Ground Support Equipment for Small Satellites

John L. D'Ausilio[1]

INTRASPACE Corporation, North Salt Lake, UT

Traditional Electrical Ground Support Equipment (EGSE) is composed largely of custom designed equipment costing hundreds of thousands of dollars and usable only for the spacecraft for which it was designed. Due to the budgetary restraints associated with small satellite programs, non-traditional alternatives were explored. This paper describes an approach to EGSE design which utilizes commercially available products and costs considerably less than traditional equipment. Reusability of the equipment for subsequent spacecraft designs or for integration into ground station equipment is discussed. Various options are available for enhancement of the system to accommodate different telemetry rate and storage requirements, simulation needs, and input/output capacities.

Acronyms

EGSE	Electronic Ground Support Equipment
A/D	Analog to Digital
D/A	Digital to Analog
TLM	Telemetry
EISA	Extended Industry Standard Architecture
DMA	Direct Memory Access
RAM	Random Access Memory
SBC	Single Board Computer
SCSI	Small Computer Storage Interface
OBP	On-Board Processor
TCP/IP	Transmission Control Protocol/ Internet Protocol

I. Introduction

Requirements for EGSE for small satellites vary as widely as requirements for the satellites themselves. The example system described in this paper was developed over a nine month period for use with a small, three-axis stabilized spacecraft with a full complement of sensor equipment (Figures 1 and 2). The basic requirements for the system included command upload with scripting capability, telemetry download with raw and engineering displays, static simulation of about 60 analog signals from avionics equipment on board, and dynamic stimulation of the simulated avionics signals for attitude control algorithm testing. Additionally, the capability to record and analyze telemetry and memory dumps was desirable.

II. Available Architectures

The cost constraints on this project forced the engineering team into a software approach to the design. Ideally, this system would consist exclusively of commercially available equipment and would require a minimum of custom-designed components. The EISA (PC-clone) bus was the first to be considered, but the lack of available in-cabinet I/O configurations and the limited capabilities of the interrupt and DMA structures made it undesirable for this approach. Examination of the VMEbus,

1 Manager , Ground Support Systems

Multibus, and VAX architectures showed all to be acceptable, and a VME system manufactured by Motorola Microsystems was chosen. This system provided an easy to implement multiprocessor system utilizing UNIX System V for the main processor and the pSOS+ real-time kernel for the slave processors. A development system was provided which enabled engineers to develop and debug portions of the software under UNIX and then port it across the VMEbus to the slave processors local memory and start it under pSOS+. Additional processors could be added to enhance the capabilities of the equipment for launch and post-launch support of the mission.

III. Equipment Configuration

The EGSE consists of two single board computers, a storage subsystem, an array of analog and digital I/O boards, and an interface cabinet (Figure 3). The system is contained in a twelve-slot 6U VME tabletop enclosure, with the interface cabinet as a separate unit adjacent to the EGSE. The main processor is an MVME147 board which consists of an MC68030 microprocessor at 20 MHz, eight Mbytes of dynamic RAM, a Small Computer Standard Interface for storage devices, four asynchronous/synchronous serial interfaces, and an Ethernet interface. It runs the UNIX operating system, and functions as the user interface to the command, telemetry and simulation code as well as running the dynamic simulation code. The storage subsystem includes an 150 Mbyte SCSI hard disk, and an 120 Mbyte QIC cartridge tape system for backup and archiving. The slave processor is an MVME 143 board which consists of an MC68030 microprocessor at 20 Mhz, four Mbytes of dynamic RAM, and two asynchronous/ synchronous serial interfaces. It runs the pSOS+ real-time kernel, and functions as the control processor for the I/O cards and the telemetry processor. The I/O system is a set of five Burr-Brown MPV-904 sixteen channel D/A boards, plus an MPV-901 A/D board and an MPV-930 digital I/O board. These boards provide the analog interfaces to simulate all on-board sensors and systems. The interface cabinet provides resistive loads to simulate actual spacecraft EPS loads, and provides a junction/ breakout-box system to facilitate testing of flight hardware and simulation of failure modes.

IV. Software Design

The software was written in C by a two-man team over a six-month period. It consists of the user-interface module and the dynamic model module running on the UNIX side, and a dispatcher module and telemetry module running on the pSOS+ side. There are also drivers under pSOS+ for telemetry, momentum wheel, earth sensor, and discrete analogs interface (Figure 4).

The initial effort was the telemetry code running under pSOS+. A driver was written which, after initialization, waits for a three-byte sync sequence in the telemetry stream. Once synchronized, the driver assembles a 1000 bit telemetry frame and sends a signal to a task under pSOS+. This task synchronously makes a device read call to the driver and is passed a pointer to the telemetry frame. The frame is copied into a shared memory area, the frame buffer freed, and an interprocessor signal is sent to the telemetry display/user-interface task under UNIX. The UNIX task displays the frame in engineering or raw units, and optionally logs the frame to a disk or tape file.

Next to be implemented was the interface to the analog and digital I/O boards. Initially, separate tasks were written to service each class of interface. This resulted in four tasks to be scheduled under pSOS+ with dynamically shifting priorities. It was decided to rewrite the tasks as drivers, and provide a central dispatch routine to service all I/O with the exception of the telemetry. Each driver is initialized, at which time it acquires a segment of shared memory and a queue for commands. The dispatch routine receives commands and data via queue from the UNIX side, and steers them to the appropriate driver. The driver performs the requested function and returns data via the shared memory structure, which is accessible on the UNIX side. This scheme avoids problems in start-up sequencing and provides a cleaner internal organization.

Once the basic analog and digital interfaces were running, separate drivers for three sensors were devised to better distribute the processor load. The earth sensor and momentum wheel simulations were rewritten as drivers and implemented under pSOS+. The sun sensor simulation was rewritten as a stand-alone UNIX task utilizing one of the serial interfaces on the UNIX processor.

The command interface was implemented as a stand-alone system on an HD64180 based single-board computer in order to provide test capabilities for the on-board processor subcontractor ahead of the date when the entire EGSE was ready. This SBC took a serial stream of ASCII-HEX commands from a serially connected terminal and outputted a manchester-encoded command stream to the command interface on the processor. Command entry and script facilities were added to the telemetry user interface and the command interface box was attached to a serial port on the UNIX processor.

V. Utilization

The EGSE described is used in a variety of ways at many points in the testing of the spacecraft (Figure 5). At the lowest level, the equipment is used to help develop and verify interfaces to the various commercial sensors. The sensor manufacturers documentation is used to develop a simulation running on the EGSE, then the simulated signal is used to test and debug the on-board processor (OBP) interface . After the OBP is completed, the OBP/EGSE combination is used to test each of the actual sensors. The system is brought up in full simulation, then the sensor signal is substituted for the simulation signal be jumpering at the interface cabinet. Finally, during system integration and testing, the EGSE serves as a limited ground station providing display and logging of telemetry data and pre-stored command sequences for testing the spacecraft. The equipment will also be utilized post-launch, with the development OBP, to test command sequences and anomaly correction plans before uplinking to the spacecraft.

VI. Expanding the EGSE

This design lends itself to enhancement in a number of ways. One modification which is undergoing evaluation is the addition of a second MVME147 card, running pSOS+, and associated storage subsystem to provide real-time logging and decommutation of a high-speed telemetry dump channel (~1 Mbit/s). By providing this card with it's own disk storage, a fast preallocation storage scheme could be used instead of the normal UNIX filesystem to provide much higher logging speeds.

Addition of suitable RF equipment would enhance the testing capabilities and provide a complete, although not very user-freindly, ground station for operation of the spacecraft. With additional enhancements to the telemetry and command user interface code, such as limit checking of telemetry, validity checks on command streams, and perhaps additional analysis tools as required by the particular mission, the EGSE is transformed into a full-blown ground station.

Another enhancement easily achieved is interconnection of the EGSE with additional computing equipment via the Ethernet port. A scheme was discussed in which the EGSE described here would be connected via Ethernet and TCP/IP to a VAX system running the COMET ground station software. The VAX would be used as the ground station operationally, but would provide telemetry and command interfaces during test. This would constitute almost a complete end-to-end test for the spacecraft OBP and ground station. Also, the dynamic model could be initialized to a state vector and started from the VAX, which would then read the telemetry and do analysis to verify attitude control and other algorithms.

Reuse of the EGSE equipment for subsequent st spacecraft was a consideration in it's initial design. The telemetry and command subsystems are table driven, and all screen images are built from text files which are easily modifiable. The serial devices used for command and telemetry are usable up to ~1.5 Mbit/s. By adding more processor cards, a vast amount of CPU resource can be added at a reasonable cost.

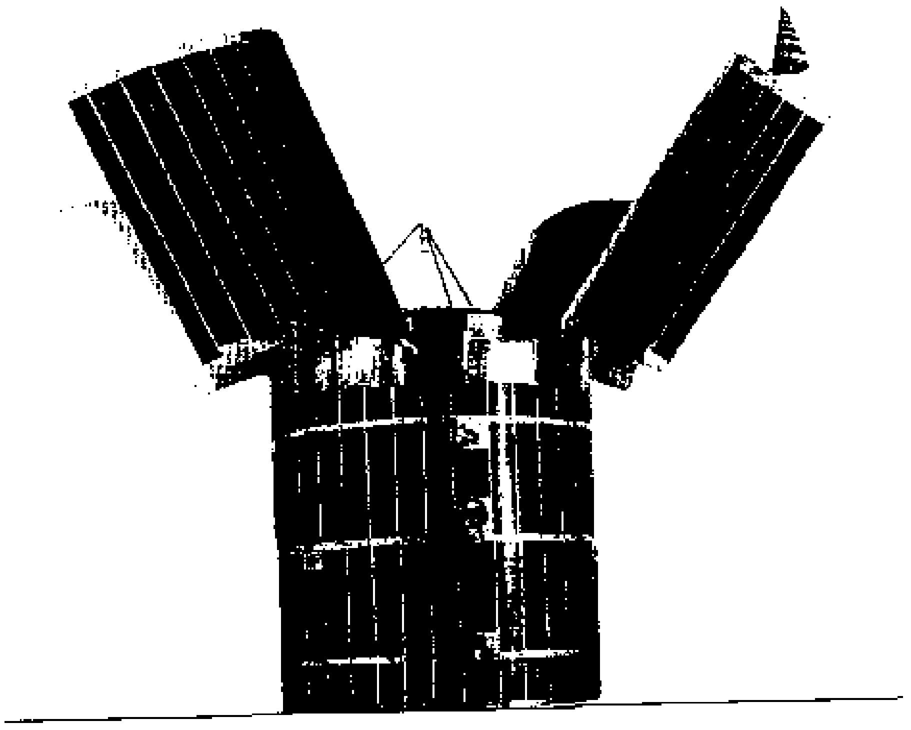


Figure 1

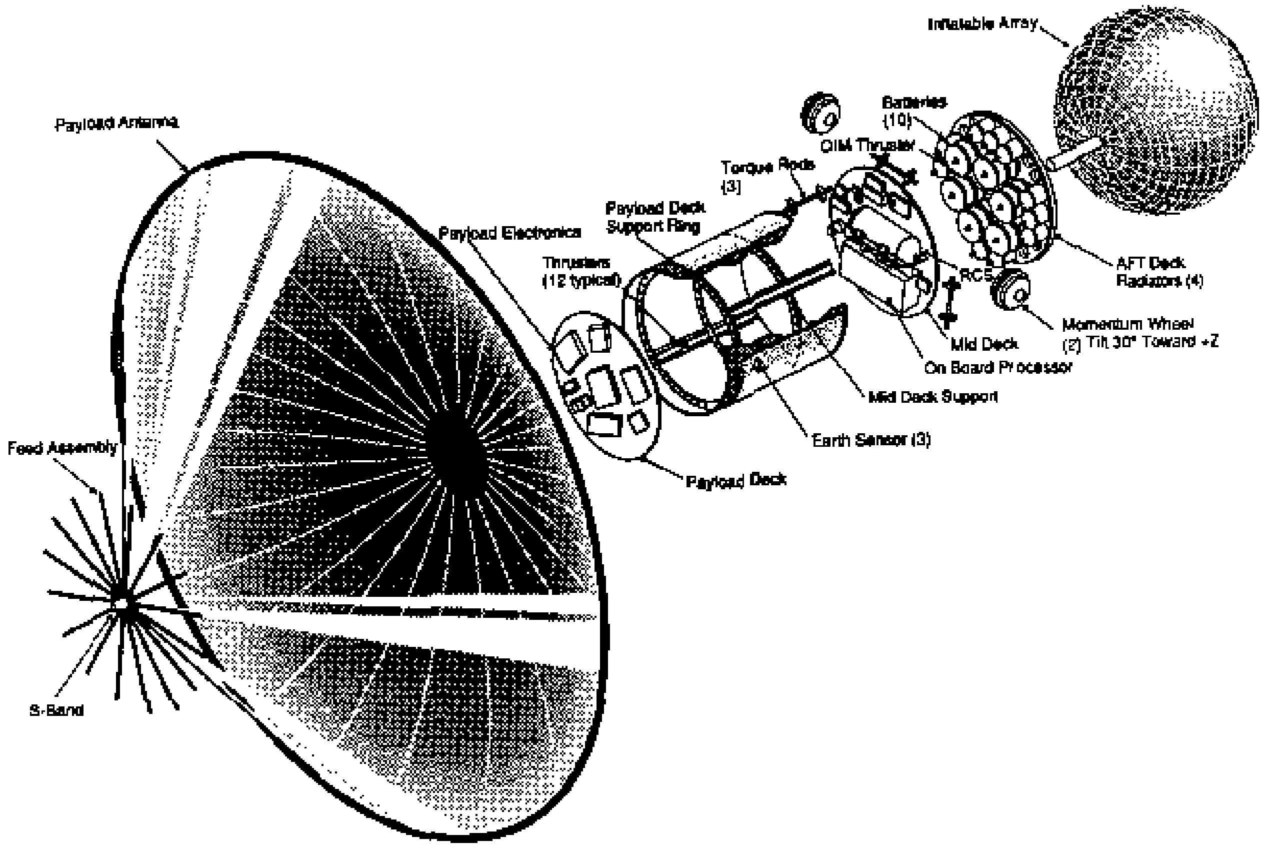
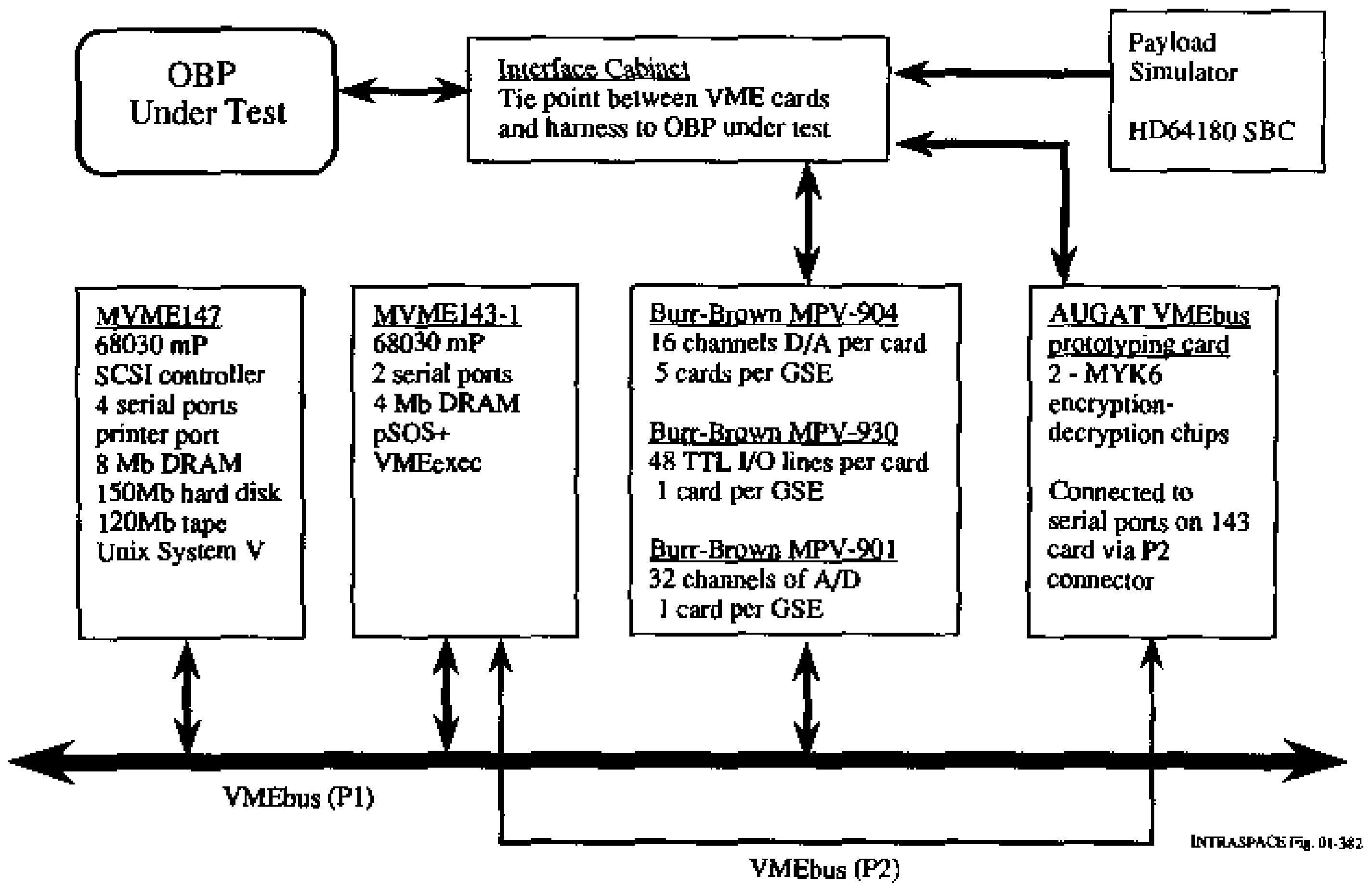


Figure 2

GSE Equipment Configuration



The MVME147 card handles the user interface and dynamic model code under Unix System V

The MVME143 card handles the analog and telemetry interface code under the pSOS+ real-time kernel

The various tasks communicate using message queues, shared memory segments, semaphores and events.

Figure 3

EGSE Software Flow

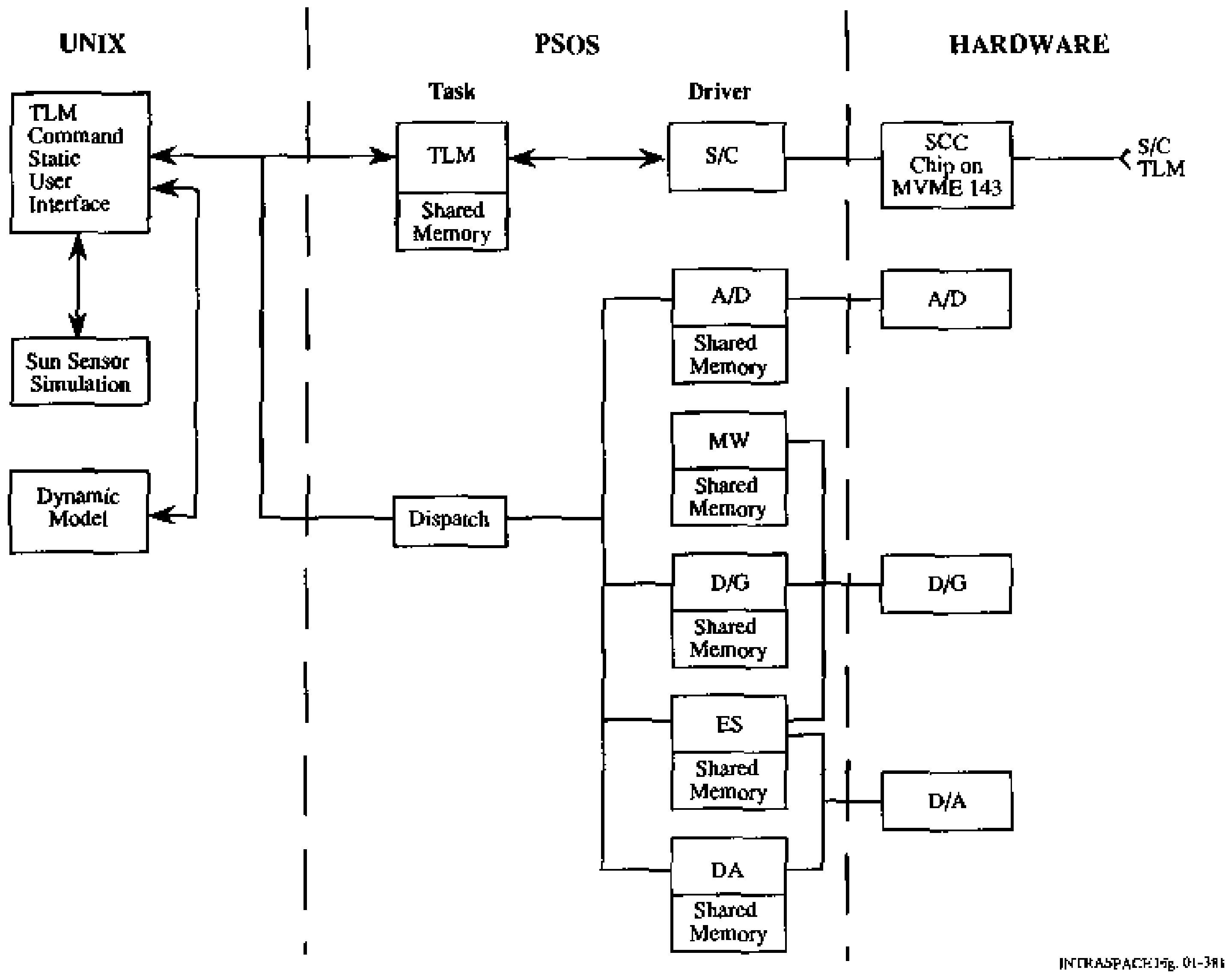


Figure 4

GSE Test Setup

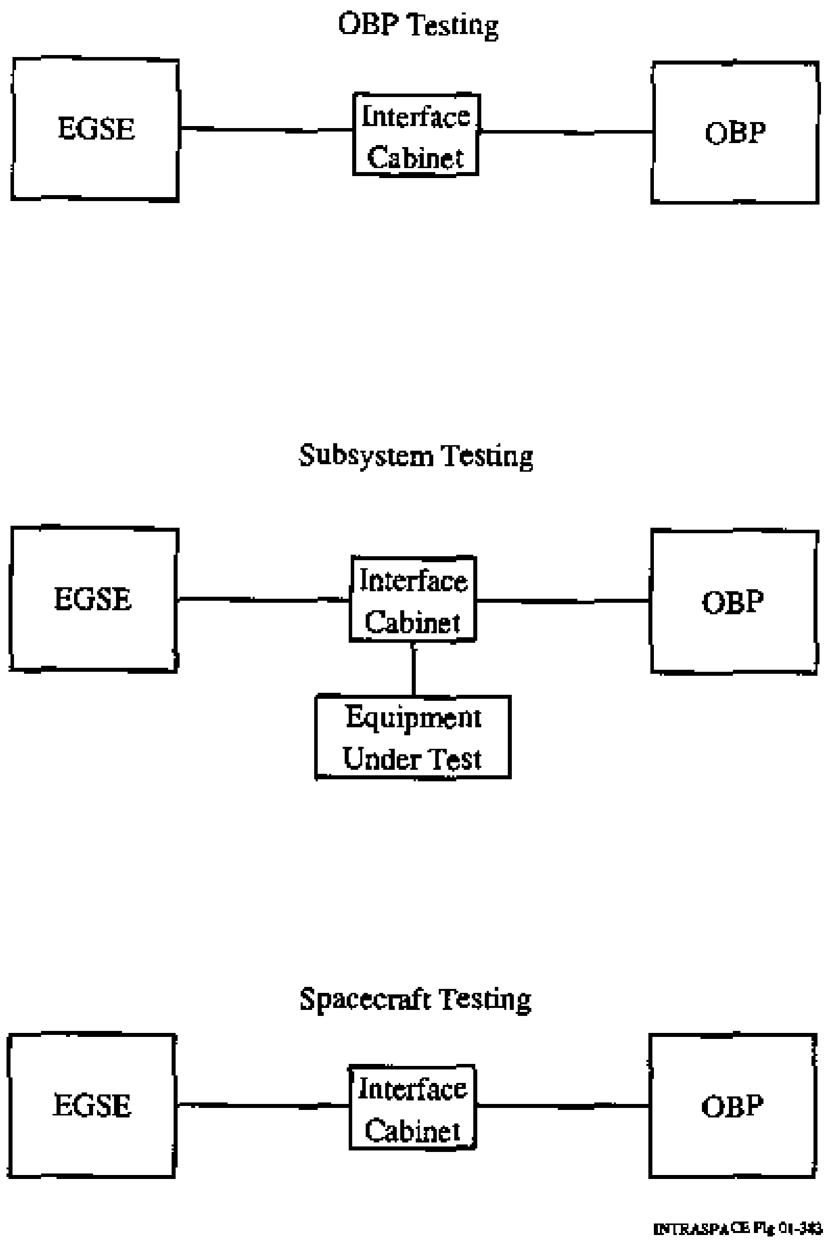


Figure 5