DESIGN OF MULTI-PLATFORM CONTROL SOFTWARE FOR TELEMETRY SYSTEMS

FARID MAHINI
Microdyne Corporation
491 Oak Road
Ocala, Florida 34472

ABSTRACT

This paper discusses the requirements and design of a multi-platform system software for control, status, calibration and testing of a telemetry system.

KEYWORDS

Telemetry System, Control Software, Multi-platform, Distributed Application Services, Native Look And Feel, Windows, UNIX, Macintosh.

INTRODUCTION

Today's advanced telemetry systems are partially, if not completely, remote controllable. Some systems are made up of distributed telemetry sites that are operated from remote command centers. Other systems may require fault isolation and corrective measures. Yet another system may require to run on more that one platform or to interface with other platforms. Such a diversity in telemetry systems presents a difficult challenge for designing control software that can be used for various systems. Microdyne Corporation has designed SysApps, System Application Software, a multi-platform telemetry system control software that can be deployed on a cost-effective mix of existing and new platforms. The following section presents the design of a multi-platform control software that can be adapted to accommodate many systems.

DESIGN CONSIDERATIONS

Control software that is capable of adapting to various systems must address the following requirements:

- Modularity
- Multi-platform
- Distributed Remote Access
- Remote Transfer Rate

Every system is comprised of various equipment where each equipment requires its own specific control interface. Non-Modular control software tends to be system dependent and does not lend itself to changes in system configuration. In contrast, modular software is made of an array of self contained modules. Each module performs all necessary tasks related to a specific device. Therefore appropriate module can be installed to accommodate any device changes in system configuration.

Off-the-shelf PC-based computers are used to control small telemetry systems. However, larger systems employ RISC-based UNIX workstations to command and control their system. Historically, control software designers were forced to design two separate software programs; one for PC-based computers and another for UNIX computers which led to long development schedules and larger financial burden. In superior designs, cross-platform development tools are used to produce complete application portability across operating systems and graphical user interfaces (GUI).

Historically the failure or success of remote control software has been the ability to provide control and status at a rate consistent with equipment performance and end-user requirements. In an attempt to provide some level of remote control, commercial software has taken the approach of extending the application screen from local site to the remote site. This technical approach limits the speed of the application and is not adequate for auto tracking antenna control in which updates in the millisecond range is required.

To increase the update rate, the SysApps application runs identical software on both local and remote platform. This approach allows the remote computer to update the application graphics and limits the communications to updating character information located inside of text boxes within the graphical display. This minimal information is converted to ASCII characters which are then transmitted to the equipment location via microwave links, fiber optics, land lines or satellite links.

The dominant limiting factor in a remote control software system is usually the baud rate, in which the application transfers information from the local site to the remote site. This

factor challenges the designers to transfer a minimum amount of information with the maximum effect.

The constant demand on communications technology today, is to provide maximum information transfer in the least amount of time. This demand has increased the speed of the modems and has helped remote control systems to some extent. Diminishing this effect is the increased amount of information to be transferred on newer instrumentation.

Remote control and status is a major consideration in many industries and specially in the telemetry industry as demonstrated by the ongoing RSA (Range Standardization and Automation) program. Many telemetry systems have a software control and status system in place and many more systems will be programmed to include software control and status systems. In an attempt to provide an overall solution SysApps has been designed and structured to accommodate:

- Any size system from a single receiver with a cupped dipole antenna to a major tracking site configured with 30 plus telemetry receivers and multiple tracking stations.

- Simultaneous multi-platform systems. Users can use their existing hardware. Remote can be a UNIX based computer system and local can be a ISA bus compatible computer.

- Configuration Management in the form of the system configuration control files and mission logs. This electronic mission log automatically logs AOS (Acquisition of Signal), LOS (Loss of Signal) and any real time system configuration change such as receiver frequency.

DESCRIPTION

The SysApps designed by Microdyne Corporation is written in C++ object oriented language taking advantage of many features of C++ such as encapsulation and inheritance. It uses a multi-platform object oriented Application Programming Interface (API) to generate multi-platform software, complied separately under each platform. Its is capable of communicating with another SysApps software running under a different platform. Sysapps is completely portable across most UNIX RISC platforms, OpenVMS, Windows 3.1, Windows NT, Windows 95, OS/2, and Macintosh. All graphical user interfaces take on the look-and-feel of the native platform. SysApps also supports many international languages. As illustrated in figure 1, SysApps is comprised of five main sections: Task Processor/Arbitrator, Interface Controller, Distributed Application Services (DAS) Controller, Test Port Controller, and Device Modules (see figure #1).

Each Device module handles mission critical data logging in addition to all graphical user functions such as data entries, message / confirmation dialogs, display updates, native platform look-and-feel preference and internationalized language preference. All user interface objects (graphical and text) are stored in a platform-independent resource file.

The Interface Controller interfaces with National Instruments IEEE-488 card and RS-232/RS-422 serial ports. The Interface Controller module is the only interface to system equipment. Th module is the link between SysApps and other platform-dependent device drivers.

The Distributed Application Services (DAS) Controller is utilized when the telemetry site and control center(s) are at different locations. The Sysapps software must be installed at both telemetry site and control center in order to communicate with each other via the DAS Controller. Use of platform-independent abstracts provides the means for two SysApps runnin on different platforms to communicate with each other.

In many telemetry sites, configuration files are handed down from the command and control center prior to any mission (such as TMATS format, chapter 9 of IRIG). These mission configuration files contain programming parameters for various test, telemetry and tracking instruments within the system. The instrument parameters can be RF frequency, IF bandwidth, data bandwidth for a telemetry receiver, position and acquisition mode of a tracking antenna, d format and data rate of a bit synchronizer, signal routing through a switch matrix, etc. The Tes Port Controller is responsible for parsing and extrapolating the mission configuration files (or calibration files). The extracted programming parameters are then routed to the Task Processor/Arbitrator for distribution.

The Task Processor/Arbitrator is responsible for maintaining all dynamic data exchange (DDE links with device modules, Interface Controller and DAS controller. Any communication with Interface Controller module is passed through the Task Processor/Arbitrator module. In the cas of remote access where distributed services is enabled, the Task Arbitrator diverts the Interfac Controller data link to DAS controller. In addition, the Task Processor handles all system configuration, system setup, loading / saving log files and mission files, AGC logs and system device inter-dependencies (master/slave).

SUMMARY

Telemetry control software must be able to evolve with ever changing telemetry system designs and increasingly demanding requirements. The number of distributed telemetry systems have grown and system designer need a software package that will meet their total hands-off remote control requirement now and in future. Multi-platform software design eliminates many incompatibilities during system design and provides an additional step towards achieving standardization in remote control software design.

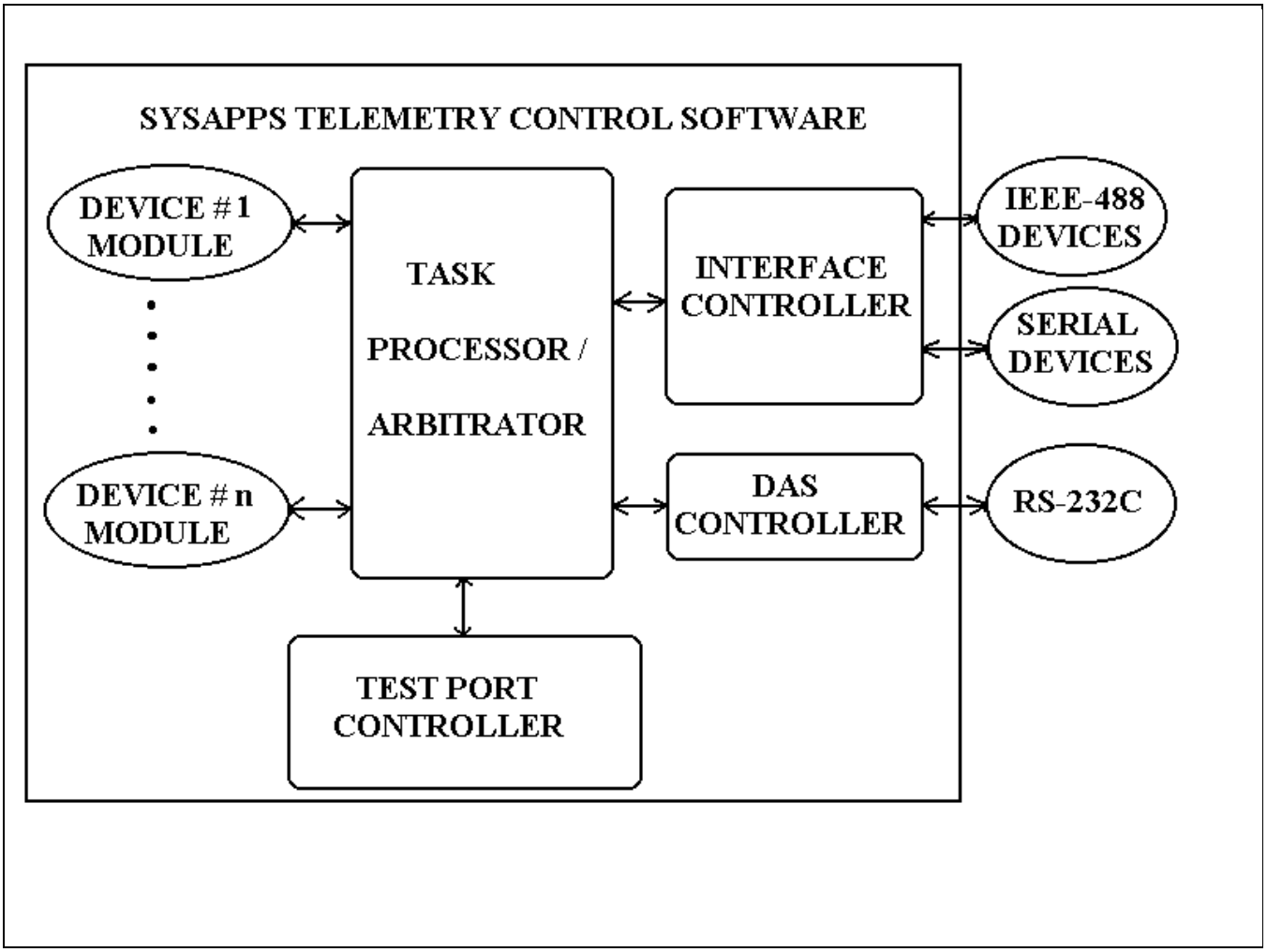
SYSAPPS TELEMETRY CONTROL SOFTWARE
DEVICE #1 MODULE
DEVICE # n MODULE
TASK PROCESSOR / ARBITRATOR
INTERFACE CONTROLLER
DAS CONTROLLER
TEST PORT CONTROLLER
IEEE-488 DEVICES
SERIAL DEVICES
RS-232C

Figure 1