table, no database access is required to retrieve this information.

3.3.4. Manual Data Entry

If a trend graph is selected for manual data entry, the user can use a dialog box to enter the value(s) of a vital sign data. The data is stored in a two-dimensional array of linked lists, indexed by the bed number and the MFC code. Linked lists are used to ensure efficient usage of memory. The linked list structure has five fields: one for time stamp, three for triple-value data, and a pointer pointing to the next element. New data is added to the end of the linked list after the dialog box is dismissed.

Since it is not likely that the nurses will enter manual data every minute, the manual data will be displayed with the half-hour data only. If single value data is saved in the linked list structure, the second and the third data fields will be marked

by *no data* flags. It is assumed that when manual data entry is selected for a vital sign, the data is not available from the DLC. However, if the DLC is still collecting data for that particular vital sign, the data collected by DLC will simply be ignored by the graphical display routine.

3.4. Graphical Representation

Graphs can display large amount of data so that critical elements can be inspected in a minimal amount of time [Laborde *et al*, 1989]. The graphical display of vital signs is an important aspect of VSMS because it is the only direct access to integrated patient data in a comprehensive format, and it is an useful tool for medical decision making.

3.4.1. Transformations in Graphs

The VSMS graphical display allows the user to adjust the ranges of time and magnitude axes displayed on the screen by using a dialog box. The user can also use two scroll bars to move along the x-axis and y-axis. Therefore, translation and scaling transformations are essential to the graphical display of data; if the user scrolls along one of the axes, each point $P(x,y)$ is is moved by either d_x units parallel to the x-axis, or d_y units parallel to the y-axis, to the new point $P'(x',y')$ according to the following equation.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \end{pmatrix} \tag{1}$$

Similarly, if the user reduces the range of either or both axes, and the size of the viewport is fixed, then the data points should be stretched, or shrunk, to fill the viewport by the multiplication

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \tag{2}$$

where the original point $P(x,y)$ is scaled to the new point $P'(x',y')$. The values s_x

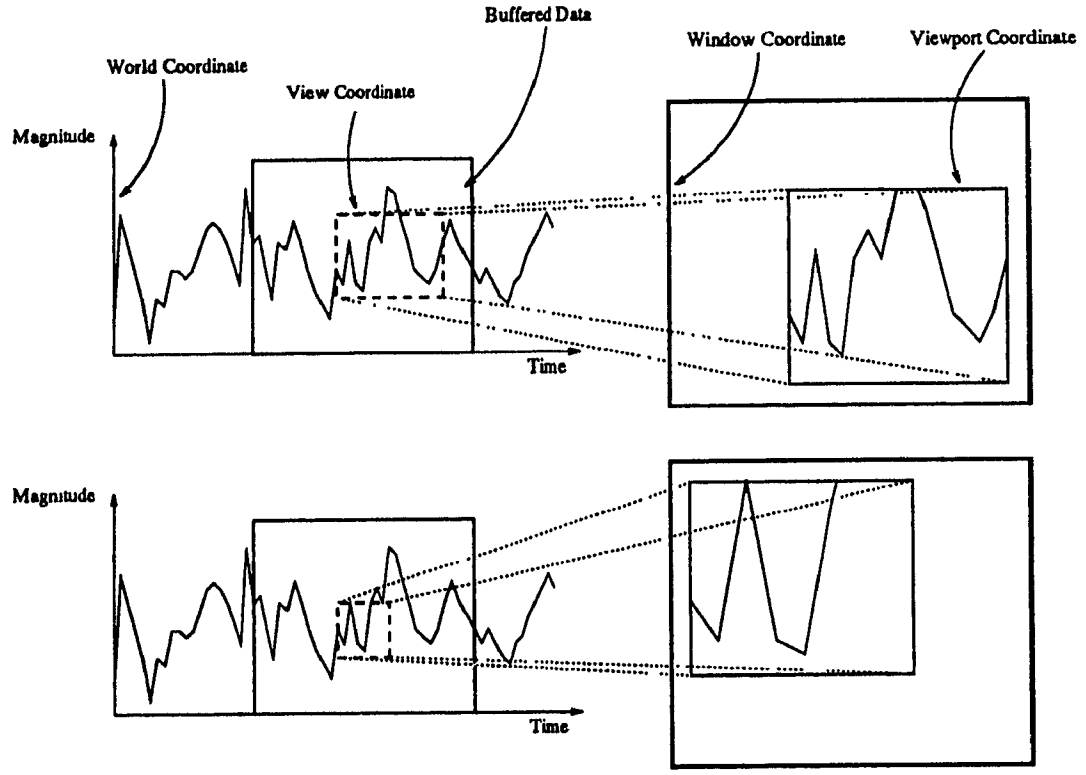

Figure 3.6 Coordinate Systems and Transformations

and s_y are the scaling factors to stretch, or shrink, the point $P(x, y)$ along the x-axis and y-axis respectively.

Displaying graphical data not only involves transformation in the same coordinate system, but also changing the coordinate system as shown in Figure 3.6. The original vital signs data is in the world coordinate where one unit on the time axis is taken as the time between two data points, i.e. two seconds for second data, one minute for minute data, and thirty minutes for half-hour data. When the user specifies the display ranges, a view coordinate is defined in the world coordinate where displayed data is bounded above and below by the maximum and minimum magnitudes, as well as bounded left and right by the start and end times. The view coordinate is then projected to the viewport coordinate by using equation (2) to perform the transformation. A viewport is an area inside a window used for

displaying the graph. The scaling factors in equation (2) are given by:

$$s_x = W \cdot \left(\frac{T_{unit}}{T_{end} - T_{start}} \right), \tag{3}$$

and

$$s_y = H \cdot \left(\frac{1}{M_{max} - M_{min}} \right) \tag{4}$$

where W and H are the width and height of the viewport respectively, T_{end} and T_{start} are the start and end time of the displayed data respectively, T_{unit} is the time between two consecutive data, and M_{max} and M_{min} are the maximum and minimum magnitudes respectively. Since the viewport coordinate is defined inside the window coordinate, a translation using equation (1) is required, where the d_x and d_y are the horizontal and vertical displacements of the viewport coordinate from the window coordinate. Another coordinate not shown in Figure 3.6 is the screen coordinate upon which the window coordinate rests. Fortunately, Presentation Manager performs the translation from window coordinate to screen coordinate automatically.

Since the data in the world coordinate are unbounded in the time domain, scrolling is impossible for an infinite number of data. Therefore, two hundred data points for each vital sign are buffered using an array, with a maximum of fifty data points for each vital sign being displayed at the same time. Another result of buffering the data is that the redraw time is reduced by minimizing the data retrieval time. Scrolling along the vertical and horizontal axes is thus equivalent to moving the view coordinate inside the buffered data. The viewport coordinates of the displayed data are stored in an array of structures with the array index as the time unit, and structure members as the viewport coordinates. Therefore, translation along the time axis is achieved by incrementing, or decrementing, the array index, and translation along the vertical index is done by adding, or subtracting, the product of s_y and the number of vertical units, Y_{unit}, from the viewport Y coordinate of the data.

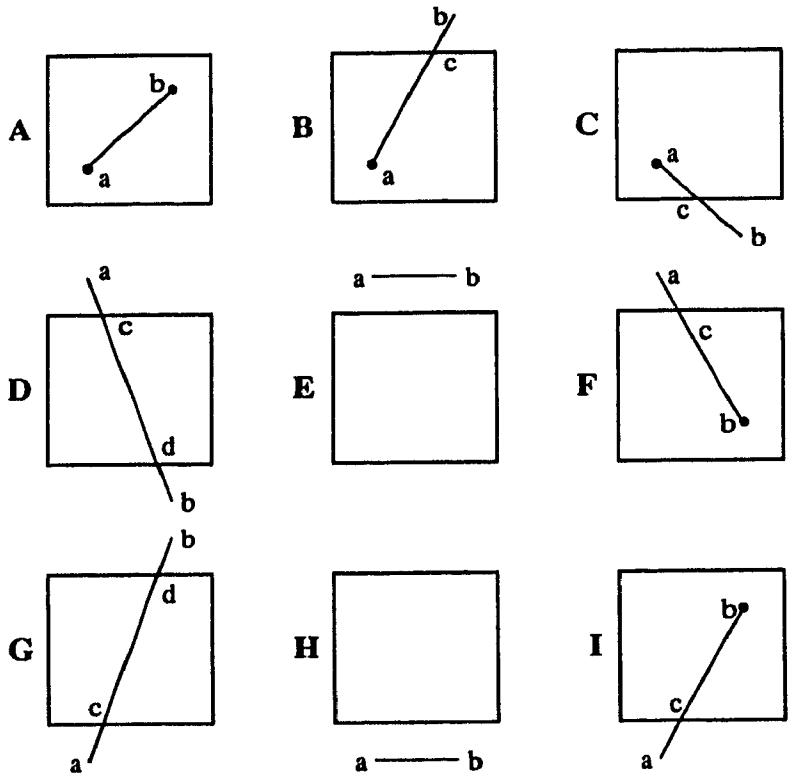

Figure 3.7 Cases of Line Clipping

3.4.2. Line Clipping Algorithm

This section discusses the clipping of lines against rectangles. The result of clipping a line against a rectangular region is always a single line segment. The use of a line clipping algorithm should be sufficient for the display of vital signs in a viewport, since the graphical display of the VSMS consists of joining the sequence of lines whose end points are the discrete data points collected from the bedside monitors. Clipping the endpoints of a line is a sub-problem of line clipping against a rectangle. If the boundaries of the clip rectangle are x_{left}, x_{right}, y_{top}, and y_{bottom}, then the following conditions must be satisfied in order for an endpoint $P(x, y)$ to lie inside the clip region:

$$x_{left} \leq x \leq x_{right}, \tag{5}$$

```
void Line_Clipping (int x_a, int y_a, int x_b, int y_b, int y_top, int y_bottom)
    {
    int x_c, y_c, x_d, y_d ;

    if (y_top ≤ y_a ≤ y_bottom)
        if (y_top ≤ y_b ≤ y_bottom)
            DrawLine (x_a, y_a, x_b, y_b)
        else if ((y_top < y_b) or (y_bottom > y_b))
            {
            Calculate coordinates of intersection, c(x_c, y_c)
            DrawLine (x_a, y_a, x_c, y_c)
            }
    else if (y_top < y_a)
        if (y_bottom > y_b)
            {
            Calculate coordinates of intersections, c(x_c, y_c) and d(x_d, y_d)
            DrawLine (x_c, y_c, x_d, y_d)
            }
        else if (y_top < y_b)
            Do nothing
        else if (y_top ≤ y_b ≤ y_bottom)
            {
            Calculate coordinates of intersection, c(x_c, y_c)
            DrawLine (x_c, y_c, x_b, y_b)
            }
    else if (y_bottom > y_a)
        if (y_top < y_b)
            {
            Calculate coordinates of intersections, c(x_c, y_c) and d(x_d, y_d)
            DrawLine (x_c, y_c, x_d, y_d)
            }
        else if (y_bottom > y_b)
            Do nothing
        else if (y_top ≤ y_b ≤ y_bottom)
            {
            Calculate coordinates of intersection, c(x_c, y_c)
            DrawLine (x_c, y_c, x_b, y_b)
            }
    }
```

Figure 3.8 Line Clipping Algorithm

and

$$y_{bottom} \leq y \leq y_{top}. \tag{6}$$

Since the horizontal axis represents time, and only data points within the time limits

of the viewport are retrieved from the circular queues, therefore, condition (5) for the endpoints to lie inside the x coordinate boundaries are always satisfied. As shown in Figure 3.7, there are nine cases to be considered when clipping a line with endpoints a and b against a rectangle. If the entire line lies inside the clipping region (case A), both end points of the line lie inside that clipping region also. If point a lies inside and point b lies outside (cases B and C), then the line intersects the clip rectangle and the intersection point c is computed. Similarly, if point a lies outside and point b lies inside (cases F and I), again the line intersects the clip rectangle and the intersection point c is computed. If both points a and b are outside the clip region, the line may intersect the clip rectangle (cases D and G), or may not (cases E and H). If the line intersects the clip rectangle, two intersection points c and d must be calculated. A clipped endpoint should be distinguished from an unclipped endpoint since the former case does not display the marker while the second case does. Figure 3.8 presents the line clipping algorithm discussed above. `DrawLine` (x_a, y_a, x_b, y_b) is a procedure that draws a line between two points $a(x_a, y_a)$ and $b(x_b, y_b)$.

3.4.3. Vital Signs Display

Patient data is first stored in an array capable of buffering two hundred data points before the graphs are displayed on the window. An array of flags, indexed by the MFC value, is used to indicate whether a vital sign is selected for display. If a vital sign is selected, the flag also indicates whether the data is collected by the DLC, or entered manually by the nurse. If the displayed data is collected by the DLC, data retrieval may either come from circular queues, or from PDMS database, and most likely from both. Data is first retrieved from the circular queues. If the time stamp of the requested data is within the range of the time stamps of the head and tail of the circular queue, then the data point is searched by moving backward from the tail of the circular queue. When the time stamp of the requested data matches that of a data point in the circular queue, the MFC code is then used to find the desired data. If more data is needed, or if the data queue has no valid data, the

PDMS database is queried. A detailed description of the PDMS relational database implementation is presented in Fumai's thesis [Fumai, 1992], and therefore, issues of the database implementation, and querying technique will not be discussed here. The same procedure just described above is used for retrieving manually entered data from the linked list.

Once the data is in the buffer, and the view coordinates are given, two consecutive data points are mapped to the viewport coordinates using the transformation described in Section 3.4.1., and they become the endpoints of a line in the graph, which is clipped by the clipping algorithm presented in Section 3.4.2 using the viewport as the clipping region. By repeating this procedure for all the data points, the graphical representation is displayed in the viewport.

Instead of making repeated calls to the line drawing function to draw a piecewise linear graph in the viewport, a Presentation Manager function, *GpiPolyLine*(*hps*, *lCount*, *aptl*), is used to draw *lCount* lines on the presentation space addressed by the handle *hps*. The coordinates of the endpoints of connected lines are placed in an array of structures *aptl*. *GpiPolyLine* is functionally equivalent to the following:

```
for (Index = 0; Index < lCount; Index++)
     GpiLine (hps, aptl + Index) ;
```

where *GpiLine* is the Presentation Manager function for drawing a single line from the current position to *aptl*[*Index*]. Although they are functionally equivalent, *GpiPolyLine* performs the looping deep within a device driver. Therefore, one single *GpiPolyLine* call is much faster than multiple *GpiLine* calls.

3.4.4. Visual Codings in Graphs

Redundant coding is used in the graphical representation of vital sign data. Each graph is coded using a combination of line color, line style, marker color, and marker type. Eight line styles and eight marker types listed in Table 3.4 are available for coding.

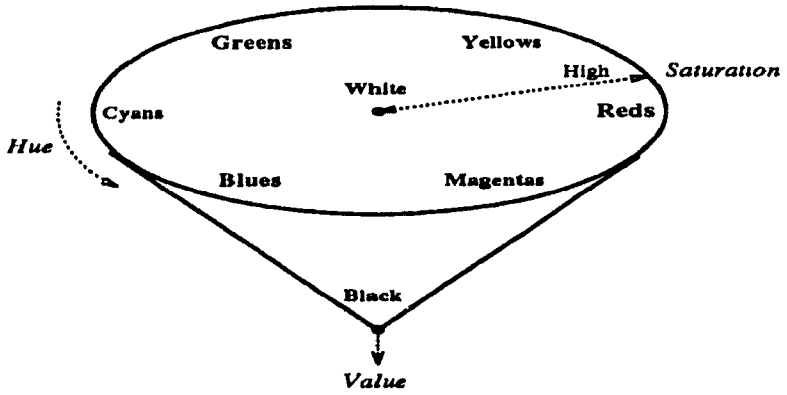

Figure 3.9 HSV Color Model

Table 3.4 Available Line Styles and Marker Types for Coding

Line Styles	Marker Types
Solid	Circle
Dot	Dot
Double Dot	Plus
Dash-Dot	Cross
Dash-Double Dot	Diamond
Short Dash	Filled Diamond
Long Dash	Six Point Star
Alternate	Eight Point Star

The Hue/Saturation/Value (HSV) model is used in describing and visualizing color in the graphical display. This model is preferred over other models, such as the Red/Green/Blue (RGB) model, because of its natural perceptual features, and the independent control of lightness and chromatic contrast [Robertson, 1988]. Figure 3.9 shows the HSV model's perceptual color space. *Hue* is the angle measured around the vertical axis, with red at 0°. The value of *Saturation* is 0 at the center, with a maximum value of 1 at the edge of the cone. *Value*, or *Brightness*, takes on values from 0 to 1 also, with black at the vertex, where *Brightness* is 0, and 1 is at the center of the cone's base. After the HSV attributes are specified by the user, the HSV color space can be mapped to the hardware-oriented RGB color space by using the geometry of the perceptual color space. If the attributes of the RGB model have values from 0 to 1, the algorithm for the color space conversion is given

```
void HSV_To_RGB (float Red, float Green, float Blue,
                 float *Hue, float *Saturation, float *Value)
    {
    int   i ;
    float f, p, q, t ;

    if (Saturation = 0)              /* Achromatic case */
        {
        Red = Value ;
        Green = Value ;
        Blue = Value ;
        }
    else                             /* Chromatic case */
        {
        if (Hue ≥ 360)
            Hue = Hue mod 360 ;
        Hue = Hue/60 ;
        i = Floor(Hue) ;             /* i = the largest integer ≤ Hue */
        f = Hue − i ;
        p = Value × (1 − Saturation) ;
        q = Value × (1 − (Saturation × f)) ;
        t = Value × (1 − (Saturation × (1 − f))) ;
        if (i = 0)
            {Red = Value; Green = t; Blue = p;}
        else if (i = 1)
            {Red = q; Green = Value; Blue = p;}
        else if (i = 2)
            {Red = p; Green = Value; Blue = t;}
        else if (i = 3)
            {Red = p; Green = q; Blue = Value;}
        else if (i = 4)
            {Red = t; Green = p; Blue = Value;}
        else if (i = 5)
            {Red = Value; Green = p; Blue = q;}
        }
    }
```

Figure 3.10 Algorithm for HSV to RGB color space conversion

in Figure 3.10.

Although the line and marker colors can be assigned different color codes, experiments have shown that excessive use of color codes will impede the rate of object recognition instead of helping it. Recognition rate of a trend in a window with twelve vital signs data has dropped from 2.7 seconds, using different color codes for

line and marker, to 1.6 seconds, using same color code for both. Therefore, lines and markers are assigned the same color codes initially, but in some special cases where two, or more, graphs are using the same line color, then the user will have to assign a different marker color to each graph in order to differentiate them. Furthermore, adequate hue difference is maintained to improve the accuracy of recognition, and a legend is used to improve the recognition rate.

Each attribute of the HSV model can take on sixty four discrete values. Therefore, a maximum number of 64^3, or 262,144, colors are available for selection. When color coding is combined with line styles and marker types, sufficient space is available among graphs so that a high density of data can be displayed on the display window. Another advantage of using line styles and marker types is that if the color coding should fail, (e.g when the user is color-blind, or printing a black and white hardcopy) they provide additional visual codings to the graphs.

3.5. User Interface

The design of an user interface for the VSMS is described in this section. The interface is built using the OS/2 Presentation Manager which provides an user-friendly window environment. The user interface design considerations described in Section 1.2.4 are closely followed in the development of the VSMS user interface.

3.5.1. The OS/2 Presentation Manager

The VSMS runs under the OS/2 operating system using the Presentation Manager environment. OS/2 Presentation Manager is a collection of dynamic link libraries (DLLs) that extend the functionality of OS/2 to include window management and graphics [Petzold, 1989]. DLLs provide a way to use the same code simultaneously without making it part of any single application, and thus increase the programming

efficiency [Southerton, 1989]. The OS/2 operating system was developed for personal computer based on the Intel 80286 and 80386 microprocessors. It exploits the hardware by providing program isolation, memory management, task management, interprocess communications and timer services [IBM, 1988].

The Presentation Manager user interface and application program interface (API) are part of IBM's Systems Application Architecture (SAA). SAA is a collection of guidelines aimed at setting standards for user interfaces and API [IBM, 1988]. The main goal of SAA is to facilitate program portability among different IBM platforms, and the Presentation Manager user interface may become a common sight on IBM minicomputer and mainframe terminals.

The Graphics Programming Interface (GPI) of the Presentation Manager is device independent. An application does not have to identify an output device in order to use it [Petzold, 1989]. Therefore, the programmer can use the same code to display graphics and text on virtually any output device. Device independency protects Presentation Manager programs from obsolescence even though video technology is advancing rapidly.

Presentation Manager uses a message-based architecture in which programs get information from the operating system through messages. It uses a system of queues and application window procedures to process messages. Input from the mouse or keyboard, selection from a menu, resizing a window, and repainting part of a window are all delivered to a program in the form of messages. All the applications in the system share the same system queue which receives messages from the system, other applications, or the timer. An input router is used to direct messages to their appropriate application message queues [IBM, 1988].

The Presentation Manager provides a consistent user interface across applications because all the elements of user interface such as menu and dialog box are built into the Presentation Manager rather than into each individual application [Petzold,

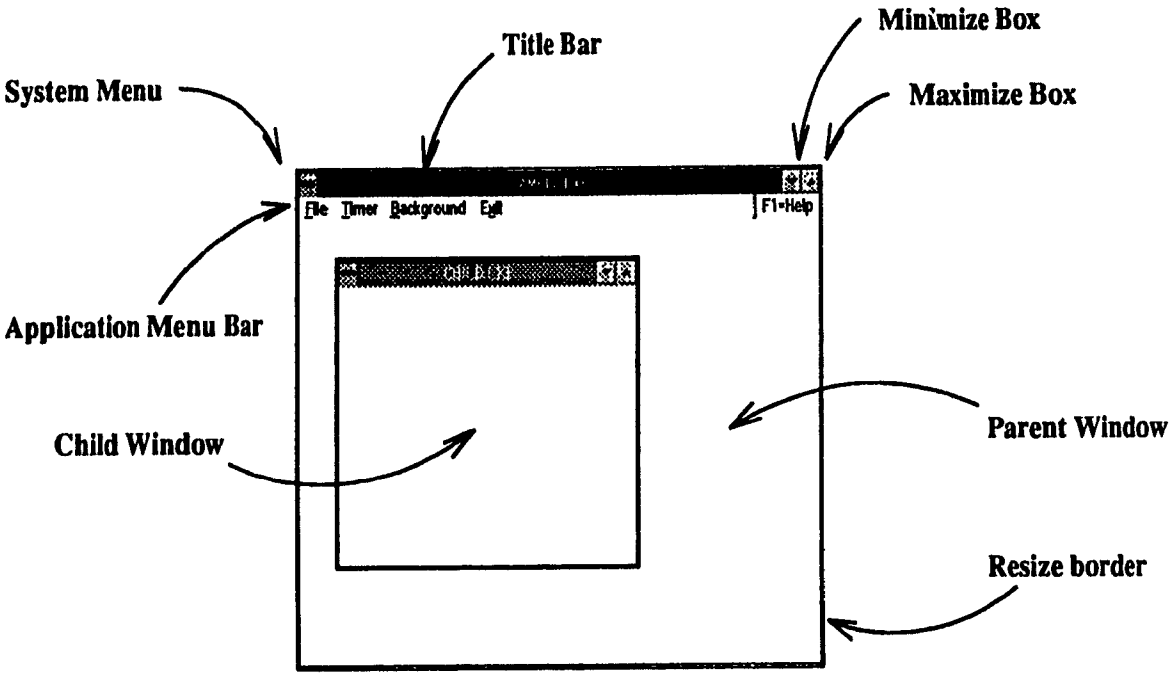

Figure 3.11 A Sample of Presentation Manager Window

1989]. Therefore, the learning time for a new Presentation Manager program can be significantly reduced if the user has experience with another Presentation Manager program.

Printing under Presentation Manager involves creating a presentation space, and opening a device context [Southerton, 1989]. The device context describes a physical output device and sends this description to the presentation space. The presentation space is an internal data structure that contains the necessary information on the drawing environment such as current color, cursor position, fonts, and page size, to exchange graphic data with a device context. When a presentation space is associated to a device context, graphics output directed to the presentation space will appear on the related device.

Figure 3.11 shows a sample window created using Presentation Manager. The system menu is always placed at the top left hand corner of the window. A pull

down menu will appear on the screen in response to a click on the system menu box, where the system level commands such as resizing, moving, and closing the window are shown as menu items. The menu bar is used for application specific menu items. It can be used to create a single level permanent menu, or pull down menus, and to invoke dialog boxes when the menu items are selected.

Resizing the window can be done by using the iconic user interface, menu item selection, or direct manipulation. Menu selection method is available through the system menu as discussed. The minimize and maximize icon boxes are on the upper right hand corner of the sample window. A window can be minimized to a single icon, or maximized to occupy the entire screen by clicking on these icons presenting a down arrow for the minimize box, and an up arrow for the maximize box. A thick sizing border can be created around the window so that the user can use the mouse to drag on the border to resize the window.

A vertical scroll bar and a horizontal scroll bar can also be created at the right hand edge and the bottom of the window. The contents of the client area which may contain a graphical or text display can be scrolled vertically or horizontally by clicking on the scroll bars. Hardware binding is also possible to enable the use of the arrow keys on the keyboard.

All of the above components of a Presentation Manager window can be created by using a series of flags called frame creation flags when the window is first created. Control windows are additional means of input to Presentation Manager programs. They are child windows that take the form of objects such as buttons, scroll bars, list boxes, and text entry fields. A control window processes inputs from the mouse and keyboard, and notifies its owner of input events.

Composite interaction tasks are performed in Presentation Manager using dialog boxes. A dialog box is a special window that contains various child control windows as shown in Figure 3.12. Each control window in a dialog box is used to

perform a basic interaction task. Dialog boxes are generally used to obtain user input beyond that which can be easily handled in a menu.

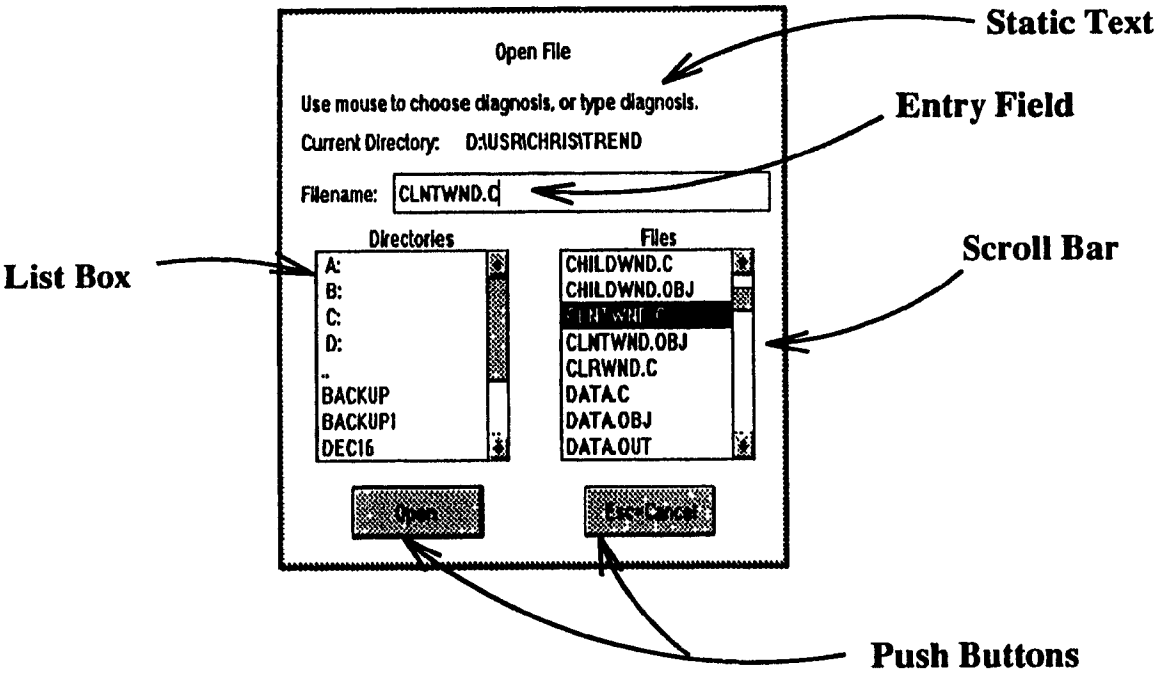

Figure 3.12 A Sample Dialog Box

3.5.2. Dialog Design

Consistency of the user interface is maintained in the conceptual model, functionality, and sequencing of dialog design. State diagrams are useful to identify inconsistency in the design, and they are also used to minimize the number of different syntactic structures because it has been shown that users can learn a user interface faster, and make fewer mistakes if the interface has a simpler syntactic structure [Foley *et al*, 1990]. There are mainly three types of syntactic structures used for the interface in VSMS. The first type is used to allow the user to enter single or

multiple values which upon acceptance may require redrawing of the graphical output, e.g. changing the time range of the viewport. The second type of syntactic structure is used for displaying information in a dialog box only, e.g. a dialog box for the help menu. The dialog box will be dismissed after the user has finished with the displayed information. Finally, there are those options which do not require input from the user, but may also affect the graphical display, e.g. imposing a grid on the viewport. Selection of these options causes redrawing of the display immediately. A combination of these basic structures can also be used for more complicated interactions.

The main menu of the VSMS display allows the user to create multiple windows for simultaneous graphical display of vital signs from the fourteen beds, and to bring up a settings control window which allows the user to change the graphical settings either globally, or individually for each bed. Figure 3.13 shows part 1 of the state diagram of the user interface for the vital signs display window. All the states in the diagram, except the one for printing, allow the user to enter values specifying changes to the display. Printing does not require input from the user because currently it uses the viewport coordinates, time scale, and time range as default values. Therefore, the hardcopy of the graphical output is the same as the one displayed on the screen. Future improvements could enable the user to specify a different scale and time range for hardcopies.

The user can either select vital signs from a list of available vital signs, or deselect them from a list of currently displayed vital signs. In case of equipment failure, or for other reasons, the user can select part of the vital signs data to be entered manually instead of collecting data from the CarePort using DLC. The procedure of selecting vital signs for manual data entry is similar to selecting vital signs for display.

For manual data entry, the user can either select or enter the name of the vital sign which is on the list for manual data entry, and then enter the value of the vital

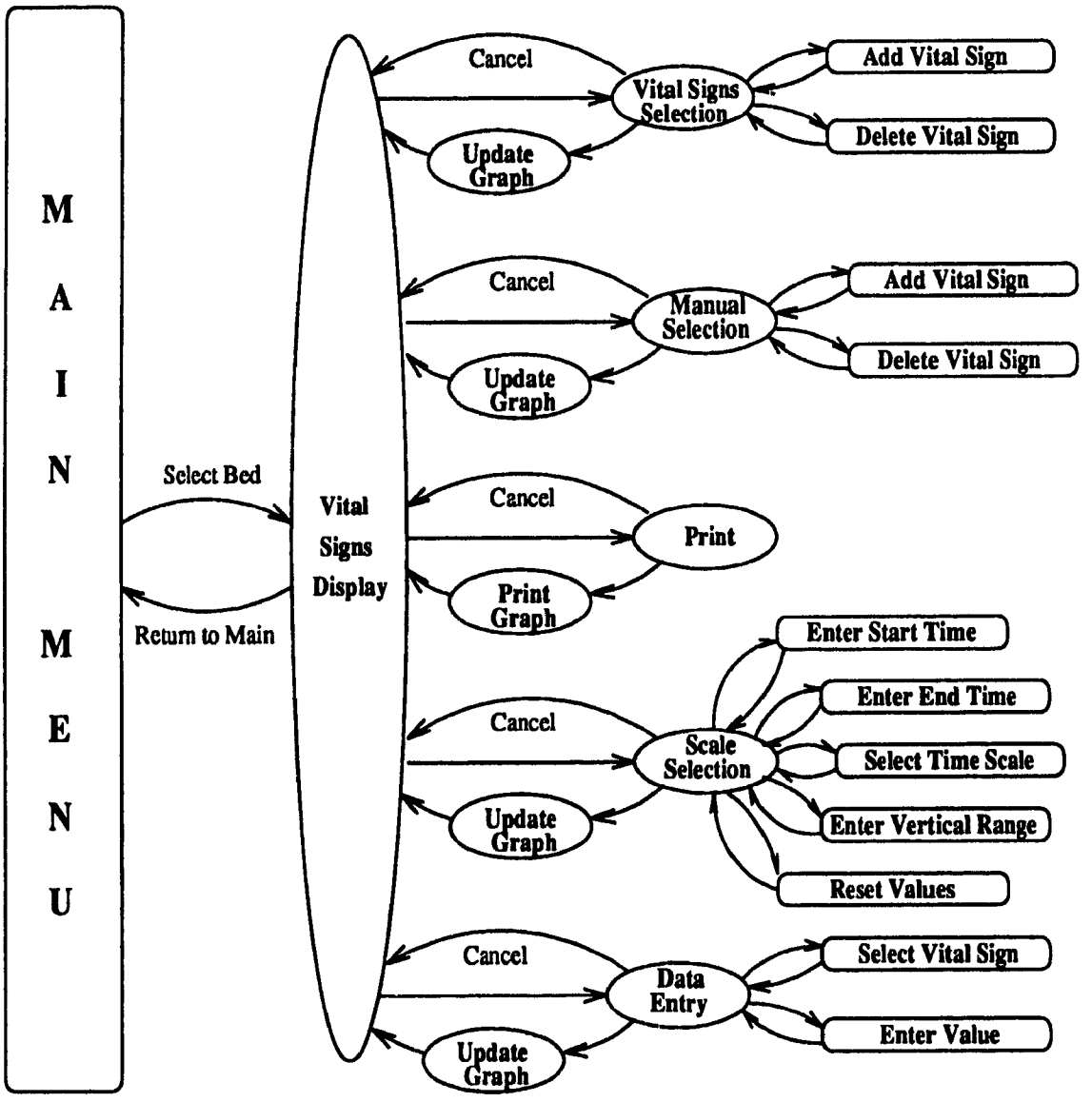

Figure 3.13 State Diagram of Vital Signs Graphical Display, Part 1

sign. The date and time of the entered data is taken as the current date and time.

If the user wants to display more data in the window, or to increase the resolution of the graph, he can change the time range (horizontal axis) and the data value range (vertical axis) of the viewport. The user can also change the time scale to display half-hour, minute, or second data on the screen. The initial values of

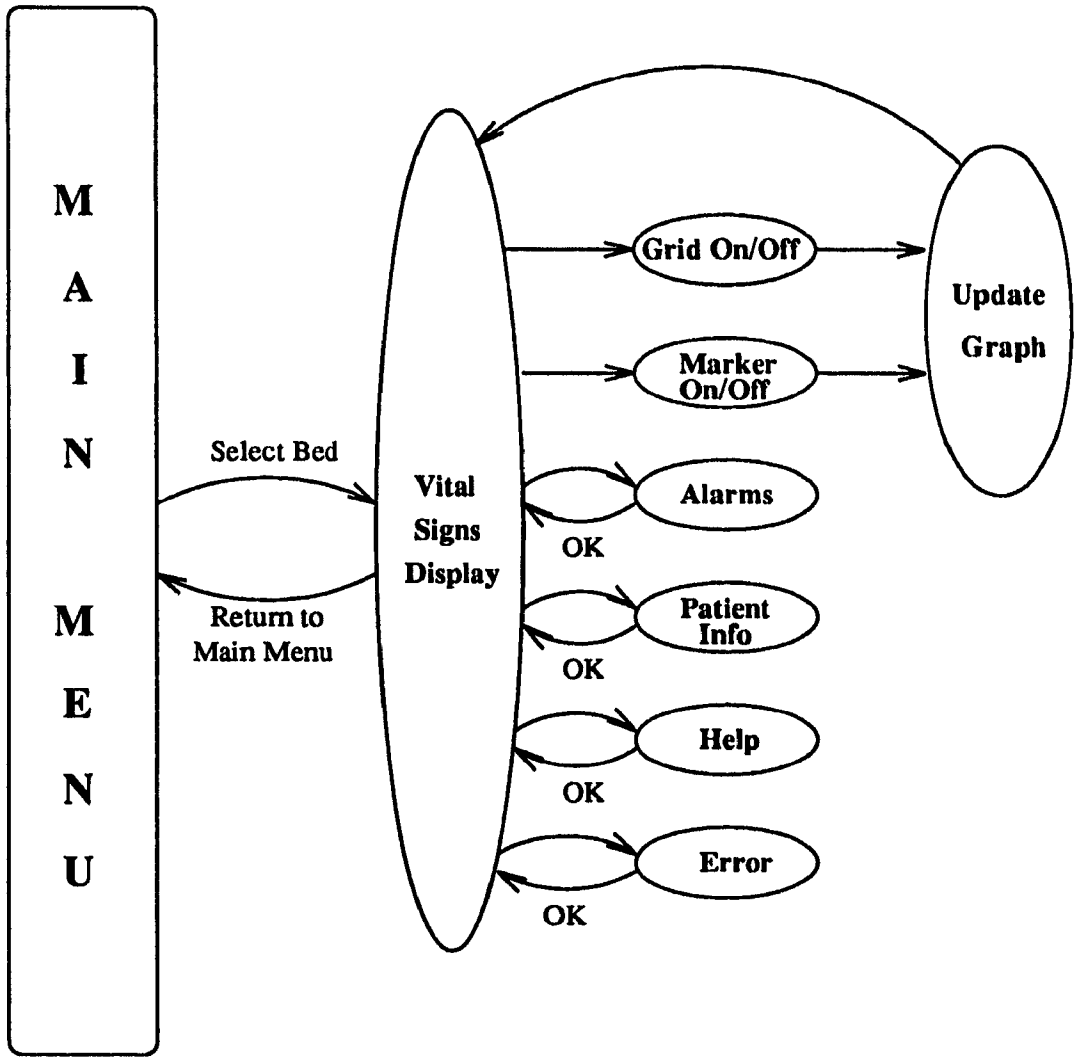

Figure 3.14 State Diagram of Vital Signs Graphical Display, Part 2

the time scale and display ranges for both horizontal and vertical axes are saved so that the user can reset these values to their initial values after changes have been made. All the states described above can return to the neutral state by rejecting the changes. If the changes are accepted, the graphical output will be updated, or printed, to reflect the changes.

Figure 3.14 shows part 2 of the user interface for the vital signs display window. As shown in the state diagram, selections such as imposing, or removing a grid

as well as showing, or hiding markers on the graph will cause immediate changes on the output without confirmation. These options are simple actions and are reversible by selecting the opposite options. Popping up a dialog box for confirmation every time these options are selected would be frustrating to the user rather than being helpful. The other states in the diagram are responsible for alarms review, patient information display, help information, and error message feedback. These functions are either invoked by the user when needed (e.g. help facility), or triggered automatically by the system (e.g. error messages), and they are dismissed after the user has finished with the displayed or feedback information. Therefore, they use the second type of syntactic structure discussed earlier.

The state diagram of the graphical display settings control window is shown in Figure 3.15. Although the state diagram is more complex, it is merely a combination of the syntactic structures already presented. This settings control window provides maximum control over the visual coding of vital signs display by allowing the user to change the color and line style, as well as the color and type of marker for each individual vital sign graph. Furthermore, the user can make changes to the visual codings of data on a particular bed, or make changes globally to all the beds in the ICU. Normally, a global change to visual codings is recommended because when displaying multiple beds simultaneously, consistent visual codings reduce the recognition time for vital signs data. However, since different windows for different beds could normally also display different combinations of vital signs, one set of visual codings may be poor in one case, and good for the others. Therefore, it is desirable to be able to assign codings to each bed locally also. When the control window is being dismissed, the user can either make changes to the visual codings, or simply disregard the changes. Multiple changes to different vital signs and beds are possible without dismissing the control window.

It is obvious from the observation of the state diagrams that two types of windows are required for the VSMS besides the Vital Signs Monitoring Main Win-

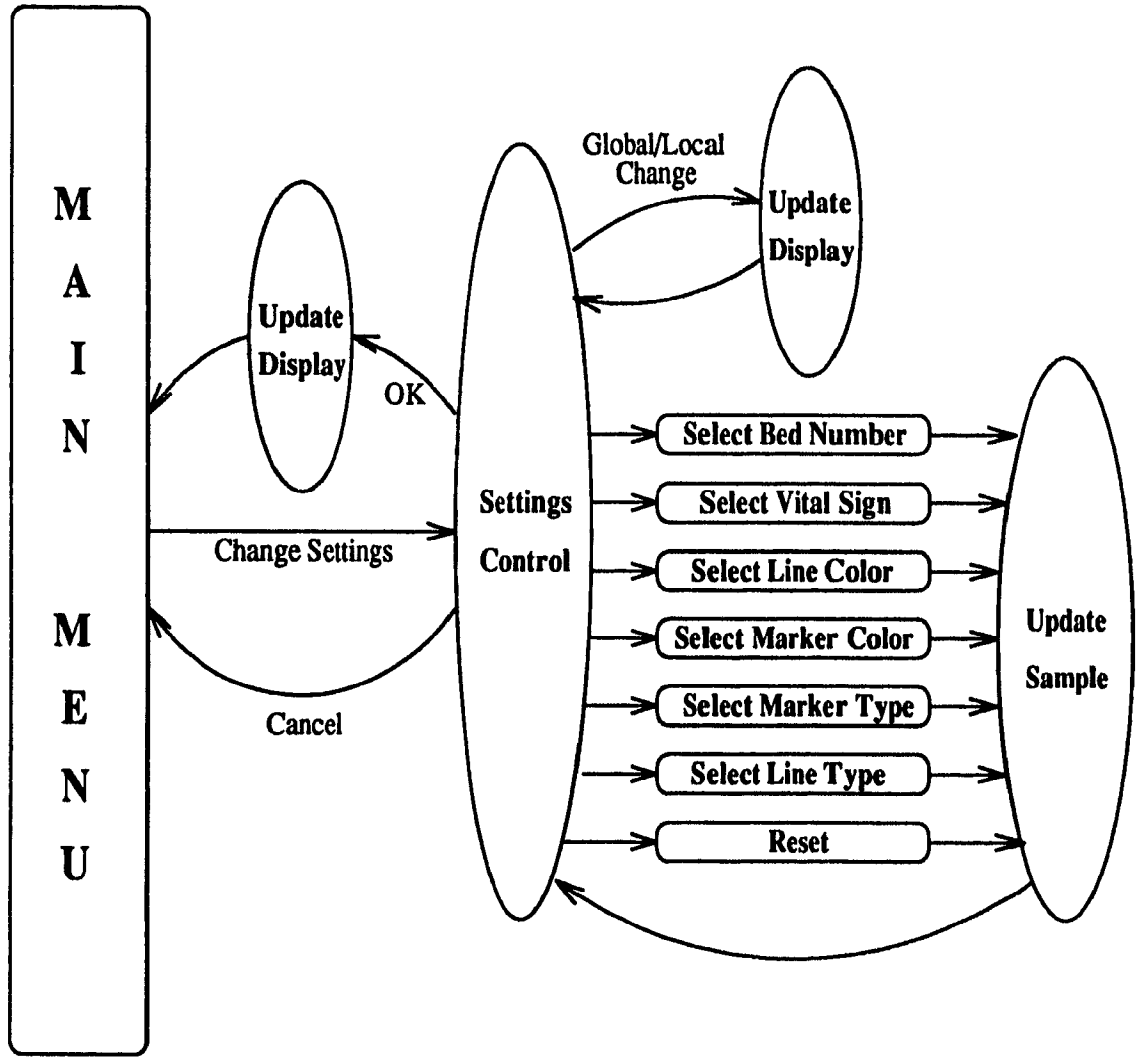

Figure 3.15 State Diagram of Graphical Display Settings Control Window

dow. One window is used for changing the settings of the graphical display, and one window is required for the vital signs graphical display for each of the fourteen beds. Therefore, the VSMS Main window is created using Presentation Manager primitives as shown in Table 3.5. The Main window uses the desktop window as its parent to make the frame a top-level window.

Similarly, Table 3.6 and Table 3.7 describe the creation of the children windows. These tables show the use of various dialog elements provided by Presentation

Table 3.5 Description of Vital Signs Monitoring Main Window

Vital Signs Monitoring Main Window	
Parent Window	Desktop Window
Frame Creation Flags	Title Bar Sizing Border System Menu Minimize & Maximize Boxes Task List Menu Bar
Menu Items	Settings Bed Selection
Child Windows	Settings Control Window Graphical Display Windows for Fourteen Beds

Table 3.6 Description of Settings Control Window

Graphical Display Settings Control Window	
Parent Window	Vital Signs Display Main Window
Frame Creation Flags	Title Bar Fixed Window Border Minimize Box
Dialog Boxes	Help Error
Control Windows	Horizontal Scroll Bars (Color Selection) Push Buttons (OK, Change, Reset, Cancel) List Box (Vital Signs Selection) Radio Buttons (Bed Selection) Radio Buttons (Line Style, Marker Type) Radio Buttons (Line/Marker Color) Radio Buttons (Global/Local Change)
Sibling Windows	Fourteen Graphical Display Windows

Manager such as menu selection, dialog box, and control windows. The constructions of the dialog boxes are described in tables from Table 3.8 to Table 3.14.

Windows in the Presentation Manager are organized using a parent-child relationship. A child window is always displayed within the area of the screen occupied by its parent, and it remains in the same position relative to the parent unless explicitly moved. All the children of a parent window will be hidden, minimized, or destroyed, if the parent is hidden, minimized, or destroyed. Windows with a common parent are called sibling windows. Sibling windows can overlap each other on the screen.

Table 3.7 Description of Graphical Display Window

Vital Signs Graphical Display Window	
Parent Window	Vital Signs Display Main Window
Frame Creation Flags	Title Bar Sizing Border System Menu Minimize & Maximize Boxes Task List Menu Bar
Menu Items	Vital Signs Selection Scale Selection Manual Data (Select Vital Signs, Enter Data) Options (Grid On/Off, Marker On/Off) Print Patient Information Alarms Help
Dialog Boxes	Vital Signs Selection Scale Selection Vital Signs Selection for Manual Data Entry Data Entry Print Confirmation Patient Information Alarms Display Help Error
Control Windows	Vertical Scroll Bar Horizontal Scroll Bar
Sibling Windows	Settings Control Window Thirteen Graphical Display Windows

Table 3.8 Vital Signs Selection Dialog Box

Control Windows	Descriptions
List Boxes	List of available vital signs List of displayed vital signs
Push Buttons	Add vital sign Delete vital sign OK Cancel

Table 3.9 Scale Selection Dialog Box

Control Windows	Descriptions
Radio Buttons	Time Scale
Entry Fields	Begin Time End Time Maximum Vertical Value Minimum Vertical Value
Push Buttons	OK Cancel Reset

Table 3.10 Data Entry Dialog Box

Control Windows	Descriptions
List Box	Vital Signs Selection
Entry Fields	Data Value Time
Push Buttons	OK Cancel

Table 3.11 Alarms Display Dialog Box

Control Windows	Descriptions
List Box	Listing of Alarm Messages
Scroll Bar	Scrolling Alarms
Push Button	OK

Table 3.12 Print Confirmation Dialog Box

Control Windows	Descriptions
Static Text Field	Message for Confirmation
Push Buttons	OK Cancel

Table 3.13 Help Dialog Box

Control Windows	Descriptions
Static Text Field	Message for Help
Scroll Bar	Scrolling Help Display
Push Button	OK

Table 3.14 Patient Information Dialog Box

Control Windows	Descriptions
Static Text Fields	Patient Name Gender Birth Date Address Telephone Physician Name Hospital ID Number ICU Bed Number Date & Time of Admission Memo
Push Button	OK

4. Implementation, Results, and Future Extensions

The algorithms used for data management and graphical display of data have been presented in Chapter 3. The conversion of these algorithms to program source code is straight forward, and therefore, will not be discussed here. However, some implementation issues such as interprocess communication, and interthread communication will be presented in the next sections. The performance of the system will be discussed, and possible improvements to the system are suggested. Future extensions to the PDMS are proposed in the final section.

4.1. Implementation

4.1.1. Interprocess Communication

The commands from PDMS processes are passed to *command_line* thread through a character string stored in the shared memory. Shared memory is potentially the most efficient and flexible method for exchanging data between processes, because once the data is written to memory, it does not need to be copied or otherwise processed by the operating system, and all the data in the segment can be freely accessed by any sharing process. A named shared memory segment is allocated by specifying the desired segment size, and the name assigned to the shared segment using the OS/2 file-naming convention in the fictitious directory \SHAREMEM\. A separate selector for the allocated segment is required for each process to access the shared segment. Since many processes have access to this shared memory command

buffer, system semaphores are used to coordinate activities among processes such that only one process can access the command buffer.

Table 4.1 Command String Format

Command	Definition
a#_	Add bed # to list of active beds
c#@	Change bed # to bed # @
d#_	Delete bed # from list of active beds
k__	Check CarePort unit status
l#_	List instrument status of bed #
m__	Dump contents of memory buffers to disk files
r#_	Resume data acquisition for bed #
s#_	Suspend data acquisition for bed #
u__	Reset and test CarePort Unit
w__	Status of all logical sources

Table 4.1 gives the list of available commands supported by *command_line*. The first character in the command string represents the command to be executed. The second character is the bed number associated with the command. A second bed number is required for some of the commands, and it is indicated by the third character. After the command has been executed, the first character will contain the error code of the executed commands.

Since the burden of managing shared memory segments falls upon the processes that use them, semaphores are used to construct the handshaking protocol for serializing access to shared data. The semaphores that are used for interprocess communication are shown in Table 4.2. A semaphore can be in one of two states: set, or clear. A set semaphore indicates that a thread should stop and wait for the semaphore to be cleared by another process, and a clear semaphore means that the thread can continue. The global name of a semaphore is assigned by using OS/2 file-naming conventions and begins with the prefix \SEM\. A semaphore handle is used to reference a semaphore, within a process.

When a PDMS process wants to access a patient's data, it requests the exclusive control of *Qsem*. After the modifications are done, the process then relinquishes

Table 4.2 Interprocess Communication

Name of Semaphore	Semaphore Handle
\SEM\ICU\CMDCTL.SEM	CMDsem
\SEM\ICU\EXECMD.SEM	EXEsem
\SEM\ICU\RESULT.SEM	RESsem
\SEM\ICU\READ.SEM	Qsem
Shared Memory	**Shared Segment Selector**
\SHAREMEM\ICU\COMMAND	CMDsel

its control. The handshaking for accessing the command buffer is more complicated. *CMDsem* is used to indicate whether the shared command buffer is being accessed or not. When *CMDsem* is cleared, then the command buffer is free to be accessed by any process. *EXEsem* and *RESsem* are used for passing a command to the shared memory and returning the error code respectively. *EXEsem* cleared indicates that the command in the shared memory is ready for the *command_line* thread to execute. Once the *command_line* thread has executed the command and placed the error code in the shared command buffer, it clears *RESsem* to signal the PDMS process to act upon the error code. The handshaking protocol for command execution is shown in Figure 4.1.

4.1.2. Interthread Communication

Since the *command_line* and *read_port* threads share the same serial port, they must cooperate with each other to avoid serial port access at the same time. When commands are issued, the *command_line* thread has to obtain the answer from CarePort through the *read_port* thread, in order to return the proper error code to the calling process. The *POLLDATAsem* and *LINKCTLsem* semaphore handles function as completion flags between the two threads. Each time a request message is sent to the CarePort, *command_line* sets the semaphore *LINKCTLsem* using *DosSemSetWait* and waits for it to be cleared by the *read_port* thread using *DosSemClear* upon the arrival of a command response. Similarly, arrival of a polled data message will clear the *POLLDATAsem* enabling the command which requested this data in the *command_line* thread to continue its execution.

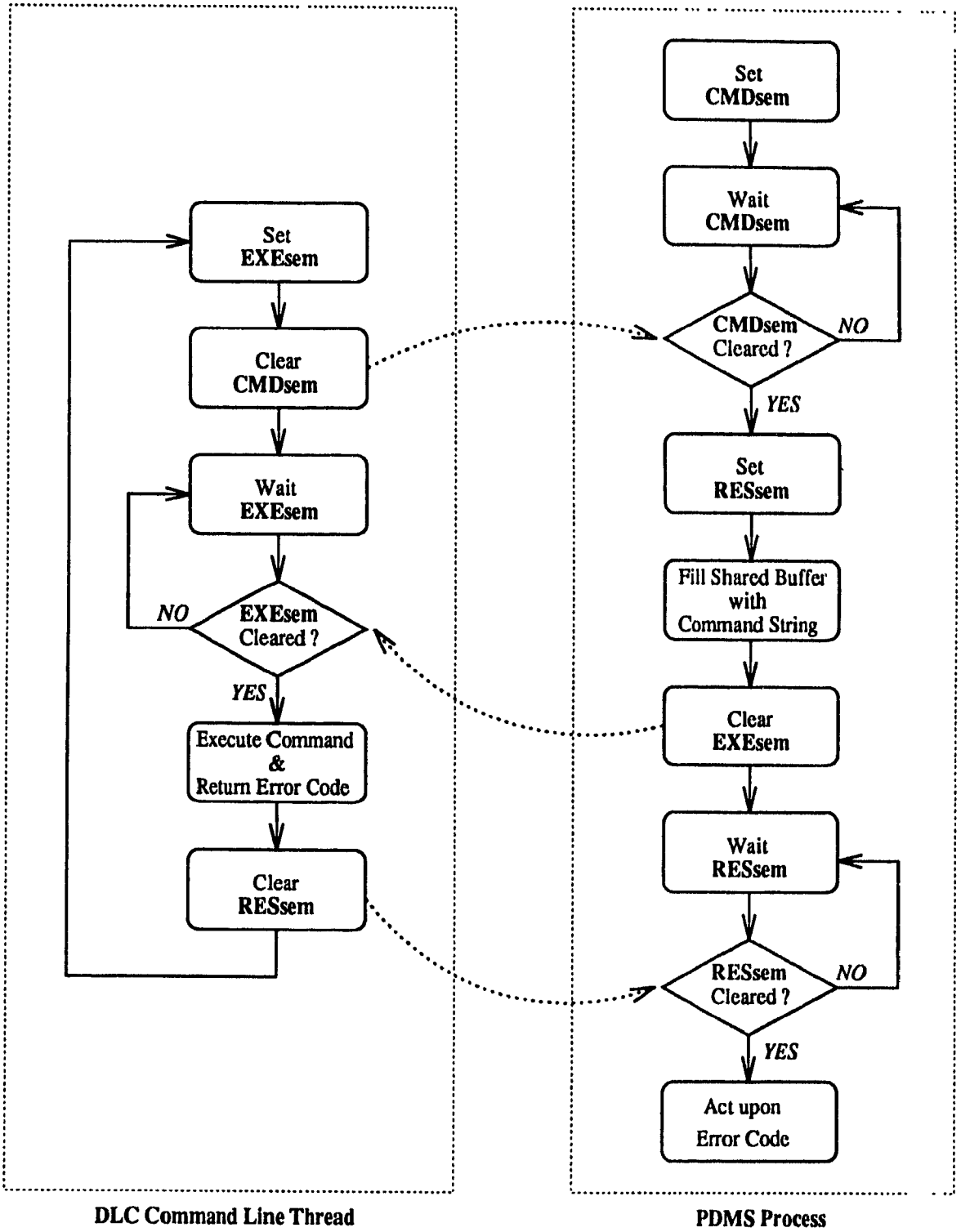

Figure 4.1 Command Handshaking Protocol

Pipes are used to control the flow of data coming from the CarePort unit, and they remove the need of semaphores for the proper execution of *accum_params* and *alarm_handler*. Table 4.3 gives the pipe handles used for reading and writing to the pipes. Data is written into the pipe using *DosWrite* with the proper *_pipeW* handle, and data is read out using the *_pipeR* handle. This is used to communicate data messages received and decoded by *read_port* thread to *accum_params* and *alarm_handler* using the *PV_pipe* and *AT_pipe* respectively. Since the HP78534A physiological monitors are currently used to display the actual waveforms of the selected vital signs at the bedside, the wave data is not supported currently by the PDMS and the *WV_pipe* is not used by any thread. However future database extensions may accommodate wave data.

Table 4.3 Interthread Communication using Pipes

Type of Data	Handle for Reading	Handle for Writing
Wave	*WV_pipeR*	*WV_pipeW*
Alarm	*AT_pipeR*	*AT_pipeW*
Parameter	*PV_pipeR*	*PV_pipeW*

The semaphore handle *COM1sem* is used to ensure that only one thread has direct access to the serial port. The thread intending to gain access to the serial port must obtain the exclusive control of *COM1sem* through the *DosSemRequest* function call first. After all the necessary operations are performed, the thread will clear the semaphore using the *DosSemClear* function.

4.1.3. User Interface

Windows and dialog boxes for the VSMS user interface are constructed from the state diagrams, and the descriptions of window frame creation flags, parent-child window relationships, menu design, and dialog elements presented in Chapter 3. The implementation fully supports the use of mouse, keyboard, arrow keys, and function keys, in order to accommodate multiple skill levels. A consistent hardware binding is used in the user interface to improve the retention rate of experienced

users. Table 4.4 shows the use of key bindings to complement the mouse input functions. Each menu item is is assigned a unique letter, such that the user can use the **Alt** key to bring the input focus to the menu bar, and then type the letter to select the desired menu item. The assignment of letters to menu items uses the first letter, the first consonant, or a subsequent consonant of the menu item.

Table 4.4 Key Bindings for Windows and Dialog Boxes

Keys	Descriptions
Enter	accept inputs from entry fields; select highlighted menu item, if menu bar is activated; activate push button having input focus
ESC	cancel inputs; dismiss dialog box; dismiss pull down menus
Tab	move input focus to the next input (group of inputs)
Alt	activate menu bar
Arrows	move the input focus within a group of inputs; move selections in menu bar; activate scroll bars
Control	keyboard accelerators

In addition to the above mentioned hardware bindings, keyboard accelerators are implemented for experienced users. These accelerators provide shortcuts to the menu items by typing the letter assigned to a menu item while pressing down the **Control** Key. Function keys are also available to minimize memorization by using the **F1** key to designate the first menu item, **F2** the second, and so on. For experienced users, keyboard accelerator is the preferred and the fastest way for menu selection.

Figures 4.2 to 4.8 illustrate the implementation of the windows and dialog boxes for the VSMS. The VSMS main window is shown with a graphical display window in Figure 4.2, and the menu items described in the previous chapter are implemented in the menu bar of the window. Figure 4.3 shows that a maximum number of 4 patients is ideal for multiple window display within the VSMS main window. The graphical display windows are created as the children windows of the

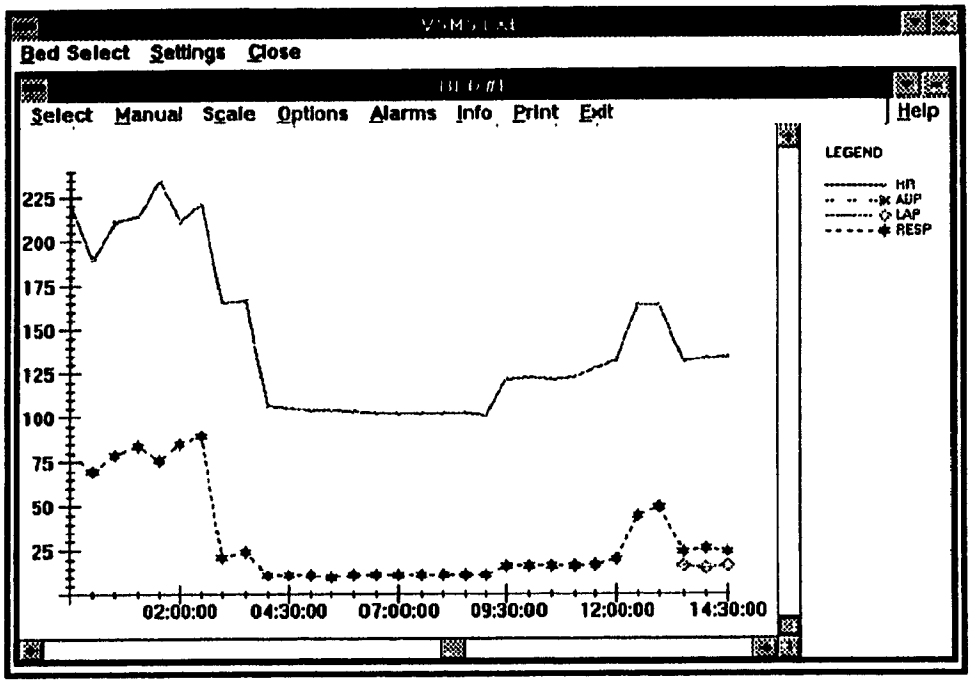

Figure 4.2 VSMS Main Window and Graphical Display Window

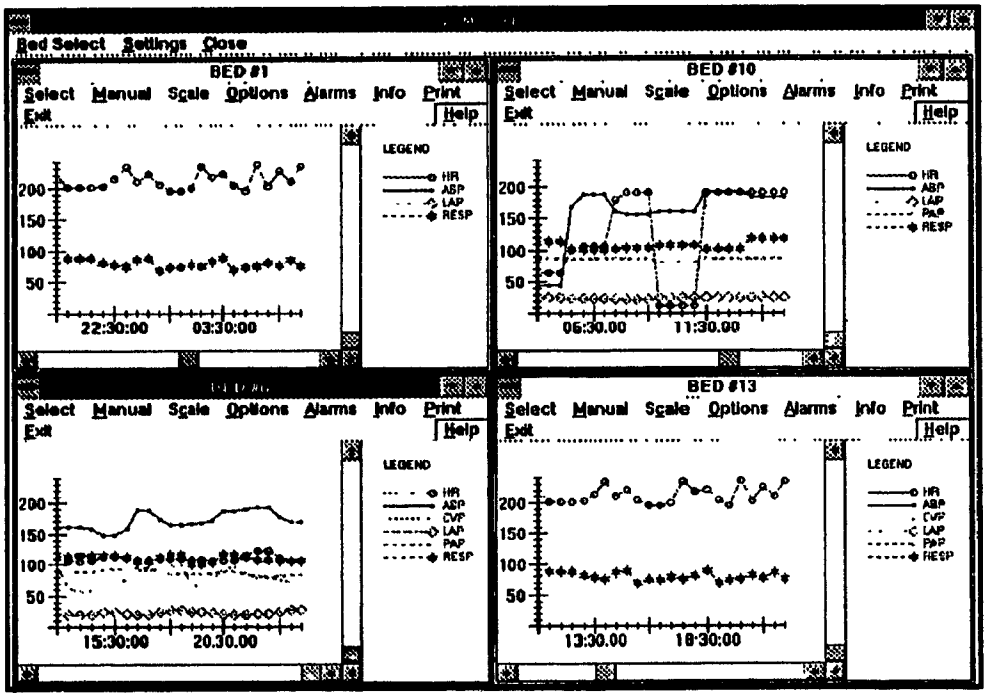

Figure 4.3 Tiled Graphical Display Windows

main window so that when switching to other PDMS windows, VSMS with all its graphical display window can be reduced to an icon by clicking on the minimize box of the main window. Figures 4.4, 4.5, 4.6, 4.7 and 4.8 show the Control Settings, Trend Selection Scale, Patient Information and Manual Data Entry windows.

4.2. Results

This section presents the performance evaluation results of the VSMS. The evaluation was performed on an IBM PS/2 model 80 computer, with an Intel 80386, 16 MHz, microprocessor, 8 Meg RAM memory and a 8514A high resolution display adapter, running version 2.0 of OS/2. The VSMS consists of approximately 7,000 lines of source code developed using the Microsoft C compiler version 6.0. For execution the VSMS module requires about 247Kbytes of RAM memory.

Figure 4.9 shows the time required for data retrieval from the PDMS database and from the circular queues. The time reported here is the retrieval time for fifty data points. Response times are recorded for the retrieval of data from one vital sign to twelve vital signs. It is found that the retrieval time relates linearly to the number of vital signs. The database retrieval time provides the upper bound of the response time, 429 ms/graph, when all the data points are saved in the database. On the other hand, the optimal retrieval time is 28 ms/graph, when all the data is in the circular queue. It is obvious from the figure that buffering the data can improve the performance dramatically, when the user uses the scroll bars to view the graphs because it avoids database retrievals while processing data buffered in the circular queue.

The redraw speed of the graphs was measured for different numbers of vital signs as shown in Figure 4.10. It was found to be directly proportional to the number of vital signs. The same evaluation was performed on two other machines

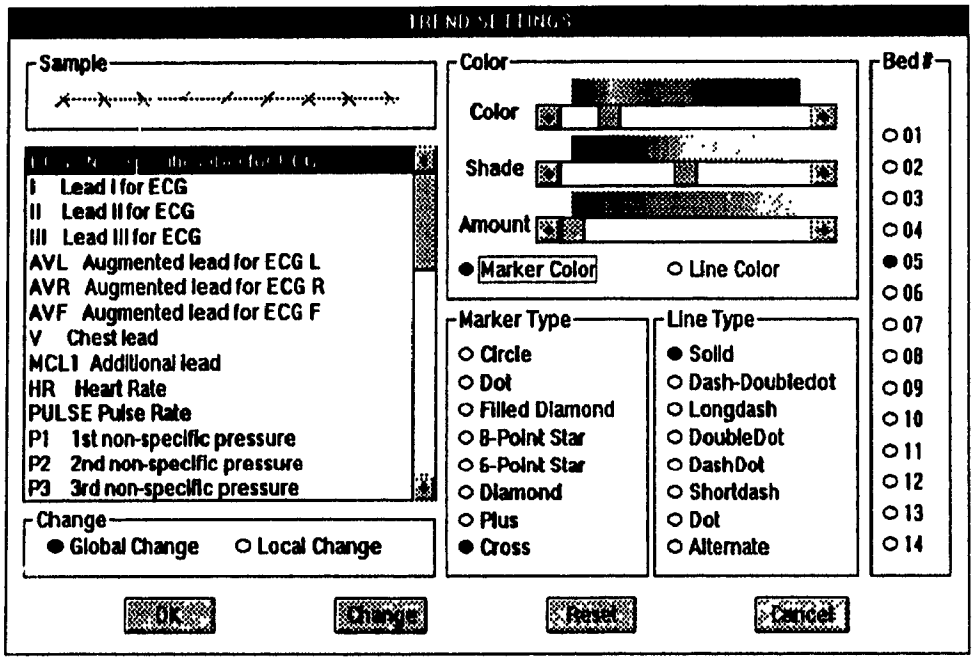

Figure 4.4 Graphical Display Settings Control Window

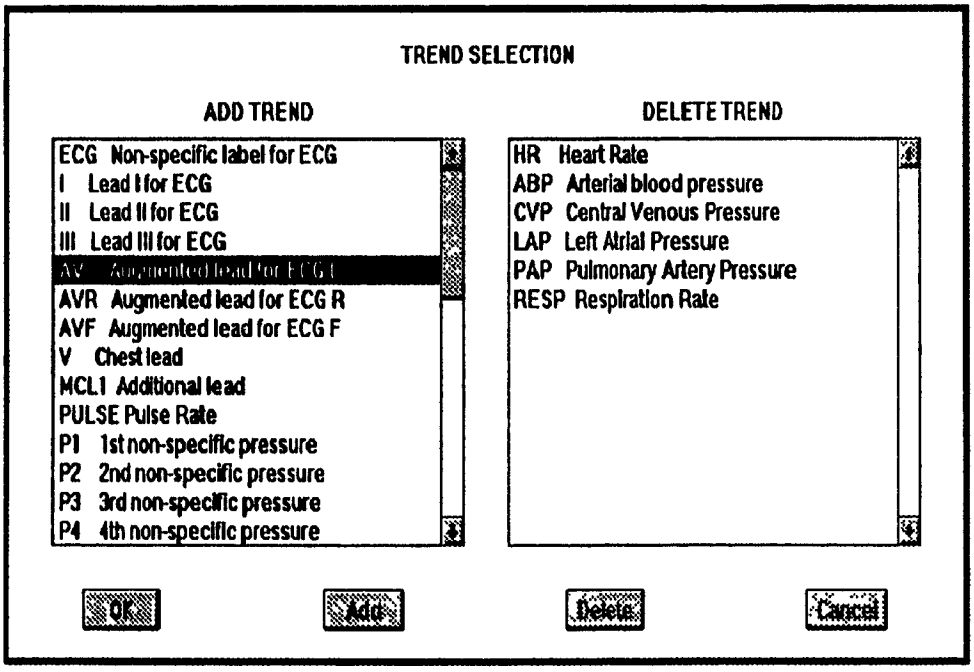

Figure 4.5 Vital Signs Selection Dialog Box

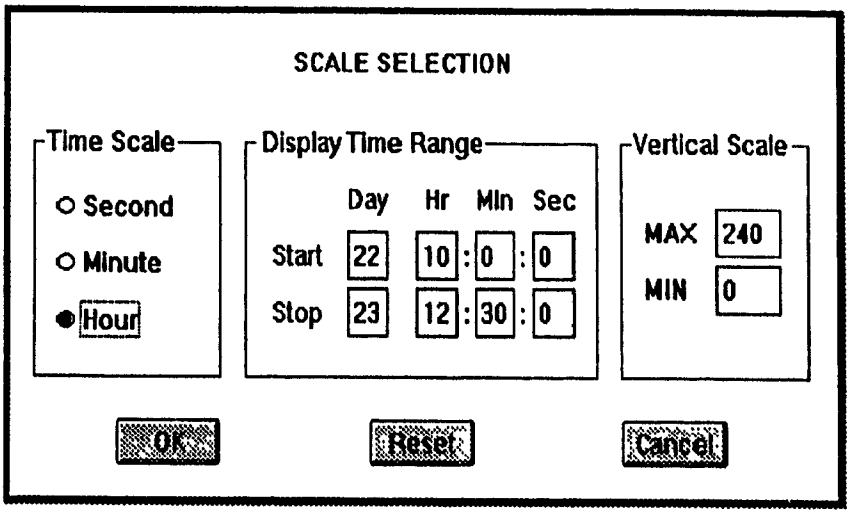

Figure 4.6 Scale Selection Dialog Box

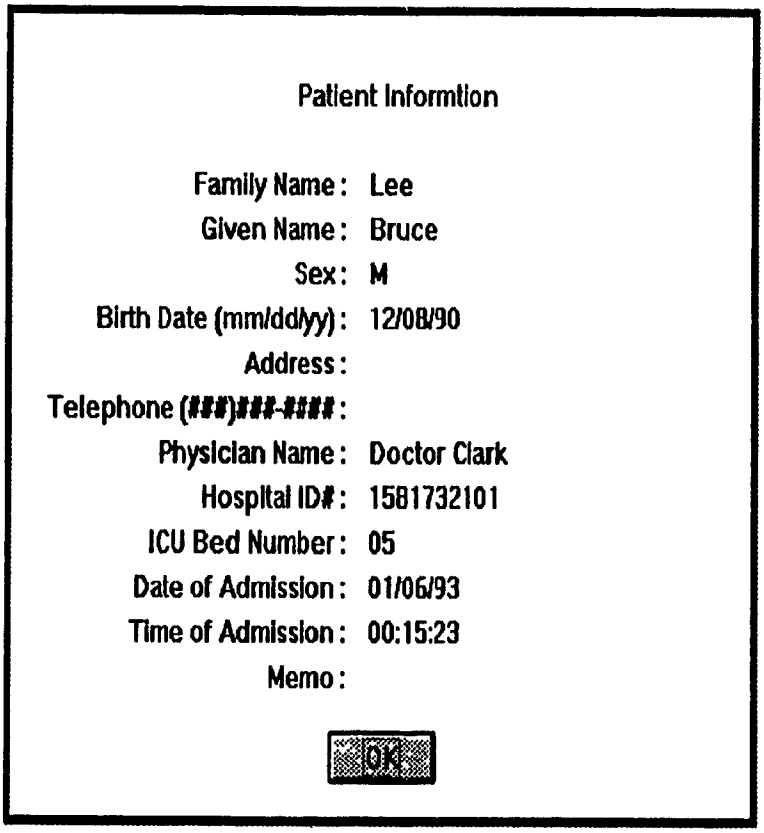

Figure 4.7 Patient Information Dialog Box

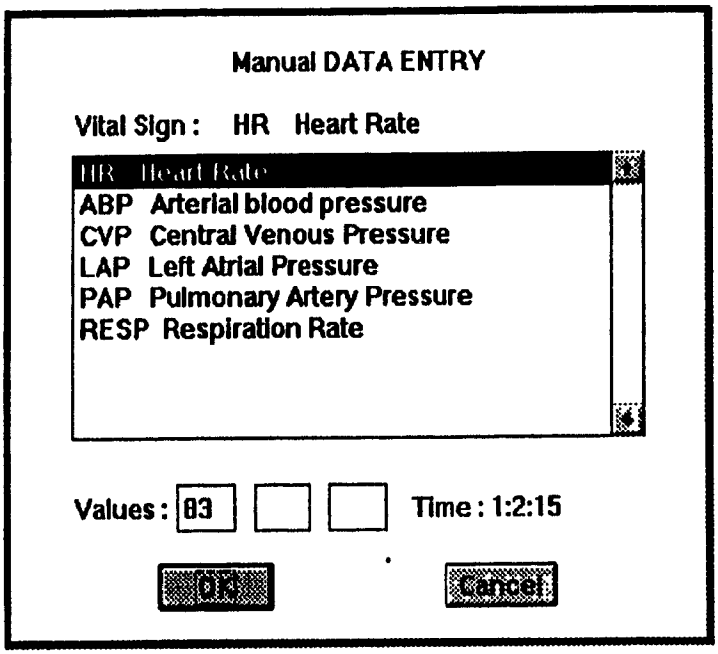

Figure 4.8 Manual Data Entry Dialog Box

for comparison. The redraw rate of the IBM PS/2 model 80 is calculated to be 22.50 ms/graph from Figure 4.10. An IBM PS/2 model 50 has a slower redraw rate of 30.36 ms/graph as would be expected. The best performance was obtained by using an Intel 80486 33MHz microprocessor based personal computer with 14.64 ms/graph. The results obtained for the graphical display is very satisfactory, since the redraw speed for a display with ten vital signs is only 0.22 *seconds* using the IBM PS/2 model 80.

Figure 4.11 shows the redraw speeds of graphs performed by an algorithm using the *GpiPolyLine* function and another algorithm using the *GpiLine* function, both running on the IBM PS/2 model 80. If the *GpiLine* function is used, the redraw rate will slow down from 22.50 ms/graph to 36.54 ms/graph. Therefore, the *GpiPolyLine* provides a 38.4% gain of performance.

The minimum requirement of RAM storage for the memory resident data

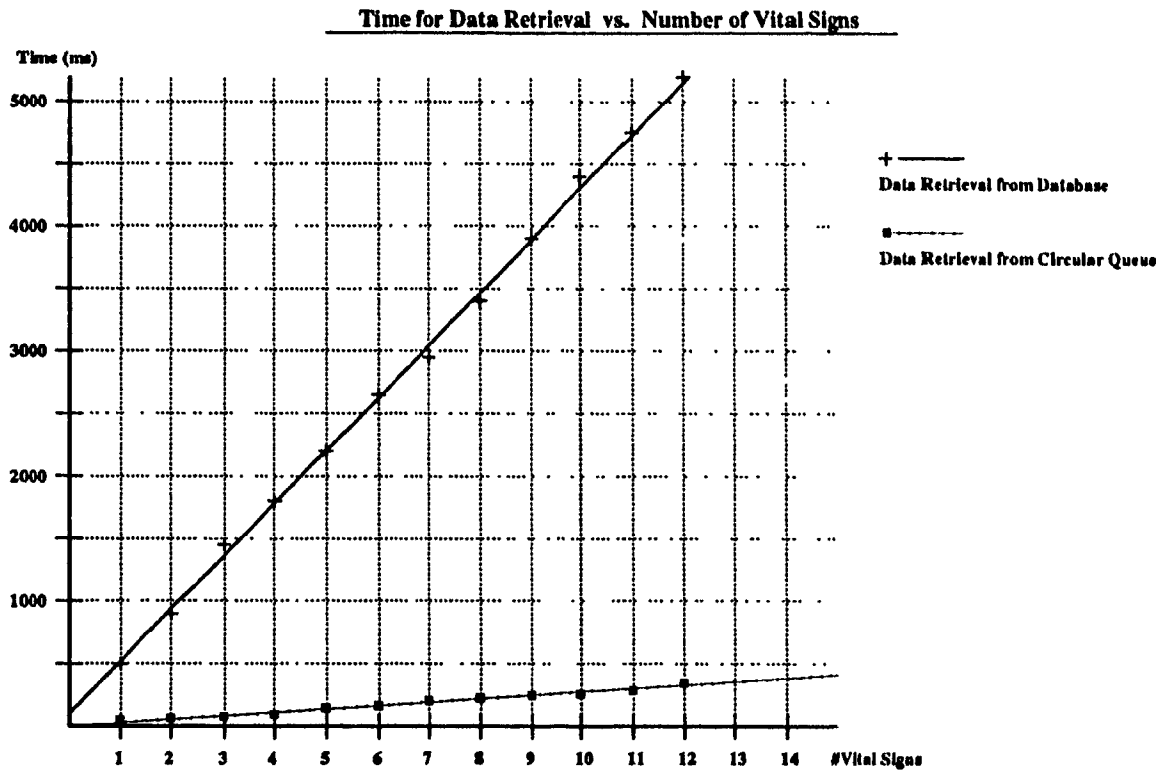

Figure 4.9 Response Time for Data Retrieval

structures is 247 Kbytes. The memory requirement for the display data buffer only is 60.94 Kbytes. The buffer currently can hold data for a maximum of 13 vital signs (single or triple value), with 200 data points each, and support simultaneous monitoring for 4 patients. This provides sufficient and satisfactory performance, since simultaneous display of 13 graphs, or more, is [Fumai, 1992] undesirable because of screen clutter. Simultaneous viewing of four patients is optimal as shown in Figure 4.3 due to the current limitations of the physical display hardware. When more windows need to be opened, the performance tradeoff is only one mouse click to close one of the currently displayed windows. The state of a display window is saved before it is dismissed. Therefore, a display window will re-appear the same as before it was last dismissed. Alternatively, display windows can be partly or totally covered by other windows, when more than four windows are opened at the same time, and it makes no difference to the display time whether the window is

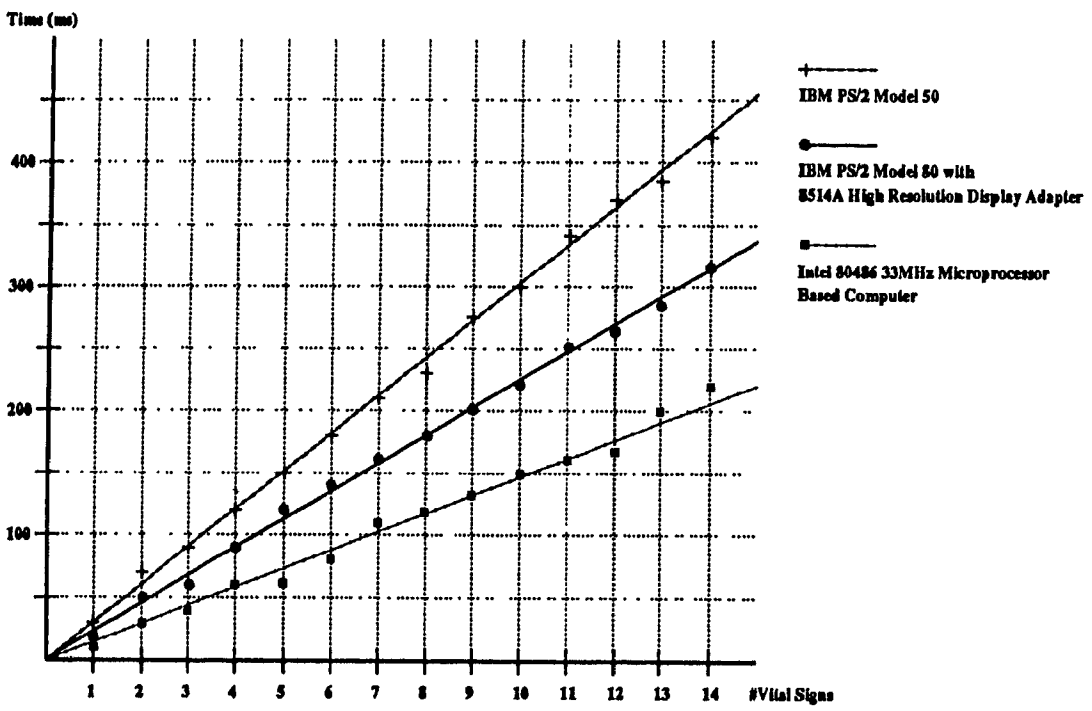

Figure 4.10 Time for Displaying Graphs

closed, or opened but covered by other windows since a covered window does not get displayed.

The memory usage of the patient table for 14 Bed Status Records is 2.83 Kbytes. The second and minutes data queues can each hold 120 data points, with 39 bytes per data point, for 14 patients. Therefore, the second and minute data queues each require 63.98 Kbytes of memory. The half-hour queue holds data for a maximum period of two days (96 data points), for 14 patients, and requires 51.19 Kbytes of memory. The current version of the VSMS has recently been installed for Beta testing in the ICU and the preliminary comments from medical and nursing uses are encouraging. More extensive testing with the integration of updated versions of the other PDMS modules such as the Vital Sign Expert System and the Bedside Workstation will follow in the near future.

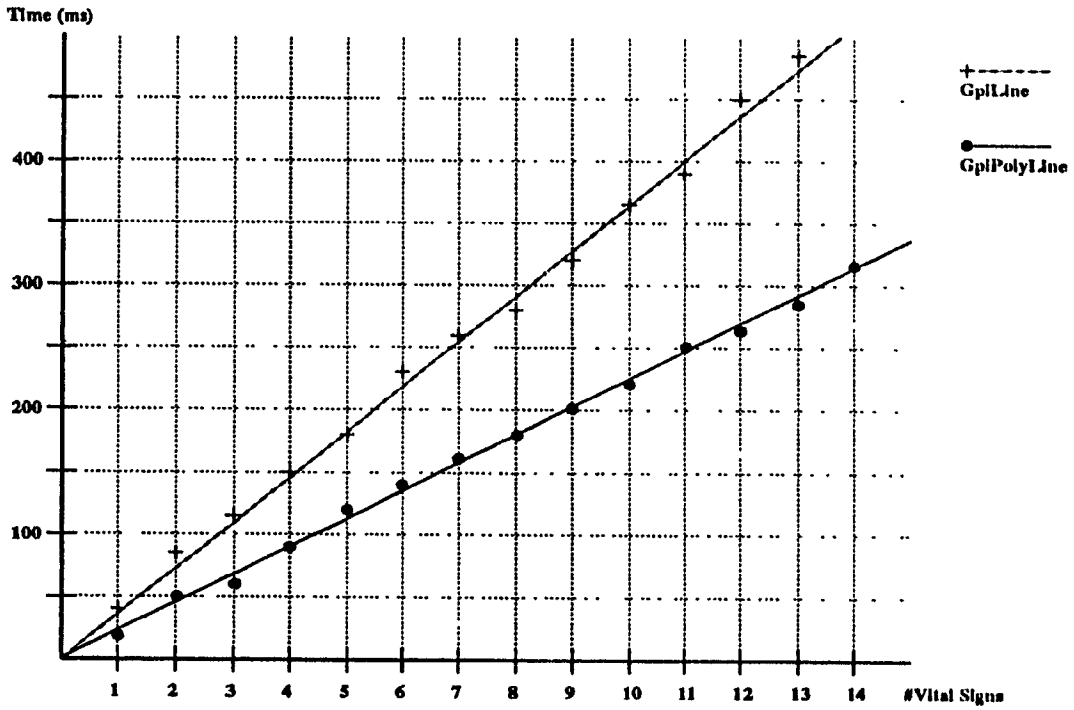

Figure 4.11 Performance Comparison of GpiLine and GpiPolyLine

4.3. Future Extensions

The original PDMS design was for use on a central station, communicating to the CarePort and managing real-time data. With the emergence of bedside terminals, some restructuring of the PDMS design is required. A client-server architecture can be used for the implementation of a distributed database.

The current VSMS software design assumes that the graphical display window is running on the same host computer as the DLC, and thus is able to access real-time data from the shared memory segment. Running the VSMS in its present form on a separate computer on the network can only access the data in the distributed

database for graphical display. However, this current performance of the system is still functional, because the DLC saves the minute data every five minutes, and half-hour every thirty minutes to the PDMS database. Therefore, a maximum of one half-hour data point or five minute data points data can be missing from the display windows on computers other than the host computer. A more comprehensive solution would involve the use of distributed shared memory in the network so that all real-time data would become accessible to all the modules on the network.

The PDMS is currently not linked to the Hospital Information System (HIS) of the Montreal Children's Hospital. Administrative data of patients stored by PDMS is likely to duplicate the data already existing in the HIS. Therefore, it is desirable to support data communication between PDMS and HIS to provide data sharing, and efficient use of computer resources. Additional modules could be integrated into the PDMS such as the ordering of medications from the pharmacy.

The possibility of computer-supported cooperative work (CSCW) is also envisaged for the PDMS. Collaborative computing utilizes computer networking, data communications, concurrent processing, and windowing environments to provide better coordination between parallel processes. CSCW could allow two or more medical staff in different departments to cooperate on a particular task by using a shared workspace on each of their computers [Ishii & Miyake, 1991]. This new collaborative computing capability will also have a significant impact on future software systems.

5. Conclusion

This thesis has presented the design and implementation of a Vital Signs Monitoring System (VSMS) for a Patient Data Management System (PDMS) in the pediatric intensive care unit at the Montreal Children's Hospital. A literature review of recent developments in medical systems, and a survey of current studies in user interface design were first presented. The literature survey was followed by an overview of the hardware and software configurations of the PDMS being developed, and a description of each individual module in PDMS.

The functionality of the VSMS was summarized, and the software structure of the system was discussed. Multi-tasking processes of VSMS were described for patient data acquisition and data management. The graphical interface development using the OS/2 Presentation Manager was then detailed. Sample results were presented to indicate the system's performance, and thus it's suitability for use in the ICU setting. Improvements to the current implementation, and future extensions to the PDMS were also suggested.

Bibliography

[Bailey, 1988] D. Bailey, "Computer Applications in Nursing : A Prototypical Model for Planning Nursing Care", *Computers in Nursing 6(5)* (1988) pp 199-203

[Bailey, 1984] P. Bailey, "Speech Communication: The Problem and Some Solutions", in *Fundamentals of Human-Computer Interaction* (A Monk, ed.), Orlando: Academic Press (1984) pp. 193-220.

[Bertin, 1981] J. Bertin, *Graphics and Graphic Information Processing*, New York: de Gruyter (1981).

[Bertin, 1983] J. Bertin, *Semiology of Graphics*, Madison, WI: University of Wisconsin Press (1983).

[Bishop, 1990] C. W. Bishop, "A new format for the medical record", *M.D. Computing 8(4)* (1990).

[Bongartz, 1988] C. Bongartz, "Computer-Oriented Patient Care : A Comparison of Nurses' Attitudes and Perceptions", *Computers in Nursing 6(5)* (1988) pp. 204-210.

[Bradshaw *et al*, 1989] K. E. Bradshaw, D. F. Sittig, R. M. Gardner, T. A. Pryor, & M. Budd, "Computer-based data entry for nurses in the icu", *M.D. Computing 6* (September-October 1989) pp. 274-280.

[Brown & Krishnamurthy, 1990] P. H. Brown & G. T. Krishnamurthy, "Design and operation of a nuclear medicine picture archiving and communication system", *Semin Nucl Med 20(3)* (1990) pp. 205-224.

[Burkes, 1991] M. Burkes, "Identifying and Relating Nurses' Attitudes Toward Computer Use", *Computers in Nursing 9(5)* (1991) pp. 190-201.

[Chaney *et al*, 1989] R. J. Chaney, F. M. Shipman, & G. A. Gory, "Using hypertext to facilitate information sharing in biomedical research groups", *Proceedings of the 13th Annual Symposium on Computer Applications in Medical Care, Washington, D.C.* (1989) pp. 350-354.

[Chang, 1984] B. L. Chang, "Adoption of innovations: Nursing and computer use", *Computers in Nursing 2* (1984) pp. 229-235.

[Chnadrasekhar & Price, 1989] A. J. Chnadrasekhar & R. N. Price, "Interactive video in medical education", *Proceedings of the 13th Annual Symposium on Computer Applications in Medical Care, Washington, D.C.* (1989) pp. 350-354.

[Collet *et al*, 1989] C Collet, N. Fumai, M. Petroni, A. S. Malowany, J. Panisset, F. A. Carnevale, R. D. Gottesman, & A. Rousseau, "A Patient Data Management System for an intensive care unit", *Proceedings of the IEEE Pacific Rim*

conference on Communications, Computers, and Signal Processing (June 1989) pp. 594-597.

[Collet *et al*, 1990] C. Collet, L. Martini, M. Lovin, N. Fumai, M. Petroni, A. S. Malowany, F. A. Carnevale, R. D Gottesman, & A. Rousseau, "Real Time Trend Analysis for an Intensive Care Unit Patient Data Management System", *Proceedings of the 3rd Annual IEEE Computer-Based Medical Systems Symposium* (June 1990) pp. 337-344.

[Collura *et al*, 1992] T. Collura, E. Jacobs, R. Burgess, & J. Turnbull, "The Epilog System - Automated Long-Term EEG Monitoring for Epilepsy", *Computer 25(9)* (September 1992) pp. 5-14

[Coutaz, 1985] J. Coutaz, "Abstractions for User Interface Design", *Computer 18(9)* (September 1985) pp. 21-34.

[Cushing, 1903] H Cushing, "On routine determination of arterial tension in operating room and clinic", *Boston Medical Surgical Journal 148* (1903) pp. 250.

[Dasta, 1990] J. Dasta, "Computers in critical care: opportunities and challenges", *DICP 24(11)* (1990) pp. 1084-1092.

[EMTEK, 1988] EMTEK, Tempe, AZ, *The EMTEK System 2000: Cost Savings and Benefit Realization* (1988).

[Foley *et al*, 1990] Foley, Van Dam, Feiner, & Hughes, *Computer Graphics : Principles and Practice*, Addison-Wesley (1990).

[Ford, 1990] J. Ford, "Computers and Nursing: Possibilities for Transforming Nursing", *Computers in Nursing 8(4)* (1990) pp. 160-164.

[Fumai *et al*, 1991] N. Fumai, C. Collet, M. Petroni, K. Roger, A. Lam, E. Saab, A. S. Malowany, F. A. Carnevale, & R. D. Gottesman, "Database design of an intensive care unit patient data management system", *Proceedings of the Fourth Annual IEEE Symposium on Computer-Based Medical Systems, Baltimore, Maryland* (May 1991) pp. 236-239.

[Fumai, 1992] N. Fumai, "A Database for an Intensive Care Unit Patient Data Management System", *Master's Thesis, McGill University, Montreal, Quebec* (1992) pp. 32-42.

[Gardner, 1990] R. M. Gardner, "Patient-Monitoring Systems", in *Medical Informatics: Computer Applications in Health Care* (E. H. Shortliffe, & L. E. Perreaault, eds.), Reading, MA: Addison-Wesley (1990) pp. 366-399.

[Gardner *et al*, 1989] R. Gardner, R. Bradshaw, & K. Hollingsworth, "Computerizing the intensive care unit: current status and future opportunities", *Journal of Cardiovascular Nurse 4* (1989) pp. 69-78.

[Goldberg *et al*, 1989] M. Goldberg, J. Robertson, G. Belanger, N. Georganas, J. Mastronardi, S. Cohn-Sfetcu, R. Dillon, & J. Tombaugh, "A multimedia med-

ical communication link between a radiology department and an emergency department", *Journal of Digital Imaging 2* (May 1989) pp. 92-98.

[Greenes & Brinkley, 1990] R. A. Greenes & J F Brinkley, "Radiology Systems", in *Medical Informatics: Computer Applications in Health Care* (E. H Shortliffe, & L. E. Perreaault, eds.), Reading, MA: Addison-Wesley (1990) pp 324-265

[Hammond *et al.*, 1991] J Hammond, H Johnson, R Vanes, & C Ward, "A qualitative comparison of paper flowsheets vs a computer based clinical information system", *Chest 99* (January 1991) pp. 155-157.

[Hartson & Hix, 1989] H. R. Hartson, & D Hix, "Human-Computer Interface Development: Concepts and Systems for Its Management", *ACM Computing Surveys 21(1)* (1989) pp. 7-25.

[Hendrickson & Kovner, 1990] G. Hendrickson, & C. T. Kovner, "Effects of Computers on Nursing Resource Use : Do Computers Save Nurses Time ?", *Computers in Nursing 8(1)* (1990) pp. 16-22.

[Hewlett, 1988] Hewlett Packard, Manual Part No. 78580-42308-6, *Hewlett Packard PDMS/CareNet System: Programmers Reference Guide* (1988)

[Hoadley, 1990] E. D. Hoadley, "Investigating the Effects of Color", *Communications of the ACM 33(2)* (February 1990) pp 120-125.

[Hudson, 1985] L. Hudson, "Monitoring of critically ill patients: Conference summary", *Respiratory Care 30* (1985) pp. 628.

[Hughes, 1988] S. Hughes, "Bedside terminal: Clinicom", *M.D. Computing 5* (January-February 1988) pp. 22-28.

[IBM, 1988] International Business Machines Corporation, Armonk, New York, *Operating System/2 Programming Overview* (1988).

[Ishii & Miyake, 1991] H. Ishii, & N. Miyake, "Toward an Open Shared Workspace: Computer and Video Fusion Approach of Teamworkstation", *Commun. ACM 34 (12)* (1991) pp. 36-50.

[Jacobson *et al*, 1989] S. F. Jacobson, M.E. Holder, & J. F. Dearner, "Computer Anxiety Among Nursing Students, Educators, Staff, and Administrators", *Computers in Nursing 7(6)* (1989) pp. 266-271.

[James *et al*, 1990] J. C. James, N. S. Gatenberg, & G. R. Hageman, "A sample computer system for physiological data acquisition and analysis", *Computers in Biology and Medicine 20(6)* (1990) pp. 407-413.

[Krieger *et al*, 1991] D. Krieger, G. Burk, & R. Sclabassi, "Neuronet - A Distributed Real-Time System for Monitoring Neurophysiologic Function in the Medical Environment", *Computer 24(3)* (March 1991) pp. 45-55.

[Kuhn *et al*, 1990] K. Kuhn, W. Doster, D. Roesner, P. Kottmann, W. Swobodnik, & H. Ditschuneit, "An integrated medical workstation with a multimodal user

interface, knowledge-based user support, and multimedia documents", *Proceedings of the Third Annual IEEE Symposium on Computer-Based Medical Systems, Chapel Hill, NC* (June 1990) pp. 469-476.

[Laborde *et al*, 1989] Laborde, J., Dando, W. A., & Hemmasi, M., "Computer Graphics : A Tool for Decision Making in Nursing", *Computers in Nursing 7(1)* (1989) pp 15-20.

[Lodding, 1983] K N. Lodding, "Iconic Interfacing", *IEEE Computer Graphics and Applications 3(2)* (1983) pp. 11-20.

[Malowany *et al*, 1989] S. Malowany, C. Collet, N. Fumai, M. Petroni, A. S. Malowany, F. A. Carnevale, R D. Gottesman, & A. Rousseau, "Database Aspects of a Patient Data Management System for an ICU", *Proceedings of the Canadian Conference on Electrical and Computer Engineering, Montréal, Québec* (September, 1989) pp. 586-589.

[McDonald & Barnett, 1990] C. J. McDonald & G. O. Barnett, "Medical-Record Systems", in *Medical Informatics: Computer Applications in Health Care* (E. H. Shortliffe, & L. E. Perreault, eds.), Reading, MA : Addison-Wesley (1990) pp. 181-218.

[Milholland, 1986] K. Milholland, "Assessing the impact on nursing practice of CDMS", *International Journal of Clinical Monitoring and Computing 3* (1986) pp. 191-197.

[Milholland, 1988] K. Milholland, "Patient Data Management Systems (PDMS): Computer Technology for Critical Care Nurses", *Computers in Nursing 6(6)* (1988) pp. 237-243.

[Ozbolt *et al*, 1990] J. Ozbolt, I. L. Abraham, & S. Schultz II, "Nursing Information Systems", in *Medical Informatics: Computer Applications in Health Care* (E. H. Shortliffe, & L. E. Perreaault, eds.), Reading, MA: Addison-Wesley (1990) pp. 244-272.

[Paganelli, 1989] B.E. Paganelli, "Criteria for the selection of a bedside information system for acute care units", *Computers in Nursing 7(5)* (1989) pp. 214-221.

[Panisset *et al*, 1989] J. F. Panisset, D. Lambidonis, N. Khoury, S. Malowany, A. S. Malowany, F. A. Carnevale, R. D. Gottesman, & A. Rousseau, "An Intensive Care Unit Patient Data Management System", *Proceedings of Graphics Interface '89, London, Ontario* (June 1989) pp. 275-282.

[Perreault & Wiederhold, 1990] L. E. Perreault & G. Wiederhold, "System Design and Evaluation", in *Medical Informatics* (E. H. Shortliffe & L. E. Perreault, eds.), Reading, MA: Addison-Wesley (1990) pp. 151-178.

[Pesce, 1988] J. Pesce, "Bedside terminal: Medtake", *M.D. Computing 5* (January-February 1988) pp. 16-21.

[Petroni *et al*, 1991] M. Petroni, C. Collet, N. Fumai, K. Roger, F. Groleau, C. Yien A. S. Malowany, F. A. Carnevale, & R. D Gottesman, "An automatic speech recognition system for bedside data entry in an intensive care unit", *Proceedings of the Fourth Annual IEEE Symposium on Computer-Based Medical Systems, Baltimore, MD* (May 1991) pp 358-365.

[Petzold, 1989] C. Petzold, *Programming the OS/2 Presentation Manager*, Microsoft Press (1989).

[Prakash *et al*, 1991] O. Prakash, N. Govindarajan, S. Meiyappan, & K S Sundar, "Workstations for the operating room and critical care", *Proceedings of the 13th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Orlando, FL* (November 1991) pp. 1226-1227

[Price & Chandrasekhar, 1989] R. N. Price & A. J. Chandrasekhar, "Department of medicine local area network: A strategic solution for the 1990's", *Proceedings of the 13th Annual Symposium on Computer Applications in Medical Care, Washington, D.C.* (1989) pp. 462-466.

[Probst, & Rush, 1990] C. L. Probst, & J. Rush, "The Careplan Knowledge Base", *Computers in Nursing, vol.8(5)* (September/October 1990) pp. 206-213.

[Rice, 1991] Rice, J. F., "Display Color Coding: 10 Rules of Thumb", *IEEE Software 8(1)* (January 1991) pp. 86-88.

[Robertson, 1988] P. K. Robertson, "Visualizing Color Gamuts: A User Interface for the Effective Use of Perceptual Color Spaces in Data Displays", *IEEE Computer Graphics and Applications* (September 1988) pp. 50-64.

[Roger *et al*, 1991] K. Roger, C. Yien, C. Collet, N. Fumai, M. Petroni, A. S. Malowany, F. A. Carnevale, R. D. Gottesman, & A. Rousseau, "A Nursing Workload Manager for a Patient Data Management System", *Proceedings of the 13th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (1991) pp. 1371-1372.

[Roger, 1992] K. Roger, "A Nursing Workload Manager for a Patient Data Management System", *Master's Thesis, McGill University, Montreal, Quebec* (1992) pp. 22-33.

[Safran *et al*, 1990] C. Safran, D. Porter, C. D. Rury, F. R. Herrmann, J. Lightfoot, L. H. Underhill, H. L. Bleich, &W. V. Slack, "Clinquery: searching a large clinical database", *M.D. Computing 7(3)* (1990) pp. 144-153.

[Scarpa *et al*, 1992] R. Scarpa, S. C. Smeltzer, & B. Jasion, "Attitudes of Nurses Toward Computerization : A Replication", *Computers in Nursing 10(2)* (1992) pp. 72-80.

[Scherrer, 1990] J. R. Scherrer, "New architectures destined for hospital computer networks opening the medical world to more communication facilities of every kind", *Schweiz Med Wochenschr 120(49)* (1990) pp. 1866-1871.

[Schroeder & Carter, 1989] M. A. Schroeder, & J. Carter, "Development of a Database Management System for an Obstetrics Unit", *Computers in Nursing 7(3)* (1989) pp 112-118.

[Schwirian *et al* 1989] P. M. Schwirian, J. A. Malone, V. J. Stone, B. Nunley, & T. Francisco, "Computers in Nursing Pratice: A Comparision of the Attitudes of Nurses and Nursing Students", *Computers in Nursing 7(4)* (1989) pp. 168-177.

[Shneiderman, 1982] B. Shneiderman, "The Future of Interactive Systems and the Emergence of Direct Manipulation", *Proceedings of the NYU Symposium on User Interfaces* (May 1982) pp. 1-27.

[Shneiderman, 1987] B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Reading, MA: Addison-Wesley (1987).

[Shortliffe & Barnett, 1990] E.H. Shortliffe & G.O. Barnett, "Medical Data: Their Acquisition, Storage, and Use", in *Medical Informatics: Computer Applications in Health Care* (E. H. Shortliffe, & L. E. Perreaault, eds.), Reading, MA: Addison-Wesley (1990) pp. 37-69.

[Smith, 1992] V. J. Smith, "Monitor-based Software Application Programs Meet the Need for Fast, Accurate Information at the Bedside", *Computers in Nursing 10(2)* (1992) pp. 51-59.

[Smith & Svirbely, 1990] J. W. Smith & J. R. Svirbely, "Laboratory Information Systems", in *Medical Informatics: Computer Applications in Health Care* (E. H. Shortliffe, & L. E. Perreaault, eds.), Reading, MA: Addison-Wesley (1990) pp. 273-297.

[Sommerville, 1989] I. Sommerville, *Software Engineering*, Reading, MA: Addison-Wesley (1989).

[Southerton, 1989] A. Southerton, *Advanced OS/2 Presentation Manager Programming*, Addison-Wesley (1989).

[Staggers, 1988] N. Staggers, "Using Computers in Nursing: Documented Benefits and Needed Studies", *Computers in Nursing 6(4)* (1988) pp. 164-170.

[Staggers, 1991] N. Staggers, "Human Factors : The Missing Element in Computer Technology", *Computers in Nursing 9(2)* (1991) pp. 47-48.

[Strikland Jr., 1991] Strikland Jr., "Development of an information system to assist management of critically ill patients", *Proceedings of the Forth Annual IEEE Symposium on Computer-Based Medical Systems, Baltimore, MD* (May 1991) pp. 70-77.

[Subramanian, 1989] S. Subramanian, "OB information management system: A microcomputer solution", *Proceedings in the IEEE Engineering in Medicine and Biology Society, Seattle, WA* (November, 1989) pp. 2005-2006.

[Swezey & Davis, 1983] R. W. Swezey & E. G. Davis, "A Case Study of Human Factors Guidelin s in Computer Graphics", *IEEE Computer Graphics and Applications 3(8)* (November 1983) pp. 21-30.

[Thimbleby, 1990] H. Thimbleby, *User Interface Design*, New York, NY: ACM Press (1990).

[Thomson, 1984] N. Thomson, "Human Memory: Different Stores with Different Characteristics", in *Fundamentals of Human-Computer Interaction* (A Monk, ed.), Orlando: Academic Press (1984) pp. 193-220.

[Tolbert & Partuz, 1977] S. Tolbert, & A. Partuz, "Study shows how computerization affects nursing activities in ICU", *Hospitals, J.A.H A 51* (September, 1977) pp. 79-84.

[van Cott & Kinkade, 1972] H. van Cott, & R Kinkade, "Human Engineering Guide to Equipment Design", *U.S. Government Printing Office, Washington, DC* (1972) pp. .

[Warner, 1981] J. R. Warner, "Principles of Device-Independent Computer Graphics Software", *IEEE Computer Graphics and Applications 1(4)* (October 1981) pp. 85-100.

[Waterworth, 1984] J. Waterworth, "Speech Communication: How to Use It", in *Fundamentals of Human-Computer Interaction* (A. Monk, ed.), Orlando: Academic Press (1984) pp. 193-220.

[Wiederhold & Perreault, 1990] G. Wiederhold & L. E. Perreault, "Hospital Information Systems", in *Medical Informatics: Computer Applications in Health Care* (E. H. Shortliffe, & L. E. Perreaault, eds.), Reading, MA: Addison-Wesley (1990) pp. 219-243.