TELEMETRY GROUND SYSTEM DEVELOPMENT VIA THE COMPONENT APPROACH

N94-23949

Karen E. Thorn
Software and Automation Systems Branch, Code 522
Mission Operations and Data Systems Directorate
NASA/Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

NASA's deployment of major space projects such as the Earth Observing System (EOS) will demand increased functionality and ground-based telemetry processing performance well above current capabilities. At the NASA/Goddard Space Flight Center, custom hardware and software components have been developed and combined into a unique architecture to address this problem. The hardware components utilize Application Specific Integrated Circuits (ASICs) developed specifically to support NASA's telemetry data systems needs and designed to handle data rates up to 300 Mbps. A generalized set of software components, called the Telemetry Processing Control Environment facilitate the rapid construction of control and monitoring functions for the ground-based telemetry processing systems. This combination of hardware and software elements enables rapid construction of flexible, cost-effective telemetry processing systems capable of meeting the performance requirements facing NASA in the coming decade.

Key Words: Object-Oriented, VLSI, CCSDS, Telemetry, XWindows, C++

1. INTRODUCTION

The goal of today's spacecraft telemetry processing system developers is similar to the goal of early television pioneers. Technology has provided a revolutionary system for transmission of information. The goal is the very broad distribution of this technology to information providers and consumers. Three elements are required to meet this goal; they are the:

- development and adoption of standards to provide a framework of commonalty
- development of engineering capabilities that meet these standards, and
- development of low cost, easy to use, reliable products that apply these engineering advances.

Development of standards in the area of space telemetry systems is well underway, meeting the first of these three elements. The second element is being met through recent advancements in hardware and software technology. The final element is being met through the development of reliable products. This paper discusses the generic hardware components and the software components developed for ground-based telemetry processing system monitoring and control. The paper describes the telemetry data, stream ground-based processing, the problem incurred as telemetry processing requirements change, the component approach proposed to solve this problem, and future direction of the work in this area.

2. TELEMETRY DATA STREAM

Telemetry data stream processing can be broken into two elements-- execution, subdivided into pre- and post-execution, and data transfer. Pre-execution begins as spacecraft orbit and attitude data are provided to the Payload Operations Control Center (POCC). This data and the spacecraft's mission objectives are used at the POCC to produce command loads, sets of machine codes for operating the spacecraft. Post-execution, the science and engineering data collected and returned to the ground is processed and transmitted to scientists and engineers. Both pre- and post-execution data transfer is handled by an entity called the Space Ground Network. The Space Ground Network uses both ground-based communication links and domestic satellites for data transfer.

3. THE PROBLEM

The technology employed by today's Space Ground Network cannot handle the data rate and throughput requirements of tomorrow's complex missions. Many current ground-based telemetry processing systems are mini-computer-based with telemetry processing performed in software. This approach constrains data rate and throughput. The problem imposed by limited processing power can be illustrated by considering the data rate and throughput requirements of future missions that incorporate imaging technology. Very large volumes of data will be generated at very high rates. Both will quickly overwhelm current telemetry processing systems.

These functional and performance requirements are driving technology and development of future processing systems employing fast hardware components and embedded telemetry processing.

4. THE SOLUTION

4.1 HARDWARE

As part of an effort to reduce the costs of NASA ground-based data processing systems, a suite of generic hardware and software components has been developed by the Microelectronics Branch at GSFC to capture, process, and distribute spacecraft telemetry. In this effort, a functional component approach was adopted. The philosophy underlying this approach is to combine basic hardware elements into larger functional components that are easily configured into a low-cost, high-performance, high-reliability telemetry processing system. Two key elements supporting this approach are Very Large Scale Integrated (VLSI) circuits, and an advanced real-time software environment. These elements utilize telemetry distribution standards, such as Consultative Committee for Space Data Systems (CCSDS) standards, to build generic telemetry processors and in so doing, significantly reduce costs to only those incurred by hardware replication.

The VLSI circuits are the backbone of the functional component approach to developing low-cost, high performance telemetry processing systems. VLSI technology enables the processing systems to handle very high data rates, up to 20 Mbps, with future developments raising that rate to 300 Mbps. The VLSI Level Zero Processing (LZP) Prototype System is an example of the application of this approach to telemetry processing system development. The LZP architecture uses a functional components approach based on the Versa Module Eurocard Bus (VMEbus) with multiple microprocessors running concurrently. Telemetry data flows through a custom built pipeline where parallel processing supports the very high data rates.

Tightly coupled with the VLSI-based hardware is state-of-the-art real-time software technology. Together, these technologies allow the developers to design and implement highly functional, flexible data systems. Modular Environment for Data Systems (MEDS) is an example of an environment that helps to design data systems on a standard hardware platform. MEDS supports the basic software functions needed in all telemetry processing systems:

- setup application specific hardware and software,
- process telemetry data based on setup parameters,
- monitor the processing, and
- supply network support for remote operator interfaces and data transfer.

The infrastructure, provided by the real-time MEDS environment and the hardware allows timely design and construction of general purpose telemetry processing data systems. A control and monitoring environment for these generalized systems is easily implemented with a mapping between each hardware component and a corresponding control and monitoring software component. This control and monitoring is layered on the messaging scheme employed by all hardware components. In addition, a set of general purpose commands can be interpreted and executed by each custom designed component. (See Figure 1.1)

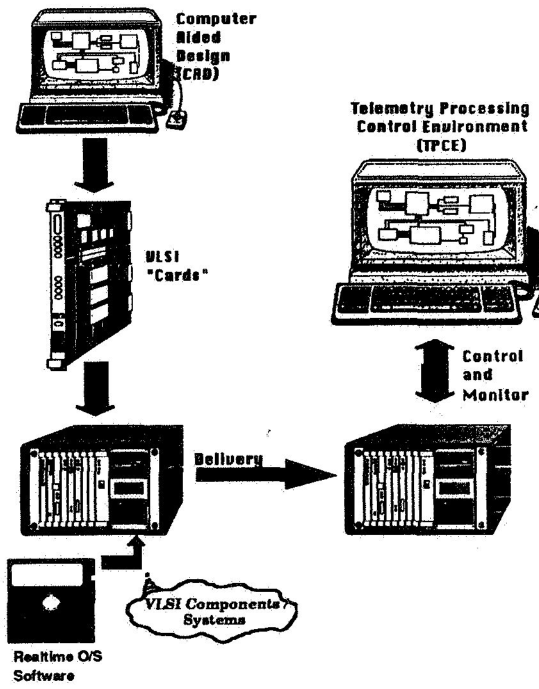

Figure 1.1 Building a Telemetry Processing System

4.2 SOFTWARE

The Telemetry Processing Control Environment (TPCE) provides a generalized suite of functions for control and monitoring. This suite allows the user to define processing actions to be taken by the hardware, to edit and store sets of hardware control information, to graphically and dynamically display telemetry processing status, to create and manipulate directories and catalogs of processing control parameters, to command the hardware, to establish and manage network communications, and to manage the TPCE application itself. All communications between the TPCE and the MEDS-based hardware systems are done through the MEDS-defined protocol.

The TPCE was designed using object-oriented methodology and the software was implemented using C++, an object-oriented extension to the C programming language. The TPCE platform is a color graphics workstation based on the X Windows System and OSF/Motif. In selecting the platform, emphasis was placed on system usability and there was no requirement for real-time operating system environment. The user interface component of the TPCE system was generated with a User Interface Management System (UIMS) developed at NASA/GSFC called the Transportable Applications Environment Plus (TAE+).

The TPCE software allows the user access to any of eight TPCE subsystems described in the following paragraphs.

4.2.1 Communications

The communications subsystem is used to establish information exchange links between the workstation and the data system hardware. The communication package allows concurrent connection to multiple MEDS-based processing systems and manages the transmission of MEDS messages. The package uses TCP/IP sockets with network event notification provided by TAE+.

4.2.2 Subsystem Editor

The catalog editor allows users to specify complete or partial hardware configuration sets and to store those sets at the local workstation for later execution. Each configuration set, called a catalog, contains the specific setup definitions for each card to be configured.

4.3.2 Monitoring

The status monitoring subsystem handles the display of status information to the user. Status information for each of the hardware cards is gathered and displayed in a textual form. Future upgrades to the status monitoring capabilities will include graphical displays. The update rate for status display can be set by the user.

4.2.4 Control

The control subsystem allows the user to send commands to the hardware cards. This subsystem handles initialization, initiation, execution, and termination of a data system session. Typical commands load setup data for a card, or enable a card using the setup data currently loaded, or reset data accounting statistics.

4.2.5 Event Log

The operator log provides a time-sequenced account of the actions taken by the user, the subsequent errors or warnings encountered, and any corrective measures that were taken in response to problems. Print and archival facilities for the log are also available. Developers and integrators of the data system can verify system configuration and monitor the installation, integration and execution.

4.2.6 User Preferences

The user preferences manager allows the user to tailor the interface to his/her satisfaction. Display panels can be rearranged via a window manager and foreground and background color schemes can be modified. The user has full control of the environment with which he/she is working.

4.2.7 Real-time Service

The real-time data subsystem allows the user to obtain Quick-look processed data, as well as real-time data packets, as the data is being processed through the system. All data is gathered and saved on the local workstation for future playback. Data accounting statistics, such as total packets with good quality data or with errors, are gathered and displayed to the user as the real-time data packets are processed.

4.2.8 Network Configuration

A network manager is the mechanism through which the network of MEDS-based systems is configured, i.e., the locations of specific MEDS-based system. The manager also monitors network activity, reporting problems to the TPCE for resolution, and allows the user to change the network configuration.

5. TYPICAL OPERATIONS SCENARIO

A typical scenario for TPCE system operation covers configuring, commanding, and monitoring the VLSI hardware. A TPCE user determines the type of information to be processed through the system, and configures the system hardware to process a specific type of information as required. The hardware components are configured using the specifications given in the corresponding catalogs. The necessary hardware configuration catalogs are then downloaded from the local workstation or loaded from local disk storage. Once the setup parameters have been specified, the operator may establish a communications session (with the front end or "box") and initiate the telemetry processing session using those parameters. The data accountability for the current processing session is monitored via returned status. Real-time or post-session telemetry data products may be viewed.

6. PROCESSING SCENARIO

6.1.1 Spacecraft

Hardware components on-board the spacecraft package telemetry data into packets. Packets are encoded for data quality evaluation at ground sites. A fixed or variable number of packets are combined into frames with header information. This header contains information used at the various ground sites for time-ordering the packets and deleting redundant packets. These frames are combined into blocks that are used by the NASA Communications (NASCOM) Network for transport.

6.1..2 Data Processing

The reverse of the previous process is performed on the ground. The Frame Synchronizer Card performs synchronization on the frames to verify the correct ordering of data packets. Next the frames are transported through the Reed Solomon Decoder card that detects and corrects errors in the frames. The Packet Processor Card then processes each of the packet, combining them into datatakes. Datatakes are captured by the Data Capture Card and earmarked for distribution to various science and operations facilities. (See Figure 1.2)

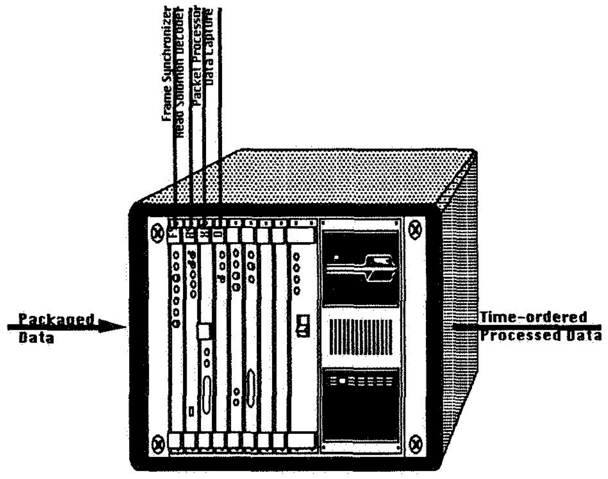

Figure 1.2 Typical Configuration including input and output products

7. TPCE SYSTEM DESIGN

The hardware components are object-oriented by nature (with similarities in configuration and actions):

- each card has a group of setup and status parameters.
- each card can accept and react to all the MEDS commands,
- each front end contains a group of cards,
- each front end can accept and react to all the MEDS commands.

This allows for a direct mapping of the control and monitoring software components to their hardware counterparts.

The top level view of a system is that of a front end within which the user may execute a data processing session. The communication paradigm remains the same across front ends, allowing each front end to handle the routing of commands to be processed to the hardware components. Each front end may be slightly different based on the cards in its configuration and may have additional actions the operator may invoke, such as retrieving real-time data or stored data. This view of the system allows each front end to be represented in the TPCE system as an object.

Each card is also represented as an object in the TPCE system. Generic commands, such as enabling and disabling, are accepted by every card. However, each card processes the commands differently in a manner appropriate to the purpose of the card. In the following paragraphs, italicized words are indicative of the software object representing the entity described.

Each *Card*, the software object representing the hardware card, has a configuration for setup and status data. The TPCE system provides for the configuration parameters to be stored in a file and loaded into the system upon startup. These configuration files contain the naming conventions for the parameters, the default values of those parameters, and the parameter's type. In addition, the *Card* structures have methods, a function on a particular software object, to read and write their appropriate data in forms compatible to the software's function.

One or more editors are associated with each card. *Editors* allow views of information to be presented to the user. The *StatusEditor* and *SetupEditor* for a card present the status or setup data to the user in a graphical format, using the TAE panels for display. View files contain the mapping between the card parameters and their corresponding display entities. The maintenance of separate configuration data makes recoding unnecessary when changing the view of the data.

A *CardEditorManager* was developed to allow the creation a user-defined number of editors at start-up time. The environment set at startup specifies the number of each type of editor to be created. A pool of editors is maintained during application execution. As editors are requested they are removed from the pool of available editors, and are released when software processing is complete.

Card structures are also created and placed in a pool. This pool is managed by a *CardManager*. These card structures are assigned to the front end object as needed and returned to the pool after the session with the front end ends.

A *CatalogManager* allows the user to modify catalogs of Card setup parameters using a *SetupEditor*. The updated configuration may be saved to the workstation's local storage or downloaded for execution during a telemetry data session. The *CatalogManager* also uses the *Card* structure to save the parameters. The *CatalogManager* is responsible for local file storage and directory hierarchy traversal.

The communications segment of the TPCE software, the *IFMapper*, handles the interprets both incoming and outgoing messages. Outgoing messages are converted to a byte stream and are then sent to the *CommGate* for transmission upon command request from the operator. The *CommGate* handles the socket communications from the workstation to the front ends to which sessions have been established. Incoming messages are read as a byte stream and decomposed into the format used throughout the TPCE software. These messages are passed through the *IFMapper* for processing by the application.

These messages in internal format, called *MedsMsgs*, have two segments: a header containing information about the data to follow, and a body containing data. Subclasses of the *MedsMsg* class handle different types of messages, including status, command, catalog, and response messages.

8. ANALYSIS

The TPCE system and associated hardware have a number of advantages and disadvantages. The functional approach to data system design is cost-effective from the hardware stand-point. The cost of building end-to-end data systems using this approach is replication cost instead of repeated design and construction cost. The adoption of standards and hardware designs based on standards have enabled repeated hardware replications. Processing is tailored to specific needs using hardware configuration specifications. The same data is used during the different processing stages, however each component extracts only relevant pieces.

The first advantage to the functional approach is enhanced capability. Future missions such as the Earth Observing System (EOS) will require telemetry processing systems to handle much higher data rates and volumes. Using prototyped VLSI components, a rate of up to 300 Mbps will be handled.

The second advantage is TPCE's modularity and flexibility, provided through an object-oriented design. In a prototyping environment where changes to software occur daily, the effect of software modifications can be localized. In addition, software can be reused. The functional component approach allows for easy tailoring of processing environments to the target mission.

An example of this localization of change can be seen in the design of the *Card*. Using the approach of editable files to hold Card configuration information provides the data system developers with an advantage of localizing the effects of hardware configuration changes. The front end hardware configuration can be changed on the fly without modifying the TPCE software. This also simplifies testing of the hardware components and integration testing of the completed system.

Another advantage to the TPCE system is fast execution. Using this editor pool scheme explained earlier, fast execution time can be achieved. There is, however, a trade-off between execution speed and higher memory usage, since all editors are maintained throughout the execution session.

Although there are many advantages to the current processing system, there are also disadvantages. One disadvantage is the MEDS-environment structure, the hardware's run-time environment. Word boundary overrides restricted workstation processing power. The volume of data accepted by the workstation posed processing power restrictions. These problems are being addressed in the next version of MEDS. For hardware components to be compatible with the

MEDS-based system, they must conform to the structure of the MEDS environment. Provision must be made for commanding and status monitoring using the MEDS system format. This requirement introduces some inefficiencies into the system, and network overhead during transmission.

Another disadvantage is the affect of an object-oriented methodology on execution time. The design imposes some structural inefficiencies affecting execution. For example, the use of *StatusEditor* setup files, while very efficient in terms of allowing changes to the configuration, introduces some overhead. Also, file management and memory usage overhead may be detrimental to some telemetry data processing applications.

A final disadvantage is TPCE's use of user interface. Any user interface will incur processing limitations, driving the execution speed of the application. The problems incurred will be addressed in the next version of the TPCE software.

9. SUMMARY

The TPCE system and the corresponding hardware it controls and monitors is the future in telemetry data system development. To fully develop the potential of future space missions, NASA's telemetry data systems must do more than simply meet specific technical requirements. They must provide reliable, low cost, modular systems which NASA and the user community can tailor in size and performance. These systems must allow growth and expansion in the years to come. Also, the push toward automated data driven operation of NASA's next generation telemetry data handling systems require standard system functional components that are virtually turnkey in operation. Even though these systems are a tightly integrated mix of hardware and software elements, they are but single elements in a large, highly complex Space Ground Network for telemetry data handling. The functional components approach and the Telemetry Processing Control Environment were designed to meet the processing needs of future missions.

10. REFERENCES

1. NASA/GSFC Code 521, 521-SRD-001, High Performance VLSI Telemetry Data Systems, September 1990.

2. NASA/GSFC, TAE Plus User Interface Developer's Guide Version 5.1, April 1991.