



Generic system for the acquisition, archiving and analysis of physiological data

C.J. Hall^a & A.S. Holder^b

^aDepartment of Applied Computing and Electronics, Bournemouth University, Bournemouth, UK

^bResearch Monitoring Systems, Wilton, Salisbury, UK

ABSTRACT

This paper describes a generic data acquisition, data archiving and analysis system designed for use in physiological monitoring research projects. Cost is minimised by exploiting the functionality and price-performance of current micro-computer workstations. The system provides researchers with data analysis/visualisation functions driven through a standard Graphical User Interface (GUI).

The system uses Apple™ computers - the low-cost, high function PowerBook (lap top) as a bed side monitoring station and the high-end Quadra with large screen for the subsequent manipulation and study of the data. Specialist hardware has been minimised and reduced to signal conditioning, analogue to digital conversion and safety isolation. This data acquisition unit is slaved to a lap-top which controls sampling/timing functions, data storage to hard disk and real-time graphical display of physiological parameters on its LCD display.

At the end of the monitoring period the data are then networked to a host micro-computer for long-term archiving to an optical disk. Using a large colour screen, visualisation tools allow researchers to navigate through the data looking for significant features. The tools provide assistance via standard feature recognition functions and through software hooks. The system has been designed generically hence the number of channels, sampling rates, analysis functions may be configured with ease.

INTRODUCTION

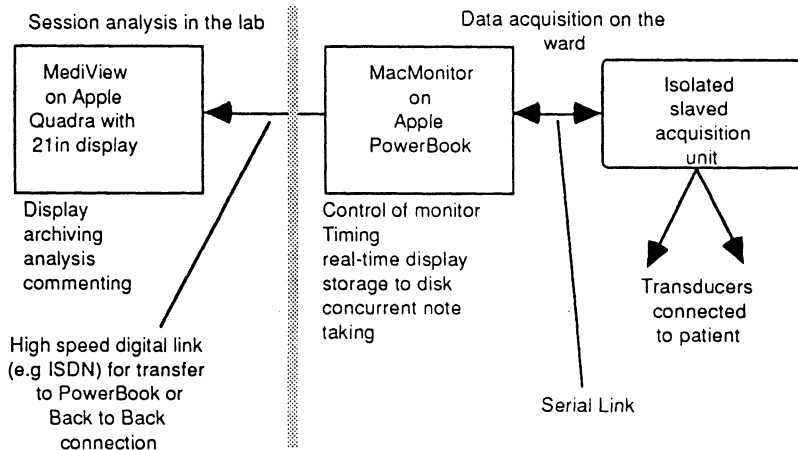
The work described in this paper arose out of a project with an academic paediatrics research unit who were interested in setting up a large scale monitoring project to advance their work on Sudden Infant

Death Syndrome (SIDS). Their need was to conduct overnight monitoring (12 hours) of the majority of babies born in the hospital before they went home. These data would then be archived and viewed by the researchers to identify any events of interest. Essentially the project posed two problems: firstly to produce an unobtrusive and low-cost method of acquiring the overnight recording; and secondly to provide the clinicians with a way of efficiently navigating their way through the mass of data to identify periods of relevance.

It was clear to the authors at the outset that this type of monitoring facility is very commonly required in medical research studies, hence one of the primary design goals of the system was that it should be generic and be capable of application to other studies with minimal change. Any changes which are necessary should ideally be capable of being made by non-computer experts without any detailed knowledge of either the system architecture or programming.

OVERVIEW OF SYSTEM

The two primary goals of flexibility and low-cost led immediately to two design decisions. Firstly to restrict specialist hardware to a minimum using simple, standard and low-cost interfaces to the computers, and secondly to exploit the power and functionality of readily available microcomputers.

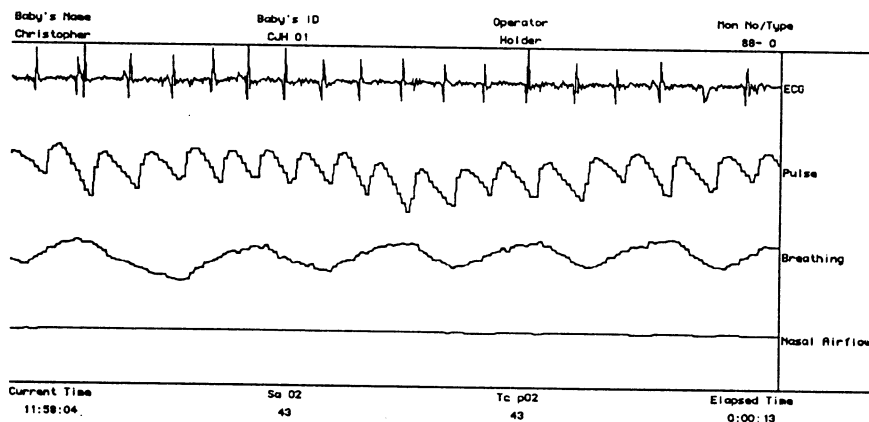


At the Cot Side. At the cot/bed side there is a dual unit consisting of an acquisition unit which is slaved to a lap-top computer. The acquisition unit is of A4 footprint so sits neatly under the lap-top computer. All interaction with the nurse or doctor conducting the monitoring is through the lap-top's screen.

The slave unit contains all signal conditioning, and sampling circuitry, but its timing is subordinate to the lap-top. It also provides all the necessary isolation to comply with current standards. In the initial study six physiological parameters are sampled - ECG, pulse, respiration, airflow, transcutaneous skin-oxygen levels and blood oxygen saturation. These data are communicated on demand to the lap-top via a simple serial link.

The software on the lap-top conforms to the standard Apple GUI guidelines and interacts with the operator. Initially it seeks confirmation of time and date, checks characteristics of the monitoring study and asks for opening information such as patient name/id, doctor's name, and opening notes. It is one of the design features that this information, in addition to notes added dynamically during the monitoring session are inextricably linked with the monitoring data and hence are always available to anyone viewing the data. They cannot be detached nor is there the possibility of other subject's notes being substituted or confused.

Typical screen image captured during monitoring



When the monitoring session is in progress, traces of the channels being acquired are displayed continuously on the LCD display, giving a very useful display of vital signs to the nurse or doctor. Concurrent with the display of these traces the lap-top also controls the slave unit to obtain samples, archives the data to disk and allows the operator to key in additional timed notes indicating events of interest or the administration of medication.

At the close of the session the operator has the opportunity of adding a final note to the recording. The compressed and complete file containing one or many recording sessions complete with attendant notes and session information may then be communicated to the host computer in the laboratory.



The Viewing Workstation. The laboratory workstation used is an Apple Quadra with 21inch colour screen. This computer is equipped with a 128Mbyte exchangeable magneto-optical disk drive for long term archiving of the monitoring session files. At its heart the Quadra has a Motorola 68040 processor so can be equated in power to a 80486DX/66 PC. The viewing software (MediView) is multi-windowed and enables the researcher to view several monitoring sessions simultaneously, allowing easy comparison between cases. MediView has built into it several analysis functions and hooks to allow user defined functions to be added.

Each of the large number of overnight studies needs analysis and comment by the clinicians. They required a way of navigating through the volume of data to identify periods of interest which may last for less than 2 minutes in a 12 hour study. In earlier studies of shorter periods, recordings were output on analogue chart recorders and the clinician flipped through the charts in their search for relevant sections. This could take tens of minutes even with short recordings. In the full scale study this time is not available.

The philosophy behind the design of MediView is to provide the user with a 'big picture' of the data and recorded session. Through this big picture of the total session the researcher can then 'home' in on features or events thought to be of significance. Their comments can then be entered on-line and archived with the full recording on the optical disk. Through the software 'hooks' additional automatic analysis tools may be added to further speed up the task of the clinician.

SYSTEM DESIGN

Earlier sections have given a general description of the system and the philosophy behind its design, this section provides a detailed description of how the flexibility and genericity has been built into the system.

Design objectives

- to minimise hardware complexity, hence cost of slaved acquisition unit
- consistent and standard multi-windowed user interface using well established GUI
- system to be generic allowing for use to be extended to other medical research projects by staff with limited computing or system knowledge.
- system to be extendible with hooks to enable additional analysis features to be added
- to enable concurrency to allow parallel processing of functions and enable foreground/background processing

Although to many the ubiquitous IBM PC or clone with 'C' would have been the obvious choice the authors strongly believes that the Macintosh family possesses a far more powerful and coherent systems architecture and that Ada as a high level language provides a quicker and more effective route to develop quality software. A further implicit objective is therefore

- demonstrate suitability of the Macintosh system architecture and the Ada language for real-time microcomputer projects.

THE SYSTEM ARCHITECTURE

The slaved data acquisition interface. The acquisition unit is modular in design and is built around a low cost Mitsubishi microprocessor control module. This communicates with the controlling computer using a simple protocol on a standard RS422 link. On demand the control module acquires samples from the various signal conditioning and interface modules with which it can be connected. Modularity is at the design level with logical modules being held in library form on an ECAD system. As variations or new versions are required new configurations can be produced very quickly with new PCBs being produced. This logical rather than physical modularity ensures more compact units with lower power consumption and greater reliability due to the elimination of connectors between modules. Although signal conditioning is determined by the hardware, all timing and sampling rates remain under the control of the lap-top hence can be changed very easily.

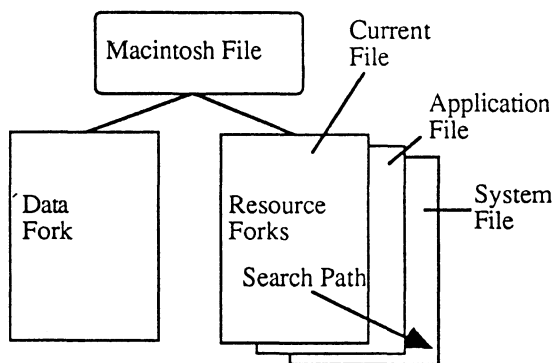
The software. There are two application programs - MacMonitor which resides on the lap-top and performs the data-acquisition, display and archiving functions, and MediView which is located on the main viewing machine which allows the user to view monitoring sessions. The following discussion examines how the objectives have been satisfied

Consistent and standard multi-windowed user interface using well established GUI. All Macintosh computers use a common graphical user interface and all software should conform to a tight set of guidelines as to how user interaction should behave. This interface is implemented mainly at low level within ROM based routines hence achieving a very crisp and totally consistent behaviour. The origins of this GUI can be traced to the original 'Star' project at the Palo Alto labs in the 70s. This consistency ensures a common look and feel to all applications minimising the learning curve of users. The clinicians in this project were already used to using a variety of Macintosh applications.

System to be generic allowing for use to be extended to other medical research projects by staff with limited computing or system knowledge. To achieve this objective requires the simple philosophy to be adopted when designing the software of logically separating the control structures and operations from the objects on which they operate. Essentially this means

not embedding within the code of the program any definitions or structures which it will operate on. If this can be achieved change to the objects can be performed in isolation of the logic which performs the operations. Although simple in concept this is by no means easy to achieve and is a subject of much recent interest in the current and topical debate on object oriented design. In this system this is achieved partly by using the generics facility within the language Ada, but mainly through exploiting the object oriented nature of the Macintosh System architecture.

Again arising out of the early work at Palo Alto the Macintosh from the outset encouraged the logical separation of objects from the control structures. In the Macintosh environment these objects and the associated methods are called resources.



Any Macintosh file (above) has two parts known as the data and resource forks. For a typical data file the resource fork is empty whilst for a typical application file the data fork is empty. Resources are basically the data objects such as windows, strings of text, images, graphical control buttons which an application manipulates and the 'methods' or code procedures which operate on them. When a resource (either an object or a method) is specified by the running application the operating system searches the current file's resource fork first, if one is not found there it then searches the applications own resource fork and finally it resorts to the system's own resource fork. This order establishes an important principle that if an alternative resource is not provided by the user or application the system default will be used. An example - The application creates a new window, the windows size, colour shape will be specified in a the resource fork of the application. The code or 'method' governing the way the window will be drawn is left to the system's own default method in the system's resource fork. If out of perversity we wanted an unusual behaviour for a window we could develop our own method and add this as a resource in the application or data file resource fork. This would then override the system's default.

Resources can either be variations of existing system defined resources or user defined. In the case of MacMonitor and MediView two additional sets are created over and above the standard. The first contains a specification of sampling rates, channel names, display characteristics, scaling factors etc. for each of the variations of monitoring studies. As new variations are introduced the only change required is the addition of new versions of this basic resource. A great benefit of this technique is that these resources may be created with very limited computer knowledge. A utility program called a resource editor can be used against a template to allow technicians to create a new version or modify an existing one.

The same application defined resource technique is used for the addition of session information and notes added dynamically during a recording. These are added to the data file as 'note' resources and form a linked list. They become an integral part of the file and MediView allows the user to step through these notes moving to points of interest in the recording.

System to be extendible with hooks to enable additional analysis features to be added. The concept of resources is used again. Default analysis methods are incorporated into the application as standard resources. For example detection of the QRS complex in the determination of beat-to-beat heart rate variation from the ECG signal, or the filtering and zero crossing detection of breathing movement to determine respiration rate. Alternative or additional methods may be added later by the addition or substitution of resources.

Enable concurrency to allow parallel processing of functions and enable foreground/background processing. Ada was originally designed by the DOD in the US for military large real-time systems so in a sense its use in this application is a 'swords to plough-shares' operation. The nature of the language and its use in this application is outside the scope of this paper other than mentioning two features.

Firstly the language is designed at the outset for concurrent time-critical applications and provides a multi-tasking and inter-task communication facility termed the 'rendezvous'. In MediView each window related to a session being viewed is designed as an autonomous task which enables many analysis functions to be performed in parallel with the human interaction. This background processing allows many analysis functions to be performed with less wait time by the user.

A second feature of the language is its strong typing and provision of generics. They allow the reuse of software modules in a flexible manner. The strong typing results in a less forgiving compiler than C (an understatement) but it invariably means a quicker route to a working and reliable quality piece of software.



EVALUATION AND FUTURE DEVELOPMENTS

The system is now in operation in β form within the original SIDS monitoring project and trials are underway to validate its performance prior to switching completely from the original analogue recording method. This switch is anticipated very shortly.

Several other variations of the basic system are now being introduced for separate studies within the original research group. Versions are also being introduced into other projects in Jamaica in the W. Indies and Hanover in Germany. Many of these subsequent studies are not connected with SIDS and involve monitoring other sets of physiological parameters with different sampling rates and characteristics. The ease with which these studies have been accommodated demonstrates the success of the original design strategy. Very little change has been necessary to the original design of the software, the majority being accomplished through modifying or adding new resources.

The success of the project has demonstrated the suitability of the Macintosh system architecture to support applications of this type. The software now extends to the equivalent of 15,000 lines of Ada and although no detailed quantitative analysis has been performed to date, the project confirms Ada to be an effective language for system implementation. The language itself is large and hence the learning curve is longer than for 'C', however limited comparison of equivalent run-times and code sizes for this implementation show very little difference. Where Ada does score very heavily is in allowing a more robust and logical expression of a problem solution and in speeding up the time to produce a working and maintainable piece of software.

This first stage of the project has delivered to the clinical researchers the ability to record and view large quantities of time-series data relatively quickly, the next stage of the project will be adding intelligent analysis features to further reduce the time taken to view recordings.

BIBLIOGRAPHY

1. Apple Computer Inc. *Technical Introduction to the Macintosh Family*. Addison Wesley, 1987
2. ANS *Reference manual for the Ada programming language*. American national standards institute. 1983
3. Barnes J.G.P. *Programming in Ada* Addison-Wesley 1989
4. Tomkins, W.J. *Biomedical Digital Signal Processing*. Prentice-Hall 1993