Autonomous Control and Data Acquisition for Advanced Satellite Systems

Tim Turner
Gulton Industries, Inc.
Data Systems Division
6600 Gulton Ct., N.E.
Albuquerque, NM 87109
(505) 345-9031

ABSTRACT

Autonomous operation is rapidly becoming a requirement for most new spacecraft systems. An autonomous spacecraft greatly simplifies the ground station processing and monitoring requirements, freeing ground station capabilities for other important tasks. The T^2C^2 (Telemetry, Timing, Command and Control) System has been conceived and architected to facilitate spacecraft autonomy. The T^2C^2 architecture is ideally suited for on-board closed-loop control, redundancy management, housekeeping and other autonomous functions. This paper provides an overview of the T^2C^2 architecture and its applications in the design and implementation of an autonomous spacecraft.

INTRODUCTION

Earth ground stations have historically been architected to provide the majority of the command and control functions required by a satellite system. The ground station is typically in continuous contact with the spacecraft, issuing uplink commands and receiving payload and housekeeping telemetry. The decommutated telemetry is then distributed to the various users, analyzed and corrective action taken if required. Present and future spacecraft are becoming increasingly complex to the point that a subsystem is more complex than the entire spacecraft of past generations.[1] This heightened complexity demands an increase in ground station data processing and monitoring requirements which multiplies the ground station computing and personnel requirements accordingly.

In order to minimize the cost and complexity of the ground station operation, while still allowing increased spacecraft performance, autonomous systems must be placed on board the spacecraft. The addition of intelligent processing to a properly designed system allows autonomous operation of two weeks to six months or more without ground station intervention. This is accomplished by allowing the spacecraft to monitor and control itself, i.e. sensing fault conditions and controlling the system reconfiguration to restore proper operating conditions.

The T^2C^2 (Telemetry, Timing, Command and Control) System has be architected to provide the spacecraft designer with the necessary tools to implement various levels of autonomy.

AUTONOMY

The definition of autonomy and the distinctions between automation, autonomy and machine autonomy are currently nebulous at best. Reference 2 describes autonomy on a system level as:

> Autonomy is that attribute of a system that allows it to operate without external control and to perform its specified mission at an established performance level for a specified period of time.

This definition is careful to avoid inflated expectations of autonomy by pointing out that a system is only autonomous within a strictly defined operational environment.

Examination of an autonomous system at a lower level will show that it is an extension of an automated system. The automated system can perform a preprogrammed sequence of actions based upon the occurrence of a particular event such as the reception of a certain input or the matching of a time-code word to a preprogrammed execution time tag. However, should this automated system encounter a fault condition, it will be unable to successfully complete that preprogrammed sequence.

The automatic system may now be extended to include autonomy by adding redundancy, fault tolerance and on-board intelligence to adapt its behavior to changing external and internal conditions.[2] This enables a system to monitor its own performance and make changes in itself, allowing continual operation. For example, a system may reconfigure itself to use a redundant subsystem upon sensing a failure. The ability to sense faults and implement changes and the availability of redundant systems are the keys to autonomy.

SYSTEM ARCHITECTURE

When implementing an autonomous spacecraft, the architecture of the command and telemetry system is of prime importance. To be effective, the system must be able to carry out the basic autonomous control process which is outlined in Reference 3 as follows:

(1) Sense and analyze the state of internal or external quantities which are inputs to the control process.

(2) Derive and command a response by the system that meets an appropriate objective.

(3) Act to element the response.

It is easy to see that this could become an iterative process in which the results of the previous autonomous command are examined and additional commands derived if necessary.

Obviously, the requirement to implement this autonomous process has an effect on the command and telemetry system architecture. First, the system must be able to request the sensor information needed to monitor a condition or make a decision. Second, the requested data must be analyzed and a decision made as to what action, if any, is required. Third, once the on-board computer finds a command necessary, it must be able to execute it through the spacecraft's command system. Finally, these three requirements must be tied together into a compatible system in which all three subsystems achieve their purpose and work together in harmony. Figure 1 shows at the system level how the T^2C^2 System ties together the major subsystems.

Telemetry

The first architectural requirement of an autonomous system is that the system be able to obtain sensor information on its internal and external environment. Internal data is required to determine if the system itself is functioning properly, while external data is needed on the process under control. Clearly, it is the telemetry system which must provide the digitized data to the computer.

When formulating the T^2C^2 System concept, it was decided that the system must first function as a "classical" command and telemetry system. This provided a simple, basic system upon which to build an autonomous system as well as a default/fail-safe framework which would continually provide ground communication should an error condition exist. With this in mind, the T^2C^2 telemetry subsystem will first and always be able to run from preprogrammed formats independent of computer control. These formats may access all available sensors for downlink to the ground station if programmed to do so.

The requirement for autonomy introduces the necessity of a computer which must have access to this data. A separate telemetry system dedicated to the computer would be prohibitive in cost, weight and size. Yet, if given access to the same telemetry system, it

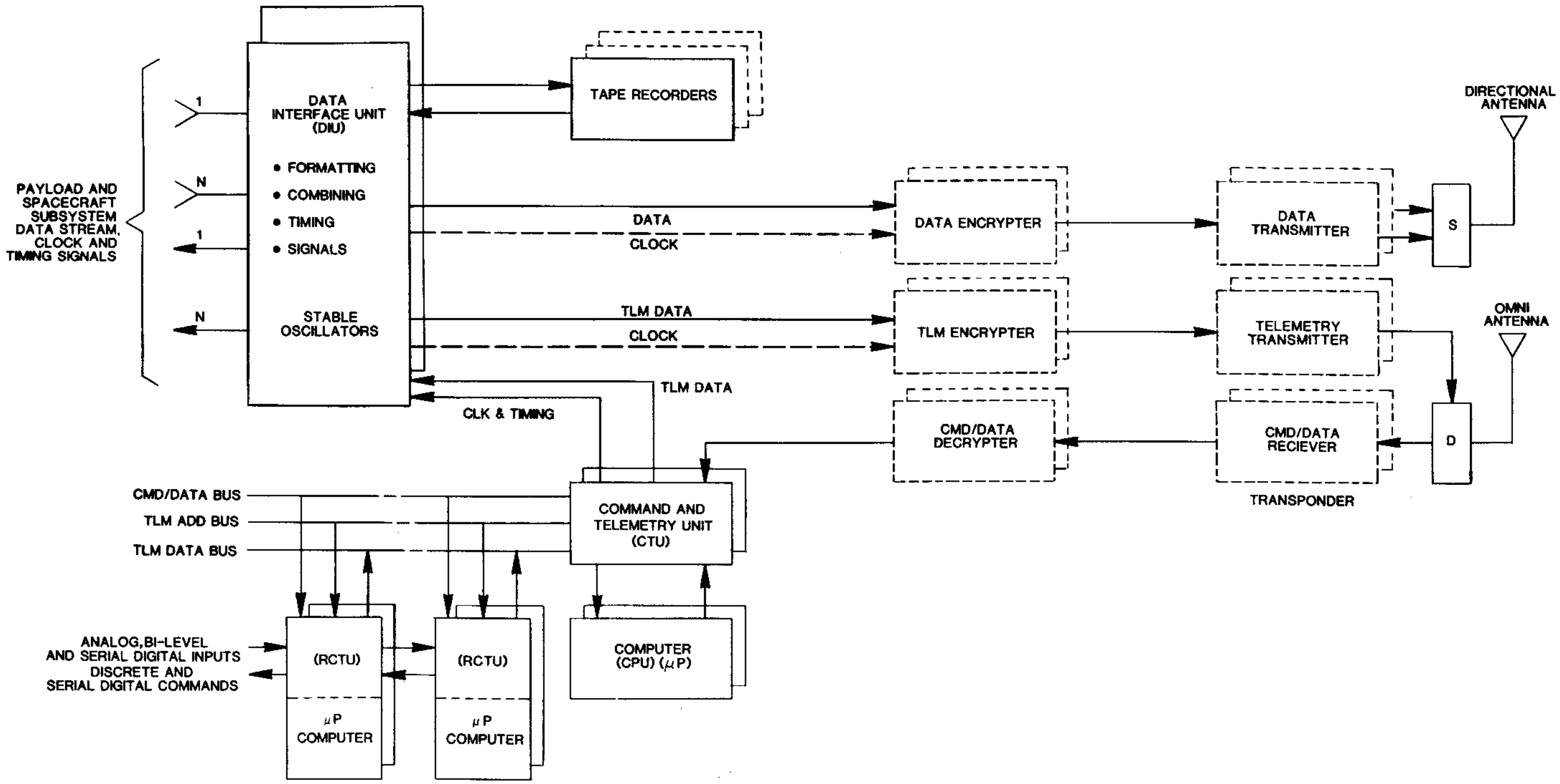

FIGURE 1
T^2C^2 SYSTEM BLOCK DIAGRAM

must obtain its data on a noninterference basis to meet the fail-safe monitoring requirement. T^2C^2 solves this problem in two ways. First, the telemetry request and reply buses operate at twice the speed required by the telemetry format control. This allows the central computer to interleave its request with the standard format requests issued by the telemetry processor. This process is Illustrated in Figure 2. In doing this, the computer may rapidly access whatever telemetry channel it requires independent of the telemetry processor operation.

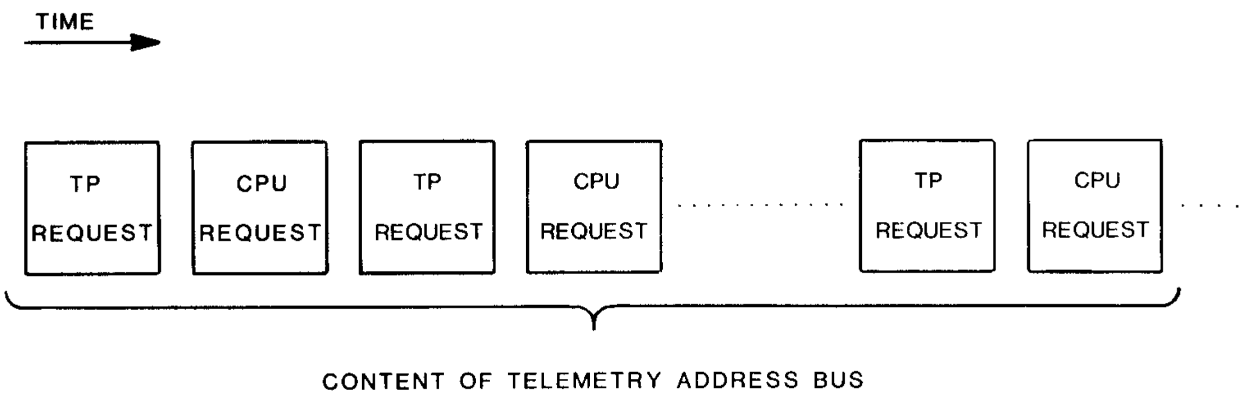

FIGURE 2
TP/CPU TELEMETRY REQUEST INTERLEAVING

Secondly, a remote microcomputer may also obtain telemetry data on a noninterference basis. The remote micro is allowed to access any of the local telemetry channels. This may be done anytime the remote telemetry processor is not busy with a bus-oriented request from the central computer. This provides an ideal situation for designing remote, autonomous subsystems.

The T^2C^2 telemetry system is controlled by the telemetry request and reply buses. The buses were separated, rather than time multiplexed, due to the redundancy requirements of an autonomous system. Without this separation, certain failure modes make system diagnosis and correction impossible.

Processing

The second system architecture requirement of an autonomous system is the ability to derive and configure an appropriate system based on the telemetry data. An intelligent processing element such as a mini or microcomputer will most often meet this requirement. As mentioned before, this processor must never interfere with or take priority over the ground station command and telemetry requirements.

The ability to put a computer into space which meets the rigorous environmental conditions has been a major stumbling block to autonomous spacecraft systems. Ten-year missions will require redundant, radiation-hard, fault-tolerant computers. Of course, these computers must also be extremely size and power efficient. Only recently have there been large strides made in the direction of achieving these goals.

The T^2C^2 system provides processing capabilities for an autonomous system in two different ways. First, each remote unit may include its own redundant microprocessor. This processor may execute complete control over all of the remote’s interfaces. For example, the remote unit and microprocessor may be dedicated to a power-control subsystem. This unit can then continually monitor and service the necessary sensors and instruments to keep the spacecraft within operational requirements.

Secondly, T^2C^2 provides autonomous processing capabilities by allowing a microprocessor or larger minicomputer to be located at the CTU. This computer may control the whole system or only keep tabs on one of the autonomously functioning remotes while performing data processing requirements. The CPU must communicate with the remotes over the command and telemetry buses, again on a noninterference basis. It is interesting to note that by implementing both the remote and central computers, a dual redundant autonomous system is obtained. The redundant remote and redundant microcomputer implement an autonomous subsystem while the redundant central computer autonomously controls the entire system or takes over a remote computer’s responsibilities, should it be required.

Command

The third system architecture requirement for an autonomous system is the ability of the computer to carry out system commands based on its decisions. The T^2C^2 command system also functions first as a classical command system. Priority is always given to the uplink command stream allowing uplink commands to be executed without delay. However, with a maximum uplink data rate of 2,000 bps, the computers have a great deal of time to insert commands into the executable command stream. Of course, during autonomous operation, the computers will not have to contend with uplink competition.

The computers, given this capability, now have the means to complete the autonomous control process. Having read the telemetry data and ascertained that an action is necessary, the computer may now issue a command to be executed. For example, the central computer may command a remote to issue a high-level discrete command to turn on a tape recorder. At a later time, further commands may be issued to record, rewind or playback.

REDUNDANCY

Obtaining a .99 probability of success on a ten-year space mission is only possible by implementing redundancy. Redundancy allows the computer controlling the system to sense a faulty subsystem and replace it with a working backup unit.

The T^2C^2 command redundancy diagram is shown in Figure 3. This diagram illustrates the multiple paths a command may take through the redundant CTU, bus and RCTU. The path a command takes is selected dynamically by bit fields in the command format. Should a command not be executed, it would be noted in telemetry and another path chosen. For example, the command may be re-executed using the redundant bus. This diagram illustrates that T^2C^2 is block redundant. Each major subsystem is duplicated and cross-strapped as required by autonomous applications.

The T^2C^2 telemetry system is also block redundant. However, due to power concerns, only one set of telemetry circuitry is powered-on at one time. For this reason, the path used by the telemetry system is selected by command.

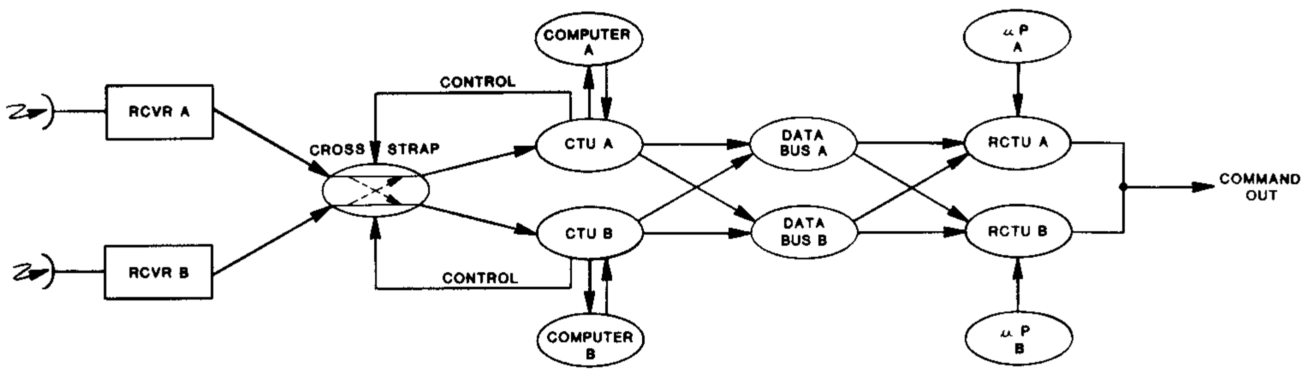

FIGURE 3
T^2C^2 COMMAND REDUNDANCY DIAGRAM

SYSTEM IMPLEMENTATION EXAMPLE

As an example of an autonomous subsystem, a simplified power control subsystem is illustrated in Figure 4. The major components include the solar array, the battery charger, the battery, a load, the RCTU and a microprocessor. The simplified command and telemetry interfaces are defined as follows:

(1) Charge Current Sensor
(2) Battery Temperature Sensor
(3) Battery Voltage Sensor
(4) Load Current Sensor

(5) Charge Rate Select Command
(6) Charger ON/OFF
(7) Load ON/OFF

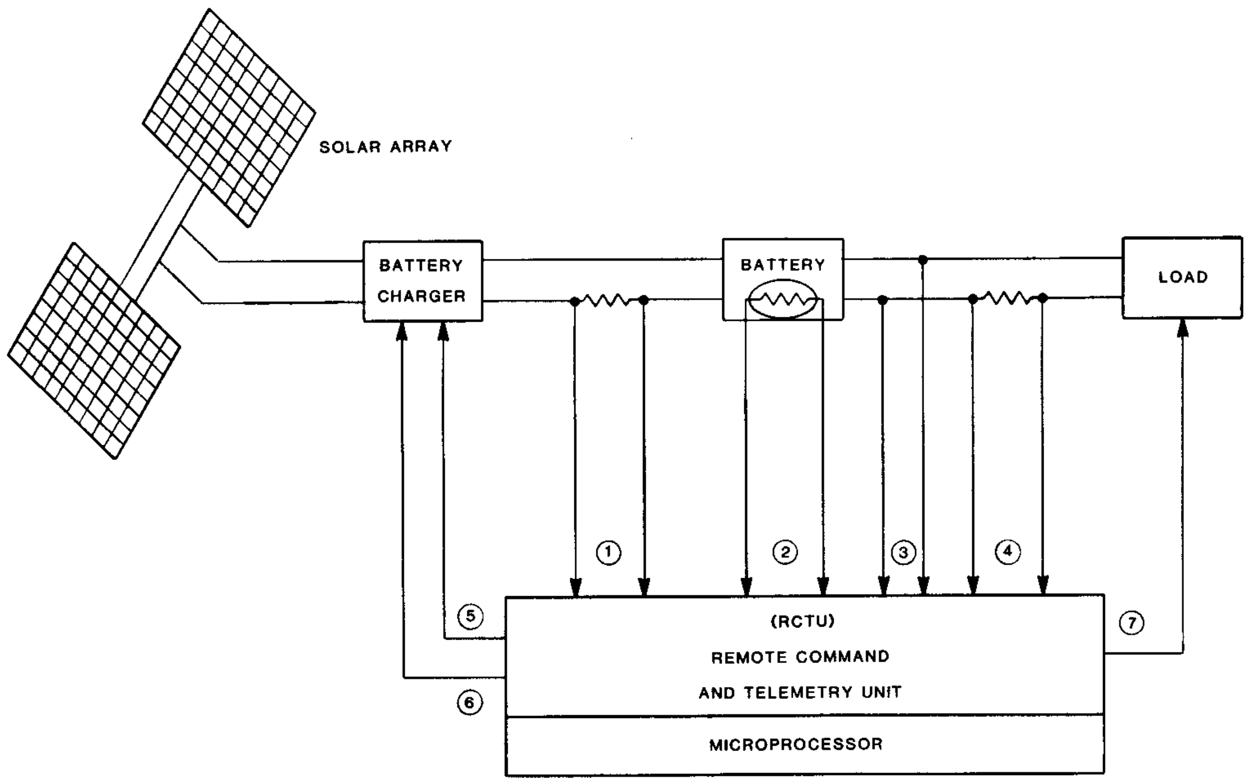

FIGURE 4
POWER CONTROL SUBSYSTEM BLOCK DIAGRAM

If implemented, this system would include redundant units and cross-strapping. However, for the purposes of illustration and clarity, it is shown as a simple single-string system. In addition, this system would also include the ability to switch each of the major components out of the string, providing the ability to isolate a failed component.

Since this imaginary spacecraft is now flying through space without ground support, one of the things it must do is make sure it always has power to continue functioning. The autonomous power control subsystem will readily take care of this since the microprocessor has the capability to monitor and compare the necessary voltages and currents. The microprocessor will use this data to control the battery charger and the load.

This system's responsibilities would include monitoring the batteries and controlling the battery recharging process. To do this, the micro would monitor the load current (4) and calculate the integral to determine the battery state-of-charge. When this value reaches a predetermined limit the battery recharge sequence will be initiated. Battery temperature (2) and voltage (3) will be used to determine the charge limit curve which will control the battery charger. The battery charger is then programmed and turned on. The micro can verify charging by the presence of charge current (1) and turn off the charger when appropriate. The battery state-of-charge can then be reset to 100% and the recharge sequence terminated.

Should the battery charger fail during the charging process, the microcomputer would immediately note this by the lack of charge current. This would put the computer into a fault recovery mode which would systematically attempt to recover to the desired operation mode. One possible algorithm would be to turn off the failed charger, switch the unit off-line to provide isolation, switch in the redundant unit and reinitialize the battery charging subroutine.

While this is a fairly simple example of autonomy, it serves to illustrate how a remote can be used to autonomously control a subsystem. This same system could concurrently control battery reconditioning, power margin determination, load management and calculate and monitor power efficiencies.[1] Additional remote units may be used on the same spacecraft to autonomously implement the Attitude Control System, Propulsion System, or Communication System.

CONCLUSION

In summary, the architecture of the command and telemetry system is extremely important when implementing an autonomous spacecraft. The controlling computer must have the ability to obtain needed data through the telemetry system as well as issue commands. Autonomous operation must never interfere with or take precedence over any needed ground control. The computer must also be able to switch in redundant subsystems in order to obtain a high probability of mission success. These fundamental autonomy requirements all drove the architecture and design of the T^2C^2 system.

REFERENCES

1. Turner, Philip R., “Autonomous Spacecraft Design and Validation Methodology Handbook, Issue 2,” Volume 1, 7030-4 (Rev. A) Jet Propulsion Laboratory, Pasadena, California, August 1984.

2. Turner, P.R., “Autonomous Spacecraft System Control Topics,” JPL D-1413. Jet Propulsion Laboratory, Pasadena, California, March 1984.

3. “Autonomous Systems: Architecture and Technology,” JPL D-1197. Jet Propulsion Laboratory, Pasadena, California, February 1984.

4. Fernandez, M. “An Overall Digital Processing Architecture For An Autonomous Spacecraft,” 82-2389, AIAA COMPUTERS IN AEROSPACE IV CONFERENCE, Hartford, Connecticut, October 1983.