MICROPROCESSORS FOR SATELLITE TELEMETRY
-A UNIVERSAL APPROACH

PETER H. CAMPOLI
Spacetac, Inc.
Bedford, Massachusetts

S. K. BREDE
President
Analytix Electronic Systems, Inc.

Summary. A system has been developed which can be flexibly tailored to meet most telemetry requirements. The concept results in higher reliability, greater modularity, and lower cost than classical approaches. It also offers more system capability.

The development was realized by adaptation of the telemetry requirements to a computer system architecture. Such an adaptation is made possible by the use of a microprocessor as a device controller.

The concept has been successfully applied to a specific telemetry requirement for use on a satellite. The results have far-reaching implications on the future of spacecraft electronics. Reductions in overall system hardware are possible while increasing the functional capability.

Introduction. In today’s space electronics market there is an ever increasing need for telemetry systems which are low cost, low risk and meet the required specifications. To comply with these needs for the majority of systems, a telemetry supplier would have to maintain an inventory of every conceivable telemetry requirement, new and old, which could be used off-the-shelf. This is not a practical solution. Another approach would be to develop a system which includes all requirements in one telemetry system and could, therefore, be used in any application regardless of the specification. Such a system would not be practical due to size and development cost.

A reasonable solution, however lies in a compromise. This paper traces the development of a concept consisting of a general purpose “manifold” to which plug-in modules may be added, thereby tailoring the manifold to a specific requirement. The result is a lower cost, minimum risk, totally compliant system which will service the majority of telemetry needs. A definition of requirements is formulated below, an architecture is justified, and a

---

1 This work was sponsored by the Office of Naval Research, Naval Research Laboratory. The author wishes to acknowledge guidance received from Messrs. P. Wilhelm and R. Eisenhauer, both of NRL. The author also wishes to acknowledge the technical contributions of Spacetac’s chief engineer, Mr. N. Elliott.
description of operation is presented to demonstrate the adaptability of the architecture to varying requirements. Finally, an application of the system is described in detail.

Requirements of a Generalized System. In the classical sense, a telemetry system samples many data sources, one at a time, in a predetermined order, converts the sampled data to sets of binary coded pulses (words) and transmits these words (usually organized into groups called frames) to another location. No two systems perform the identical functions, yet there is a degree of commonality that allows us to sort functions into two classes: (1) those that are common to most telemetry systems and, (2) those that differ from one to another. The functional elements most frequently found in telemetry systems are:

1. Control- to time and synchronize all other functional elements
2. Formt generation- to determine the order of data samples
3. Data input- to multiplex various types of data and convert it to binary coded pulses
4. Data output- to transform these pulses to a serial bit stream suitable for transmission
5. Interfaces- to service external systems such as remote multiplexers
6. Power converters

Of these, the control element and power converter can be classified as common to most systems, while the remainder change in one way or another from system to system. When realized in hardware, the control element is usually distributed throughout the maze of all other functional elements and this is where design complication usually occurs.

An optimum architecture for a telemetry system then would be one in which a control section and a power converter form a general purpose manifold which accepts unique plug-ins that tailor the system to each unique application.

The architecture most closely fitted to this requirement is that of a centralized computer with bussed functions which communicates to a variety of peripheral devices. Only recently has such an approach become realizable. With the advent of microprocessors, computer systems can now be implemented in miniature; a necessity for most satellite applications.

\(^2\) Definitions relating to this subject can be obtained in the “Aerospace Telemetry Standards,” MIL-STD-442.
**Description of the General Purpose System.** A processor-based, general purpose manifold was designed which forms a control nucleus applicable to most telemetry systems. It is presented in Figure 1. All functions which appear to the left of the system bus interface are controlled by software. A parallel bus connects the common processing circuitry to an array of plug-in devices shown to the right of the interface, completing a structure that meets a specific telemetry requirement.

![Diagram of Generalized Telemetry System Architecture](image)

**Figure 1 - Generalized Telemetry System Architecture.** Dedicated service requests from each device and service grants from the microprocessor enable a dialog between the common and applications-oriented hardware.

The mechanical, electrical and software design philosophy is dictated by the operational requirement that a periodic, synchronous request for data, issued by a timer, initiates a word processing cycle— a sequence that includes sampling an input channel and loading the data into the output (downlink) serializer. This process must be dependable (jitterless) and form the heartbeat of the telemetry system.

Following this philosophy, a library of physically separable, plug-in devices would include as a minimum:

1. A module containing a timing countdown chain and a downlink serializer. These functions are combined into one device because the telemetry output bit stream is synchronized and shifted by timing signals from the countdown chain.
2. A format storage device that specifies the ordering of inputs to be sampled.

3. An input multiplexer that samples a data channel and converts the sampled data to a digital word.

4. Interface devices which communicate in a unique fashion to external systems. Some applications might include a command link.

Other types of devices can be added to the system, constrained only by the bus design. This flexibility makes the system adaptable to a wide variety of applications.

Software for the microprocessor is divided into short, high speed, software modules in a main line executive program. Each module corresponds to a specific task which is initiated by a device requesting service. Service requests are prioritized in accordance with the importance of each request to the telemetry process. Firmware for the software modules is contained in programmable read only memory (PROM). Instructions in PROM are accessed by an interpreter which utilizes a random access memory (RAM) as a scratch pad.

Although word processing is paced by the downlink serializer, all transactions over the system bus between the microprocessor and devices are asynchronous. Asynchronous operation is implemented by a combination of dedicated and bussed signals. There is a dedicated service request line from those devices which require interrupt capability and there are dedicated service grant lines by which the microprocessor selects the device intended for a bus transaction. Each service request line is assigned a priority that corresponds to its device function. The system bus is organized with parallel address and data lines driven by tri-state devices. Each transaction on these lines is completed in a single bus cycle mainly for high speed operation. During each cycle, synchronization signals establish handshaking between the microprocessor and interface devices.

Typically, the microprocessor telemetry system produces a serial output from selected data inputs in the following manner. The timing countdown chain requests data from the Microprocessor. The microprocessor fetches a word processed during the previous word time and loads the downlink serializer. The serial output bit stream is produced automatically in the requesting device. The microprocessor then fetches the input select address for the next data sample from the format storage device. The address is presented to the input multiplexer by the microprocessor. The input multiplexer selects and converts, if necessary, data that was sampled. The microprocessor is notified when a conversion is complete and it accepts the converted data. This data is stored in the working storage memory until the next downlink serializer request is made. Variations in the operations described are possible and can be made under software control.
The processing “duty cycle” is the ultimate limitation on the functional complexity and speed (downlink bit rate) of any specific application. On applications to date, the “on” time is a small percentage of the available time between words. Accordingly, the microprocessor selected uses a high speed Schottky microcontroller. It consists of an interpreter with an eight-word instruction set and eight, 8-bit interface vector (IV) bytes which serve as input/output devices. The microcontroller with supporting circuits, Figure 2, constitutes the microprocessor. The firmware is contained in programmable read only memories (PROM) referred to as the program storage memory. A random access memory (RAM) is used as a scratch pad by the microprocessor and is called the working storage memory. Communications between interpreter and interface vectors is established over the local IV bus. Communications to the system bus from the microprocessor are made by four IV bytes. Sixteen address and sixteen data lines are provided on the system bus to accommodate high data transfer rates. Data lines are bidirectional and the address lines are unidirectional, outbound from the microprocessor. Two IV bytes are dedicated to the working storage memory for address, control and data. One IV byte outputs a binary code to a one-of-sixteen decoder for the selection of a device by a dedicated service grant line. The eighth IV byte monitors the binary output from a priority encoder. This encoder indicates which device is requesting microprocessor service, and it arbitrates in the event of multiple service requests.

When a device requests service, a power strobe is activated, powering the interpreter, the IV bytes and the program storage circuits. A pulse at power turn-on resets the interpreter program counter to zero. Processing begins when reset is released. The interpreter reads the priority encoder to identify which software module is needed for processing. Then it loads the appropriate IV bytes and grants service to the selected device. Third, it starts the sequencer which controls the transaction. After this the bus transaction is carried out independently of the microprocessor. Directionality of the bus data transfer to or from the IV bytes is controlled by a signal level presented to the sequencer. The microprocessor continuously monitors the sequencer until completion of bus activity. Upon completion, the microprocessor terminates service grant and resumes processing until all requirements of the current software module are complete. If no service requests are pending at that time, all strobed devices are powered down under software control.

The processing “duty cycle” is the ultimate limitation on the functional complexity and speed (downlink bit rate) of any specific application. On applications to date, the “on” time is a small percentage of the available time between words.
Figure 2 - Microprocessor Implementation. Handshaking with devices is controlled by bussed control functions from the sequencer. Data-on-line (DOL) to and Data Acknowledge (DACK) from the selected device form a closed loop which governs all information transfers.

A Typical Application - REQUIREMENTS  The generalized telemetry system described in the preceding section has been used for a classified satellite program under contract to the Naval Research Laboratory. Requirements for the system, from here on referred to as the Central Telemetry Controller (CTC), are as follows:

1. Pulse Code Modulated (PCM) output bit stream in real time
2. Delayed transmission through the use of a suitable storage medium
3. Fixed and alterable telemetry formats
4. Remote multiplexing and conversion of data (no central multiplexing)
5. Control outputs for the manipulation of other systems
6. Serial camnd inputs which load memory and change operating parameters such as bit rate and operating mode
7. Reconfiguration by relay commands
Interaction of these operations results in a sophisticated requirement as indicated by the following performance characteristics:

Once the system is powered up via relay closure, all operating parameters are initialized to a predetermined state. Alteration of these parameters may be made by instruction type commands which enter the CTC through a dedicated serial link from the command system. Two basic modes of operation are possible within the CTC; real time and delayed.

In the real time mode, transmission occurs as the data samples are taken (or within the delays imposed by processing). Ordering of data at the telemetry output is established by codes in the format storage memory. Two types of storage exist, fixed memory and an alterable memory. One or more fixed formats are contained in integrated circuits programmed by fusible link prior to launch, and the alterable format is loaded into an external core memory by the command system acting through the CTC. The core memory, functionally labeled the Bulk Storage Memory, is general purpose and is used for a variety of applications described later in this paper.

Each code word in the format storage specifies the address of one remote multiplexer and the address of one input channel of that multiplexer. The remotes are all connected to the CTC by a serial bus, termed the telemetry bus. The remote units are specialized by data types. They multiplex, condition and convert analog, bilevel or serial inputs to serial digital data. The telemetry bus is bidirectional and is time shared in such a way that no arbitration for the bus or handshaking is necessary for its use. Serial data is supplied to the CTC by the remotes and inserted into the telemetry output.

Format organization consists of lines and pages. A line is 40 words long and a page consists of from 1 to 64 lines depending on the format programmed in memory. Subcommutation exists to depths of 2, 4, 8 and 16, again under format control. Each line is divided into two segments. The first 8 words, referred to as the header, consists of sync codes and various status bits which indicate CTC and command system modes. The remaining 32 words of a line are telemetry data.

In the delayed mode, data samples are stored in the bulk storage memory until the CTC is instructed to transmit them to the ground. These samples may be taken simultaneously with the real time data under a completely separate format. When stored data is to be transmitted to the ground it is interleaved with real time data on a line basis into the output bit stream.

Data for delayed transmission may come from a memory associated with the command system as well as from the bulk storage memory. Control signals from the CTC cause
serial data from the command storage memory to be placed in the telemetry bus mentioned earlier, for direct placement into the CTC telemetry output.

**A Typical Application - IMPLEMENTATION.** The performance requirements of the CTC are indeed complex. However, application of the generalized microprocessor approach was relatively simple. In fact, only a small portion of the total processing capability was used.

Devices on the system bus (Figure 3) correlate with the CTY requirements listed earlier:

1. Timing and output module - a countdown chain which divides a clock to one of four bit rates selectable by ground command via the microprocessor, and a telemetry output device which converts parallel data to a serial output
2. Bulk storage memory interface
3. Fixed format storage memory containing 2048, 12 bit words
4. Remote telemetry bus transceiver
5. Command storage memory controller
6. Instruction command interface
7. Status monitors to sense internal configuration

Although relative complexities vary from device to device, the bus interaction is consistent and simple. Referring to Figure 4, the format storage memory containing input selection codes for telemetry data samples is discussed. High density, Electronically Programmable Read Only Memories (EPROM) are used for the storage elements. These elements are installed on carriers for easy replacement. Thus the ordering of input sampling can be changed up until the time of launch. 2048, 12-bit codes of format are provided for the CTC.

The format storage device is activated (powered up) by a service grant from the microprocessor. Not all devices are power strobed, but for the EPROMs, being high powered and having a low operational duty cycle, strobing is a distinct advantage. The address present on the CTC bus is read by the format memory at power turn-on and an input select code is immediately presented to the CTC bus data lines. A “data-on-line” control signal is generated by the microprocessor sequencer shortly after power turn-on. “Device acknowledge” is returned (after a delay for data settling) by the bus interface
**Figure 3** - CTC Application of the Generalized Architecture. Processing for the CTC requires 2048 words of program storage and 48 words of working storage. Four devices require service request capability and all eight require service grants.

controller to indicate that data is stable on the CFC bus. The microprocessor latches this data upon receipt of “device acknowledge” and deactivates the device.

Software main line coding, as previously described, is organized into discrete, high-speed software modules. Each module is dedicated to a particular service request. In the CTC there are four circuits requiring service requests and in the order of descending priority, they are:

**Figure 4** - A Typical Device on the CTC System Bus. The format generator associated with classical telemetry systems is here replaced by microprocessor-format storage interaction.
1. Initializer

2. Downlink serializer

3. Telemetry bus receiver

4. Instruction command interface

Processing for these requests and the nature of their functions are highlighted in the CTC operational description which follows.

Application of power to the CTC causes the microprocessor to undergo an initialization operation. A delayed service request initiates the operation automatically shortly after power to the CTC has stabilized. Initialization sets all operating parameters in the microprocessor working storage memory to a predetermined state. This operation is a complete software module in the main line executive. Henceforth, instruction commands from the ground control the CTC operating parameters until power is removed. The highest priority has been assigned to the initialization service request. The CTC would not perform correctly were this routine bypassed.

Once initialized, the CTC assumes the real time, fixed format operating mode. Downlink serializer service requests trigger the CTC operation. These requests come at the telemetry word rate and cause the microprocessor to power up and load data into the downlink serializer. The software which responds to downlink service requests is called the “word time” software module. This module causes the microprocessor to respond first by retrieving, from working storage, a word processed during the previous word time request. A bus transaction loads this word into the downlink serializer.

Processing then begins on the next word, which is specified by a word counter in working storage. If the next word is part of the header (described earlier in this paper) then the microprocessor constructs a sync code or status word. If the next word is data, the microprocessor accesses format storage to obtain the address of the data channel to be sampled. The microprocessor loads this “input select code” into the telemetry bus transceiver device. The microprocessor then returns itself to an idle state by deactivating the power switch under software control. The transceiver device transmits the input select code to a remote multiplexing unit. Data is sampled and converted to binary code by the remote unit and returned on the (serial) telemetry bus to the receiver portion of the bus transceiver. When all data is shifted in, the receiver generates a service request. The microprocessor is again powered on by the service request, takes returned data from the receiver, and loads it into working storage where it remains until the next word time request. The receiver service request is assigned priority three because its function is less critical to the CTC operation. The microprocessor returns to the idle mode once the process is complete.
Operational modes are changed by instruction commands from the ground. The instruction command interface device receives and stores serial commands from the command system. The interface generates a service request when the command is validated and the microprocessor receives the command information by a CTC bus transaction. In addition to instructing operational mode changes, these commands also load the bulk storage memory. The service routine for the instruction command interface device is contained in a separate software module. Service requests from this device are assigned priority four, because command receipt is considered to be an off-line process.

All other operational modes represent only variations of the real time, fixed format mode. For instance, the real time, alterable format mode differs from the fixed format version only in that the bulk storage memory is used as a source of input select codes. For the delayed mode of operation, two additional activities are performed with the real time process. First, input selection codes are obtained from the BSM via its interface device, and the converted data is stored in the bulk storage memory instead of being transmitted when first processed. Second, in the downlink output real time lines are interleaved with lines of stored data. The bulk, command, and working storage memories are all sources of data for the delayed mode.

The microprocessor handles the complicated performance requirements of the CTC quite easily. Processing requires only six percent of the time between telemetry words at the fastest bit rate. The CTC thus has plenty of room for expansion of its existing capability.

**Conclusion.** It has been demonstrated in the foregoing example that the generalized telemetry concept illustrated in this paper is indeed viable. The success is due to the utilization of a microprocessor to centralize general control functions. The advantages are impressive. The system is realized with fewer parts, is more reliable, is general purpose and has more capability than non-microprocessor telemetry systems.

There are limitations, however. The size, weight, and power are not dramatically improved. The central telemetry controller described in this paper, for example, weighs nine pounds, occupies 317 cubic inches and consumes three watts of power. These statistics are not unreasonable considering that the controller is 100% redundant.

Microprocessor applications nearly always feature far more capability than the immediate requirement. This additional processing power can be utilized by controlling other spacecraft systems with the telemetry processor, without notably increasing total system size, weight, or power. Partitioning of the spacecraft electronics takes on new dimensions when faced with the alternatives offered as more powerful architectures evolve. Many detailed studies will be necessary to optimize the tradeoffs.