A GENERIC TELEMETRY HOST COMPUTER SOFTWARE ARCHITECTURE

Robert K. Buell
Manager New Programs and Design Technology
Fairchild Weston Systems, Inc.
Data Systems Division
P.O. Box 3041
Sarasota, Florida 33578

ABSTRACT

A generic software architecture has been developed to support the typical functionality required of the host computer in a telemetry ground station system. The architecture provides sufficient flexibility to permit support of the wide spectrum of requirements typically placed on such systems, while at the same time providing a structural shell which helps to minimize the complexity of applications software. The general issues addressed by this architecture include:

- The need to interface to a wide variety of telemetry front end equipments.

- The need to provide a convenient consistent, and efficient operator interface to the integrated telemetry system.

- The need to support a variable amount and wide range of applications specific processing.

- The need to be adaptable across different sizes of host computers.

- The need to be adaptable across different host computer systems.

This paper defines, at a high level, the architecture that has been defined and the general data structure concepts required to make it work. It further addresses the standardized operator interface supported by the architecture and finally, summarizes the benefits that have been demonstrated to be derived through the use of this standardized approach in the development of telemetry host computer software.

INTRODUCTION

The Data Systems Division of Fairchild Weston, is a major supplier of turnkey telemetry data acquisition and analysis systems. These systems span a large range of capabilities, ranging from the acquisition of data from a single input PCM data link to the simultaneous acquisition of data from multiple PCM, FM, and 1553 data links, and the subsequent playback and analysis of the acquired data by multiple simultaneous users.

In order to be able to efficiently produce systems with such a wide variety of capabilities, and recognizing that the cost of such systems is being driven more and more by the cost of the software in them, it became obvious that a generic telemetry ground station software system architecture would have to be developed. This architecture would be required to support the acquisition of data from a variety of telemetry front end configurations, would be required to be adaptable to a wide range of sizes of computers or computer systems, would be required to support the inclusion of application specific software on a project by project basis, and would be required to provide a consistent software environment across systems in order to minimize future efforts to be expended in the maintenance of ground station software. Such an architecture has been developed and has proven to be an effective aid in the design development, and maintenance of custom telemetry ground stations.

SYSTEM REQUIREMENTS

The basic functionality provided by a telemetry system ground station typically will include some subset of items from the following list:

- Control the acquisition of data into the ground station.
- Record the acquired data, or some subset of that data, to a mass storage device.
- Compress the acquired data by application of user specified algorithms.
- Perform engineering units conversions on the acquired data.
- Perform limit checking on the data being acquired and notify the users of limit violations in realtime.
- Perform user specified processing on the acquired data and make the results available for display.
- Display the data being acquired to one or more users in realtime.
- Display, in post-realtime, data that has been previously acquired to mass storage.

The above list of items can be broken down into four general categories; the acquisition, processing, recording, and display of data. Each of these four categories then defines a functional requirement which must be able to be satisfied by a system if that particular requirement exists in the application.

In order to facilitate the design and development of modular software to implement the system functional requirements, four interfaces have been defined which provide the vehicle for software in each functional area to communicate with software in other functional areas. The interfaces are: the recording data buffers, the display data buffers, the recorded data files, and the operator interface. Figure 1 illustrates the relationships between each of the major processing functions and the defined interfaces. It is the definition of these interfaces, and the associated functional breakup of the software that defines the ground station system software architecture.

Data Acquisition Functional Requirements

The data acquisition function has three principal requirements; the acquisition of data from one or more input data channels into memory, the control of the data acquisition process, and the monitoring and reporting of the current status of the data acquisition process. In a typical large system, data is acquired simultaneously through two data channels. The data acquired through the first channel is routed to the recording data buffers and data acquired through the second channel is routed to the display data buffer. The specific sets of data acquired through each channel may be identical or different depending upon application requirements. Additionally, since the channels operate independently, any number of input channels from one to n may be implemented to acquire data simultaneously, depending upon application specific requirements. It is important to note that the data acquisition software has no requirement to have any knowledge of the content or format of the data being acquired through any of the input data channels. It’s primary functional requirement is to move data from an input data channel into predefined buffers in the ground station computer memory.

Data Recording Functional Requirements

The data recording function has only one principal requirement; the transfer of data from the data recording buffers to disk or magnetic tape. In performing this transfer the data recording software is responsible for implementing the recording file format. Just as in the case of the data acquisition software, the data recording software does not require any knowledge of the content or format of the data being recorded. It’s sole requirement is to move data from buffers in the ground station computer memory to a mass storage device. The data recording function is responsible for performing all device specific initialization

and general housekeeping associated with the opening and closing of files and the recording of multi-volume data sets.

Data Processing Functional Requirements

The requirements to be satisfied by the data processing function are variable from system to system, and depend not only upon application specific requirements, but also upon the particular complement of hardware present in the system.

One set of requirements is imposed when a data preprocessor is not present in the telemetry system. When present in a telemetry system, a preprocessor will typically perform engineering unit conversions on the incoming data samples and tag each sample with an identifying value. That tag may subsequently be used to specify the specific memory location that engineering unit converted value is to be written to in the display data buffer. If a preprocessor is not available in a particular system, software must be provided to decommutate the data in realtime, perform engineering unit conversions as required, and move the engineering unit converted data into the appropriate memory locations in the display data buffer. This software must in fact emulate some of the functions that would have been provided by the missing preprocessor. The advantage of following this approach is to permit the structure of the display data buffer to remain fixed regardless of the telemetry hardware complement in the system.

A second set of requirements to be addressed are those imposed by the specific application. Typically this will involve software to perform realtime derived parameter calculations, where user applied algorithms are applied to the incoming data samples and the resultant data values are made available for display. Data to support these calculations is typically read from the display data buffer since it’s structure makes it very easy to rapidly locate the current data value associated with any specific parameter. The recording data buffer can be used as a source of information however if system timing and the structure of the data in that buffer permit. In this case the data samples are read from that buffer before it is written to disk or tape. Other types of application specific software may also be easily incorporated into a system utilizing these same interfaces.

Data Display Functional Requirements

The requirements to be satisfied by the data display function also fall into two categories; those supporting the display of data during realtime, and those supporting the display of data in post-realtime. In the realtime environment all data to be displayed is obtained from the data display buffer which is maintained either by direct data acquisition from a preprocessor or by software emulation of the preprocessor. Regardless of how the display buffer is maintained, the realtime display software can expect to find the most recently

processed value for any parameter of interest in a preassigned location in that buffer. In addition to the most recent value, information is also found in the buffer pertaining to limit violations which have occurred for each parameter. The display software can therefore access current value and limit violation information for all parameters of interest from the display buffer and can run asynchronously with the actual data acquisition software. The processing associated with updating multiple displays for multiple users will therefore not interfere with the data acquisition process as long as the data acquisition process runs at a higher priority than the display software.

In the post-realtime environment all data to be displayed is obtained from the recorded data files which have been previously written by the data recording software. At this point the format of the data within the data area of the recording buffers does become important because the display software will have to interpret that data in order to display it.

SOFTWARE SYSTEM INTERFACES

There are four fixed interface definitions which provide the glue that holds the entire system architecture together. Three of these interface definitions are the definitions of data structures which serve as conduits through which data may move from one functional processing set of software to the next. Each of these interfaces serves as a barrier which is used to enforce the modularity imposed by the system architecture. In all cases a set of functional processing on one side of the interface writes data into the data structure while the complementary set of functional processing on the other side of the interface reads data from the data structure. The crucial point in the design is that, in realtime, the reading and writing of data through the interface data structure is performed totally asychronously, and thus the two sets of functional software cannot adversely impact each other in the sense of generating wait states or race conditions which might degrade realtime performance. The fourth interface is a definition of how the system user will communicate with the system software. While not as crucial to the actual operation of the system as the other three interface definitions, this interface is nevertheless important in that it makes it possible for the system users to interface with several software elements which actually run independently and yet perceive those elements to be part of an integrated system.

Recording Data Buffers

The recording data buffers provide the data path between the data acquisition function and the data recording function. In some cases they may also provide a data path between the data acquisition function and the data processing function. When this situation occurs there are usually multiple recording buffer areas such that the recording function can read data from one set of buffers and the processing function can read data from a different set of buffers, thus preserving the isolation between the functional processing areas. Figure 2

illustrates the format of a recording data buffer. It consists of three areas; the preface, data area, and appendix. The preface is comprised of a fixed number of words which contain information that will subsequently be useful when the data is read back from mass storage. Buffer count, word counts, and status words are the kind of information typically included in the preface. The data area of the buffer is a fixed length for any given data acquisition run, but may be adjusted from run to run. This feature allows the user to adjust the overall size of the recording buffer for purposes of maximizing data throughput to the recording media. It is important to note that the format of the data within the data area of the buffer is not constrained and may therefore be easily adjusted to respond to application specific requirements. In one instance the data may represent tagged engineering unit converted data while in another it may be raw position correlated data as provided by a frame synchronizer. Finally, the appendix area is a fixed length for any given data acquisition run but may be variable from run to run. This area is similar in purpose to the preface except that it will normally be unused. It exists to provide a mechanism for the user to insert application specific information into the recorded data stream.

Display Data Buffer

The display data buffer provides the data path from the data acquisition and data processing functions, both of which may write into the display buffer, to the realtime data display function, which reads from the buffer. Figure 3 illustrates the structure of the data display buffer, also frequently referred to as a Current Value Table. This buffer can be thought of as a two dimensional array in memory which typically contains the most recent value of all parameters of interest, and a current count of the upper and lower limit violations that have occurred for that parameter. The array is structured such that the index into one dimension of the array is the tag that is associated with each parameter known to the system, and the index into the other dimension of the array defines the specific type of data found in that location. Thus a current value or a current limit violation count for any specific parameter will always be found in a fixed location in the array for any given data acquisition run. That location may be changed from run to run if required by merely redefining the tag associated with the parameter. It is important to note that from the point of view of a generic software architecture, the size, shape, and contents of the display buffer are unimportant. It is the concept of being able to always locate a specific piece of data in a known memory location during realtime that is important.

Recorded Data Files

The recorded data files provide the interface between the realtime data acquisition and data recording functions and the post-realtime data display function. They will typically be formatted according to the standard file system on the ground station computer, and their data content will include a single header record followed by a long series of recording data

buffers recorded end to end. The header record normally contains information used to identify the particular data acquisition run which created the data, and any other information which may be useful to analysts processing the data in post-realtime. The use of a standard file system is important in this application as it permits the data files to be generated on tape or disk, and further permits those data files to be easily copied from one medium to another.

Common User Interface

The generic software architecture that has been described is centered around the concept that the major functions in a telemetry ground station system can be modularized and isolated from each other by a set of predefined interfaces between those functions. A software system implemented under this architecture will therefore necessarily generate several different software elements all of which must communicate with the system users but which are prohibited by the architectural constraints from using common code to implement that communication. It is therefore necessary to define a common user interface such that the users of the system will perceive it to be an integrated entity, instead of a collection of independent but related functional processes. Once again, from the point of view of the generic architecture, it is not the implementation level details of this interface that are important. The important issue is that such an interface is defined and adhered to by the system designers.

SUMMARY

The generic system architecture presented in this paper offers system designers a high level of functional modularity and design flexibility. It accommodates the design and implementation of systems supporting a minimal set of functional processing requirements up through systems supporting a very complete set of functional processing requirements. The architecture does not make use of any of the unique features of various computer architectures and thus can be implemented on any of those architectures.

The architecture as described herein has been used by the Data Systems Division of Fairchild Weston in the implementation of several telemetry system ground stations over the past two years. It has proven to be adaptable to systems running on a wide range of DEC VAX processors, ranging from the VAX 11/730 at the small end to multi-computer clusters of VAX 11/785s at the high end, and has easily accommodated the inclusion of a wide range of system specific applications processing functions into the various delivered systems.

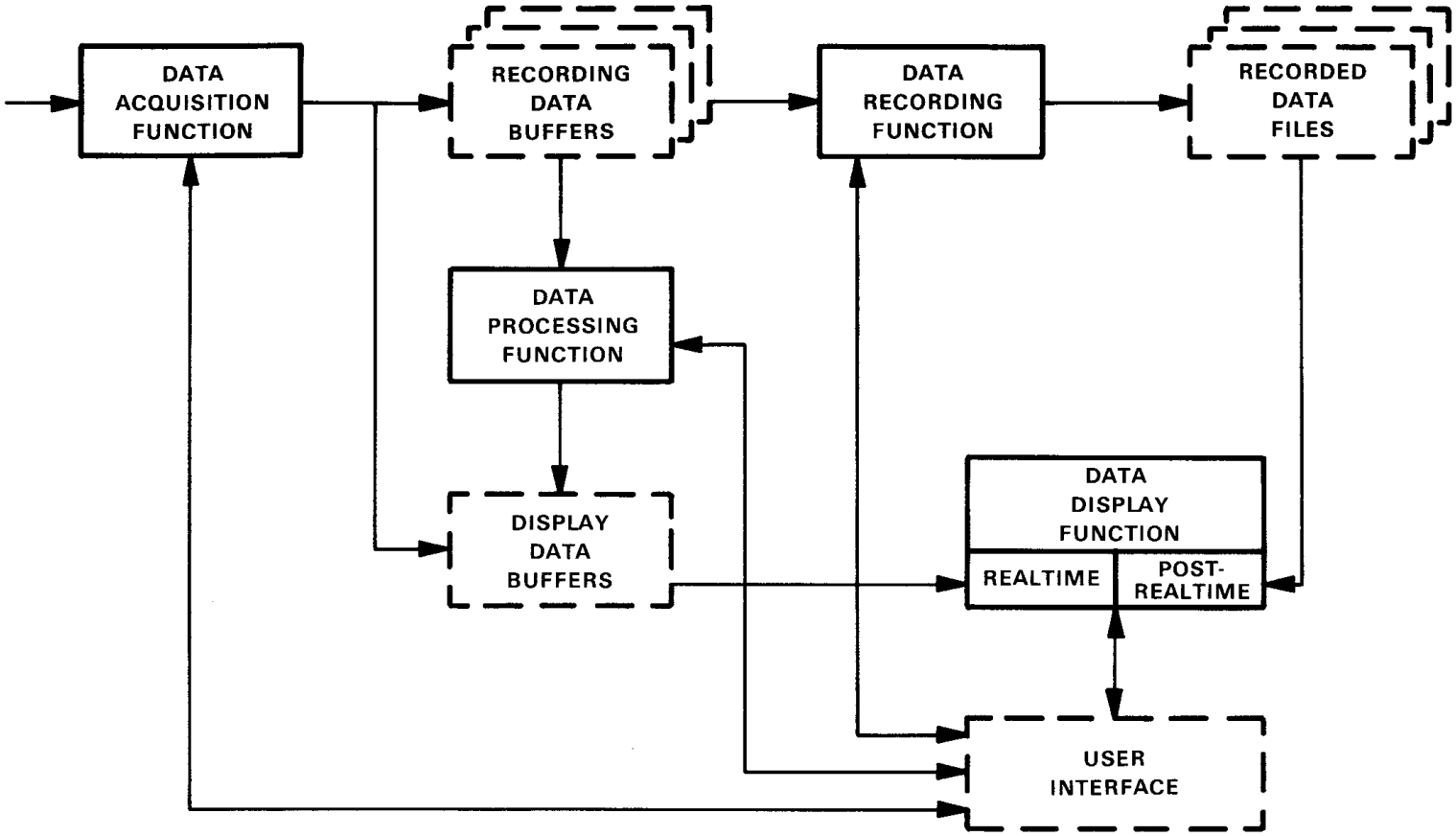

FIGURE 1
TYPICAL TELEMETRY GROUND STATION
DATA FLOW

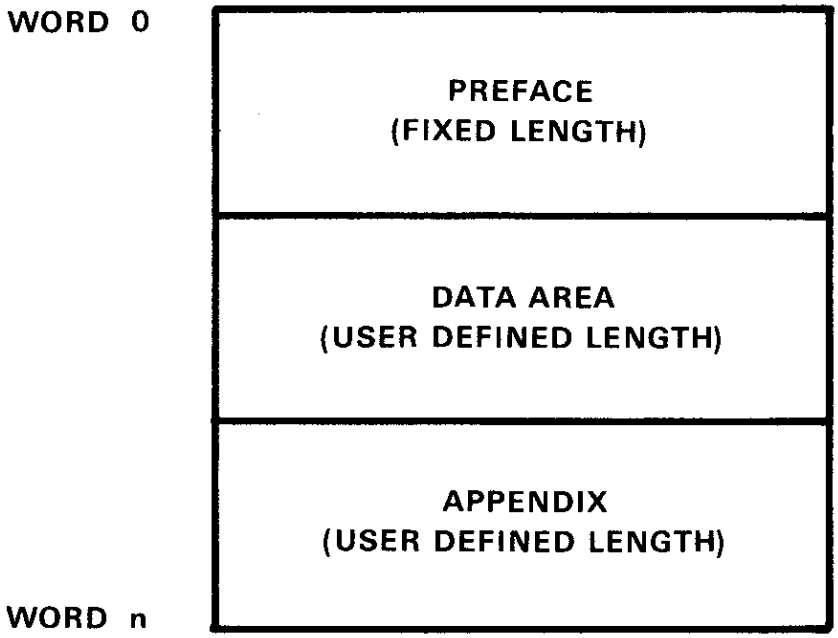

FIGURE 2
RECORDING DATA BUFFER FORMAT

TAG 0	TAG 1	• • •	TAG n
CURRENT VALUE	CURRENT VALUE	• • •	CURRENT VALUE
UPPER LIMIT VIOLATION COUNT	UPPER LIMIT VIOLATION COUNT	• • •	UPPER LIMIT VIOLATION COUNT
LOWER LIMIT VIOLATION COUNT	LOWER LIMIT VIOLATION COUNT	• • •	LOWER LIMIT VIOLATION COUNT
SPARE	SPARE	• • •	SPARE

FIGURE 3
DISPLAY DATA BUFFER FORMAT