Virtual Instrument Simulator for CERES

John J. Chapman
NASA Langley Research Center
21 Langley Blvd. MS 423
Hampton, VA. 23681, USA

Abstract

A benchtop virtual instrument simulator for CERES (Clouds and the Earth's Radiant Energy System) has been built at NASA, Langley Research Center in Hampton, VA. The CERES instruments will fly on several earth orbiting platforms notably NASDA's Tropical Rainfall Measurement Mission (TRMM) and NASA's Earth Observing System (EOS) satellites. CERES measures top of the atmosphere radiative fluxes using microprocessor controlled scanning radiometers. The CERES Virtual Instrument Simulator consists of electronic circuitry identical to the flight unit's twin microprocessors and telemetry interface to the supporting spacecraft electronics and two personal computers (PC) connected to the I/O ports that control azimuth and elevation gimbals. Software consists of the unmodified TRW developed Flight Code and Ground Support Software which serves as the instrument monitor and NASA/ TRW developed engineering models of the scanners. The CERES Instrument Simulator will serve as a testbed for testing of custom instrument commands intended to solve in-flight anomalies of the instruments which could arise during the CERES mission. One of the supporting computers supports the telemetry display which monitors the simulator microprocessors during the development and testing of custom instrument commands. The CERES engineering development software models have been modified to provide a virtual instrument running on a second supporting computer linked in real time to the instrument flight microprocessor control ports. The CERES Instrument Simulator will be used to verify memory uploads by the CERES Flight Operations TEAM at NASA. Plots of the virtual scanner models match the actual instrument scan plots. A high speed logic analyzer has been used to track the performance of the flight microprocessor. The concept of using an identical but non-flight qualified microprocessor and electronics ensemble linked to a virtual instrument with identical system software affords a relatively inexpensive simulation system capable of high fidelity.

Key Words : Satellite/Simulation/Microprocessor/Software/Instrument

1. INTRODUCTION

The Clouds and the Earth's Radiant Energy System (CERES) instrument will soon fly on several earth orbiting platforms (See Ref. 1). The first to be launched will be NASDA's Tropical Rainfall Measurement Mission (TRMM) currently planned for November, 1997 . Four more CERES instruments will be launched on NASA's Earth Observing System (EOS) satellites, two on EOS AM-1, which is scheduled for launch in June 1998 and two on EOS PM-1 scheduled for December, 2000.

The CERES instrument measures top of the atmoshere radiative fluxes (See Ref. 2) using microprocessor controlled gimballed radiometers. The standard commanding of the instrument will be done by the Flight Operations Team (fig.1) using CERES instrument functional modes chosen for the daily operational scenarios predetermined by the Science , Mission Operations and Project Management Teams. The commands consist of mnemonics grouped in long or short command formats or sequences of commands called scan tables which modify the instrument behavior, for example the scan limits may be adjusted for sun avoidance during normal scanning profiles, or special instrument calibrations or diagnostics. The CERES instruments are built by TRW, Redondo Beach, CA. for NASA, Langley Research Center. The engineering model of the instrument, normally used for ground based testing after the flight production models are launched was incorporated into the TRMM spacecraft, thus saving the expense to build one of the flight models. This decision left the Flight Operations Team without a testbed instrument for the ground based investigation of potential instrument anomalies. The CERES instrument simulator will allow testing of any memory patches or custom commands prior to uploading by telemetry link to the orbiting CERES instrument.

2. SIMULATOR DESCRIPTION

The architecture of the embedded microprocessor portion of the simulator (fig 2) is exactly the same as the flight system. The simulator was assembled following the schematics of the flight electronics. It consists of twin 80C186 microprocessors on separate cards running the TRW flight software. Direct memory access (DMA) between the microprocessors and shared RAM for the telemetry linkage is used to communicate with the spacecraft or Bench Checkout Unit (BCU). The Ground Support Equipment (GSE) software running on the BCU 80586 PC serves as the instrument monitor. This software was originally developed at TRW to provide a housekeeping screen of the telemetry parameters being monitored by the CERES instrument microprocessors while under test. Test instrument commands to be verified by the simulator are loaded into the GSE/BCU PC from the Long Command Editor, then sent to the ICP via the telemetry link (which is accomplished by a coaxial cable for the simulator) to the Spacecraft Interface card. The GSE/BCU housekeeping screen then displays the current instrument mode (fig.3), elevation and azimuth gimbal data, and microprocessor memory checksum data of the uploaded patch. Since the simulator has no mechanical gimballed components, a second fast 80686 (PC) interfaced to each microprocessor scanner control port provides a separate virtual azimuth and elevation gimbal. Engineering software models of the elevation and azimuth gimbals have been modified to provide a fast, virtual instrument running in real time, interrupt linked to both the ICP and DAP scanner control ports.

3. HARDWARE MODIFICATIONS to the FLIGHT CIRCUITRY

Since space qualified memory integrated circuits used for the flight articles are fabricated with radiation hardened components and then environmentally tested, they are prohibitively expensive for a ground based simulator. A set of bare circuit cards identical to the flight design was obtained from the flight item vendor to fabricate a functionally identical electronic platform. The cards consist of the Instrument Control Processor (ICP), Data Acquisition Processor (DAP) with two Digital Interface cards and one Spacecraft Interface card. For the simulator, inexpensive commercial grade integrated circuits, in plastic packages but equivalent in function to the flight parts were used throughout and installed in sockets where possible.

The N80C186-25MHz microprocessor chips chosen for the simulator are manufactured in plastic leadless chip carriers (PLCC) which fit into custom emulation sockets. The custom sockets provide for an external logic analyzer to be connected as a monitor during testing. The flight microprocessor boards were designed for Surface Mount Technology (SMT) devices. The difference in pinout of the SMT footprint and the PLCC equals the mirror image rotated by 90 degrees. Thus the PLCC component for the ICP & DAP microprocessor must be inverted and rotated 90 degrees and the proper pin matching is achieved through the custom microprocessor sockets. A current-limited power supply was used to power the cards for trials. The completed simulator uses ferroresonant transformer power supplies with built in rectification, filtering and voltage regulation. Memory chips designed for use in PC static random access memory (SRAM) 32K X 8 and Eraseable Programmable Read Only Memory (EPROM) 8K X 8 with device architecture, logic polarity and access time were chosen to mimic the flight components.

The unmodified flight code is programmed in several EPROM chips installed on small memory adaptor boards mounted above the ICP and DAP cards, hence the simulator has an identical memory map as the flight memory system. Other modifications include a manual reset switch combined with a watchdog timeout inhibit function and a non-flight grade fiberoptic to TTL converter for the TRMM interface between the Spacecraft Interface Card and the GSE/BCU. The ICP to DAP data and address interconnect lines are provided by inexpensive ribbon cables. Other card to card wiring was made short to maintain good switching signal profiles. The other CERES sensors which comprise the remaining 220 channels of instrument sensors such as, covers, azimuth brake, solar avoidance and component temperatures are synthesized by preloaded PC memory lookup files set to their nominal levels. They are interfaced to the ICP and the DAP instead of the A/D converter cards. Both the ICP and DAP continually check sensor status (such as the azimuth brake) and react to these inputs accordingly to protect the instrument.

4. FLIGHT SOFTWARE/HARDWARE INTEGRATION & SYSTEM DEBUGGING

The baseline flight code was originally developed by TRW with the use of a dedicated emulator for each processor to facilitate debugging of the code. Since the simulator uses identical circuitry and runs unmodified CERES flight code it was felt an emulator was not necessary and could even unnecessarily complicate the integration process. The baseline microprocessor flight code was programmed into 27C64 EPROMS and the checksums were verified prior to the boot-up trials. The CERES Flight Software documentation and notes from a CERES flight software workshop held earlier were used to identify major milestones in the ICP and DAP boot-up and initialization cycles. A Tektronix Prism multichannel logic analyzer was alternately connected to the ICP adapter socket address and data connections as well as certain critical lines such as Address Latch Enable (ALE), the microprocessor machine state indicator S0,S1,S2 and clock lines. Debugger/disassembler software available from Tektronix was loaded in the analyzer memory and 8K trial samples of address/data pair bytes were triggered and then displayed on the analyzer screen. Unique address/data byte pairs corresponding to the boot-up milestones were observed to verify the extent of the code execution. A Tektronix 684A 4 channel digital storage oscilloscope was also used to monitor certain flight code activated lines such as the memory and peripheral chip select lines to help debugging and to verify memory and peripheral function. The analyzer samples are stored and displayed by the debugger software. The analyzer selection menu includes filtering the sampled data sets to provide displays in either hardware mode, which shows instruction fetches, memory read/write and I/O operations, software mode (which shows the machine instruction mnemonics), or subroutine call mode (which shows only the subroutine calls and the physical address). The boot-up portion of the flight code consists entirely of 80C186 assembly language, for which the hardware and software debugging screens are well suited. The initialization and main loop portion of the flight code is written in C++. It consists of a series of callable subroutines which are readily recognized by the analyzer in the subroutine display mode. The logic analyzer can store only 8K of samples, thus the choice of suitable code for the address or data trigger is important to capture the desired segment of the executed code. Once both processors are in their main control loops, their subroutine calls (see Tables 1 and 2) are synchronized within a 10 msec sample window midpoint and both loops are repeated simultaneously every 10 msec. For the DAP, the important calls used for the simulator include the Acquire Elevation Data subroutine, which reads the pseudo-encoder output and the Elevation Scan Control subroutine which sends rate and position commands to the elevation control port based on data from the scan tables in SRAM. The ICP similarly controls the azimuth control port with the Acquire Azimuth Data and Azimuth Control subroutines. The ICP also controls the telemetry interface port feeding the GSE/BCU. The logic analyzer and the BCU display both provide the means to verify that the ICP and DAP do indeed communicate and properly execute the CERES flight code at the 16 MHz clock speed just as the flight instrument.

5. VIRTUAL INSTRUMENT MODELS & LINKAGE TO MAIN LOOP

The control model diagram of the Matlab engineering model of the elevation scanner is shown in fig. 4. The original non-linear model was developed by engineering teams at NASA LaRC and TRW. Simulator specific modifications were made to convert the non-linear engineering analysis model into a fast, real time executable virtual instrument using the Matlab Real Time Workshop. The resultant model was developed to be indistinguishable from the mechanical scanner to either the ICP or DAP. A linear model has also been developed as a means to verify the non-linear model, plots of the two models differ only in the overshoot at the scan inflection points. Both real time executable models run on a 200 MHz 80686 PC and interact with the simulator microprocessor I/O port through an interface card and some added pulse conditioning circuits. The interface card provides a bi-directional parallel transfer of the rate, direction and position, interrupt driven by the microprocessor. An interrupt service routine was developed for processing strobed input/output (I/O) via the Metrabyte PIO-12 I/O card. A sample Elevation scan profile table is given in Table 3, and plotted in Fig. 5. Each inflection point in the scan corresponds to new rate, direction and position commands. Fig. 5A shows a short scan profile. In the event that the error between commanded position and actual position exceeds 500 counts for a duration of at least 100 milliseconds, the instrument will be placed in the STOW mode and the motor drive disabled. The azimuth gimbal model shown in fig. 6 is very similar to the elevation model, the difference being the range (350 deg), rate (4 to 6 deg/sec), and specific values for friction, torque and gain. This unique feature allows the investigation of scanning commands in a virtual instrument simulation scenario without costly mechanical servo hardware or servo driver circuits in the simulator. The adjustment of the Matlab input file

parameters such as drag or servo driver gain allows modification of servo performance to reflect instrument related behavior such as the effect of mechanical wear or long term radiation effects on electronic components..

6. CONCLUSION

The CERES instrument BCU monitors, displays and archives the important parameters such as the current and previous mode (See Fig. 7) of operation, recent commands processed and ICP and DAP memory ROM and RAM checksums. The elevation and azimuth position and gimbal stastus data are displayed in realtime also. This provides for benchtop testing of new elevation & azimuth scan profiles, instrument scan mode, inflection point positions, direction, torque, error in counts & degrees, status, stall and drive enabled/disabled. The test commands may be up-loaded into memory and the telemetry output is recorded on removable hard drive disks which can be played back for performance verification or transferred to the operations team, specific parameter plots can also be obtained from the system. Files from the logic analyzer can also be saved and transferred on disquettes.

Software modifications (memory patch) may thus be tested prior to uploading to measure the impact on both ICP & DAP functioning. The simulator will also be used as a training aid to provide operational familiarity to instrument technicians and engineers.

Table 1: Main Loop Calls by Instrument Control Processor

Synchronize to 100 Hz Clock
Acquire A/D Converter Data
Change Sub-mux Address for next sample
Read Azimuth Position and Error Values
Perform Azimuth Control Function (includes brake)
Check Spacecraft Safe-Hold Signals
Perform Solar Avoidance Check
Check for Telemetry Messages
Process Telemetry Commands
Process Current Internal Sequence
Wait until 100 Hz Sample Period Midpoint
Communicate with DAP (DMA with Shared Memory)
Perform all remaining TRMM Interface Tasks
Calculate another piece of Code Checksum
Keep a running Calculation Min./Max. Execution Times
Strobe Watchdog Timer

Table 2: Main Loop Calls for Data Acquisition Processor

Synchronize to 100 Hz Sample Clock
Acquire A/D Converter Data
Change Sub-mux Address for next sample
Read Elevation Position and Error Values
Execute Elevation Scan Profile
Perform Cover Control
Perform Closed Loop Cintrol of Detector & Blackbody Temperatures
Perform Bolometer Bridge Balance
Update SWICS Intensity Port if necessary
Wait until 100 Hz Sample Period Midpoint
Communicate with ICP (DMA with Shared Memory)
Calculate another piece of Code Checksum
Calculate Min./Max. Execution Times
Strobe Watchdog Timer

Table 3 Elevation Scan Profile

Inflection Point	Sample Number	Starting Position Command	Rate & Direction Command
0	0	3237	0 deg/sec
1	50	3237	63 deg/sec
2	279	29532	0 deg/sec
3	305	29532	250 deg/sec
4	317	35317	0 deg/sec
5	342	35317	-250 deg/sec
6	354	29532	0 deg/sec
7	380	29532	-63 deg/sec
8	609	3237	0 deg/sec
9	659	3237	0 deg/sec

Acknowledgements

Special thanks to the following people for their important technical assistance:
Tom Evert of TRW Corp.
James L. Donaldson of Computer Sciences Corp.
Michael C. Holloman of NASA, LaRC
James B. Miller of NASA, LaRC
Bryant Douglas Taylor of NYMA Corp.
Paul Sakaguchi of TRW Corp.
Christopher J. Slominski of Computer Sciences Corp.

References

1. Bulletin of the American Meteorological Society, Vol. 77, No.5, May 1996
2. Bulletin of the American Meteorological Society, Vol. 76, No.11 Nov. 1995
3. http://asd-www.larc.nasa.gov/ASDhomepage.html

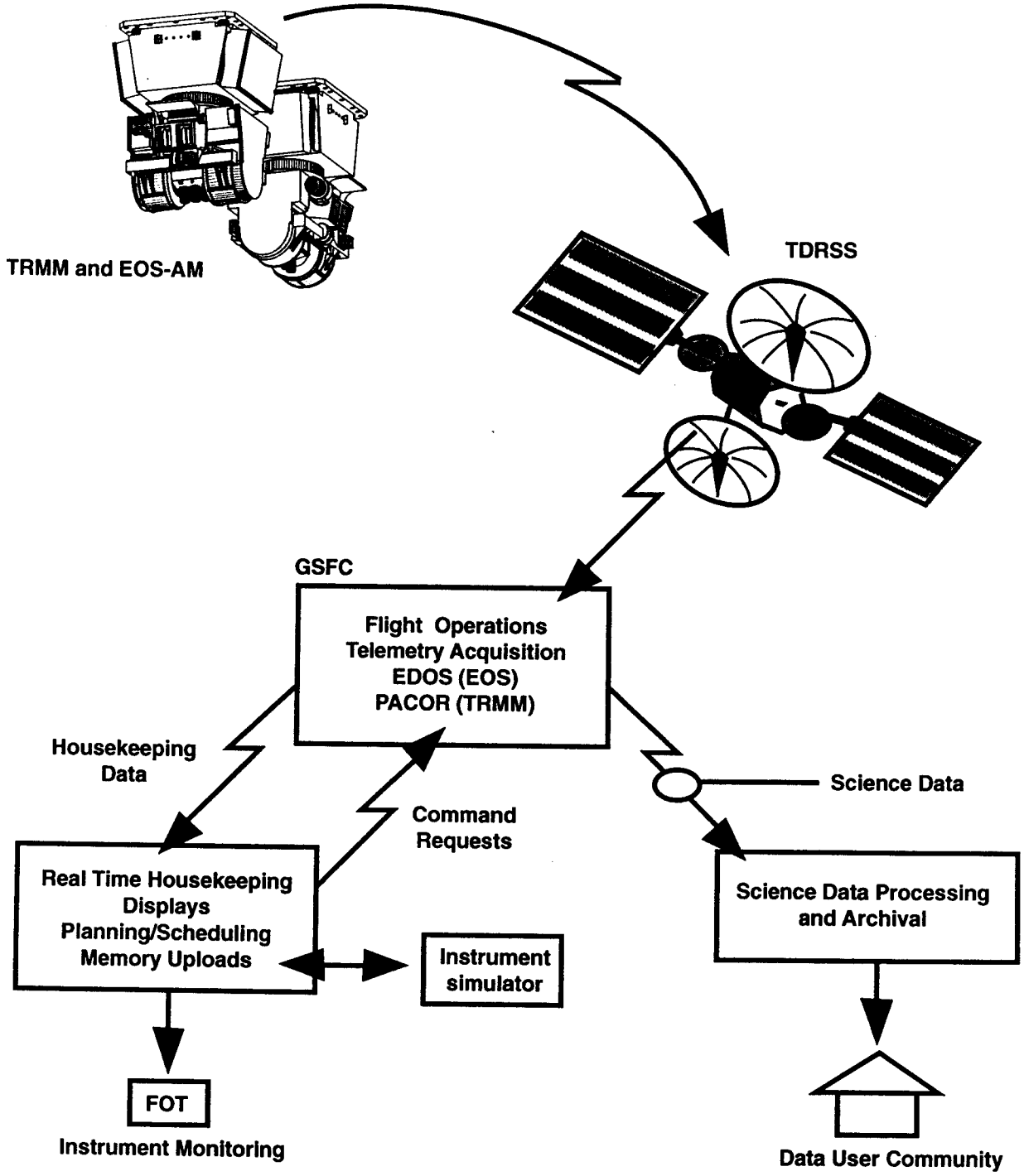

Figure 1. CERES Flight Operations/Instrument Simulator

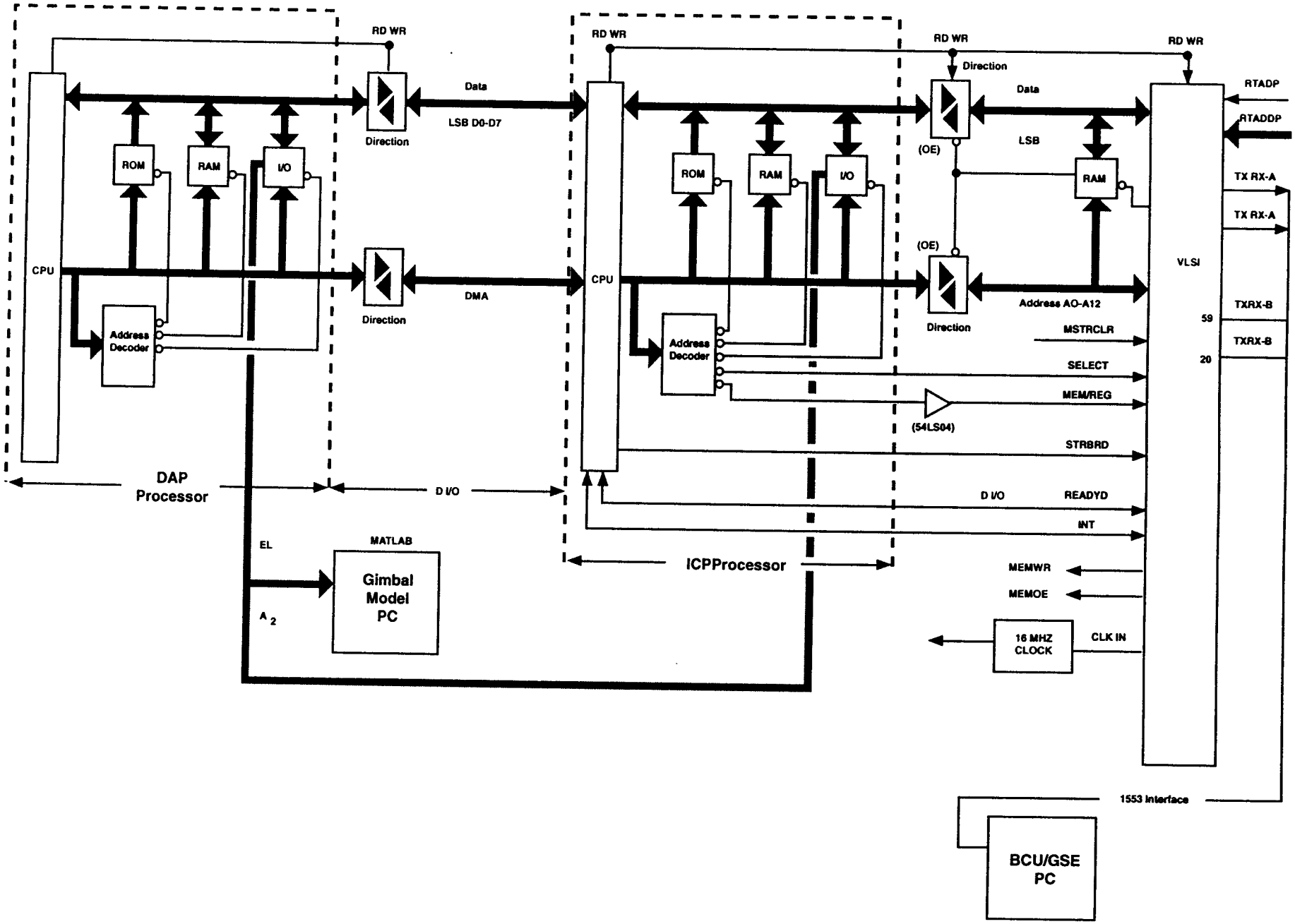

Figure 2. CERES Simulator Architecture

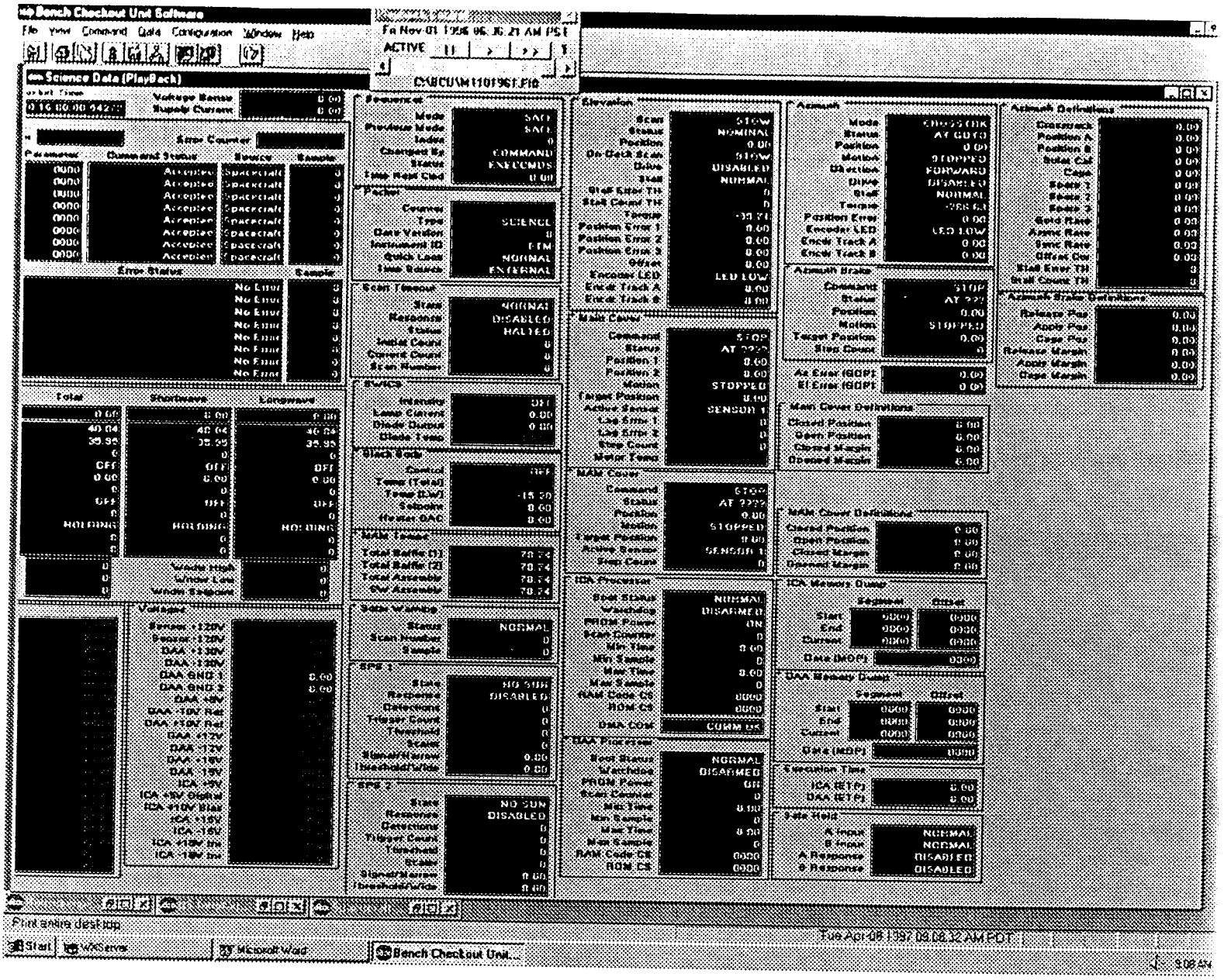

Figure 3. BCU/GSE Instrument Screen

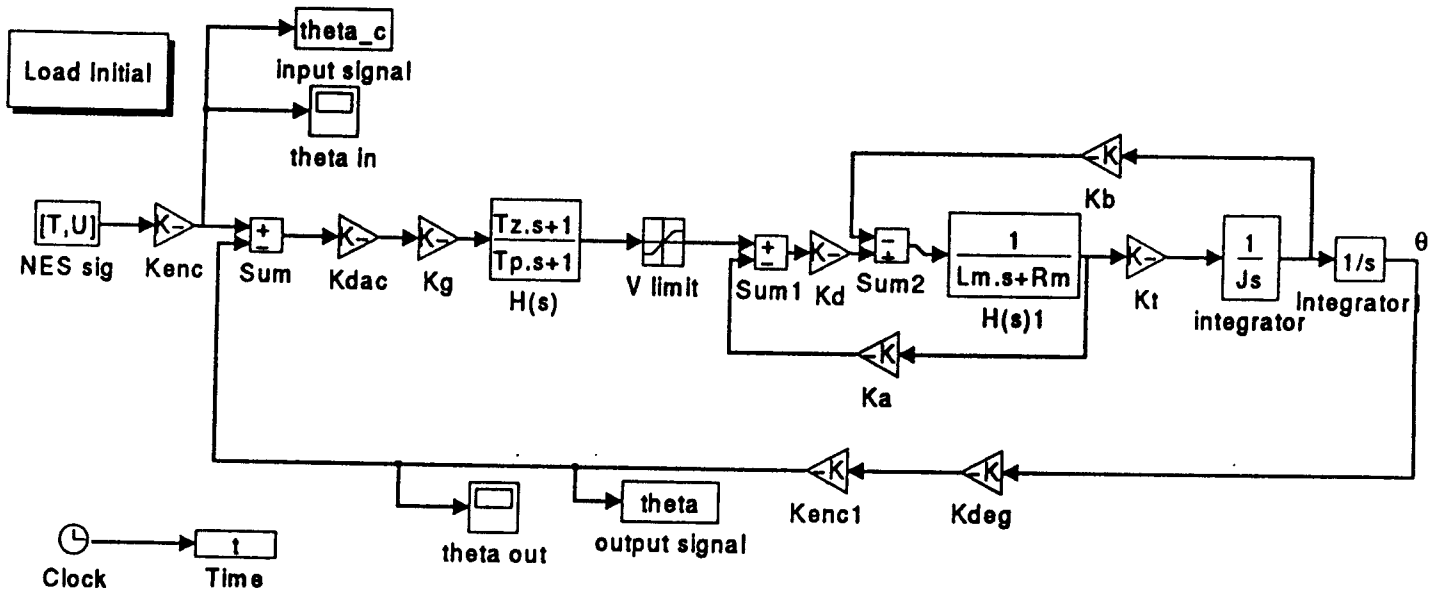

Figure 4. Linear MATLAB Model

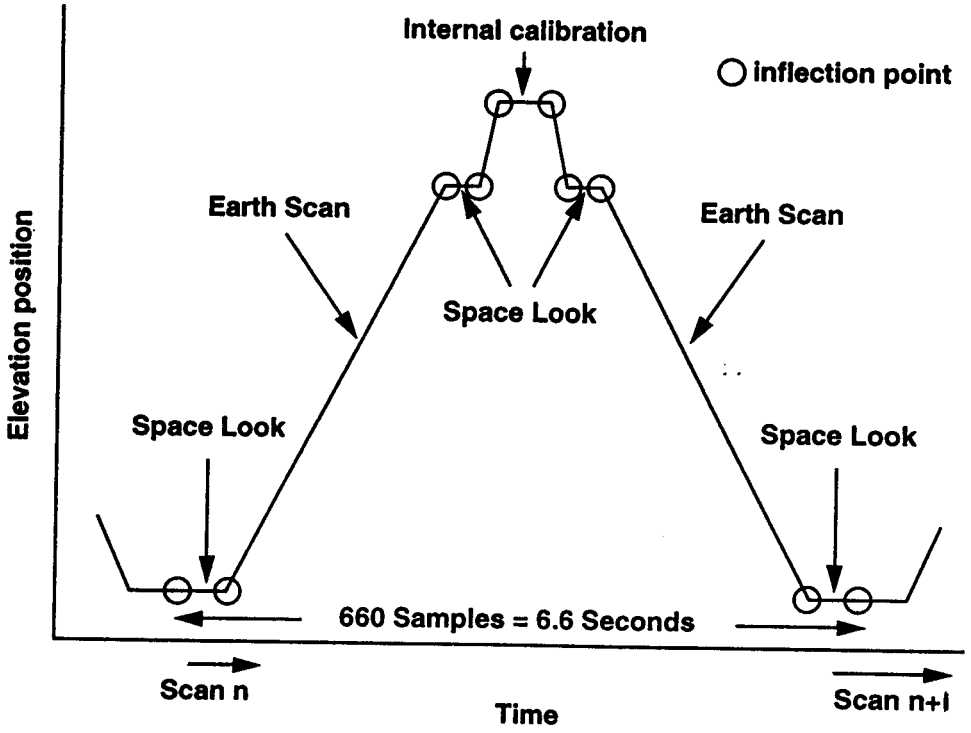

Figure 5. Elevation Scan

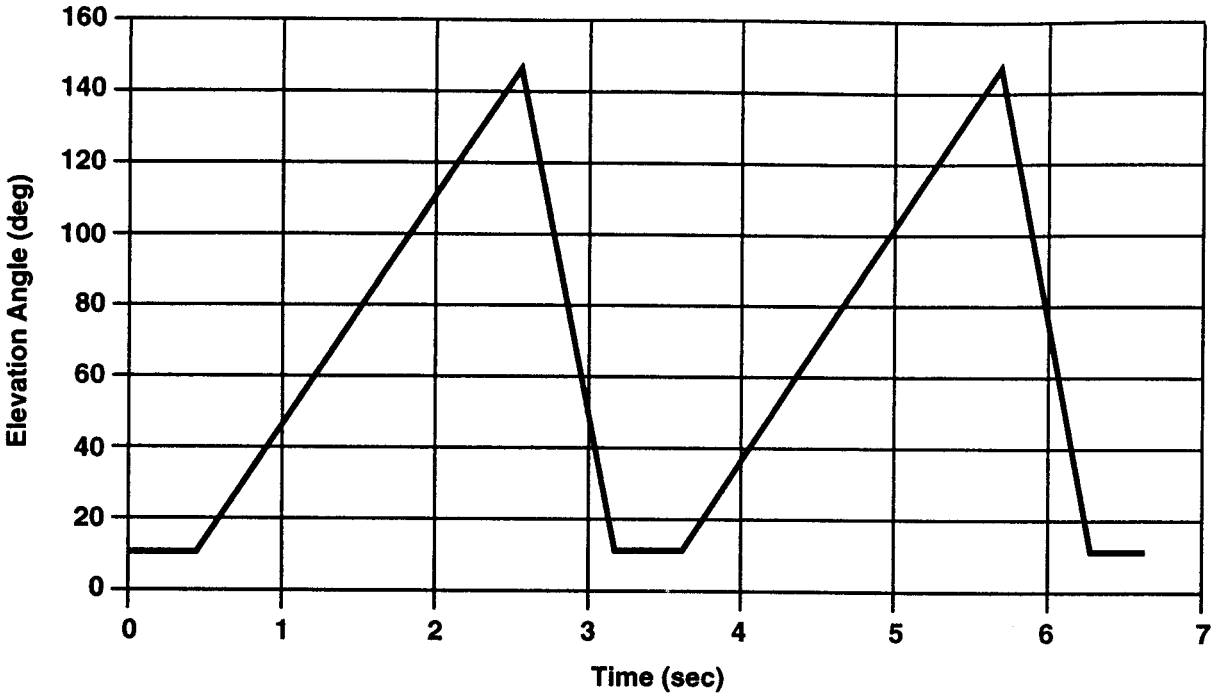

Figure 5a. Elevation Short Scan Profile

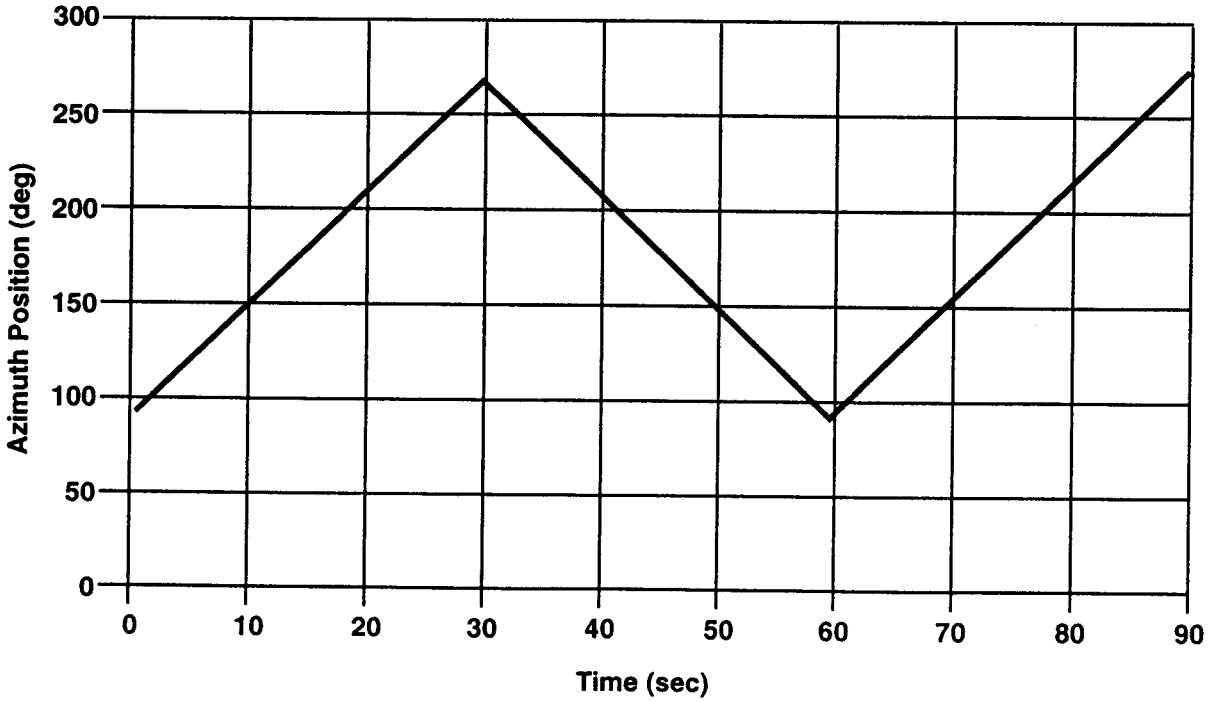

Figure 6. Azimuth Scan

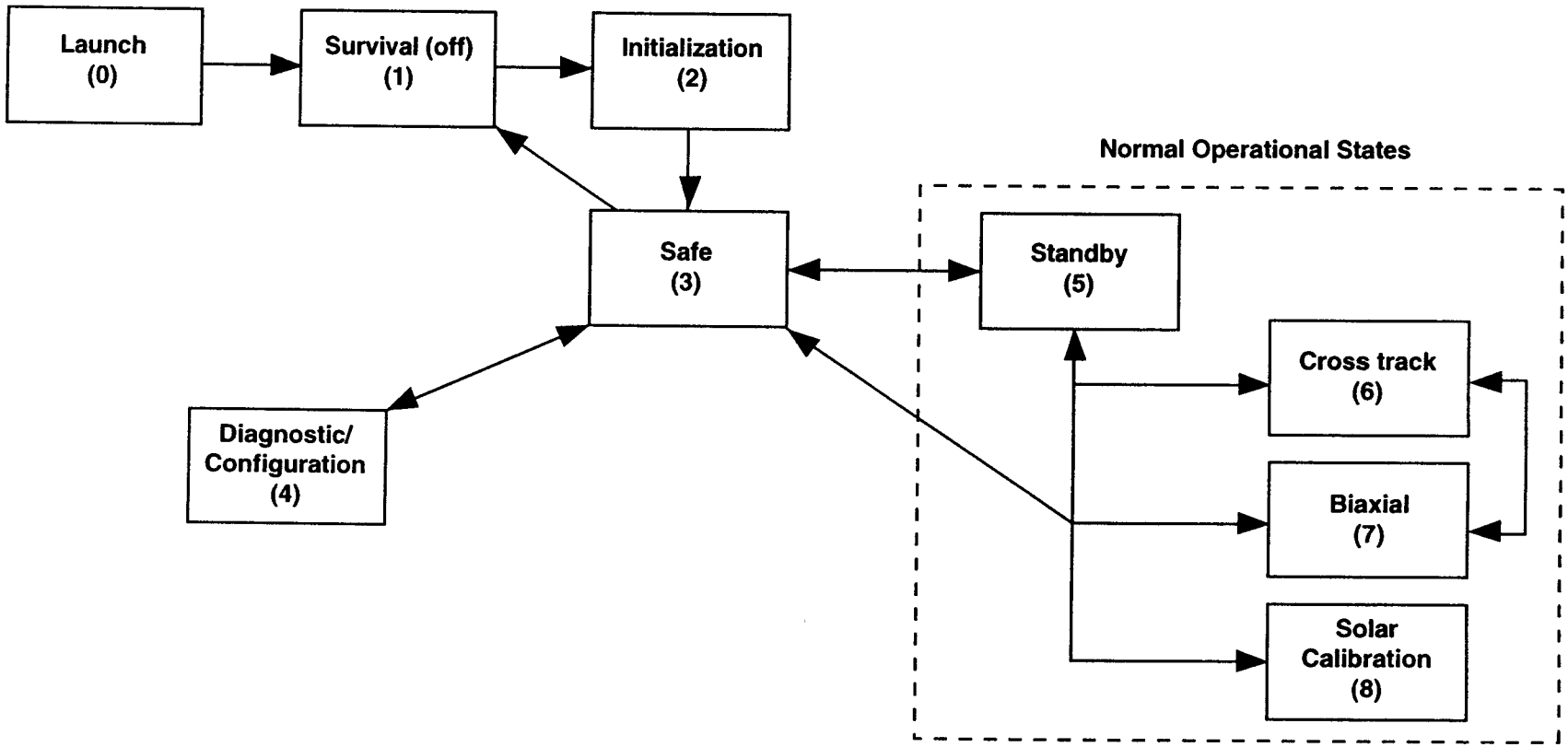

Figure 7. CERES Instrument Modes