An Integrated Payload Development and Operations System (IPOS) for PI Class Missions

Gerald B. Murphy
Design_Net Engineering Group LLC
1968 Mountain Maple Ave.
Highlands Ranch CO
80126
303-683-6544

Elaine Hansen
Director Space Grant College
University of Colorado at Boulder
Campus Box 520
Boulder CO
80309-0520
303-492-3141

Abstract. The IPOS approach recognizes that the real problems in achieving schedule, cost, and science return for PI class missions can only be dealt with by re-examining the basic architecture of payload and satellite systems. We must re-evaluate how they are built, integrated and operated. Reducing integration time and risk, lowering operations cost while increasing science return are absolutely necessary for today's new, low cost, fast turn-around missions. IPOS is systems engineered to match an innovative *architecture* with both an implementation *methodology* and enabling *technologies*. These are designed with recognition of the pitfalls that have historically been problems in satellite/payload integration, providing features that enable the PI to focus resources on development of the science instruments. IPOS is scaleable and targeted toward the UNEX, SMEX, MIDEX and Discovery class missions with a single responsible PI. It is designed to make it practical for the PI to integrate their payload with existing S/C or to "buy a ride" on commercial class vehicles.

To achieve these objectives, IPOS defines the entire science payload as a single "PI instrument" making integration with the S/C simpler, and giving the PI control of the entire payload both during development and throughout the mission. IPOS achieves low cost and reliability by using an enabling software bundle called the End-to-End Mission Operations System or "EEMOS" and hardware technology already developed by NASA, DoD, and the commercial sector. IPOS provides each of the Co-I instrument developers with power, command/data, and thermal control, plus enhanced centralized science data processing. IPOS is not simply another integrated payload, it is system engineered be an "end-to-end" set of services, and designed to give the user flexibility in its implementation without redesign. It can be configured for a variety of missions from simple to complex, with soft to hard environmental requirements, and with simple to redundant implementations.

Introduction

In today's world the PI class missions are selected from an elite group of highly competitive proposals. Anyone who has participated in this process understands that the squeeze between available dollars, a tight development schedule, aggressive science goals, and the need for managing not only the payload but the spacecraft procurement and integration has made this a risky effort, not just in proposing but in executing the job. Somehow, we must develop a system that lowers this development risk, allows more scientists and institutions to participate in the process, and enables PIs to focus more on the science aspects and instruments of the mission and less on the common hardware and software details common to all missions. More time and dollars should be focused toward developing advanced sensors and enabling those sensors to operate more intelligently and more autonomously.

The development of the IPOS system grew out of the extensive experience of the authors, lessons learned in how we *shouldn't* do things, and a desire to fix those problems once and for all. IPOS recognizes a broad class of mission, instrument, and environmental requirements; seeks the root causes of failure, schedule, and cost overruns; and provide services to the PI that dramatically increase their chance for success. The IPOS design seeks to achieve six major goals:

1) Simplify building science payloads by freeing instrument teams to concentrate on sensor devel-

opment--not power supplies, CPUs, interface documents, nor simulator development, etc.;

2) Present an environment for instrument scientists and subsystem designers that is *identical* from the first simulator interface test through mission ops;

3) Reduce by at least a factor of five the time (and cost) needed for integration of a science payload and the integration of that payload to the S/C system

4) Provide for gradual migration of autonomy to reduce mission operations cost;

5) Allow a broader spectrum of University participation in missions by providing much of the systems engineering and specialist engineering in a pre-designed yet tailorable package;

6) Enable certain missions that would otherwise not be possible by reducing development time, cost, and technical risk.

Achieving such goals requires more than simply "integrating" all of the instruments with a single processor to make a simpler interface to the spacecraft. What is needed is an integrated set of payload support services available prior to starting the design phase. That system must be robust, simple, and flexible. IPOS provides these integrated support services by providing: 1) a new payload subsystem architecture; 2) a support system for payload development; and 3) technologies which enable this architecture, and 4) an integrated data system that provides the users with a full set of operations services throughout payload development, test, and operations.

In the following sections we examine the hardware and software components of flight segment of IPOS and illustrate how they work together to change the way a payload is developed, integrated and operated.

IPOS Hardware Architecture

The IPOS architecture has the following features:

1) A central Payload Data Processor (PDP), contained in a payload subsystem package and coupled to individual Instrument Control Boards (ICBs) contained within each instrument. This enables a clean software firewall between code which handles the details of each instrument's operation (managed by the individual instrument controller and developed by the instrument team) and the PDP code, which handles the S/C communication, command and data management, packaging of the science data blocks, and inter-instrument processing.

2) A switched architecture PDP system bus based on ANSI standard RaceWay protocol and FPGA or ASIC interface chips. Bandwidth is not limited by bus contention. More processors are easily added for more horsepower with out the complication of symmetric multiprocessing system design. The switch fabric topology can be tailored to the processing and reliability needs of a mission.

3) A choice of high speed (selectable up to 1000Mb/s) and low speed (1Mb/s) serial links for the instrument to PDP data links. Selection of the interfaces to use can be made on a instrument by instrument basis with all links using simple serial protocols. The PDP has random access to all instrument data stored in global memory for data prioritization or compression, generation of "virtual instruments", implementation of autonomy rules, and enhanced science return;

4) A redundant central power system with ac distribution controlled by the Advanced Instrument Controller (AIC) chip developed by Phillips Laboratories and flight heritage power supply designs. This allows control and monitoring of each instrument's power and thermal status. IPOS also provides a power module for the instrument side of the interface giving regulated voltages, with an auxiliary output and separate ground for HV supplies. This subsystem controls all grounding and helps ensure EMC.

5) A modular system capable of evolving gracefully with time. This enables the system to easily take advantage of superior technology components as they become available.

IPOS Methodology

1) IPOS provides an End-to-End power, thermal, and data component for the payload, complete with a ground development unit. This enables a payload to quickly start development. Interfaces are defined, capabilities are defined, constraints are understood. The first development step (systems engineering) is complete.

2) IPOS recognizes the separation of responsibility between the PI and the instrument developers and provides each with the appropriate tools for their job. Projects always proceed with lower risk and

higher probability of schedule success when interfaces are clean and responsibilities well-defined.

3) IPOS allows a single interface between the S/C and payload and easily defined interfaces between the instruments and the payload subsystem. The ICD's are simple and the integration is simple. IPOS even provides interfaces to the instruments so the hardware has assured compatibility.

4) IPOS does not wait for integration to test compatibility. Providing a ground development system (GDS) identical to the flight system and complete with its operations and data system software assures all testing can be done prior to delivery to the PI.

5) IPOS not only gives the instrument developers the power module, the instrument controller, and a GDS, it also provides training in key software tools standardized across the payload. For example the use of the EEMOS to control an instrument and display data from the beginning of its design through mission ops provides across the board standardization and the EEMOS training is provided when the GDS is first delivered.

6) With its architecture, reliability engineering, parts selection, testing program, packaging, controlled interfaces, and redundant power system, IPOS builds in a level of payload *reliability* usually only achievable at a high cost and after a lot on non-recurring engineering.

7) Last, and very importantly, IPOS provides for the gradual evolution and migration of autonomy. Lower mission operation costs and better science return can be obtained with more autonomous operation, but autonomous operation must be achieved gradually, therefore, IPOS includes the EEMOS software tools which allow autonomy to be developed over time.

The IPOS Hardware Technologies

In order to enable the architecture and methodologies of IPOS to be effective, certain enabling technologies are needed. Some of the difficulties with previous attempts to achieve this level of integration have been do the compromises necessary because the technology just wasn't mature enough. The following list illustrates some of the IPOS technologies. It is worthy to note that IPOS itself does not develop new technology, it takes advantage of investments already made by the government and commercial sectors, selecting those appropriate for this application.

1) The Advanced Instrument Controller (AIC) developed by Phillip Lab is an augmented 8051 family controller packaged in 3D High Density Interconnect technology. The AIC has environmental robustness superior to most other technologies, is in a package as small as a postage stamp, and consumes very little power. Its processing capability is well matched to the needs of 90% of instruments being developed today. The AIC is provided (as a mezzanine card) to all instrument developers.

2) A centralized power conditioning system using a 80V AC distribution provides robust EMC compliant interfaces, high efficiency, and central control of instrument power as well as instrument thermal balance. This power system has the unique property that it powers the AIC board within an instrument even when the instrument is off.

3) RaceWay switched Architecture (an ANSI standard originally developed by Mercury Computing) allows for reliability, expandability, low latency, high coherence, and ease of S/W integration. Symmetric multi-processor systems are difficult to develop and test. The RACE bus allows simple expandability with adjustable redundancy and cross strapping. All processors on the bus have access to global memory, a Solid State Recorder.

4) The Mongoose V (R3000 based RISC processor) provides a space hardened processor capable of handling most instrument requirements. The RaceWay bus allows you to use more than one if needed. Options for future higher power expansion include the R6000 and Power PC.

5) High speed fiber channel interfaces allow the integration of high bandwidth data processing capability (up to 150Mbytes/s throughput).

The IPOS Hardware Overview

With this introduction and background into IPOS, it is appropriate to look at the top level hardware components and architecture. Figure 1 is an illustration of the IPOS definition of "integrated payload."

Several things should be emphasized. First, IPOS is *distributed*, that is, part of IPOS is located in a central housing referred to as the Integrated Payload Subsystem Control Module (IPSCM), and part of it is located *within* the instrument. This is true for both power component and the data component.

Secondly, The IPOS system is designed to be modular so that it can be sized to mission requirements by simply adding component boards which are already designed and whose layout is complete. IPOS can support the number and complexity of instruments required for a broad range of missions. The basic building block of this IPOS modularity is an IPSCM capable of handling 1 to 7 instruments depending on the power requirements and the degree of integration of the each instrument.

Note the two key components that make up the IPOS subsystem: a data component; and a power / thermal control component. These blocks and their components are described in more detail below.

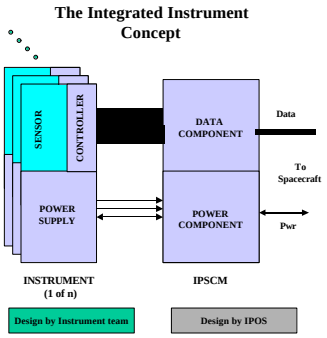


Figure 1

Data Component

Boards that make up the IPSCM PDP includes the following:

1) Processor board—holds the Mongoose V, on-board cache memory, interface to power control board, UARTS for serial interfaces, Ethernet port, and RaceWay interfaces;

2) Global Memory—Up to 2Gbyte of storage capability, this is really a Solid State Recorder used in a new way;

3) Instrument Interface Board—has the serial and fiber channel to RaceWay interface modules for the instruments;

4) S/C and GSE interface—contains high speed serial data link to GSE computer and a link to the S/C subsystem;

5) RaceWay switch module—the bus that ties it all together, this module may consist of one or several switches configured in a topology based on mission level requirements.

The interconnectivity of these modules through the RACEway bus is illustrated in Figure for a simple configuration.

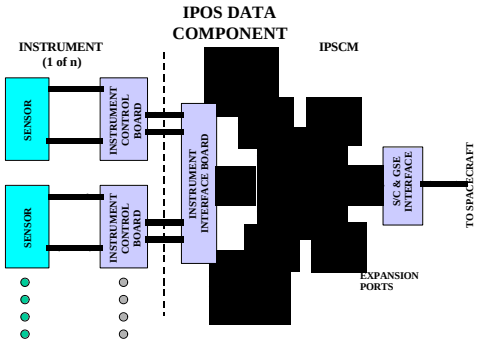


Figure 2

Table 1 summarizes the general specifications for the IPSCM data block or PDP. A key capability of this module is its *ability to link to another identical module* (see figure 2). These modules will have the capability of running independently (as many as desired can be chained together over the RaceWay bus) creating a multiprocessor system. Likewise, both the processor configuration and the switch fabric topology can be configured for levels of redundancy and reliability dictated by mission requirements. Figure 2 illustrates only one six port switch. In reality a number of ports can be linked together into a "switch fabric" whose topology can determine how many processors or auxiliary nodes such as instrument interface boards are used. The spare ports (2) on the baseline configuration allow a simple ring or connection through an intermediate node into a star configuration. The reader can immediately see the advantage of this switched architecture computer. All six ports can operate simultaneously, each capable of 160 Mbyte/s peak transfer rates.

Table 1 PDP Specifications

Main Processor	Mongoose V(baseline), up to 40 MIPS each, 1 to n available
Hardness	1 Mrad, LET, 80MeV-cm2/mg, latchup immune
Operating system	VxWorks
I/O	To GSE: Fast Ethernet To S/C: Configurable To other nodes: RaceWay
Instrument I/O	1Mb/s to 1000Mb/s data port Synchronous Serial cad port
Spacecraft Interface	Serial, selected for specific mission
Memory	SSR memory, configurable 4 Mbytes on board cache with hardened SRAM and EDAC
User Software	EEMOS (see later section for detailed description)
Redundancy	Configurable as required; any number of processors may be operated in parallel

Just as IPOS is modular in its Payload Data Processor at the level of Switch Topology or number of processors, IPOS is also modular and scalable to the number and type of instruments. We shall examine two boards in a little more detail to gain some insight into how the PDP acquires and manages the instrument data.

Instrument Interface Board

The Instrument Interface Board (IIB) can accommodate either very modest data rates or very high data rates from individual instruments. This is accomplished by utilizing two types of serial interfaces, a low speed bi-directional link which is a default and always present, and a scalable high speed fiber channel link which is optional. (See Figure 2) Each of the instrument interfaces on the IIB has a low speed synchronous serial channel (1Mb/s) for commands and housekeeping data and may have an optional high speed (up to 1000Mb/s) link for high data transfer speeds. The low speed channel is active at all times even when the instrument is off, the high speed link is only active when the instrument is on. All of the serial data links, no matter what their speed, follow a similar data transfer protocol.

Global Memory

The use of global memory is a very important part of the IPOS architecture. It is really a Solid State Recorder configured to act in a very special way so we give it a new name to emphasize that it is much more than just a recorder.

Global memory serves several important functions. The first is that, unlike most central data processing units, the PDP does not "gather" instrument data, instead the instruments write their data directly to global memory where it becomes available to the PDP for further processing. The second is that global memory stores these data until the autonomy "engine" of EEMOS accesses it to identify trends, spot events, or simply filter data according to priority when too much is being collected to send down. These data are available for PDP processing for some fixed interval of time after which global memory is used for its third function. It is also the storage location for data which have been prioritized, compressed and arranged into Science Formatted Data Blocks (SFDBs). Although global memory serves several different functions, in reality the hardware is almost identical to the traditional solid state recorder.

The global memory is segmented into circular buffers each assigned to a different instrument data interface.

Instrument Control Board

The Data component which ties the instrument to the IPSCM lies within the instrument and is called the Instrument Control Board (ICB).

The ICB is based on the Advanced Instrument Controller (AIC) developed by Phillips Laboratories. Specifications for the AIC are outlined in Table 2. The design has the advantage that its radiation hardness can be procured to several levels. For the baseline IPOS, a low level of hardness is used coupled with a chip level watchdog for LET events. Should some missions require instruments with a higher level of reliability, the hardened version can be used. Consuming only 50 milliwatts nominal (at 5 MHz clock speed) the instrument controller can afford to be on all the time (even with the instrument OFF). The AIC actually consumes much less than that because its clock is throttled down at

instrument turn off and then back up at instrument turn on. This rather unique approach (having an instrument controller active even when the instrument is off) is made possible by the central power / thermal control module which discussed in the next section.

Figure 3 represents a block diagram of the AIC based Instrument Control Board. Its interface (syschronous serial) to the IPSCM has already been discussed and is part of the AIC hybrid. The board, given to the instrument team as a mezzanine card may also have the high speed fiber channel link on it if that option is used.

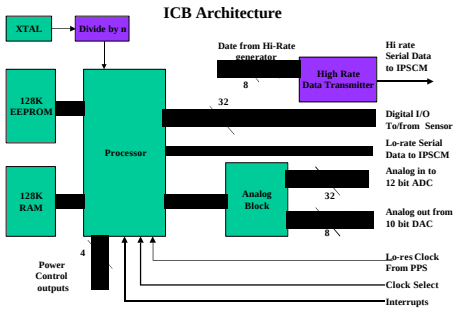


Figure 3

Table 2 AIC specifications

General

Supply Voltage	+5, +3.3
Power dissipation	50mw (@5MHz) scale by speed
Temp range	-120C to +80C
Oscillator	Internal or external (varactor or crystal)
Clock speed	500 kHz to 16 Mhz selectable
Physical size	footprint = 1.4" x .9"

I/O:	
32 user I/O	digital, definable, bit/byte addressable
32 Analog input	to 12 bit ADC, random access, sequential access, continuous channel sample modes, 1mV/bit
8 Analog output	from 10 bit DAC
4 Async I/O	Mux'd, single ended, full duplex, 8 or 9 bit programmable, framing error detect, baud rates selectable, separate TX/RX for each channel
2 sync I/O	Mux'd, single ended, internal clock, separate data out/in and clock for each
4 outputs	power FET control outputs

Processor:	
Instruction set	8051 super-set
Bus	internal 8 bit
Counter /timers	3 16 bit, up down, programmable clock out
RAM	128K internal SRAM
EEPROM	128K internal EEPROM
Interrupt	4 level interrupt priority

Power / Thermal Component

Just like the data component, the Payload Power Subsystem (PPS) is distributed having part of the component in the IPSCM and part in the instrument. This subsystem is designed to provide centralized power distribution and control for PI class missions. It provides a single redundant interface to the S/C power system and switched power for all of the instruments. It is designed to work with the IPOS central data system providing power and thermal status by using the same AIC based microcontroller discussed above. All power components are to be manufactured by Battel Engineering with a high degree of space flight

heritage. The subsystem is sized to supply up to 100 watts for up to eight instruments and is modular.

When the payload is activated, *all of the instrument control boards are automatically switched on* and placed in standby (low power) mode. Although main instrument power is off, the central unit maintains knowledge about the temperature, controls survival heater power, and can monitor relay or other key state parameters of all the instruments. The low power overhead of the AIC makes this practical. This centralized knowledge allows for ease of management of all payload power. Likewise, since the controllers are already active, they can watch the instrument power come up, and act immediately should any off-nominal condition arise. Figure 4 is a system level diagram of the PPS.

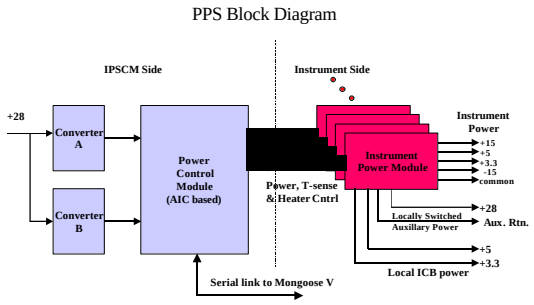


Figure 4

As shown above there are two parts to the IPOS PPS, the first is the power supply and control module which mounts in the IPSCM. The second is the Instrument Power Module which is provided to the instrument developer as part of the IPOS system (just as the AIC board is provided) and mounted within its chassis. The instrument power module is a small hybrid unit packaged like off-the-shelf power converters used by many instruments today. Specifications for the PPS are given in Table 3

Table 3 PPS Specifications and Functions

- Dual redundant supply with AIC based control unit housed in the IPSCM
- Provides centralized control and monitoring of all payload power
- Isolation of all secondary power with centralized grounding and EMC control
- Uses 80V AC distribution with post regulation for clean and isolated power.
- At start-up, +5 volt central bus powers all instrument AICs plus the power control AIC
- Supports up to 7 instrument modules with total power of 100 watts
- Input voltage range : 21V to 36V
- Monitors 3 temperatures per instrument (one at instrument power board) and controls 2 survival or replacement heaters per instrument while instruments are off.
- Uploadable current limits (by mode) for each instrument
- Background power consumption (instruments off) due to AIC's less than 300mw
- additional control modules can be stacked to provide more power for more payload
- Instrument side of the interface provides ±15, +5, and +3.3 regulated outputs with additional, separately switched and grounded 28V auxiliary supply for HV, motors, etc.
- Point-to-point connection of instruments with 15 wire, 26 ga interface utilizing miniature connectors.

IPOS Data Flow

Data transfers from the instrument are routed to the global memory address space (virtual memory mapped) and stored for use by the PDP and ultimate downlink. One of the important distinctions to keep in mind about the IPOS architecture is that instead of a central data processor *collecting* data from each instrument, the instruments *write* their data directly to global memory asynchronously with each other and other data operations. The PDP's job is to keep track of where that data is, by managing certain data pointers, process the data (e.g. prioritization or compression), and control the instrument clocks.

The IPOS hardware architecture enables EEMOS software to perform its functions efficiently and accommodates a wide variety of instruments and data flows.

When the types of realtime data acquisition systems embodied by a number of different payload elements are examined, their requirements and characteristics can be classified according to the following 4 dimensional parameter space: data rate, frame size, data mode, and timing accuracy.

DATA RATE: Instruments may have average data rates that range from less than a kilobit/s up to

many megabits/s. In addition to an average rate, there are some instruments whose data handling requirements are driven by their burst rate.

FRAME SIZE: A frame is defined as a self contained or logical unit of data from the instrument, e.g. a string of time-ordered samples, a complete sampling of a number of instrument sensors, or single readout from a CCD, etc. Within the PDP, data are compared or processed on a frame by frame basis. Different types of instruments can have frames of many different sizes.

DATA MODE: Data mode refers to the manner in which the instrument takes data, that is, fixed time sampled, event driven (high energy particle or photon counters are examples), or variable over a wide range being sometimes low and at other times quite high. To coordinate observations of instruments operating in different modes, the data structures and protocols must be organized such that the system "doesn't care" what kind of data mode an instrument may have.

TIMING ACCURACY: For some instruments, highly accurate time tags are required for certain "events" while others need timing information attached only occasionally because samples are regular or data are simply not sampled that often. Still others need to correlate timing of their data with that of another instrument.

The following sections offer a glimpse into the details of the data handling capability of the IPOS system.

Data Transfer from Instrument to PDP

All data transfers from a particular instrument are assigned to a specified address segment in global memory allocated according to the size and frequency of their data frames. Each of these address segments is managed as a circular buffer. The PDP controls the upper bits of the address to these buffers while the instrument control processor controls the lower order bits, writing each frame into as sequential segment of the circular buffer. The circular buffer size for each instrument is allocated in such a way such that all instruments can store approximately the *same time period's* worth of data before the buffers "wrap" around. This allows equal "look back" intervals in time by the PDP for all instruments. For instruments with more than one data interface, (e.g. they use both the low speed and high speed serial interfaces), the PDP controls each stream of data and its addressing separately giving each a circular buffer of the appropriate size.

Instruments select a *fixed frame size* for each their interfaces (the frame transfer *rate or format* may depend on the instrument state). Frame sizes for low and hi rate streams may be different.

Header information in each instrument data frame (whether low rate or high rate) tells the IIB "on ramp" chip what global memory address to begin writing the data, the size of the data frame, the instrument ID and status (for later processing), and instrument clock at the start of the frame, (16 bits of coarse time and 16 bits of fine time) Up to 16 bits of "fine time" may be added *within* the data frame for high accuracy data tagging if required by the instrument but the low resolution time bits are only stamped at the beginning of the frame. There can be up to 4 formats within the instrument frame dependent on instrument mode.

Certain parameters in the header (Instrument time, memory address) are latched by the interface chip and read by the PDP to create tables that track of the data for later retrieval. Once written into global memory the data may reside there for the "look-back" interval and available for random access by the PDP before being packaged into the Science Format Data Blocks (SFDBs) containing a complete set of payload data for the given time interval.

It is important to note that the IIB interface chip allows the *switch topology to be transparent to the instrument*. Likewise, very high data speeds can be accommodated with relatively little horsepower required from the PDP's microprocessor. All of this transfer is managed in hardware over the RaceWay bus.

In summary, the manner in which data is transferred between the instrument and the PDP:

- Allows different instruments to have different frame sizes;
- Accommodates multiple frame formats for each instrument's serial stream;
- Allows instruments to vary their frame rate at will since each is separately time tagged;
- Accommodates various timing accuracy requirements with the use of coarse and fine time clocks;
- Preserves integrity of the data in such a way such that the PDP may access it, compare it with data from other instruments, further

process it, or apply autonomy rules based on its values or changes in values.

Data Timing Accuracy

A difficult and often intractable problem in designing a real time data system is the issue of time keeping. This is made difficult because different instruments usually have different accuracy and resolution requirements ranging from no requirement at all to correlation with other data down to fractions of milliseconds or even microseconds. Since the IPOS is being designed as a general "data acquisition and analysis" system, it must have the capability to correlate different data sets in time, interpolate data sets, and it must do so with a high degree of flexibility. IPOS is designed to satisfy the following requirements:

1. It must allow diverse payload elements to time-tag their data to an accuracy of at least 100μs.
2. It shall allow all data frames within reasonable "look back" periods (perhaps the time interval between S/C ground contacts which may be 12 to 24 hours) to have a unique time-tag.
3. It must minimize the downlink telemetry space required for time-tags.
4. It shall provide the capability to *easily* time-correlate data sets from different instruments.
5. It must be flexible in its hardware implementation and not depend on the characteristics of a specific processor.
6. IPOS must be able to relate its internal clock to an external one which ultimately tied to GMT.

The timing system breaks the 32 bits of clock data into two distinct resolutions. A low resolution clock, T1 and a high resolution clock, T0. Each is 16 bits. T1 is *synchronized* to the master crystal on the PCM. T0 clocks are *generated on each individual AIC board and may run at different frequencies*.

Two dividers set the clocks (reference Figure 5). First, there is a divide by 'n' (8 bits) coming right out of the crystal. This determines LSB of the T0 clock. For the AICs 'n' but be greater that 12. Although it is not necessary, selecting 'n' to be the same on the power control AIC and the instruments' AICs means *that each have a high frequency clock running at the same rate*.

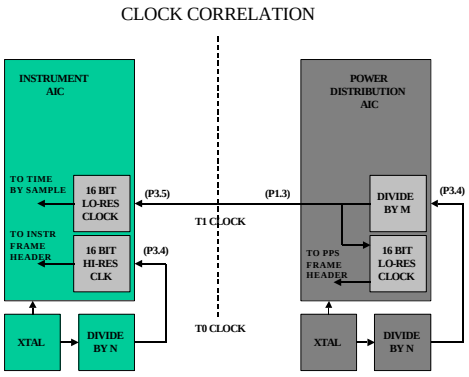


Figure 5

The second place where a choice must be made is a 'software' divide by 'm' inside the power controller AIC. 'M' is a 16 bit divide so can be any power of 2 up to 65,536. The value of 'm' determines the low frequency output clock rate T1 distributed *to all* of the instruments. By selecting 'n' and 'm' one can determine:

1. the highest resolution time tag available (hardware limit is $1/12^{th}$ the xtal frequency);
2. the longest time before an instrument's time stamp "rolls over" and is no longer unique;
3. the amount of overlap (if any) between the low resolution clock and the high resolution clock(s).

Although each instrument time-stamps its data frames, it is important to <u>align</u> all of the instrument clocks so that the PDP can later correlate data time. This is done by an interrupt sent by the Power Controller AIC to each of the instrument controllers *before* the instruments are turned on.

Timing is also preserved though possible upsets or PORs and can be synchronized to GMT once per ground contact.

The PDP keeps track of instrument data frames by their lo-res time (T1). Since the T1 time word is synchronized across all instruments, T1 is used by the PDP to correlate the data across the payload and with GMT.

<u>Processing Within the PDP</u>

Before discussing the EEMOS in detail, it is useful to have an overview of the data operations. Figure 6 is a flow chart representation of the PDP processing. There are four basic steps: 1) an instrument writes data directly to the global memory; 2)

the PDP has an opportunity to operate on that data with the EEMOS software; 3) the PDP "blocks" the data into SFDBs in order of the data priority and applying compression algorithms as required; 4) data are stored in a separate global memory circular buffer awaiting packetization and transmission to the ground.

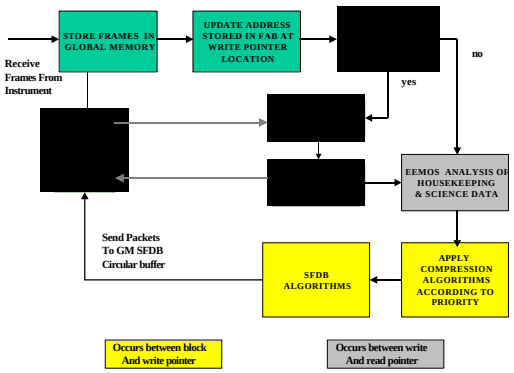


Figure 6 Instrument Data Processing by PDP

While in global memory, data from the instruments are stored in circular buffers with a separate buffer assigned to each instrument low speed and high speed stream. In addition to these buffers there in another circular buffer which contains the entire payload data in the SFDB format and ready for packetization and shipment to the ground.

After being written to the instruments circular buffer, data for each instrument reside within global memory for a specified "look-back time" during which the PDP may access it to perform any one of a number of operations:

1. Using EEMOS tools, the PDP may look at certain values of engineering data or science data applying rules;

2. The PDP may compare one instrument data with another in order to identify an event, or determine the priority of a data frame;

3. Data may be examined and prioritized, then placed back in memory with a priority flag;

4. Data may be examined for trending;

5. Data may be retrieved and compressed according to a pre-determined algorithm in preparation for its blocking and eventual transmission to the ground;

7. Engineering unit transformations may be applied and the data written to an Ethernet port immediately after reception in global memory.

In the next section we examine the detail data/command services that the EEMOS provides.

THE IPOS END-TO-END MISSION OPERATIONS SYSTEM (EEMOS)

A major design driver for future Missions is cost. Significant cost reductions are needed in developing ground and space-based mission operations systems, in prelaunch testing, and in operations after launch. To accomplish these cost reductions while supporting today's more sophisticated, computer-literate users and enabling these users to work from their own offices, future mission operations systems must be automated, easy to use, distributed, robust, efficient, interactive, secure, and must utilize common tools and software [1].

The students and faculty of the University of Colorado's Space Grant Consortium have developed an End-to-End Mission Operations System, or "EEMOS," architecture designed to include all of these attributes, have demonstrated this system on space applications, and is preparing to demonstrate the full suite of EEMOS capabilities on a small satellite called the CITIZEN EXPLORER. In the first application, called "DATA" (Distribution and Automation Technology Advancement), many EEMOS elements were demonstrated on a Shuttle payload flown in August 1997 [2]. In the third application, the EEMOS will be part of a student-developed Earth-observing satellite planned for launch in 1999.

EEMOS Description

The EEMOS architecture supports the seven basic mission operations functions, illustrated in Figure 7, needed to operate a space mission. These seven basic functions are to: 1) *Plan and Schedule*; 2) *Command*; 3) *Monitor*; 4) *Analyze*; 5) *Provide Data Services* for data communications, security, computing, capture, storage, processing, and distribution; 6) *Develop, maintain, and train* the people, procedures, software, and hardware that make up the EEMOS; and 7) *Manage* its development and use.

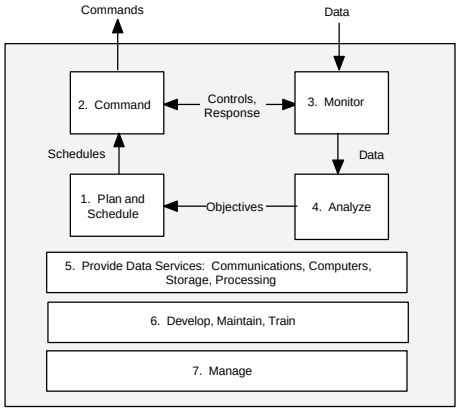


Fig. 7 The EEMOS Architecture supports the seven basic mission operations functions.

Replication of Software Tools and Applications

In the EEMOS architecture, these seven functional capabilities are replicated for different user teams, at different locations, for different uses [4]. For example, operators in the central Mission Control are responsible for: coordinating the plans and schedules of the science payload, the spacecraft subsystems, and the operations network; commanding and monitoring these system elements; receiving, handling, storing, and distributing real-time data; and analyzing the performance of the overall system.

Payload scientists, on the other hand, are concerned with the operation of their specific scientific instrument. They plan and schedule future instrument activities, command and monitor instrument activities, receive and process instrument data, and analyze the performance of the instruments and their measurements.

The operators in central Mission Control and the scientists operating their specific instrument perform the same seven functions. The same basic set of software tools and applications can therefore be used to support these diverse groups. These basic seven functions are also needed by the spacecraft subsystem teams who operate the spacecraft, and by the Flight Dynamics team that controls the spacecraft trajectory. These same seven functions are also needed on-board. A subset of these seven functions is also needed by science guest investigators and student investigators who need the same analysis tools as the other mission teams.

Distributed Operations Sites

The distributed arrangement of mission users is illustrated in Figure 8. The distributed group of scientists, consisting of guest investigators, student investigators, and distributed science data centers, is coordinated by the instrument science team. In turn, the science team, spacecraft engineering team, and the NAV team are coordinated by the Mission Control Center. Teams are not necessarily located at one site, but instead may be distributed across numerous NASA centers, universities, industries, and schools.

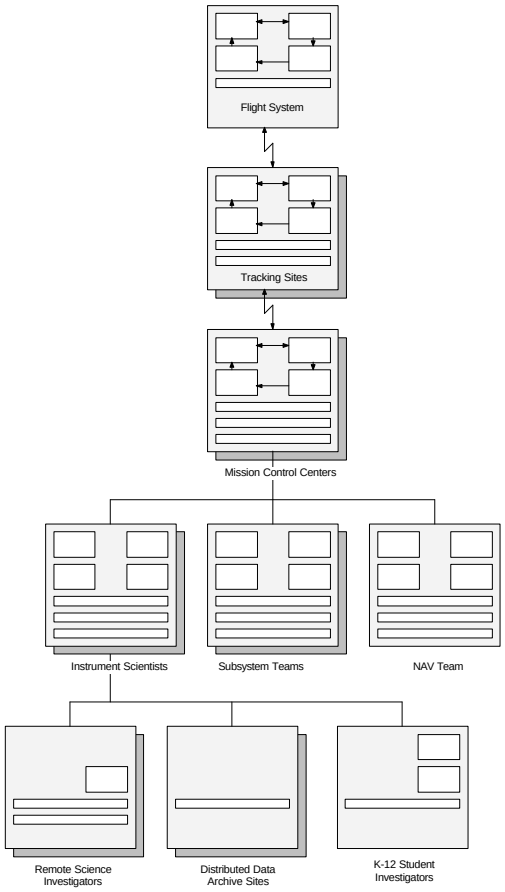


Fig. 8 Similar functions are performed by each ops team in the distributed arrangement of space and ground sites

A key feature of the End-to-End Mission Operations System approach is that it includes the on-board flight system as well as the ground system. In this EEMOS, most of the functions of the Mission Control Center are also included in the flight system. Thus, the flight system has the ability to reschedule activities, to command and monitor current operations, to process and store data, and to evaluate these data. The tools used on the flight system are a copy of the tools used in the ground portion of the EEMOS.

Consistent EEMOS Architecture Throughout Project Lifecycle

The EEMOS plays a key role in the project beginning at the start of the mission design phase and continuing through the flight mission and beyond, as illustrated in Figure 9. Throughout these phases, the EEMOS evolves to support the project's needs. In the early design phase, the EEMOS exists as a set of core operations capabilities with a simulated spacecraft and payload.

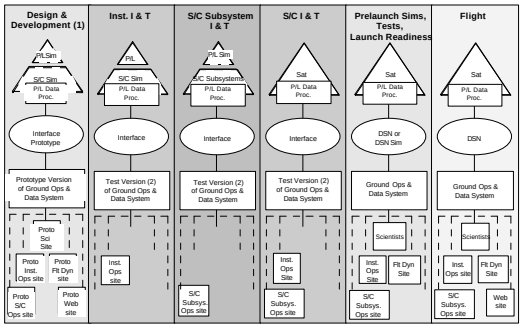


Fig. 9 The EEMOS is designed to be a part of the mission from beginning to end and to evolve as needed to support the project phases.

As the design matures, the EEMOS supports the design effort serving as an end-to-end mission testbed — with simulated instruments, spacecraft elements, and users, and with only critical operations functions and interfaces. This testbed evolves throughout design and development to include simulations of user teams, integrated software tools for the full suite of operations functions, simulations of each flight instrument and subsystem, and the ability to replace simulated flight units with their actual hardware counterparts.

A portable version of the EEMOS, as an integrated set of software tools and applications, can be located at each of the science instrument development sites. This portable EEMOS can serve to support functional and interface testing, calibration, and user training.

The EEMOS will also support spacecraft integration and test by supporting test scheduling, commanding, health and performance monitoring, data services, and analysis, and by providing a simulation of the science instruments. During this time, the EEMOS will be evaluated by the project scientists and engineers who will be able to request improvements while there is ample time to make these improvements.

The next project phase is satellite integration and test. The EEMOS will support this phase by providing the tools, applications, and support services to plan, control, monitor, and analyze functional and calibration tests. The scientists and engineers involved in this phase can evaluate the EEMOS and make suggestions for improvements while there is still time to incorporate these changes.

After the flight system has been integrated, the EEMOS with its human, procedural, software and hardware components will be tested as an end-to-end system. These tests will consist of launch simulations, mission simulations, encounter simulations, user certification exercises, and interactions with remote scientists and schools. Again, these users will evaluate the EEMOS and improvements made.

The flight version of the EEMOS will be the same as that used and evolved throughout the project. This evolution need not end after launch, but may continue as improvements are needed and as new applications are available.

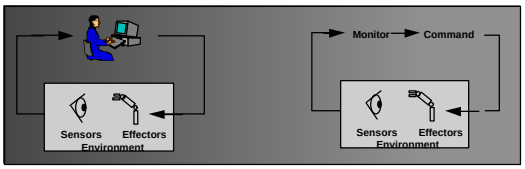
MULTI-AGENT SHARED CONTROL AUTOMATION

The EEMOS architecture facilitates end-to-end automation. Many previous space systems included automated functions, but these were usually limited to specific functions that were well-defined. Typical automation was also function-specific (e.g. attitude control). Harder-to-model functions such as subsystem health and status monitoring were not substantially automated. Significant operator attention was still required to monitor tasks and handle exceptions since it was difficult to translate operational experience to automated functionality.

On the other hand, since most space systems cannot be completely teleoperated due to communication time delays (Figure 10).

State-of-the-art space systems are therefore a mix of teleoperations and autonomy. Space systems would ideally operate fully autonomously as depicted in Figure 10. But full autonomy is difficult to achieve due to unpredictable variations of the operational environment. Significant automation advances can be made by automating tasks that are difficult to model and define in advance, but may be fairly easy to automate as operations experience is gained during a mission. The approach taken in this research is to define an EEMOS framework for flexible, evolutionary automation and for

"shared control" of space systems by people and automation.



Teleoperations *Full Autonomy*

Fig. 10 State-of-the-art automation is a mix of teleoperations and autonomy.

The multi-agent design, to implement the shared control concept, incorporates a remote agent (remote to the operator, but local to flight sensors and actuators) and a ground agent (local to operator interface) as depicted in Figure 11.

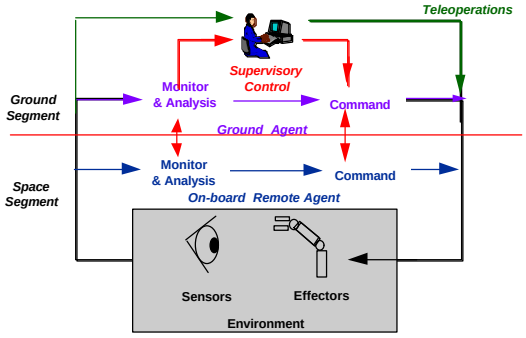


Fig. 11 Shared Control Architecture for a Mix of Teleoperations and Autonomy

The agents link event monitoring with analysis of their environment and payload state to produce goal-oriented commands to achieve specific mission objectives. The current testbed being evaluated at the University of Colorado's Space Grant Consortium includes event detection using the performance analysis tool SELMON (SElective MONitoring) [5] to detect behavioral changes from mode-based behavioral references. SCL (Spacecraft Command Language) is used for rule-based inference monitoring and for generating command reactions to events. A number of automated intermediate steps are taken by the system when linking events to commanded reactions in the current system, including: state monitoring, event detection, event classification, reaction selection, and command execution (Figure 12). Reactions can handle classifiable events, leaving those that cannot be reliably detected or classified, to be handled by the operator. This type of operator interaction with the cooperative agent system characterizes shared control. Maintenance of the system is distributed between the on-board remote agent, the operator's ground agent, and the operator.

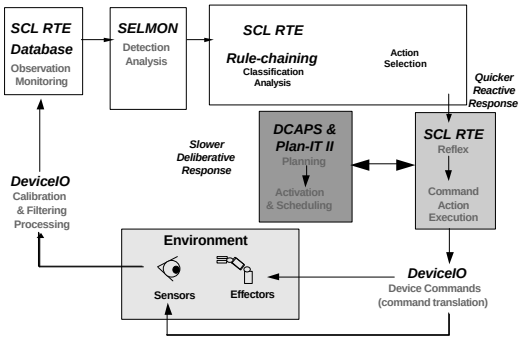


Fig. 12 The Remote Agent takes a series of steps to link monitored events to commanded reactions.

Reactive to Deliberative Agent Module Integration

A key consideration of this shared control architecture is coordination of the agents and operators to meet observing goals, short-term schedules, and realtime control requirements. In order to provide this coordination, the remote and ground agents are composed of deliberative and reactive agent functions. Reactive agent functions provide interfaces to deliberative agent functions (Figure 13). Multiple layers of agent functionality are envisioned, ranging from continuous reaction automation (e.g. digital control), to event-based automation (e.g. fault handling and opportunity recognition), to planning and resource management automation. The current EEMOS architecture includes the reactive event-based agent automation which provides for parametric control from the deliberative planning and resource management agent. SCL rules may be activated and deactivated by the deliberative agent, and likewise, constraints on device commands may be activated and deactivated. Examples of the parametric control of the event-based reactive agent by the deliberative agent include: sensor sampling rates, data calibration, event detection thresholds, classification rules, and action selection rules. Rules may be activated, deactivated, and added to the rulebase.

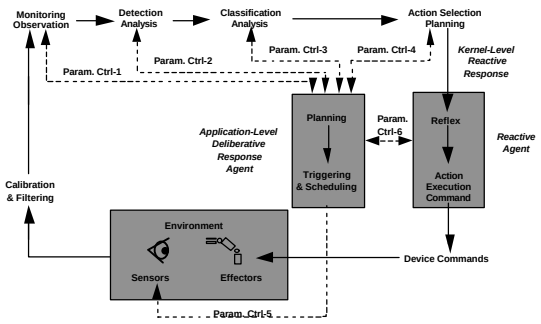


Fig. 13 Parametric Control of the Reactive Agent by the Deliberative Agent

Hierarchical agent architectures have been investigated on a theoretical and experimental level [6], and have also been used in robotics [7] and systems automation [8]. However, to our knowledge, shared control using this reactive to deliberative agent automation scheme has not yet been evaluated. One reason is that this type of control can be difficult to implement, to verify, and to determine its realtime performance. We are addressing these issues by incorporating modules for which realtime performance and correctness can be isolated in three levels — continuous digital control, event-based reaction, and goal-oriented epoch-based planning. By carefully controlling the interfaces between these levels, this type of advanced automation can be safely and cost effectively deployed.

Agent Realtime Performance

Realtime performance of this system is critical since it includes continuous control as well as event-driven and time-based automation. Reliability can be controlled through the parametric control interfaces which allow for automation activation/deactivation and updates to rules, constraints, and scripts. Reliability with respect to realtime performance is also important to the system. The implementation must include a performance range from "guaranteed" hard realtime to soft "performance oriented" realtime execution [11]. A missed deadline in a digital control loop could mean loss of vehicle control, as an example of the need for guaranteed performance. However, for the event-based agent, some latitude in the response time to an event may be acceptable — such that failure to meet the deadline does not result in total failure, but in performance degradation.

The design allows for decomposition of functionality according to timing criticality. Digital control loops may be handled as realtime kernel threads; event-based automation as soft-realtime execution sequences; and planning as best-effort background execution. The overall goal in this decomposition by functionality and realtime performance is to provide a systematic approach for verifying critical low-level modules on which higher-level automation is built, to achieve improved performance, and to enable decreased operator workload.

SUMMARY

The EEMOS system described here, with its support of evolutionary automation, will enable significant cost savings in the development phase, the pre-launch test phase, and the flight operations phase [12]. At the same time, the EEMOS will support today's sophisticated, computer-literate users, and enable these users to work from their distributed home institutions. The EEMOS is:

Automated, to react quickly to events or anomalies through its reactive agent, to react deliberately to longer-term situations through its deliberative agent, and to migrate operational experience to automated functionality.

Progressive to enable shared control by both onboard "remote" agents and by ground operators — and to enable hard-to-model functions to be automated by migrating sequences to the onboard system which have been proven through operational experience on the ground.

Responsive to unforeseen situations through the incorporation of a set of agents including a quick, reactive agent, a slow, deliberative agent, and a supervisor who is part of the ground system.

Self-Monitoring and Reporting to enable the flight system performance to be observed, detected, and classified, and for resultant actions to be selected, executed, and reported.

Easy-to-Use through the inclusion of Graphical User Interfaces which can be evaluated and improved throughout the multi-year design-development-test- integration-and operations phases of the project.

Robust due to the extended opportunity to use, evaluate, and debug the EEMOS prior to the launch, and to the architecture which reduces the total software that is used, tested, and maintained by replicating a common set of software tools and applications for each of the operations teams.

Efficient and Predictable. The performance of on-board agents will be efficient and predictable by establishing priorities for threads and protecting these threads for execution.

Distributed to involve a distributed set of users at remote sites — including university student controllers and analysts — by incorporating a distributed organization, security protocols, and a communications capability.

Evolvable. The EEMOS evolves throughout the entire pre-launch phase to ensure that it can be evolved during the flight phase as well. This evolution is largely enabled by the use of common tools and applications on the ground and on-board which enable activities to migrate from a ground user, to ground automation, to on-board automation.

Open through the use of modular software and standardized interfaces.

Interactive and Cooperative by enabling autonomous operations, giving the supervisor priority over automated operations, and giving the ground Mission Controllers priority over all functions.

Integrated Space-Ground Architecture by replicating software tools and applications for common functions on-board and at distributed ground sites.

Consistent by enabling one EEMOS to be used throughout all phases of the mission from early design, throughout development, test, integration, launch, mission operations, and post-mission analysis.

Secure by incorporating firewalls and security protocols in the communications system.

Generic to enable one EEMOS architecture with one set of tools and applications software to support a range of space missions including Earth-orbiting satellites, solar probes, communications satellites, missions to the outer planets, and beyond.

REFERENCES

[1] E. Hansen, S. Siewert, "Autonomous Operations for Small, Low-Cost Missions: SME, EOR (STEDI), DATA, and Pluto Express", Proc. on Autonomous Lights-Out Operations, 1996.

[2] E. Hansen, "Distribution and Automation Technology Advancement," NASA INSTEP Proposal, 1993.

[3] J. R. Stuart, "Teledesic Global Wireless Broadband Network: Space Infrastructure Architecture, Design Features and Technologies," NASA/Aerospace First Int. Conf. in Integrated Micro-Nanotechnology for Space Applications, 1995.

[4] E. Hansen, F. Lacy, "OSSA Payload Communication/Information Systems Analysis & Trade Study: Systems, Services, and Communications," NASA OSSA Report, 1990.

[5] Doyle, R., "Determining the Loci of Anomalies Using Minimal Causal Models," Int. Joint Conf. on Artificial Intelligence, Montreal, Canada, August, 1995.

[6] Brooks, R., "Intelligence Without Reason", In Steels, L. and Brooks, R., eds., The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1995.

[7] Russel, S. J., and Norvig, P., Artificial Intelligence: A Modern Approach, pp. 786-790, Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[8] Simmons, R., "Towards Reliable Autonomous Agents", AIAA Spring Symposium on Software Architectures, Stanford, CA, March 1995.

[9] Stankovic, J., "Misconceptions About Real-Time Computing: A Serious Problem for Next-Generation Systems", IEEE Computer, Vol. 21, No. 10, October, 1988.

[10] Stankovic, J., Spuri, M., et al., "Implications of Classical Scheduling Results for Real-Time Systems", IEEE Computer, June 1995.

[11] Burns, A., "Scheduling Hard Real-Time Systems: A Review", Software Engineering Journal, May 1991.

[12] E. Hansen, "Lowering the Cost of Spacecraft Operations: Lessons Learned from SME," Proc. of AIAA 26th Aerospace Sciences Meeting, 1988.