XTE Experiment Data System

Hardware Functional Description and Requirements

D. Gordon

MIT 64-02006.01

Rev. B

June 23, 1992

Table of Contents

1 Overview

1.1 Introduction

The X-Ray Timing Explorer (XTE) satellite, planned for a 1996 launch into a low-Earth orbit, houses three X-Ray detector systems: an All Sky Monitor (ASM), a Proportional Counter Array (PCA) and a High Energy X-Ray Timing Experiment (HEXTE). Together they will monitor the X-Ray sky, observing sources in the 2-200 Kev range and telemetering the observation data to the Earth.

The Experiment Data System (EDS), a computer-based on-board data system on the XTE satellite, will receive data from two of the X-Ray detectors (the ASM and the PCA), timestamping and reducing the data in accordance with an observation specific set of commands. The EDS is customized to meet the unique requirments of the ASM and PCA, matching them to the dedicated spacecraft capabilities.

This document describes the overall design of the EDS and its interface to the Flight Software and the XTE intruments and satellite.

1.2 References

MIT 64-03001 .... "PCA-EDS Interface Control Document", Rev. D
MIT 64-03006 .... "ASM/EDS Interface Control Document", Rev. A
MIT 64-02007 .... "ASM Functional Description & Requirements", Rev. 02
MIT 64-03005 .... "HEXTE-EDS Interface Control Document", Rev. B
MIT 64-30102 .... "Flight Software Global Design", Rev. C
MIT 64-30111.0801 .... "Typer Builder Module Specification", Rev. A
MIT 64-30111.0501 .... "Recoder Builder Module Specification", Rev. A
GSFC-735-XTE-ICD-002 .... "XTE SDS to EDS Interface Description" - May 12, 1992
MIL-STD-1773 .... "1773 Bus Specification DOD-MIL-STD-1773 (USAF)" - May 20, 1988

Harris 80C286, 82C59, 82C237
Texas Instruments............................. TMS320C25
Signetics 2692
Data Device Corporation BUS-65612

1.3 List of Acronyms and Abbreviations

ALU .... Arithmetic Logic Unit
ASM............. All Sky Monitor
BREGS .... Backend Registers
CCSDS......... Consultative Committee for Space Data Systems
CSR............. Center for Space Research
CPU .... Central Processing Unit
DMA .... Direct Memory Access
DSP .... Digital Signal Processor
EA .... Event Analyzer
ECC .... Error Correction Code
EDS .... Experiment Data System
EEPROM ..... Electronically Erasable Programmable Read-only Memory
FIFO............. First-in, First-out (usually refers to memory)
FPGA .... Field Programmable Gate Array
FREGS......... Frontend Registers
GSFC .... Goddard Space Flight Center
HEXTE .... High Energy X-Ray Timing Experiment
ICD .... Interface Control Document
I/O .... Input-Output
LSB .... Least Significant Bit
LUT .... Look-up Table
MET............. Mission Elapsed Time
MIT .... Massachusetts Institute of Technology
MSB .... Most Significant Bit
NASA .... National Aeronautics and Space Administration
PRAM .... Protected RAM
PCA .... Proportional Counter Array
PCU .... Proportional Counter Unit
PSDU .... Power Switching and Distribution Unit
RAM .... Random access memory (read-write memory)
ROM .... Read-only memory
RT .... Remote Terminal
S/C .... Spacecraft
SDS .... Spacecraft Data System
SEU.............. Single Event Upset
SSC .... Scanning Shadow Camera
TBD .... To be determined
TBR .... To be resolved
TBS .... To be specified
UTC .... Universal Time (Coordinated)
XTE .... X-Ray Timing Explorer

1.4 EDS Description and Goals

The EDS is a computing system which filters and processes digitized data streams received from X-Ray detectors, downlinking the processed data (usually in a highly compressed form) to earth via the XTE Spacecraft Data System (SDS). It is composed (see Figure 1) of eight independent and parallel processors, called Event Analyzers (EAs); two EAs are dedicated to the ASM and six EAs are dedicated to the PCA. The EAs are divided into two autonomous subgroups, which receive independent services from the spacecraft. Each EA subgroup contains its own "System Manager" which is linked to the SDS via a 1773 fiber optic bus providing the uplink path through which the EAs are commanded.

The EDS must be capable of accepting PCA data at the maximum expected instrument data rate: no greater than 700 Kevents/second. It must accept ASM data at the maximum rate specified by the ASM Science Requirements Document: 5000 events/second. (NOTE: Although the EDS hardware may accept events at these stated maximum rates, the EDS as a whole will have processor and mode dependent limitations which are further detailed in the Flight Software design documents and specifications.)

The EDS must process the data and downlink the aggregate of EA packet contributions to the SDS at a maximum burst rate of 500 Kbits/second. (The maximum sustained rate will necessarily be lower due to spacecraft telemetry limitations, and must be constrained by choosing sufficient data compression via the ground programmed EA modes.)

The EDS must be tolerant of single point failures and should operate (with partial capacity in some cases) if any single hardware component fails. Operation following double point failures is not guaranteed.

The EDS and individual EAs must be "rugged"; the design goal is to expect and support occasional crashes and hence reboots, due to the expected radiation environment. The hardware must support a graceful crash/reboot mechanism.

XTE is primarily a timing experiment; hence an objective of the EDS hardware is to timestamp with precision the occurrence of each instrument event.

Figure 1 - Overall Block Diagram

2 Components

As shown in Figure 1, the EDS is comprised of two varieties of hardware modules: the eight event analyzers (EAs) and the two System Managers. The EAs are computing engines; the system managers include a control store ROM, a protected RAM, an uplink interface and power regulation, switching and control circuitry.

2.1 Event Analyzer

A total of eight autonomous Event Analyzers form the processing core of the EDS. Each "EA Subgroup" consists of four EAs, one collecting ASM data and three collecting PCA data. Figure 1 identifies each EA by its subgroup (A- or B-) and then by its EA ID within the subgroup (P1, P2, P3 or A1). Four connector signals uniquely identify each EA and its subgroup; therefore EA and EA subgroup ID are determined by an EA's insertion location (which connector it mates with) within the EDS. PCA EAs differ slightly (as described below) from the ASM EAs.

2.1.1 Nomenclature

Each EA contains two microprocessors in series, a frontend processor and a backend processor. The backend processor is referred to as CPU or 80C286 (or 286). The front-end processor is referred as DSP or 320C25. There are several varieties of registers (described in more detail in later sections) which are classified as follows:

CPU Registers	Readable and/or writable by the backend processor; they control and check status of the backend processor and the uplink and downlink interfaces.
Observation Registers	Readable and/or writable by the backend processor; they control and check the status of the frontend processor, the timing generators and the event processing circuitry and instrument interface circuitry.
Pulse Registers	An assertion of any bit in these write-only registers by the flight software creates a temporary strobe to the associated hardware device or register. There is no need to deassert pulses.
Control Registers	Control register bits, readable and writable, modify hardware behavior and although usually set and reset by the processor, some are resettable by the hardware. These are described in further detail in the actual register descriptions.
Status Registers	Read-only, generally hardware conditions which the flight software must track.

2.1.2 Functional Overview

An EA functionally divides into two sections as shown in Figure 2. The "frontend" interfaces to the science instrument; while the "backend" is the link to Earth. The frontend, a slave processor to the backend, is responsible for instrument data collection and reduction. It also contains dedicated hardware which timestamps and filters instrument events. The backend receives uplink, interprets and executes ground initiated commands, sets up frontend modes, collects reduced data from the frontend, packetizes the data and downlinks science and engineering data to the ground via the SDS.

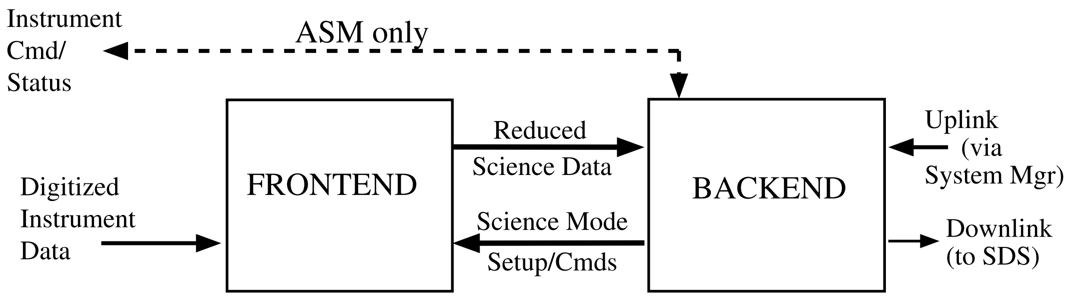

Figure 2 - EA Functional Partitioning

The backend control is governed by the portion of the flight software (FS) (see the Flight Software Global Design) which runs on the 80C286 microprocessor. The frontend code (another portion of the FS) is observation dependent and runs on the 320C25 digital signal processor.

2.1.3 Timekeeping

Timekeeping is a crucial function of the EDS. Each EA must acquire science data in an accurate and definable manner, while meeting the spacecraft realtime demands for uplink and downlink. EAs must be able to synchronize to an absolute time reference (and hence to each other) while autonomously clocking and time-stamping their own event streams.

The following spacecraft clocks, supplied by the PSDU (Power Switching and Distribution Unit), are received by each EA:

Major Tick	1.0 Hz; it is used by the backend to track "mission elapsed time" and by the frontend as a reference to which internal timing signals are aligned (see below). The Major Tick is precisely 2^{20} MicroClocks.
Minor Tick	32 Hz (31.25 ms); it is used by the backend as a realtime clock. The Minor Tick is precisely 2^{15} MicroClocks.
MicroClock	1,048,576 Hz (953 ns); it is used by the frontend to generate observation dependent programmable clocks.

Spacecraft generated clocks are all phase aligned to within 100 ns (TBR, see Appendix B). Clock distribution skew within the EDS shall be no more than 50 ns.

The S/C onboard computer tracks the difference between "mission elapsed time" and actual time (UTC) and telemeters as housekeeping information the correlation between the two. Thus the FS has no knowledge of UTC.

The clocks generated internally within each EA are:

Interval Tick	The interval tick is programmable by the backend processor. The minimum duration will be 2 microclocks; the maximum duration will be 4096 seconds (about 68 minutes). Realistically the interval tick has a lower bound of about 5 ms imposed by FS latencies. It is initially phased to the major tick and subsequently to the previous interval tick.
Timebin Clock	The timebin clock is programmable by the backend processor. The minimum duration will be 1 microclock; the maximum duration will be 4096 seconds (about 68 minutes). It will be phase aligned to the interval tick and always less than or equal to an interval period (the time between two interval ticks)..

The interval and timebin clocks serve as the "coarse" and "fine" timestamps respectively which accompany science data to earth. Since intervals are always referenced to an initial major frame tick at the start of an obervation, science data can ultimately be correlated to UTC.

2.1.4 Overall EA Description

A block diagram of the EA is shown in Figure 3. The backend is shown at the left of the diagram, with the spacecraft and debug interfaces shown on the bottom. The instrument interface, shown on the top right of Figure 3, feeds the frontend event processing circuitry. The DSP core is shown on the lower right hand side of Figure 3.

The Harris 80C286 core, along with the DMA controller (Harris 82C237), the interrupt controller (Harris 82C59) and associated ROM, RAM and registers, serves as the backend processor. An optional debug interface (shown in dashed lines) is available for development, will not be included in the final flight hardware.

The TI 320C25 DSP and its associated memory and input FIFO comprise the frontend processor; the instrument mux and event processing circuitry assemble and filter events before arrival at the DSP input FIFO. Note that the PCA and ASM EAs are slightly different. The PCA EA contains lookup table RAM written by the backend processor and read by the event processing circuitry. The ASM EA does not contain a lookup table; therefore all ASM events are received by the DSP. However, the ASM EA contains a dedicated parallel-serial converter which is used to communicate with the ASM motor/drive and the three ASM scanning shadow cameras (SSCs).

2.1.4.1 Clock Circuitry

The master clock (32 Mhz - shown in the DSP core block of Figure 3) directly drives the DSP. It's divide by two version is used to generate the "raw clock" used by the backend clock-gating circuitry (see section 2.1.4.1.1 below). Since the input clock is 16 MHz, the effective operating frequency of the 80C286 is 8 MHz. The 16 MHz clock is further divided to generate an 8 MHz version used for clocking sequential logic in the frontend.

Figure 3 - EA Block Diagram

2.1.4.1.1 Clock-Stopper

Since the backend processor utilization is expected to be less than 50%, a clock-stopper circuit gates "raw clock", shutting down the processor (thus reducing overall power consumption) during idle time. The backend processor initiates the shutdown by writing to the CPU Pulse Register. Upon receipt of the pulse, the clock stopper hardware requests a bus hold from the 80C286; upon receipt of a hold acknowledge, the clocks to the 80C286 and associated controller hardware are supressed. The clocks are restarted by the occurrence of a system interrupt or reset.

2.1.4.2 Reset Circuitry

The EA reset sources are: a power-on reset, a watchdog reset (see section 2.1.4.2.1 below), a discrete commanded reset (see section below on System Manager Board for further detail) and a push-button switch generated reset. (The switch is a debug amenity for the prototype hardware; it will not be included in the flight hardware.) All reset sources charge an RC network with a time constant of about 50 ms, which guarantees the minimum reset pulse width.

The backend directly controls the frontend reset line, which is a readable and writable in the Observation Control Register (see Register Descriptions below). The frontend reset, referred to as DSPRst also directly drives DSP hold as well as DSP reset, placing the 320C25 address and data lines into high impedance when it is held in a reset state. DSPRst initializes other frontend functions as well, as described in section 2.1.4.6.1.2 DSPRst is also activated by an EA hardware reset.

2.1.4.2.1 Watchdog Reset

The watchdog reset is a hardware mechanism provided to double-check processor sanity. Its purpose is to yank the system out of an unknown and probably crashed state. Upon reset the watchdog timer is inactive. The CPU starts up the watchdog timer as part of its initialization procedure by pulsing the "STEP_ON_WATCHDOG" bit of the CPU Pulse Register. Once the watchdog is started, it operates autonomously in hardware and cannot be stopped by the system software. The watchdog timer will reset the CPU if it is not "stepped-on" once every 8 seconds. If a watchdog reset does occur, all EA hardware is reset (CPU, DSP, peripherals and logic) including the watchdog timer itself (it is deactivated).

2.1.4.3 Backend-Frontend Interface

The DSP is completely under the control of the CPU; its RAM (both scratch and histogram buffers) are memory mapped into CPU address space as shown in Table 1. The Observation Control Register permits the CPU to directly manipulate the DSP reset line.

The DSP scratch and histogram buffers are shared between the frontend and the backend. When the 80C286 requests an access to this area, the hardware asserts a "hold-request". If the DSP is reset, a hold acknowledge will return immediately; if not, the 80C286 bus cycle will be extended until a hold acknowledge is returned from the frontend. This can range anywhere from about 350 ns up to 80 microseconds (the upper bound if a repeat instruction has just started). It may be judicious to avoid DSP repeat instructions in order to minimize backend bus cycle latencies. However, due to the bus partitioning, see section 2.1.4.4 on EA Bus Structure, a frontend access induced 80C286 bus hold will not affect DMA controller latencies.

The DSP operation scenario is as follows: With DSPRst held, the CPU loads a program into DSP RAM beginning at the reset vector fetch address (0x0000). Any data parameters required by the DSP are loaded into DSP data scratch RAM. If execution from DSP internal RAM is desired, the

DSP program must include a copy routine, which transfers the DSP code to 320C25 (the DSP) internal RAM, and then jumps to the executable start address. When the CPU completes the DSP program/data load, it releases the DSP reset line.

The Lookup Table Memory (LUT) is shared by the CPU and the frontend in the following manner: when the DSPRst line is asserted, the CPU owns the LUT busses; otherwise they revert exclusively to the event processing circuitry.

2.1.4.4 EA Bus Structure

EA modules communicate via the following busses:

P-bus This is the 286 processor private data/addr bus. It connects the 286 to the various other busses described below. Only the 286 can drive the PADR lines which are then latched onto the other EA busses as directed by the Bus_Arbitor/Controller.

D-bus This is the memory and DMA controller bus. The DMA controller is granted access to this bus by the bus arbitor state machine upon receipt of DMA_HOLD_REQ. If DMA is not active, the 286 owns the D-bus, which houses its RAM, backend registers and DMA controller registers. The 286 can access the B- and F-busses while DMA transfers occur on the D-bus.

B-bus This is a general peripheral bus on which sits the 82C59 interrupt controller, the boot ROM, the frontend registers, and frontend lookup table RAM. Note that since interrupt acknowledge cycles occur on the B-bus, they do not affect DMA controller access latencies.

F-bus This is the DSP memory/peripheral bus. The DSP RAM is memory mapped into 286 address space. A 286 access to this space generates a DSPHOLDRQ. Upon receipt of a DSPHOLDACK, the Bus_Arbitor/Controller connects the B-bus to the F-bus. The DSP core circuitry ORs in the B-bus RAM strobes when DSPHOLDACK is active.

L-bus This is the lookup table RAM private data and address busses. With DSPRst asserted, the L-bus is accessible to the 286 via B-bus transceivers. When DSPRst is deasserted, it reverts to the event processing circuitry, which drives LADR and reads LDAT.

Somewhat unconventionally, the DMA controller (82C237) does not directly communicate with the 80C286 CPU. Instead HOLDRQ is directed to a bus arbitor (see figure 3), which grants D-bus access to either the DMA controller or the 80C286. Since the DMA controller is highest priority, uplink and downlink operations are guaranteed to meet their timing windows (see sections 2.1.4.8.1 and 2.1.4.7 on uplink and downlink) and are decoupled from CPU bus-locked or interrupt acknowledge cycles.

2.1.4.5 Diagnostic Hardware

Diagnostics fall into two categories: debug hooks, and operational diagnostics. (The former very often evolve into the latter.) This specification addresses only the provisions for diagnostics and not the diagnostics *per se,* which will be addressed in the future.

Since the EAs house two "smart components", much of the checking can be automated. Towards that end, all software settable signals, as well as general status lines, are readable by the appropriate processor. The backend CPU will be capable of performing basic diagnostics resident in its boot

ROM upon power-up. The DSP, executing out of its RAM under direct control and reporting back to the CPU, can similarly run through boot diagnostics. More complex sets of diagnostics can be loaded into RAM, both during the debug phase and operationally.

A set of 3 "LEDs" are provided in the CPU control register. These will evolve into discrete downlink signals in the actual flight hardware (see section 2.2 below on the System Manager). The LEDs are used to report EA status during the boot process.

In order to enhance the debug environment, an optional plug-in board is provided which contains ROM and RAM and a serial port controller. The debug board houses the Intermetrics monitor and RAM for monitor and debug usage only. The monitor communicates via a serial port to a host workstation which runs the high-level Intermetrics source-code level debugger. Access to/from the CPU is provided via a 16-bit data bus and a set of control signals. The debug devices are memory mapped into CPU space 0xD0000 through 0xDFFFF, as shown in Table 1. One serial port on the debug board is dedicated to the ROM resident monitor; an extra serial port allows for auxilliary debug I/O.

2.1.4.6 Backend (286 CPU) Core

The backend processor, the 80C286, (documented in the HARRIS literature) is a 16 bit processor with a 24-bit address bus. (NOTE: The 80C286 powers up in "real mode" in which only the lower 20 address bits are used; and the flight software subsequently operates exclusively in "real mode".) The processor clock is a 2X 16 MHz input; therefore the effective CPU frequency is 8 MHz.

A 256 Kbyte block of zero-wait state RAM is mapped to hex address 0 - 0x3FFFF (see memory map - Table 1). A 32 Kbyte block of 2 wait state boot ROM is mapped to hex address 0xF0000 - 0xFFFFF. RAM is both byte and word accessible; no byte-write support circuitry is provided for ROM, peripherals, registers and frontend circuits. (The 80C286 supports byte reads from all devices.) When byte wide peripherals are used (such as the 82C59 interrupt controller) the least significant address bit is connected to CPU address bit 1. Therefore peripheral address zero = base address; peripheral addresss one = base address + 2, etc., and only the lower eight bits of the data bus are used.

2.1.4.6.1 Registers/Devices

The CPU accessible registers provide auxilliary circuitry which support the spacecraft and instrument interfaces. They fall into two general classes: BREGS relate to CPU operation, and uplink/downlink; FREGS, the "Observation Registers", set up the frontend timing circuitry and oversee DSP operation. Unless otherwise noted, all device accesses incur 2 wait states. Although exact addresses are shown, there is aliasing within each address space, due to incomplete decoding of the address bus.

2.1.4.6.1.1 BREGS and 286 Peripherals

Mission Elapsed Time (Ox80000 and 0x80002) - read only

Register Layouts:

Bits 15:0
MET15:0

Address 0x80000 - Mission Elapsed Time Low Word

Bits 15:0
MET31:16

Address 0x80002 - Mission Elapsed Time High Word

Mission Elapsed Time is set by the SDS via a 1773 bus command and broadcast by the system manager hardware simultaneously to all EAs within a subgroup upon receipt of the command. The SDS Interface Specification stipulates that the MET will not be updated within 100 msec of the actual major tick. The FS should therefore read these registers within this 200 msec window.

CPU Control Register (0x90000) - read/write

Register Layout:

Bit Position:	**15:7**	**5:3**	**2**	**1**	**0**
Field:	*unused*	LED[0-2]	*unused*	Dnlk_Rqst	Self_Int

Field Definitions:

Self_Int When set to 1, this control bit posts a low priority interrupt (level 7) to the CPU. This interrupt will remain asserted until explicitly deasserted by the CPU clearing this bit.

Dnlk_Rqst This control bit, when set to one by the CPU, posts a request to the SDS data system for the RS422 1 MHz serial downlink channel. Once a downlink grant is received from the SDS; the EA hardware resets this bit to zero. See section 2.1.4.7 on downlink for further details.

LED[0-2] Diagnostic LEDs (1=> ON); these bits drive indicator lights on the prototype EAs and discrete 1773 telemetry registers (see section below on the System Manager discrete telemetry interface) in the final flight hardware.

CPU Control Register (0x90000) - read/write

0xFFFFF 0xF0000	Boot ROM	64 Kbytes - 2 wait state
0xE0000	Interrupt Controller	82C59-byte wide device - 2 wait state
0xDC000 0xD8000 0xD4000 0xD0000	Diagnositic Serial Ports Monitor RAM Unused Monitor ROM (R)	2692 - byte wide device - 2 wait state 8 Kbytes - 2 wait state 8 Kbytes - 2 wait states
0xC0000	DMA Controller	82C237-byte wide device - 2 wait state
0xB0000	Repository Memory (on System Manager) (R)	IPC (Actel FPGA) Addressable Registers 2 wait state (+ arbitration latency for Repository Memory)
0xA0000	CPU Pulse (W) / CPU Status (R)	
0x90000	CPU Control Register	
0X80002 0x80000	Mission Elapsed Time (HW) (R) Mission Elapsed Time (LW) (R)	
0x70000	Frontend Histogram Buffer	Frontend Memory 0 wait state (+ DSP arbitration latency)
0x60000	Frontend Scratch Buffer	
0x5000C 0x50008 0x50006 0x50004 0x50002 0x50000	Obs Pulse (W) / Obs Status (R) Observation Control Register BinClk Cntr Duration (HW) BinClk Cntr Duration (LW) Interval Cntr Duration (HW) Interval Cntr Duration (LW)	Observation Registers Resident in Timing Generator (Actel FPGA) 2 wait state
0x40000	Look-up Table RAM (PCA EAs) or Command/Status Register (ASM EAs)	2 wait state
0x3FFFF 0x00000	RAM	256 Kbytes - 0 wait state

Table 1 - Backend Memory Map

(HW = highword, LW = lowword, R = read only, W = write only)

CPU Status Register (0xA0000) - read only

Register Layout:

Bit Pos:	**15:9**	**8**	**7:5**	**4**	**3**	**2**	**1**	**0**
Field:	*unused*	Boot_Uplk	EAID[0-2]	DnInt	UpInt	Maj_Lat	Wdog_Act	Min_Lat

Field Definitions:

Min_Lat A value of 1 indicates the Minor Tick (31.25 ms clock) arrival from the PSDU is latched by EA Hardware. This bit is cleared by a write to the CPU Pulse register (see below); it also serves as the level 4 interrupt to the 82C59 interrupt controller.

Wdog_Act This status bit is a one if the watchdog timer has been activated by the CPU (see CPU Pulse register description below).

Maj_Lat A value of 1 indicates the Major Tick (1 s clock) arrival from the PSDU is latched by EA Hardware. This bit is cleared by a write to the CPU Pulse register (see below).

UpInt A value of 1 indicates that an Uplink packet has been received by the EA from the System Manager Board. It also occurs when an uplink packet is directed at the EA, but not received properly due to a DMA blockage. UpInt is the level 2 interrupt to the 82C59 and is cleared by a write to the CPU Pulse register (see below).

DnInt A value of 1 indicates that a Downlink transmission to the SDS has occurred and terminated. This bit also causes a level 3 interrupt to the 82C59 interrrupt controller and is cleared by a write to the CPU Pulse register (see below).

EAID[0-2] Processor ID - hardwired to EA "slot", it is a function of the connector into which an EA plugs in. Bit 2 indicates the EA subgroup. The actual wire levels are TBS.

Boot_Uplk Discrete Uplink signal, set by the ground and resident on the System Manager. If set, EA looks for uplink boot instead of the System Manager ROM.

CPU Pulse Register (0xA0000) - write only

Register Layout:

Bit Position:	**15:3**	**4**	**3**	**2**	**1**	**0**
Field:	*unused*	Rst_DnInt	Rst_UpInt	Stop_Clk	Pul_Wdog	Rst_MajMin

Field Definitions:

Rst_MajMin Writing a one to this bit causes the latched versions of Major Tick and Minor Tick to be reset to zero.

Pul_Wdog Writing a one to this bit causes the watchdog timer to be restarted. If the watchdog timer is inactive, this pulse causes it to be activated, after which it cannot be deactivated by software. (A hardware reset deactivates the watchdog timer.)

Stop_Clk Writing a one to this bit deactivates the CPU clock. The clock is reactivated by a CPU interrupt or a hardware reset.

Rst_UpInt Writing a one to this bit causes the latched version of the Uplink Interrupt to be reset to zero.

Rst_DnInt Writing a one to this bit causes the latched version of the Downlink Interrupt to be reset to zero.

Repository (0xBXXXX) - read only

A read access to the respository address space generates a request to the repository arbitor on the System Manager board. The EA is granted access within 8 (TBR) seconds for a 2 (TBR) second window. This process is performed in hardware; i.e. the requesting EA's bus will be held until the repository access is granted. Since reading the repository is only planned during a boot procedure following a reset, the 8 second bus holdoff should not be a problem. Of course, no interrupts can be reasonably serviced, and the watchdog timer must not be active.

Upon receiving access to the repository, the EA will read the data word residing at address 0 of the ROM first. A counter on the repository board increments each time a read occurs, running through the entire ROM (128Kbytes - TBR) followed by PRAM (64Kbytes - TBR). An EA is expected to complete the reading of the repository well within the 2 second grant window. At the beginning of each grant, the repository address counter is reinitialized to 0. For further details, see section 2.2 on the System Manager.

DMA Controller (0xC0000) - read/write

A Harris 28C237, clocked at 8 MHz, provides the uplink and downlink DMA channels. The EA uses two of the four available independent DMA channels, channel 3 for downlink and channl 4 for uplink. (Channel 3 is higher priority than channel 4.) See the Harris data sheet for further details.

Although the DMA controller is specified as a byte device, it is configured for word-only memory transfers. CPU address lines A1-A16 are connected (through the D-bus) to DMA controller address lines A0-A15. This implies that rather than a 64 Kbyte maximum transfer size, the EA implementation allows for a 128 Kbyte transfer size. During DMA transfers, A17 is driven low, effectively hardwiring DMA transfers to the lower half of CPU RAM.

The expected word transfer rates, determined by the I/O data rates of the uplink and downlink subsystems, are: 16 microseconds/word for a downlink memory read, and 20 microseconds/word for an uplink memory write. The DMA controller is given top priority on the D-bus by the bus arbitor controller.

DMA interrupts are generated by dedicated uplink and downlink state machines resident in the Interface Protocol Actel FPGA. The 82C37 EOP line is not used by the EA hardware.

The DMA controller should be programmed for active high DACK and DREQ, compressed timing mode, and extended write.

Monitor ROM (0xD0000) - read only

The Intermetric's monitor resides in ROM beginning at address 0xD0000. The total ROM space is 16Kbytes, residing on the D-bus.

Debug RAM (0xD8000) - read/write

The Intermetric's monitor uses RAM located at addresses 0xD8000 - 0xD8FFF, residing on the D-bus. Debug RAM is both byte and word addressable.

Diagnostic Serial Ports (0xDC000) - read/write

A Signetics 2692 DUART provides serial ports A and B for debug use; see the Signetics data sheet for further details. CPU address lines 1-4 drive the 2692 address lines 0-3; the byte-wide 2692 resides on the D-bus. The serial controller interrupt is the highest priority level (IR0) of the 82C59.

Interrupt Controller (0xE0000) - read/write

The interrupt controller is an 8 MHz 82C59 (Harris), a byte-wide support chip for the 8086 family. The base address of 0xE0000 is modified by the CPU address line A1 driving 82C59 line A0. Hence the data byte read at address 0xE0002 would be the 82C59 byte addressable at address 1.

The priority controller interrupts the 80C286 according to its programmed mode; its operation details can be read in the associated data sheets. For the purposes of the FS, the 82C59 is programmed to "fully nested mode" resulting in fixed interrupt priority levels. The 82C59 should be programmed for "level-triggerred" operation. The EA interrupt sources and their levels are:

82C59 Input	**Source**
IR0	Serial Port Controller Interrupt
IR1	Interval Interrupt
IR2	Uplink Interrupt
IR3	Downlink Interrupt
IR4	Minor Tick Interrupt
IR5	*Unused*
IR6	First Flag Interrupt
IR7	Self Interrupt

Boot ROM (0xF0000) - read only

The reset vector and EA boot routines reside in the boot ROM, which consists of 32K x 8 EEPROMs residing at addresses 0xF0000 - 0xFFFFF.

2.1.4.6.1.2 FREGS and Observation Control

Frontend memory which is accessable to the backend CPU falls into two general categories: (1) the registers/memory settable by the CPU which control event processing and timing circuitry, and (2) the shared memory which is owned by the frontend during observations, and which the CPU accesses via the DSP hold-request/hold-acknowledge handshake. The lookup tables, interval and binclock counters, and observations registers comprise the first category (see the description of the frontend event processing circuitry in section 2.1.4.14 for more functional detail). The second category consists of the DSP Scratch and Histogram Memory.

Lookup Table RAM (0x40000) - read/write - PCA EAs only

The lookup table RAM, a 64 Kbyte block which is programmable by the backend CPU, serves as the Recoder and Typer Lookup Tables during event processing. The lower half of the LUT RAM is devoted to the typer codes, the upper half to the recoder (see the Recoder and Typer Encoding Module Specifications for further detail). LUT RAM is word addressable only and resides on a private L-bus, which is switched onto CPU B-bus for programming.

The CPU is granted access to LUT RAM only when the frontend circuits are reset, achieved by asserting DSPRst in the Observation Control Register.

Command/Housekeeping Register (0x40000) - read/write - ASM EAs only

Each ASM EA is equipped with a parallel/serial converter in order to send commands to and receive status from the ASM. A write to this register automatically generates a serial stream transmission to the ASM. Since this line is shared and driven by both ASM-EAs as well as the various ASM devices, it is the resposibility of the FS to direct ownership of the line. (ASM devices only drive the line when queried for status by an EA.) It is the responsibility of the ground software to insure that one and only one ASM-EA is driving the interface. The "B" side of the EDS is the primary ASM command/status interface controller; the "A" side contains a backup interface which connects only to the motor assembly of the ASM. The details of this interface are described in the ASM/EDS Interface Control Document.

Interval Count Duration Register (0x50000 and 0x50002) - read/write

Register Layouts:

Bits 15:0
ICDUR15:0

Address 0x50000 - Interval Count Duration Low Word

Bits 15:0
ICDUR31:16

Address 0x50002 - Interval Count Duration High Word

The Interval Count Duration Register resides in the Actel Field Programmable Gate Array (FPGA) which handles frontend timing and observation control (TIMGEN - the lower block within the TG-FREGS block shown at the center top of Figure 3). A functional description of the timing circuits appears in section 2.1.4.13 of this document.

The Interval Count Duration sets the value of data loaded into a 32-bit Interval Counter, used to time observations. The TIMGEN FPGA containing the interval counter resides on the B-bus as a word device.

BinClock Count Duration Register (0x50004 and 0x50006) - read/write

Register Layouts:

Bits 15:0
BCDUR15:0

Address 0x50004 - Binclock Count Duration Low Word

Bits 15:0
BCDUR31:16

Address 0x50006 - Binclock Count Duration High Word

The Binclock Count Duration Register resides in the TIMGEN FPGA. A functional description of the timing circuits appears in section 2.1.4.13 of this document.

The Binclock Count Duration sets the value of data loaded into a 32-bit Binclock Counter, used to clock the timestamp counter.

Observation Control Register (0x50008) - read/write

Register Layout:

Bit Pos:	15:12	11	10:9	8
Field:	TBMask[0-3]	*unused*	EATrigOut[0-1]	HxTrigOut

Bit Pos:	7	6	5	4	3	2	1:0
Field:	EnIntTk	*unused*	DSPRst	*unused*	Nxt_FFIntEn	Nxt_Act	Nxt_IC[0-1]

Field Definitions:

NOTE: Further explanation of the use of these signals appears in section 2.1.4.14 of this document.

Nxt_IC[0-1] Next Interval Constant - this value is loaded into the Interval Constant Latches as a result of the next interval tick. It should be loaded by the CPU between intervals, with sufficient leadtime before the next interval tick.

Nxt_Act Next Active - determines whether the next interval is a data collection interval. If set to zero, writes to the DSP input FIFO will be supressed. It is updated in the manner of the interval constant (see above); however the advanced versions (those updated by the interval tick and first_flag word write) are cleared when DSPRst is asserted.

Nxt_FFIntEn Next FFIntEn - determines whether next interval causes an interupt to the CPU when the marker propagates through the event input FIFO. It is updated in the manner of the interval constant (see above).

DSPRst DSP Reset - An assertion of this bit places the DSP into reset and hold states. It also resets the event input FIFO, clears the advanced versions of Nxt_Active (see above) and switches the LUT RAM into CPU ownership. It is set active (to a high level) by a hardware reset of the EA.

EnIntTk Enable Interrupt Tick - This initializes the interval tick generator. It is set inactive (to a low level) by a hardware reset. The FS can initialize the interval tick generator by setting this bit (with sufficient setup time before the next Major Tick). The first interval marker is set coincident with the next Major Tick. Clearing this bit immediately disables interval tick generation.

HxTrigOut Hexte Trigger Out - Serves as a trigger out to the HEXTE detector system. The pulse duration, controlled by the FS, shall be greater than 20 microseconds and less than 100 milliseconeds. An EA reset clears this bit.

EATrigOut[0-1] EA Trigger Out - Two interEA triggers service each EA subgroup. The pulse duration, as for the HEXTE trigger out, is controlled by the FS. The pulse is converted to a level on each receiving EA (see description in Observation Status Register below). An EA reset clears these bits.

TBMask[0-3] Timebin Mask Field - The timebin mask suppresses the appearance of timestamp bits in the event word on a nibble basis: 4 bits of the event word timestamp are selectively masked by each bit of the TBMask; e.g. TBM0 masks counter bits 0 - 3. A timebin mask = 0 enables timestamping; a value of 1 disables timestamping. It is set by the CPU on a per observation basis.

Observation Status Register (0x5000C) - read only

Register Layout:

Bit Pos:	**15:6**	**5**	**4:3**	**2**	**1**	**0**
Field:	*unused*	Lat_HxTrig	Lat_EATrig[0-1]	Lat_FFlagInt	Lat_IntvInt	Lat_Spillage

Field Definitions:

Lat_Spillage Latched Spillage - Indicates if the DSP input FIFO has filled (thereby shutting down event collection for the remainder of the interval). It is reset by the next interval tick. The version read as status is delayed by one interval, i.e. reading Lat_Spillage in interval #3 implies that spillage occurred in interval #2. See event processing circuitry description later in this document.

Lat_IntvInt Latched Interval Tick - Set by the occurrence of an interval tick, this signal directly drives the level 1 input of the 82C59 interrupt controller. It will remain asserted until explicitly cleared by the CPU (see the Observation Pulse Register below).

Lat_FFlagInt Latched First Flag - Set by the concurrence of a first flag (reading the first word out of the input FIFO) and FFIntEn (also propagated through the input FIFO), this signal directly drives the level 6 input of the 82C59 interrupt controller. It will remain asserted until explicitly cleared by the CPU (see the Observation Pulse Register below).

Lat_EATrig[0-1] Latched EATrig - The two inter-EA triggers each set a latch independently upon arrival at each EA. These latches must be cleared by the CPU (see the Observation Pulse Register below).

Lat_HxTrig Latched HEXTE Trigger - The HEXTE trigger-in sets a latch upon arrival at each EA. This latch must be cleared by the CPU (see the Observation Pulse Register below).

Observation Pulse Register (0x5000C) - write only

Register Layout:

Bit Pos:	**15:4**	**3**	**2**	**1**	**0**
Field:	*unused*	ClrTrigs	ClrLat_FFlag	ClrLat_Intv	*unused*

Field Definitions:

ClrLat_Intv	Writing a one to this bit generates a strobe which clears the Latched Interval Interrupt (see Observation Status Register above).
ClrLat_FFlag	Writing a one to this bit generates a strobe which clears the Latched First Flag Interrupt (see Observation Status Register above).
ClrTrigs	Writing a one to this bit generates a strobe which clears the Latched EA and HEXTE trigger inputs. (see Observation Status Register above).

2.1.4.6.1.3 Shared Memory (0x60000 - 0x7FFFF)

Two 64 Kbyte blocks are shared between the frontend and the backend processors. Assertion of DSP_Hold determines which processor can access these blocks. When DSP_Rst (a bit in the Observation Control Register) is asserted, the DSP_Hold is simultaneously asserted. This assures that the CPU has access to DSP memory for loading DSP code and parameter blocks. The DSP boot code is loaded into base address 0x60000, which the DSP sees as program space at 0x0000 (see section 2.1.4.11 below on the DSP Core). All transfers to/from the shared memory region are word only.

During DSP execution, the CPU can obtain shared memory access by requesting a bus hold from the 320C25. This is done by the bus arbitor/controller upon decode of a CPU bus request for the shared memory region. If the DSP is not reset (in which case Hold_Ack will return immediately) there will be a 350 to 500 nanosecond latency before the CPU is granted access to the front-end bus (F-bus) (assuming that DSP repeat instructions are not used). TBR: Whether the CPU can then hold onto the bus continuously is a design issue which we need to address. This will normally not be the case, because the CPU alternately accesses its own RAM, copying the histogram buffer to CPU memory. TBD then is during a memory fill using the 286 string facility, do we want to grab the DSP busses for the duration of the fill? Since there is a sizable arbitration latency, it may make sense, but it does impose a real-time constraint: the FS must assure that the event input FIFO does not overflow due to a long string operation on DSP RAM.

2.1.4.7 Downlink (to SDS) Interface

Each EA connects individually and independently to the Spacecraft Data System via a 1 MHz Serial Link (see Figure 1). The serial interface consists of 4 differential pairs:

RTS RequestToSend - Output of EA, requests data transfer to SDS.

CTS ClearToSend - Input to EA, handshake from SDS in response to RTS. Indicates that the SDS will begin accepting data.

ST SendTiming - Input to EA, a data clock, valid only when CTS is asserted - nominal clock rate = 1 MHz.

SD SendData - Output of EA, data, must be valid with respect to ST.

The interface operates as follows: An EA asserts RTS and, after some arbitration delay (dependent on the downlink requirements of the other EAs and specified in the XTE SDS to EDS Interface Description), the SDS responds with a CTS and issues 8192 ST clock pulses. For each 16 bits transferred to the SDS, the EA DMA Controller is required to read a word from its internal RAM, turning it over to a shift register resident in the IPC (Interface Protocol Controller) resident in an Actel FPGA (see "DMA" block in Figure 3).

Operation from the FS point of view is: A data packet conforming to SDS CCSDS format is written into RAM. A DMA transfer of 1024 bytes from the RAM to DMA I/O channel 3 is setup and enabled. The FS then writes to the CPU Control Register, asserting Dnlk_Rqst. When a downlink interrupt (DnInt - level 3) is received, the FS may check that the packet was successfully transferred by reading the relevant DMA controller registers (see the 82C237 specification). The FS can also check if a packet transfer has started by checking the state of the Dnlk_Rqst bit in the CPU Control Register as the hardware resets this bit upon receipt of CTS from the SDS.

The Downlink hardware functions as follows: The IPC contains two 16-bit holding register. A state machine monitors the state of the Dnlk_Rqst Control Register bit (see CPU Control Register description above). Upon assertion of Dnlk_Rqst, the state machine asserts DREQD (the channel 3 DMA request input). A DACKD (channel 3 DMA acknowledge) causes the loading of one holding register. The EA hardware then asserts RTS. Upon receipt of CTS, Dnlk_Rqst is cleared, and the loaded holding register is shifted out to the SDS, clocked by the 1 MHz ST clock. Simultaneously, the other holding register is loaded via DMA. Each time the loaded holding register is shifted out, (after 16 ST clocks) a buffer swap occurs and another DREQD occurs, initiating a D-bus RAM to IPC data transfer. Thus the required DMA transfer rate is one 16-bit word every 16 microseconds. An actual transfer, once the bus is granted, will require approximately 500 nanoseconds. A deassertion of CTS by the SDS causes a downlink interrupt (DnInt) to be generated.

2.1.4.8 System Manager Interface

Each EA communicates via a 16 bit repository data bus which serves two purposes: it allows the EA to read out of a shared memory bank containing the flight software image and default parameter blocks, and it serves as the path through which uplink data is received. It consists of the 16-bit RDAT bus and the following sets of control signals (per EA -> the system manager interface must supply a dedicated version of this signal for each EA in the subgroup):

RRQST	Output	per EA, requests the use of the Repository for copying the ROM and PRAM. This is asserted by the EA hardware upon decoding a a repository read access.
RGNT	Input	per EA, grants the use of the Repository in response to RRQST above. This is decoded by the EA hardware, allowing a repository read to continue.
UPSTB	Input	per Subgroup, strobes uplink data into the EA selected by UPID (see below)
UPID[0-2]	Input	per Subgroup, selects active EAID for uplink (see UPSTB above). Unique UPID codes are reserved for elapsed time count transfers to all EAs simultaneously. The MSB of UPID is the EA subgroup ID during uplink packet transfers, and otherwise indicates a control transfer when qualifed by UPSTB.
DWN_DIS[0-2]	Output	per EA, these are the downlink discrete lines ("LEDs") which every EA drives (in CPU Control Register). These levels are transferred to the ground by the SDS via the 1773 bus.
UP_DIS[0-1]	Input	per EA, these are the uplink discrete command lines which every EA receives. There are currently two specified lines: DIS_RESET (UP_DIS[0]), a commanded reset, and BOOT_UPLK (UPDIS[1]), boot from uplink (in CPU Status Register).

The system manager controls access to its data bus, determining when an EA can read the ROM/PRAM, and transmitting/dispatching uplink packets as they arrive from the 1773-bus. See section 2.2 on the system manager for further details.

2.1.4.8.1 Uplink Controller

The uplink interface operates as follows: The system manager board receives a CCSDS packet from the SDS 1773 bus. Each packet has a subchannel address associated with it dependent on its function code (see the SDS to EDS Interface Description for a description of the use and mapping of function codes). The system manager translates subchannel address to EAID which it outputs as UPID. Upon uplink packet detection, the system manager transmits a "start of packet" code to all the EAs in the subgroup. As each packet word arrives, an UPSTB is transmitted by the system manager along with the 16-bit data word. EAs compare EAID to UPID and if a match is detected, strobe in the uplink word, and perform a DMA transfer to CPU memory. This continues for the entire packet length (<= 64 bytes). NOTE: DMA channel 4, dedicated to uplink, should always be enabled by the EA.

If an EA is unable to accept a packet due to an internal error condition, no status is returned to the system manager. (Status is however returned to the EA via an Uplink Interrupt assertion.) The remainder of the uplink packet is transmitted to the EA but the EA hardware prevents the writing of it into RAM. The system manager does not detect an internal EA error condition (due for instance to a DMA blockage) and hence no error status is returned to the SDS. Upon termination, the system manager transmits an "end of packet" code to all the EAs. This causes the receiving EA to exit from the packet reception state, also causing an Uplink interrrupt (UpInt - level 2).

Operation from the FS point of view is: when an EA is ready to receive uplink, following its boot process, the FS sets up a 64 byte DMA transfer from the uplink channel to memory. When the Uplink interrupt is received, the FS checks that the packet was successfully transferred, reading the relevant DMA controller registers (see the 82C237 specification). If a packet was terminated prematurely or is less than 64 bytes, the CPU can determine the actual transferred packet size by reading the counter value. In order to receive all of its directed packets, and EA should always enable DMA channel 4 for uplink.

The word transfer time on the 1773 bus is 20 microseconds, requiring a DMA turnaround time of less than 20 microseconds. Since the DMA controller is the highest priority device on the D-bus, the actual latency is no longer than 2 microseconds.

A packet transfer takes about 0.8 milliseconds. (See the MIL-STD 1773 Bus Specification for more detail.) The SDS guarantees at least 30 ms between each packet transmission (as per the XTE SDS to EDS Interface Description).

2.1.4.8.2 Mission Elapsed Time Transfer

The mission elapsed time broadcast is a special case of uplink transfer from the 1773 bus to the system manager and ultimately to the EAs. A subchannel on the 1773 bus, dedicated to time-code broadcasts, is translated by the system manager into two UPIDs which are decoded by all EAs to address their Mission Elapsed Time Registers. The MET HW and LW are transferred successively by the system manager and are strobed into all the subgroup EA registers. The system manager strips all subsecond information. Since the transfer does not occur within 100 ms of a major frame tick, the MET data must be read by the FS inside this 200 ms window. See the system manager description in section 2.2 for further details.

2.1.4.9 PSDU Interface

Each subgroup receives the three differentially driven PSDU clocks via 26C32 CMOS differential receivers. The clocks (MajorTick, MinorTick and MicroClock) are issued from two independent PSDU sources. All clocks are guaranteed to be in phase to a tolerance of less than 100 nanoseconds (TBR).

The clocks are received on the System Manager boards and routed to the subgroup EAs. Also received is a signal indicating for each PSDU if it is active or on standby. A fixed priority scheme selects which clocks are used if both PSDUs are active. Details of this interface will be included in Appendix A.

The PSDU also supplies +28VDC (unregulated power), one copy/EA subgroup, which is received, regulated and distributed on the system manager board.

The PSDU must receive the following analog discrete telemetry lines from the EDS: Regulated +5 V measurment from the two EA subgroups, and the EDS baseplate temperature. Details of this interface will be included in Appendix A.

2.1.4.10 InterEA Burst Trigger Interface

Each EA in a subgroup can drive two open collector lines: T1 and T2 in Figure 1. A rising edge on this line clocks a high level data input to a latch on each EA, readable in the Observation Status Register. Appendix A shows a schematic of the physical interface.

Note that the private EA copies of trigger-out and trigger-in, although separated on the individual EA are ultimately coupled via the open collector shared trigger line. This implies that an EA asserting a trigger pulse, will trigger itself. Note also that once an EA drives the trigger, it is the responsibility of the FS running on that EA to release the Trigger line by deasserting the appropriate EATrigOut in the Observation Control Register. (The minimum pulse width is smaller than could be generated by the FS, less than 50 nanoseconds.)

2.1.4.11 Frontend (DSP) Core

The frontend DSP and its associated memory and registers are shown to the right in Figure 3 (blocks DSPCORE and DSPRAM). The DSP is a Texas Instrument 320C25 (see data sheet for more detail), containing a controller, 32-bit ALU (arithmetic logic unit), and 16-bit external address and data busses. The DSP, configured in microprocessor mode, contains 256 words of internal program memory, and 288 words of internal data RAM. It is clocked at 32 MHz and incurs a 1 wait state penalty for all external accesses. The frontend memory map (Table 2) shows the various internally addressable blocks as well as external memory and registers.

The 320C25 provides separate program and data strobes; code and data address spaces have been separated to allow for additional capacity. The Scratch RAM spaces are separate sections of RAM for data and program accesses. The 320C25 provides block move instructions which allow for shuttling of data back and forth between the code and data sections. (This shuttling could also be done by the backend CPU which has access to the DSP RAM.)

Since the 320C25 is a word-only device, its 16-bit address bus can access 128 Kbytes or 64 Kwords of memory. 64 Kwords of RAM are partitioned as follows: The lower 16 Kwords respond to a program strobe and DSP address range 0x0000 to 0x3FFF. This is the boot area and is loaded by the backend CPU (at 286 base address 0x60000) with the appropriate DSP code prior to the release of the DSP reset line by the CPU. The next remaining 48 Kwords respond to a data strobe and DSP address range 0x4000 to 0xFFFF (maps to CPU range 0x68000 to 0x7FFFF).

2.1.4.11.1 Registers/Devices

The DSP has one device mapped into its memory space:

Event Input FIFO (0x2000) - read only

The Input FIFO contains the processed event stream (see section 2.1.4.14 on Event Processing). Only the lower 15 bits contain data; the MSB (most significant bit) is devoted to a "first-flag" indicator, described in more detail in section 2.1.4.14.3 of this document. Since address decode is incomplete, DSP reads directed anywhere in the address space from 0400 to 3FFF hex cause the FIFO to be read.

Event Input FIFO (0x2000) - read only

Address	Code Space	Data Space	CPU Address Mapping
0xFFFF 0xFF00	*Internal Block B0*	Buffer RAM	0x7FFFF 0x70000 to Buffer RAM
0x8000	Unused		
0x4000	Unused	Scratch Data RAM	0x6FFFF 0x68000 to Scratch Data RAM
0x0000	Scratch/Boot Program RAM	Event Input FIFO (Read Only) *Block B1: 0x0300 – 0x03FF* *Reserved: 0x0080 – 0x0200* *Block B2: 0x0060 – 0x0080* *Registers: 0x0000 – 0x0006* *Internal Blocks/Registers*	0x67FFF 0x60000 to Scratch/Boot Program RAM

Note: *Italic type signifies memory internal to the DSP.*

Note: *The LSB of the CPU Address is ignored (CPU A1 maps to DSP A0).*

Table 2 - Frontend Memory Map

2.1.4.12 Instrument Interfaces

The instrument interfaces to the ASM and PCA consist primarily of data paths from the detectors, serial-to-parallel converters, and multiplexers (muxes) which create an "event bus" feeding the EA event processing circuitry. There are also auxilliary interfaces paths for broadcasting status (the HEXTE trigger interface) and for issuing commands (i.e. the ASM Command/Housekeeping Interface).

2.1.4.12.1 PCU/SSC Event Muxes

The PCU and SSC event muxes (resident in Actel FPGAs) both receive and parallelize serial data streams from their respective instrument sources. The basic architecture is shown in Figure 4.

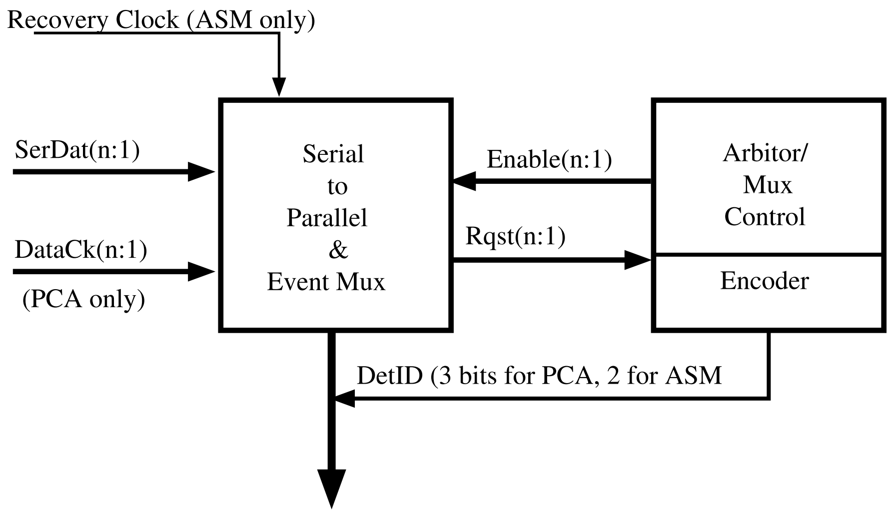

Figure 4 - Event Mux Architecture

The event muxes input "n" streams of serialized detector data, where n=5 for the PCA and n=3 for the ASM.

PCU Event Input

A "DataCk" signal (nominal clock rate = 4 MHz ± 1 MHz) accompanies each PCU stream, qualifying and clocking event stream bits as they arrive at the interface. (See the PCA-EDS Interface Control Document for further details on the PCU event interface) The data lines are held at logic zero if no event is being transmitted. A start bit (=logic one) preceeds an event and serves as the request line, after which a full event word (19 bits) is received. A stop-bit (=logic zero) always follows an event transmission.

SSC Event Input

The ASM data is received at a slower rate, and in order to minimize the number of interface wires, data is self clocked. The interface runs at 512 KHz, and clock recovery at 8 MHz. Like the PCU data, the lines are held at logic zero if no event is being transmitted. One start bit (logic one) preceeds each event word and one stop bit (=0) follows each event word. A state-machine monitors the data line, looking for a start-bit. After detecting a start-bit, the state-machine clocks itself to mid-bit and generates a receiver clock for the following bits of the SSC event word.

Event Arbitration

Upon event reception, the serial to parallel register generates a request to the arbitor. The detector requests are synchronized to the arbitor clock and considered in a fixed priority fashion, with non-requesting sources skipped. Once an event request is granted, the arbitor asserts the appropriate enable and detector ID; event processing occurs during the current microclock. For the PCA, the event request lines also cause the timestamp latches to be clocked (see description of PCA event processing circuitry in section 2.1.4.14.6).

Interface

The receivers are standard RS-422 compatible low-power CMOS 26C32 differential receivers. Termination will be provided within the EDS. The EAs are in close physical proximity, allowing for short-stub teeing of the the signals within the EDS. The termination scheme insures that a failure of one receiver will not interfere with the remaining receivers on that line. The interface details are included in Appendix A.

2.1.4.12.2 ASM Command/Housekeeping Interface

The EDS sends commands and status queries to the ASM via a bit serial line similar to the data interface describe above. For simplicity, commands and status queries are confined to 16-bit words, which presents a straightforward decode task to the ASM. As far as the backend CPU is concerned, the ASM command/housekeeping register looks like a read/write 16-bit register, memory mapped into CPU address space. However a write into the register activates a parallel to serial conversion and transmission of the bit serial stream to the ASM. The interface is self-clocked (as the SSC science data interface described above) with a data rate of 16 Kbits/second. The 512 Kbit/second clock (used to transmit science data) recovers the command/housekeeping stream at each ASM device (motor and the three SSCs).

At any point in time, only one EA drives the Command/Housekeeping interface. The primary EA is able to communicate with the Motor/Drive, and the three SSCs (see Figure 5). The secondary EA serves as a backup command system and runs only to the Motor/Drive. The stepper motor in the ASM has two windings; each winding is driven by a different EA residing in a different EA-subgroup. ASM regulated +5V is supplied to each side of the motor control from the 2 EA subgroups separately, controlled by discrete uplink (see section 2.2 on the System Manager for further detail). (28 VDC is separtely supplied to the motor windings.)

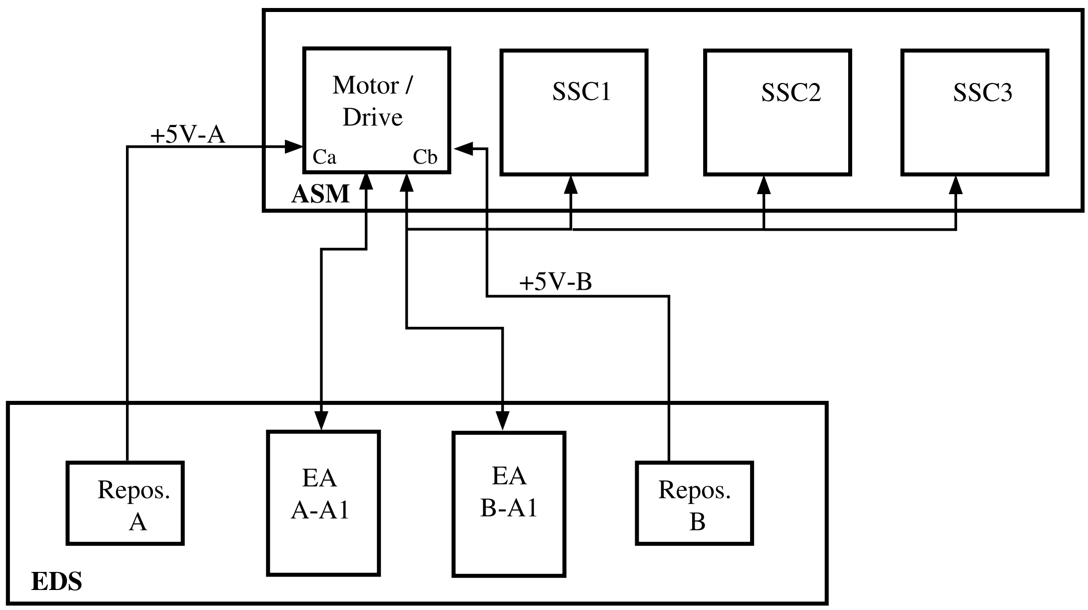

Figure 5 - ASM Command/Housekeeping

The Command/Housekeeping Interface is "owned" by the currently selected EA, at mission onset, the primary EA. This means that this EA is responsible for insuring that the bus is driven by only one source at a time; the controlling EA must also insure that, following a query, sufficient time has passed, before reading the response.

Commands

The ASM Motor Drive receives rotation commands from the EDS. The format of these commands is described in the ASM/EDS Interface Control Document.

The SSCs can be commanded to perform various control functions relating to event processing. These command options are described in the ASM/EDS Interface Control Document. (Note: Only the ASM EA on the B side of the EDS is capable of commanding the SSCs.

COMMAND CYCLE TIMING

Hardware enforces the recovery time; it the responsibility of the FS to insure that adjacent commands are separated by more than 3 ms. Commands received while the hardware is busy will be dropped. For a detailed description of the command cycle bus phases, see the ASM/EDS Interface Control Document.

HOUSEKEEPING QUERY (for both the Motor/Drive and the SSCs):

For housekeeping queries, only one piece of information can be requested at any one time. The format of the Housekeeping Query is described in the ASM/EDS Interface Control Document.

STATUS QUERY TIMING

Hardware enforces the recovery time; it the responsibility of the FS to insure that adjacent status queries are separated by more than 6 ms. Status queries received while the hardware is busy will be dropped. Similarly, the FS should not attempt to read the returned status information until at least 6 ms after the initiation of the query. For a detailed description of the status query bus phases, see the ASM/EDS Interface Control Document.

The command/housekeeping interface, like the science data interface described above is implemented with the CMOS family of differential drivers/receivers (26C31 and 26C32). Since the drivers can be selectively enabled, all bus components can be physically connected; each hook-up to the bus connects to a 26C31/C32 driver/receiver pair connected to form a transceiver. An interface schematic is shown in Appendix A.

2.1.4.12.3 HEXTE Burst Trigger Interface

The HEXTE burst trigger interface consists of two "trigger-in" signals, each corresponding to a HEXTE cluster and one "trigger-out", a logically ORed wire driven by all EAs in one of the two subgroups (see Figure 1). The trigger-out drives both HEXTE clusters. The FS controls the HEXTE trigger-out pulse width and is required to meet the 20 microsecond to 100 millisecond pulse-width specification (see the HEXTE-EDS Interface Control Drawing). HEXTE and the EDS are optically coupled; the interface schematic is shown in Appendix A.

The HEXTE trigger-in signals are ORed together upon arrival at each EA. Upon assertion of the pulse, an internal latch is set per EA (readable in the Observation Status Register); the latch is reset by the backend CPU (via a write to the Observation Pulse Register).

2.1.4.13 Timing Generator

The event time measurement circuitry is controlled by the backend CPU and interacts extensively with the event processing circuitry. There are three counters resident on each EA: an interval generator, a binclock generator and a timebin counter. The interval tick is used both to place time markers in acquired science data and to signal to the CPU when to close out a partition's worth of data, and packetize it for downlink. The binclock, as the timebin counter clock, controls the effective resolution of event time-stamping. The output of the timebin counter, if enabled by the current event processing mode (see section 2.1.4.14 on event processing), appears to the DSP along with the parallelized detector event words.

The timing generator is divided between two Actel FPGAs shown at the top of Figure 3. "Timing Generator/Fregs" contains the interval generator and the binclock generator. Both the Interval Tick and the BinClock are inputs to "Event Proc/Timebin Counter" which houses the event processing circuitry along with the timebin counter.

2.1.4.13.1 Interval Generator

The interval generator is a 32-bit counter, clocked by the MicroClock (953 nanoseconds) and programmable by the backend CPU. An Interval Duration register appears at CPU addresses 0x50000 (low word) and 0x50002 (high word), (see Backend Memory Map, Table 1). The maximum interval duration is about 68 minutes; a minimum duration of 2 MicroClocks is possible (in hardware), however the actual allowable minimum duration will be determined by FS latencies. Durations operate as follows:

Duration Programmed		Interval to Interval Spacing	
HEX	DECIMAL	in MicroClocks	in Seconds
11	17	18	17.2 microseconds
1	1	2	1.91 microseconds
0	0	$2^{32} + 1$	4096 seconds
FFFFFFFF	4.3×10^9	2^{32}	4096 seconds

Upon reset, interval tick generation is disabled. Setting the EnIntTk bit in the Observation Control Register initializes the interval generator by causing an inteval tick coincident with the next major tick. Note that the FS is reponsible for writing the EnIntTk bit with sufficient set-up time to allow deterministic generation; i.e. it must not be set within 1 microsecond of the next MajorTk onset.

The 32-bit Interval Counter is loaded with a duration count upon generation of an interval tick. Following the first interval tick, aligned to MajorTk, the interval circuitry free-runs loading a duration count every interval tick. The FS can change the duration "on-the-fly" between intervals; however the duration count load must occur before the next interval tick. For example, following interval tick #3, the FS schedules the gap between interval ticks 4 and 5, and must load the duration words in advance of tick #4 . For constant length interval generation, the FS need do nothing more than set up the duration, set EnIntTk, and let it roll.

Interval Tick Generation is halted immediately by the clearing of EnIntTk by the FS. Again, this must be done with sufficient set-up time; 1 microsecond or more before the next scheduled interval tick.

2.1.4.13.2 BinClock Generator

The binclock generator is a 32-bit counter, clocked by the MicroClock and programmable by the backend CPU. A BinClock Duration register appears at CPU addresses 0x50004 (low word) and 0x50006 (high word), (see Backend Memory Map, Table 1). The maximum BinClock duration is about 68 minutes (2^{32} MicroClocks); the minimum duration is 1 MicroClock. Note that a zero duration has a different effect on the binclock generator than the interval generator. Durations operate as follows:

Duration Programmed		Binclock to Binclock Spacing	
HEX	DECIMAL	in MicroClocks	in Seconds
11	17	18	17.2 microseconds
1	1	2	1.91 microseconds
0	0	1	953 nanoseconds
FFFFFFFF	4.3×10^9	2^{32}	4096 seconds

The BinClock duration is loaded at the interval tick (thereby phase-aligning it to the Interval Tick) and reloads every time a BinClock occurs. Effectively, it is a free-running sequence generator within the interval tick boundaries. The output of the BinClock generator is used to clock the Timebin counter, which is described below.

2.1.4.13.3 Timebin Counter

The Timebin Counter is a 15 bit binary counter, clocked by the BinClock (see above). Its output feeds the event processing circuitry, timetagging instrument events. The timebin counter is reset at the interval tick only, and otherwise freeruns, wrapping at its terminal count.

2.1.4.13.4 Timestamp Latches (PCA EAs only)

The timestamp latches, 5 15-bit latches resident in the "Event Proc/Timebin Cntr" FPGA, allow PCU events to be timestamped upon arrival rather than upon servicing by the event controller/arbitor. The timestamping occurs for each PCU as an event is received by the instrument mux and as a request for service is sent to the event controller/arbitor. When a request is eventually serviced, its latch is directed through a multiplexer into the event processing circuitry (see the PCA Event Processing description, section 2.1.4.14.6 for further detail).

2.1.4.14 Event Processing

The event processing circuitry is architecturally similar for the ASM and the PCA EAs. Both receive a wide event word from the instrument multiplexer (see section 2.1.4.12.1 above on the Instrument Muxes) along with an event qualifier bit which activates the event processing circuitry. Both timestamp the event and write the data into the event input FIFO. However, the PCA event processing involves two look-up tables accesses, event rejection, and a compression of the event word and its timestamp into 15-bits, while the ASM event processing circuitry merely sequences the event information into three 15-bit words leaving compression and event rejection up to the DSP. The event processing circuitry, along with the instrument muxes, is shown in Figures 6 (PCA) and 7 (ASM).

2.1.4.14.1 Sequence Active

Event processing is enabled by CPU assertion of the Nxt_Act bit in the Observation Control Register during the interval preceeding the desired acquisition. Nxt_Act is latched by the following interval tick; the latched version, SEQACT, enables event acquisition and is cleared when DSPRst is asserted.

2.1.4.14.2 Interval Constants

Like SEQACT, the CPU is responsible for setting the Nxt_IC[0-1] bits in the Observation Control Register during the interval preceeding an event aquisition interval. Nxt_IC[0-1] are latched by the next interval tick and become IC[0-1], the interval constants which feed the event processing circuitry. Interval constants are used primarily to adjust the addressing of the DSP buffer memory, allowing events to be written to different sections of memory depending on their acquisition interval.

2.1.4.14.3 First_Flag

The event circuitry writes a "first_flag word" into the input FIFO upon detection of an interval tick. The first_flag word is written regardless of the state of SEQACT (see section 2.1.4.14.1 above) and includes relevant interval status bits, such as SPILLAGE (see below) status for the previous interval. The first_flag word alway has bit 15 (known as "first_flag") and bit 0 cleared (=0). This allows binned mode to run "blindly", i.e. the DSP need not analyze the first_flag word because it addresses a reserved location in DSP scratch RAM, similar to FIFO_empty words (see below). One bit of the first_flag word is dedicated to FFIntEn (in the Observation Control Register). When the first_flag word is read out of the event input FIFO by the DSP, the combination of first_flag = 0 and the assertion of FFIntEn will cause a FFLAG interrupt to the backend CPU.

2.1.4.14.4 Spillage

A spillage condition occurs when the event input FIFO fills. For the PCA, spillage occurs when 1024 events (or first flag words) are awaiting DSP processing; for the ASM, when 342 events are pending. If spillage occurs, a latch is set, which prohibits further events from being written into the event input FIFO. The spillage latch is reset at the next interval tick, allowing event acquisition to resume. The first_flag word is the first entry written to the event input FIFO following a spillage.

2.1.4.14.5 Event Input FIFO Control

In addition to the spillage detection circuitry, hardware monitors the FIFO_EMPTY signal, preventing the DSP from reading an empty FIFO. If the DSP attempts to read the input FIFO while it is empty, the read strobe is supressed and the data word 0x7FFF is driven onto the FDAT bus. Thus, the DSP can run "blindly" in binned mode; a reserved location in DSP scratch RAM increments each time an empty FIFO read is attempted.

The Event Input FIFO is cleared with the assertion of DSPRst (a bit of the Observation Control Register).

2.1.4.14.6 PCA

Figure 6 shows the serial data streams feeding the instrument mux, which outputs a 22 bit event bus containing PCU event information as well as the PCU detector ID. Note that timestamping of incoming events occurs immediately upon a request for arbitor service from the parallel event latches resident in the EVINP module. Thus actual event timestamping may be several microseconds before the event is processed.

Figure 6 - PCA Event Processing

Upon detection of a valid request, the EVCONT block performs all the necessary tasks associated with an event: typer lookup (the lower half of LUT RAM) and latching, recoder lookup (the upper half of LUT RAM) and latching, and if the event is accepted, strobing of the event input FIFO write line. Both the Recoder and Typer table data (bit 15) can potentially reject an event.

The latched Typer and Recoder words feed a giant OR, along with the timetag and interval constant. A typer output (bit 0) is dedicated to timetag disable (if set, the entire timetag is zeroed). The timebin masks (if set in the Observation Control Register) can also zero the timetag in nibble increments. Typer activated recoder disable is accomplished via the three typer bits which feedback as the recoder lookup address. (See the LUTADMUX module of Figure 6 for a description of the Lookup Table addresses. The giantOR module details how the recoder and typer event words are ordered into the combinational logic.)

Note that the giant OR outputs 15 bits of information. The MSB is used to mark the first event of an interval. This is accomplished with the use of a "first-flag word". The arrival of the "first-flag word" is treated by the EVCONT as follows: All pending events (whose requests were received before the arrival of the first-flag word) are written into the input FIFO. The first-flag word is then written into the input FIFO, and simultaneously the latched interval modifier bits (e.g. interval constants, seq_active) are updated to reflect the current interval status.

2.1.4.14.7 ASM

Due to the slower ASM event rate, the EDS hardware dedicated to ASM event processing is simpler than that for the PCA. There is no event rejection circuitry; the DSP, however, can reject events. The 30-bit ASM event word along with its corresponding 10-bit timetag and interval constant is simply sequenced into three 16-bit words. This "tagged-sequenced event" consists of the following format:

WORD1:

15	14	13:12	11:0
First_flag (1)	Word_Synch (1)	Int_Const[0-1]	Pulse_height_A[0-11]

WORD2:

15	14	13:12	11:0
First_flag (1)	Word_Synch (0)	Det_ID[0-1]	Pulse_height_B[0-11]

WORD3:

15	14	13:10	9:0
First_flag (1)	Word_Synch (0)	Anode_code[0-3]	Timetag[0-9]

For all ASM event words written into the event input FIFO, the first-flag bit is set (=one). The first-flag word written into the FIFO upon detection of an interval tick has this bit cleared (=zero), enabling the DSP to determine interval marks within the event data stream. The FFIntEn bit, part of the first_flag word, gates the first_flag interrupt to the backend CPU. FFIntEn, which is set by the CPU (Observation Control Register), determines whether a " first_flag_out" for a particular interval causes a CPU interrupt.

Word_synch is set to one for word one of any event and is zero for words two and three.

The Interval Constant (Int_Const) is an arbitrary 2-bit binary code set by the CPU, and latched on a per interval basis (as described for the PCA event processing circuitry). It is set by the CPU (in the Observation Control Register) prior to interval tick generation.

The timetag is truncated to the 10 MSBs, giving ASM events an effective resolution of 32 microseconds. (See section 2.1.4.13 on the Timing Generator).

Pulse Heights (corresponding to event energy) and anode codes arrive from the SSCs; detector ID (1, 2 or 3 corresponding the SSC generating the event) is added by the EA event input circuitry. Anode code is defined in the ASM/EDS Interface Control Document.

The Input FIFO controller operates for the ASM EA much in the same way as for the PCA EA. An empty FIFO will not be read, and the hardware will generate a data word of 7FFF hex. (This contrasts with the First_Flag word which always has bits 15 and 0 cleared.) Note that both the First_Flag and FIFO-empty indicators are only one 16-bit word, as opposed to the event word, which is three 16-bit words.

Figure 7 - ASM Event Processing

2.2 System Manager

Each EA subgroup contains a dedicated System Manager board which performs miscellaneous services and communicates with the spacecraft via the 1773 serial bus.

2.2.1 Functional Overview

The System Manager block diagram is shown in Figure 8. It contains four basic components as described below. It is responsible for receiving and distributing uplink packets for an EA subgroup, appearing as an RT (Remote Terminal) on the 1773 bus. (The 1773 bus is a fiber optic version of the MIL-STD 1553 bus, a 1 Mbit/sec Manchester Encoded serial bus.) The 1773 bus is used to receive discrete commands and to send discrete downlink. The 28 VDC unregulated power supply from the PSDU is also received on the System Manager, where it is then regulated to +5 VDC and distributed through relays to the EAs and the ASM (motor control). Finally, the System Manager houses two banks of memory which are readable by the subgroup EAs: 128 Kbytes of ROM and 64 Kbytes of RAM.

2.2.2 Arbiter/Controller

The arbiter determines access to the Repository memory bank. Contending for memory access are:

a. EAs, 4/subgroup, contend to read the entire ROM and PRAM (protected RAM) areas.

b. the 1553 bus protocol chip/controller, highest priority, contends to write to the PRAM and to use the RDAT bus for transmitting uplink to an EA.

c. the ECC (Error Correction Code circuitry), lowest priority, cycles through the PRAM area when no other requestors are active.

The highest priority is granted to the 1553 bus controller because it must keep up with the 20 microsecond/word real-time uplink rate. It can interrupt an in-progress EA grant (see below).

The EAs are next in priority and are treated as follows: The grant period for an EA is two seconds (TBR). At the beginning of a grant period, the lowest EAID requestor gains the grant with the restriction that ownership of the previous grant prevents a current grant. This simple arbitration scheme is possible because of the somewhat deterministic nature of the ROM/PRAM request pattern. (An EA will only request access following a reset, or as a result of a severe crash.) The controller has a built in counter which generates the ROM/PRAM address; the counter is reset to zero at the beginning of a grant, and increments each time an EA reads a word (RRQST must act as a strobe). If an EA grant is active and a 1553 bus request arrives, the EA is held off (following the delivery of the currently accessed address to the EA) by deassertion of RGNT. When RGNT is again returned (in no more than one microsecond) the EA picks up where it left off, reading the next address out of the ROM/PRAM area. Since the local "ready" signal on each EA is gated by RGNT, the EA's CPU bus is held while the 1553 bus protocol chip delivers a data word to either the PRAM or to an EA's uplink hardware.

Figure 8 - System Manager

The ECC circuitry kicks in on an arbitration period onset when no other arbitration requestor is active. It uses the counter, and works its way through PRAM, reading the array, and writing corrections for single bit errors.

2.2.3 Memory

The memory consists of a 64K x 16 bank of EEPROM and a 32K x 16 bank of PRAM, both resident on the RDAT and RADDR busses. The EEPROM contains the flight software and default parameter blocks; the PRAM exists to augment the ROM, either with patches to the FS or loadable parameter blocks.

Protected RAM is not writable by the EAs (and hence it is protected from occasional processor insanity). It is loaded via ground command via the SDS, which then addresses (by using the specified function code for the PRAM) the PRAM, delivering a CCSDS conformant uplink packet to the appropriate RT. The system manager hardware strips the CCSDS packet headers and looks for the first word of the packet which it uses to define a memory address; subsequent words are loaded into the next consecutive location in PRAM. Transmission terminates at the end of the immediate uplink packet; subsequent PRAM loads begin with another uplink packet directed at the PRAM.

The PRAM is also protected from SEUs (single event upsets) by either an ECC scrubbing state-machine or by using extra radiation-hardened parts. This is TBD, and depends on the availability of the rad-hard RAMs. This specification assumes a baseline of non-rad-hard parts; a change to a rad-hard part only simplifies the design.

The PRAM array consists of two 32Kx8 static RAMs, and an extra 32Kx8 RAM which holds the check bits. The actual generation of the check code (a 6-bit Hamming code) will occur each time a word is written into PRAM. At each read (whether for scrubbing or as the result of a processor request), the code is regenerated and compared with the stored code. On a single bit conflict, the memory word is reconstructed and rewritten. On a double bit conflict, an error flag is set, available to the ground as part of the discrete downlink.

The address, data and control lines of the EEPROMS will be brought out to an external connector to facilitate in circuit programming. The details of this interface are TBS.

2.2.4 SDS (1773 bus) Interface

The 1773 bus interface consists of a transceiver (supplied by GSFC) which converts the fiber optic interface into the 1553 wire interface. The bus is redundant, i.e. two independent transmit and receive lines are available, but only one set is enabled at any point in time. The active transmission channel is selected by the 1773 bus master, i.e. the SDS.

The 1773 bus transceiver (1553 levels) interfaces to the DDC BUS-65612 1553 protocol chip which decodes the Manchester serial, manages synch detection and mode decode, and performs serial-parallel conversion and presents a 16-bit data bus interface to the system manager subsystem. The 1553 subchannel address appears as a dedicated output of the BUS-65612 and is used by hardware to direct command packets and to select the appropriate discrete downlink register

The two RT channels are assigned as follows: EA Subgroup A = RT 1 and EA Subgroup B = RT 2.

Within an EA subgroup, the subaddress channels allocated are:

Component	Bus Function	Subaddress
EA A1	Discrete Cmd	11
EA P1	Discrete Cmd	13
EA P2	Discrete Cmd	14
EA P3	Discrete Cmd	16
EA A1	Serial Command	1
EA P1	Serial Command	2
EA P2	Serial Command	4
EA P3	Serial Command	7
EA A1	Discrete Telemetry	2
EA P1	Discrete Telemetry	3
EA P2	Discrete Telemetry	4
EA P3	Discrete Telemetry	5
PRAM	Serial Cmd (PRAM load)	9
MET	Serial Cmd (MET update)	21
Subgroup Cmd	Discrete Cmd	10
Subgroup Status	Discrete Telemetry	1

Table 3 - 1773 Bus Subchannel Assignments

The SDS-EDS Interface Document maps the GSFC assigned Application IDs and Function Codes to the subchannels identified above.

The 1773 bus timeout value is TBS by the SDS-EDS Interface Document.

2.2.5 Discretes and Power Control

The System Manager board receives +28 V. unregulated power from the PSDU, which can be switched on and off via ground commands to the S/C. A low voltage power supply converts and regulates the raw +28 V. to +5 V. for the entire subgroup as well as one side of the ASM Motor/Control. The +5 is distributed through relays (open when de-energized) to the four EAs in the subgroup and the ASM.

Discretes are a transferred via the 1773 bus and are assigned as follows:

Discrete Telemetry Signals (a 16-bit register/subchannel address is available)

a. Each EA

   1. Three "LED" signals which are driven by the FS to display boot status; a catastrophic condition indicator if the EA is unable to downlink via the 422 interface - 3 bits

b. Power Control/System Manager (Subgroup) Status

   1. ECC Uncorrectable Error detect; signals ground to reload PRAM - 1 bit

   2. Include 1553 bus error flags from BUS-65612 (TBR).

Discrete Command Signals (a 16-bit register/subchannel address is available)

a. Each EA

   1. Boot from Uplink, read by FS in CPU Status Register - 1 bit

   2. EA Reset - 1 bit (Pulse)

   3. EA Power - 1 bit

b. Power Control/Status

   1. ASM motor winding power - 1 bit

   2. SSC HV Control - 3 bits

The Discrete Telemtry is read by the SDS every 8 seconds and is transmitted to the ground as housekeeping. Discrete commands are sent by the ground as needed.

3 Test and Verification

Test and Verification will be addressed in a later revision of this document.

Appendix A - Interface Diagrams/Schematics

Will be included in this Appendix prior to CDR:

a. PSDU Interface Schematics

b. Inter EA Burst Trigger Interface Schematics

c. Hexte - EA Burst Trigger Interface Schematics

d. PCU Science Data Receiver Interface Schematics

e. ASM Science Data Receiver Interface Schematics

f. ASM Command/Housekeeping Interface Schematics

Appendix B - List of TBRs, TBDs and TBSs

Document Reference	Section	Resolution Expected by
Spacecraft generated clocks are all phase aligned to within 100 ns (TBR - must be specified by ICD).	Timekeeping & PSDU Interface	Specification of PSDU, should be included in ICD, expected from NASA by ????
EAID- the actual wire IDs are TBS.	CPU Status Register	Wire list for EDS will specify the EAID jumpers. Expected to be resolved within MIT by the end of 1992.
The bit patterns of UPID are TBS.	System Manager Interface	UPID patterns are based on EAID specification; see above item.
The EA is granted access within 8 (TBR) seconds for a 2 (TBR) second window.	System Manager	Measurement or estimation of the 80C286 boot times along with knowledge of the amount of repository memory size which the EA will require. Should be resolved by the end of 1992.
A counter on the System Manager board increments each time a read occurs, running through the entire ROM (128Kbytes - TBR) followed by PRAM (64Kbytes - TBR).	System Manager	Repository Memory Size, relates to above item. Should be resolved during the FS breadboard stage -- around 3rd quarter 1992.
TBR: Whether the CPU can then hold onto the bus continuously is a design issue which we need to address.	Shared Memory	A design issue which should be resolved by 3Q'92 via discussion with the MIT XTE team.
The PRAM is also protected from SEUs (single event upsets) by either an ECC scrubbing state-machine or by using extra radiation-hardened parts. This is TBD, and depends on the availability of the rad-hard RAMs.	System Manager	Resolved by detailed design and breadboard debug of system manager board, by 3Q'92.
The address, data and control lines of the EEPROMS will be brought out to an external connector to facilitate in circuit programming. The details of this interface are TBS.	System Manager	Resolved by detailed design and breadboard debug of system manager board, by 3Q'92.
Similarly the 1773 bus timeout value is TBS by the SDS-EDS Interface Document	System Manager	Resolved by GSFC generating SDS-EDS ICD, should be immediate.
TBR: ??Include 1553 bus error flags from BUS-65612	System Manager	Should be resolved by 3Q'92.