AN OPEN SOFTWARE ARCHITECTURE FOR UNIX BASED DATA ACQUISITION/TELEMETRY SYSTEMS

DANIEL DAWSON - VEDA SYSTEMS INCORPORATED

ABSTRACT

Veda Systems Incorporated has recently completed the development of a completely open architecture, UNIX-based software environment for standard telemetry and more generic data acquisition applications. The new software environment operates on many state-of-the-art high-end workstations and provides a workstation independent, multiuse platform for front-end system configuration, database management, real-time graphic data display and data, logging.

KEY WORDS

UNIX
Multi-user
List-based MMI

INTRODUCTION

This paper addresses the software design from an operator's perspective and demonstrates how extremely complicated databases containing more than 100,000 parameters are maintained and utilized in real-time with complete flexibility. The paper also presents arguments in favor of standards-based system design and examples of existing state-of-the-art implementations.

Omega System Overview

The Omega system architecture is divided into two distinct categories -- front-end processing and workstation processing. The front-end processing is based on the proven ITAS/ISA card set with an optional expansion integrating VME.

The Omega system hardware architecture is based on a dual-bus, distributed-processor design in the front-end, supported by a network of distributed workstations which can set up the front-end modules as well as act as monitor and command stations. The dual-bus

design allows the provision of a completely separate environment for system control and display duties from the high speed data pathway. The high speed bus may be implemented in either VME P2 or a separate global memory interface bus for ISA backplane configurations. All system hardware modules are interfaced with the Control Bus (VME P1 or ISA) to allow independent load, run, and control operations. Any module that requires access to the data bus also has a data bus interface.

The design supports multiple front-ends as well as multiple workstations networked on a single Ethernet (or FDDI) backbone. Each component is identified by its network address, and the network as a whole is configured by maintaining the appropriate hosts file.

As with the existing line of ITAS products, the Omega system remains project-oriented. By networking the front-end and workstation resources, the Omega can be configured as needed to support concurrent project development and execution. An Omega workstation allocates front-end resources as needed, enabling multiple projects to be run simultaneously. For example, with an Omega configuration of multiple front-ends and workstations, Workstations 2 and 3 could load and run Project A using Front-ends 1 and 2, while Workstation 1 could load and run Project B on Front-end 3 simultaneously. This capability makes the ITAS Omega an extremely flexible and powerful system.

The Omega system incorporates the standards based, X-windows and OSF/Motif layers as the graphical user interface (GUI). This allows us to offer customers any level of workstation capability from low cost X-terminals to full, special-purpose graphic workstations. In addition, our customers can fully utilize their existing computer assets and operate them under the X standard on, a system network. This enables the Omega to maintain compatibility with a large base of installed systems and peripherals, as well as provide almost unlimited expansion potential.

Software Design

The heart of the Omega is its software suite based on a POSIX compliant version of AT&T System V UNIX. This real-time, multi-tasking operating system is the key to the system's power and versatility. The core of the new distributed architecture is network "daemons", or monitor processes, that reside on each front-end and workstation. These processes are responsible for the inter-process communications (IPC) across the network. They communicate by transferring Ethernet packets that conform to an Omega-specific command and data protocol. The front-end daemon is initially in an idle mode waiting for requests from the workstations to download a telemetry format, process and distribute additional parameters, etc. The workstation daemon will monitor the LAN for messages and commands from the front-ends and master workstations, maintain a list of active front-ends, and load data into local current value tables (CVTs).

The Omega workstation processing is based upon a new software architecture that mirrors the UNIX operating system on which it is built. A central Task Dispatcher module controls the activation of specific Task Modules based upon the user's requests. The Task Modules are small and easily maintained and provide specific functions to meet various telemetry processing requirements. A wide variety of Task Modules have been developed that, together, provide full setup, control, display, and command processing environment.

The use of an evolutionary List-Based architecture, based on layered lists to allow simple access to data elements, is a key design feature of Omega. The use of this system can best be shown by an example of a system that has independent data streams from a test vehicle. The Omega accepts multiple data sources from any combination of telemetry, bus, discrete and analog inputs, and time orders all parameters on the system data bus. The full list of all raw, bit and word combined, EU and Derived parameters within the system is called the Global List. From this, Data Groups may then be formed; For example:

The avionics test engineer, engine performance engineer, or safety engineer can develop separate data lists that only contain the parameters in which they are interested for any given test scenario. This will provide these two groups with effectively separate telemetry systems. Combined data can be archived at the same time and to the same device. since it may be merged on the data bus or separation may be maintained for data security.

This innovative list-based architecture has an additional advantage in that in some systems the Global List may become extremely large. Engineers may find it difficult to locate the parameter that they require from 6,000 or more different entries contained in a single flat file. (One of our aerospace customers has over 25,000 real-time parameters in, their active Global List, and the definition up to 65,000 parameters is supported). The use of the List System allows each project and user to limit the parameters in the lists used to only those that are required. This list-based technique can also be used to limit bandwidth to selected devices by creating sublists for specific archives processing, and display duties.

The system design also incorporates the idea of "minimal processing." This means that only those parameters that are being used by the active applications at any given time are actually being processed by the front-end (when EU processing is being done on the front-end). As a logical extension to this, only the data required by the superset of running applications will be packetized and broadcast on the network.

To add to the robustness of this system, a series of "health checks" are performed at pre-defined intervals. When a project is downloaded and running, the master workstation daemon will periodically request an "I'm OK" status message to each of the allocated front-ends.

General Operating Principles

The Omega System varies significantly from most telemetry and data acquisition systems in the manner by which a data stream is handled. For example, a typical PCM stream consists of a bit sync and decom pair and a set of processes required for the stream. In multi-stream systems, multiple bit sync/decom pairs are supported, but the processing and associated measurements reside in a single, format-parameter-list data base. The entire project is treated as a whole, rather than mini data bases for each processing unit.

In addition to PCM modules, a variety of bus monitor and data acquisition modules are also supported. For instance, MIL-STD 1553 streams may be merged with PCM or output from commercial buses such as ARINC 429/629. A/D modules may be used to input data from FM/FM systems. Omega encompasses a much wider range than the typical telemetry ground station affords, and each new capability is integrated within the same project-oriented data structure and man-machine interface environment.

Each project is made up of streams, parameters, and lists. Projects are created by first executing the Project Editor, and then adding components to the project with the Stream Editor, Parameter Editor, and List Editor. Projects are referred to as either Global or Local Projects. The project currently being executed for telemetry analysis is referred to as the Global Project. Only one Global Project at a time can be loaded and executed per workstation. Many projects, however, can be simultaneously active as project engineers work with their data for analysis or preparing for future tests.

Selection of measurements for archive and display is accomplished by defining a “list” of measurements. The list contains a set of limits for the data. values, a title, and a description. A complete graphically-oriented list editor is provided. A list provides an easy way to prepare a test. All measurements of interest can be stored into a data base for easy retrieval during run-time. All archive and display tasks are list-driven; the list is sorted alphanumerically and can be edited during run-time, which adds increased system flexibility.

Method of Operation

A workstation can be configured to run in two basic modes -- as a “master” workstation that allocates front-end resources, controls downloads, and maintains the active application user list, or as an “application,” workstation that runs any of the various analysis and display applications. The system is project-oriented. By adding front-end ID field to the stream definition, a project definition completely defines what front-end resources are required.

Any workstation with authority can request to download a project (i.e. to become the master workstation for that project). The project loader would send a download request to each of the front-ends used by the project. Each of the polled daemons would respond by sending its current status (whether or not it is available). If the front-end has not been allocated to another project, it will allocate itself to the requesting master workstation and send the master its current hardware configuration. If all of the required front-end resources are available, then the master downloads the appropriate telemetry setup information to the front-ends. It then initializes the active user list for the project and tells the front-ends to start sending data (initially this is just IRIG time). The requested project download can fail if the required files are not present, if the requested front-ends are not available, etc. The project loader must send an unload request to any front-ends that were allocated.

Any workstation can request to be an "application" or "user" workstation for a project. When the user invokes the appropriate command, the workstation polls each front-end to get its current status. The user is then provided with a list of the projects that are currently running. (For example, "Project F-18 loaded at 09:43:25 by ITASWS2 on ITASFE1.") The user selects the appropriate project from the active list, and the user workstation then sends an "Add User" request to the master workstation and invokes the "Copy Project from Workstation" utility. This file transfer routine will download the required project data base files from the master to the user. These files are required to ensure that the local CVT uses the same vectors as the front-ends. The user is given the option to download the Derived Data Base, the List Data Base, and the Autoload Data Base from the master.

Derived Parameter Processing

Derived parameters are an important capability, as often the units or format of the computed measurements are inadequate. In Omega, the processing of real-time derived parameter data and external system commanding is accomplished in three distinct and increasing capability levels. These levels are the operation level, the system level, and the application level.

The operation level of derived parameter and commanding is limited to processing descriptions entered by the user via a simple Fortran-like algebraic syntax-based description. At this level, real-time users can create new parameters which are functions of other real-time data parameters (or previously defined derived parameters) and constants. No programming knowledge is required to utilize this feature. The resulting derived parameters are available for display or data logging. This level of processing is not data driven. Rather, the iteration rate is set by the user and will be synchronous to the incoming data. The following example shows how a new parameter, ALT, can be calculated from two other real-time data parameters - P1 and P2.

```
ALT = (P1 - P2) ** 2.345967
```

Once identified, defined parameters may be merged and further processed similar to any parameter and can be further combined in sequential, operation-level calculations.

At the system level, a user with basic C programming experience can easily create functions which can then be called by the operation-level user. For example, if a particular test necessitates that Omega output a RS-232 message when, a derived data value exceeds a certain value, the system level programmer needs only to create a simple command output function as shown below.

```
int send_msg (val, limit)
        float val;
        float limit;
{
        if (val < limit)
                write_to_port("Altitude is too low!")
        else
                write_to_port("Altitude is nominal");
}
```

Then the operation-level user can create the following derived parameter:

```
ALT = (P1 - P2) ** 2.345967
send_msg (ALT,10000)
```

The lowest level of programmability is the applications level. Here the processing philosophy moves the location of the processing from the UNIX CPU to a front-end, high-speed dedicated processor. In addition to moving the location of the processing, the processing methodology changes from synchronous, fixed iteration rates to data-driven processing. Naturally, all parameters generated from the front-end processor are fully integrated into the previous processing levels.

As described below, the front-end processors available within the Omega environment provide performance currently unapproachable by the competition. These include advanced i860-based processor cards which, provide equivalent performance to several mini-computers on a single card.

With this processing architecture, most simple derived-parameter processing does not require the cost of dedicated front-end processors; however, with the addition of one or

more data-driven front-end processors, the system is capable of processing the most demanding real-time data streams.

Man-Machine Interface

The Omega man-machine interface is an X-window, OSF/Motif-compliant graphical user interface (GUI). In addition to the ability to create custom displays, standard features include a menu bar, a header window, a command line entry window, and a status window. These functions are grouped together in a Manager window to centralize the software control and status information. The Manager window also displays the currently selected Local and Global Project. Additional "background" Omega operating system routines include a hardware status task and a software workload task. These background tasks can be brought up as needed but are more for optimizing the system than constant display.

The user interacts with the Omega system primarily through the menu bar and command window. The two perform very similar functions, but the command window is primarily used by more experienced users to enter "shorthand" commands for system execution. The menu bar controls the same functions as the command line, but with easy-to-use point and click interaction.

The Omega system is set up by editing the various components of the telemetry data base. For PCM streams, once the bit sync parameters are defined and the decom format is programmed, the user defines the various syllable collection, format conversion, EU conversion, and user-defined measurement processing. Omega supports over 15 pre-defined data conversion formats, including exotic types such as poly-multi-sectional fits and the more mundane first through n'th order fits. Omega also supports advanced telemetry syllable encoding features: up to 32 syllables per measurement. The simulator and front-end hardware can then be "loaded." Loading the front-end actually programs the bit sync(s) and decom(s) and downloads the i860 EU conversion software. All of this is accomplished via drop-down menus from the menu bar or directly through the command window. Pointing and clicking with the integral mouse is easiest, but, for experienced users, a direct command line entry is often faster.

Once the front-end is loaded, the Omega system is processing the telemetry stream(s) and is ready to archive and display data as chosen by the user. To support the repetitive test points often found in telemetry applications, Omega provides the capability to pause the display and archive tasks while outside of test point limits. This helps save valuable computer disk storage space and wear and tear on the system printer.

To support off-line analyses, Omega includes an integrated set of utilities for off-loading data onto removable media, in both MS-DOS and UNIX formats. These utilities will enable certain users, depending on customer system specifications, to copy their data bases onto the system for conversion to ITAS format. Screen dumps, window dumps, and file prints are also available as utility options.

For display, Omega supports a Quick-Look window and a variety of other graphic display options. Quick-Look displays the measurements in a tabular format, including not only the measurement name and current value, but the description, units, limits, and optional time and raw data display. See figure 2-2. The measurement can be raw data, collected syllables, processed syllables, or the engineering unit value. The output format is selectable; the measurement name is identical throughout. This task includes an integrated control page which enables the user to dynamically change the list, iteration rate, font size, and format. As with all Omega application tasks, multiple Quick-Look routines can be activated and positioned anywhere on the display screen.

Omega Set-Up

For telemetry applications, the Omega system, is set up by editing the various components of the telemetry data base. Once the bit sync parameters are defined and the decom format is programmed, the user sets about defining the various syllable collection, format conversion, EU conversion, and user-defined measurement processing. Omega supports over 15 pre-defined data conversion formats, including unusual types such as poly-multi-sectional fits and the more typical first through n'th order fits. Omega also supports advanced telemetry syllable encoding features; up to thirty two (32) syllables per measurement with user-defined bit selection from each syllable can be programmed. Utilities may also be provided to input data from existing customer data bases into the Omega parameter data structure.

Once defined, the data base is expanded, or compiled, into an executable format suitable for the front-end hardware. This process assigns vectors into the current value table and defines how syllables are collected and converted and with which coefficients. The simulator and front-end hardware can then be loaded using the stream definition. Parameter definition may also be accomplished through utilities that allow importing data from existing host data bases.

Loading the front-end actually programs the front-end bit synchronizers, decommutators, and processors. All of this is accomplished via drop-down menus from the menu bar or directly via the command window. Pointing and clicking with the integral mouse is easiest, but for power computer users, a direct command line may be faster. Cost on utilities may even be provided to program competitor's equipment from within the Omega environment.

Defining a telemetry stream is typically the most difficult process, encountered when programming a telemetry pre-processing system. Omega eases this effort by treating the telemetry stream as a whole, which enables the user to enter the stream definition information in a graphical user environment. The process involves defining the bit stream pattern that must be captured, defining the format of the data stream(s), and defining what data conversions should be executed on the raw data stream.