Eidetic Systems Corporation

A review of spacecraft fault-tolerant computer design concepts / T. Gomi, M. Inwood

Checked 11/83

Government of Canada Gouvernement du Canada

Department of Communications

DOC CONTRACTOR REPORT DOC-CR-SP-82-049

DEPARTMENT OF COMMUNICATIONS - OTTAWA - CANADA

SPACE PROGRAM

TITLE: A Review Of Spacecraft Fault-Tolerant Computer Design Concepts

AUTHOR(S): T. Gomi, M. Inwood

ISSUED BY CONTRACTOR AS REPORT NO: 82-001

PREPARED BY: Eidetic Systems Corporation
P.O. Box 13440
Kanata, Ontario
K2K 1X7

DEPARTMENT OF SUPPLY AND SERVICES CONTRACT NO: 3ER.36100-1-0274
SN: OER81-03138

DOC SCIENTIFIC AUTHORITY: R.A. Millar

CLASSIFICATION: Unclassified

DATE: March 3, 1982

②

/A REVIEW OF SPACECRAFT FAULT-TOLERANT COMPUTER DESIGN CONCEPTS/

Technical Report No. 82-001

①
/T. Gomi /
M. Inwood
Eidetic Systems Corporation

March 3, 1982.

P
91
C655
G64
1982

DD4589357
DL4589391

C O N T E N T S

ACKNOWLEDGEMENTS

This study was performed for the Department of Communications, Communications Research Centre, Shirley Bay, Ottawa, Ontario, Canada under DSS Contract OER 81-03138.

Eidetic Systems Corporation would like to acknowledge the support of Dr. D.A. Rennels of the Jet Propulsion Laboratory of the California Institute of Technology and Dr. George C. Gilley of The Aerospace Corporation, Los Angeles, California, for their comments and cooperation obtained during the course of the study.

The authors are also indebted to R. Millar, of the Communications Research Centre, Ottawa for his guidance and support.

ACRONYMS

AASC	Advanced Autonomous Spacecraft Computer
ABI	Address Bus Interface (FTBBC/MIBB)
Ada	DoD defined Ada programming language
AE	Access Element (FTBBC/MIBB)
ALU	Arithmetic-Logic Unit
AP	Attached Processor (AASC)
ASM	Autonomous Spacecraft Maintenance
BA	Bus Adaptor (FTBBC/BIBB)
BB	Building Block
BC	Bus Controller (FTBBC/BIBB)
BIBB	Bus Interface Building Block (FTBBC)
BIU	Bus Interface Unit (AASC)
BCT	Bus Control Table (FTBBC/BIBB)
BR	Bit Replacement
Core-BB	Core Building-Block (FTBBC)
CP	Command Processor (FTBBC/HLM)
CPC	Check Bits Parity Check
CPDU	Control and Power Distribution Unit (OBDH)
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CTU	Central Terminal Unit (OBDH)
DBI	Data Bus Interface (FTBBC/MIBB)
DB	Data Bus
DE	Double error
DMA	Direct Memory Access
EBI	External Bus Interface (FTBBC/MIBB)
ECS	Error Control Section (FTBBC/MIBB)
ESA	European Space Agency
ESTEC	European Space Technology and Research Centre
FHE	Fault Handler Element (FTBBC/Core-BB)
FH	Fault Handler (FTBBC/BIBB)
FP	Format Processor (FTBBC/HLM)
FS	Fault Sequencer (FTBBC/Core-BB)
FTBBC	Fault-Tolerant Building Block Computer
FTC	Fault-Tolerant Computing
GDP	General Data Processor (AASC)
HLM	High-level Module (FTBBC)
IBI	Internal Bus Interface (FTBBC/BIBB)
IBS	Intercommunications Bus System (FTBBC)
IOBB	Input/Output Building Block (FTBBC)
IOP	Input/Output Processor (AASC)
IP	Interface Processor (AASC)
JPL	Jet Propulsion Laboratory
LAN	Local Area Network
LSI	Large Scale Integration
MCS	Memory Control Section (FTBBC/MIBB)

MCU	Memory Control Unit (AASC)
MDR	Memory Data Register (FTBBC/MIBB)
MFI	Master Fault Indicator (FTBBC/Core-BB)
MIBB	Memory Interface Building Block (FTBBC)
MIPS	Mega Instructions Per Second
NIU	Network Interface Unit (AASC)
NRZ	Non-Return To Zero
OBDH	On Board Data Handling, an ESA standard for computer based on-board housekeeping methodology (ESA)
PCE	Process Check Element (FTBBC/Core-BB)
PLA	Programmed Logic Array
RAM	Random Access Memory
RBI	Remote Bus Interface (ESA/OBDH)
RTI	Real Time Interrupt (FTBBC)
RTU	Remote Terminal Unit (OBDH)
RS	Recovery Sequencer (FTBBC/Core-BB)
SCCM	Self-Checking Computer Module (FTBBC)
SE	Single Error
SEC/DED	Single Error Correction/Double Error Detection
SGC	Syndrome Generator Check (FTBBC/MIBB)
SSI	Small Scale Integration
TM	Terminal Module (FTBBC)
UDS	Unified Data System - a JPL designation for a standard-ization of fault-tolerant on-board computer system.
VLSI	Very Large Scale Integration

1. SUMMARY

The Fault-Tolerant Building Block Computer (FTBBC), designed by the Jet Propulsion Laboratory (JPL) for satellite on-board use, incorporates extensive fault-tolerant mechanisms. An evaluation of these are made in comparison with two systems which have similar objectives; a European system developed for unmanned applications, and the preliminary design for implementation of the latest, or imminently available technology. These are the European Space Agency's On-Board Data Handling (ESA/OBDH) system and the Advanced Autonomous Spacecraft Computer (AASC). The capabilities and limitations of the FTBBC are pinpointed by reference to this comparison and to recent developments in associated technologies. The conclusions drawn from this comparison point to the direction in which future advances should lead.

Current research is investigated and its relevance to space applications is discussed. Contacts made within the space community are reported and information gained through these channels is included.

A set of design rules which has evolved during the study is appended.

2. INTRODUCTION

The Jet Propulsion Laboratory's (JPL) Fault-Tolerant Building-Block Computer (FTBBC) was initiated in the early 1970s as an exercise in designing an on-board computer which would take advantage of current developments in the fields of fault-tolerance and technology. In particular, the FTBBC was one of the first computers to incorporate extensive fault-handling functions in its hardware design. As this study reveals, the mechanisms to detect hardware faults, and confine them are extensive, whilst attempts are also made to analyze and recover from them. The flexibility of the Central Processing Unit (CPU) and the memory selection afforded by the Self-Checking Computer Module (SCCM) arrangement, for example, are essential features of any fault-tolerant computer design. Furthermore, the Unified Data System/FTBBC is the first major computer system development based on microcomputer technology that addresses issues related to fully distributed systems, either for space use or otherwise. A version of the FTBBC will be flown on Galileo (Ref.1).7

A second computer of the building-block type, the European Space Agency's On-Board Data Handling (ESA/OBDH) system, which was well defined by 1973, has been studied. This was also intended as a reusable, distributed system for unmanned satellites, and contains some redundancy and building-block features. It was the result of combined development by European countries. The Europeans made no effort to use available components and designed the system in the custom-built style traditional to the space industry. High costs and long development time were mitigated by distributing the responsibility for the various modules amongst member countries.

Development of the FTBBC computer has now been in progress for a decade. During this time research has gone ahead in fields which may well have a considerable impact on fault-tolerant concepts and methods of implementing them. Aspects which have been problematical in the past are now being approached, and solutions are being presented to concerns which were not clearly understood at the time of the FTBBC. This report is an attempt to understand and high-light these concepts, and study their relevance to the present stage of the FTBBC.

A further outcome of this study has been the formulation of the Fault-Tolerant Computing Design Rules. This is a set of criteria which, it is felt, should be applied to the design of on-board spacecraft computers. These rules have been used as a point of reference in the comparison contained herein.

An understanding of the present status of the FTBBC and current research has been gained by conducting personal and telephone interviews with Dr. D.A. Rennels and other JPL personnel and also with other active participants in the area of on-board computing. Reference has been made to the open literature available from JPL, ESA and other international institutions.

3. JPL'S UNIFIED DATA SYSTEM/ FAULT-TOLERANT BUILDING BLOCK COMPUTER

3.0 General

The FTBBC project grew out of an earlier study on the Unified Data System (UDS) at JPL (Ref.4) in which Dr. David Rennels, the chief architect of the present system, took a leading part. Whereas the UDS is a system level design, the FTBBC is a hardware implementation of a Self-Checking Computer Module, an essential component of the UDS system. Since the FTBBC is best viewed in this context, a brief outline of the UDS is given first.

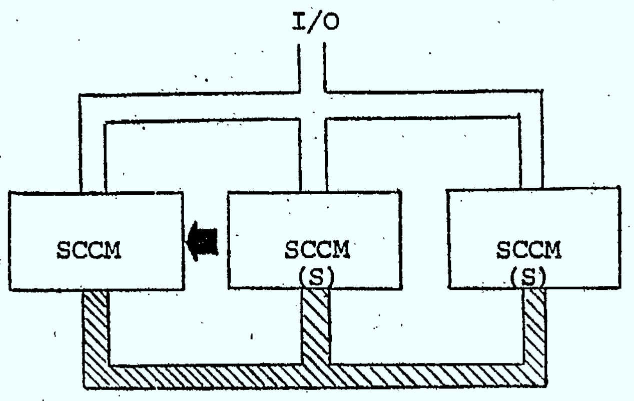
3.1 Features of the Unified Data System (UDS)

The UDS was developed as a distributed computer network architecture for spacecraft on-board processing. It consists of a set of SCCMs connected by a redundant set of intercommunication buses. Microcomputers, utilized as High Level Modules (HLM) or Terminal Modules (TM), may be combined in a variety of different configurations, as shown in Figure 1.

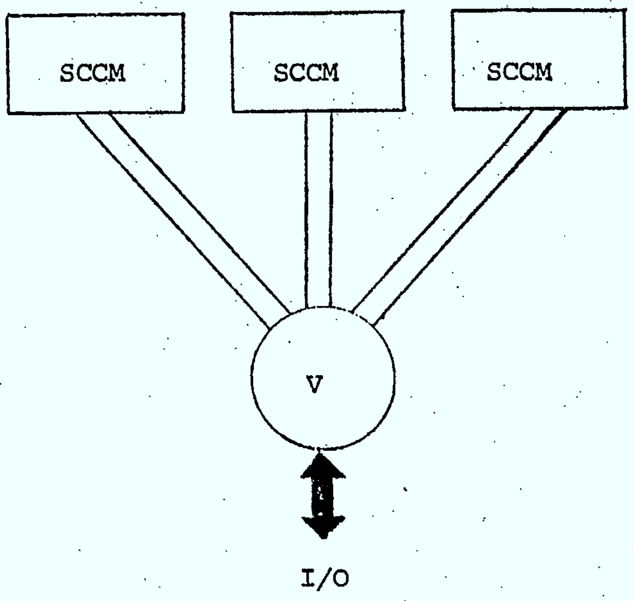
TMs are dedicated to a particular spacecraft subsystem and are responsible for control and data collection within it. A TM cannot initiate bus communications but can be commanded to fetch or deposit data into its memory. In addition, it actively supports DMA transactions and it can be accessed through several buses simultaneously.

HLMs are responsible for coordinating processing within the remote TMs, for controlling the Intercommunication Bus System (IBS), and for high level processing such as data compression and decision making. HLMs contain the same internal components as the TMs but with the addition of a Bus Interface Building Block (BIBB) programmed as a Bus Controller (BC). Unlike the TM, an HLM has no I/O circuitry apart from the IBS and only communicates with other computer modules.

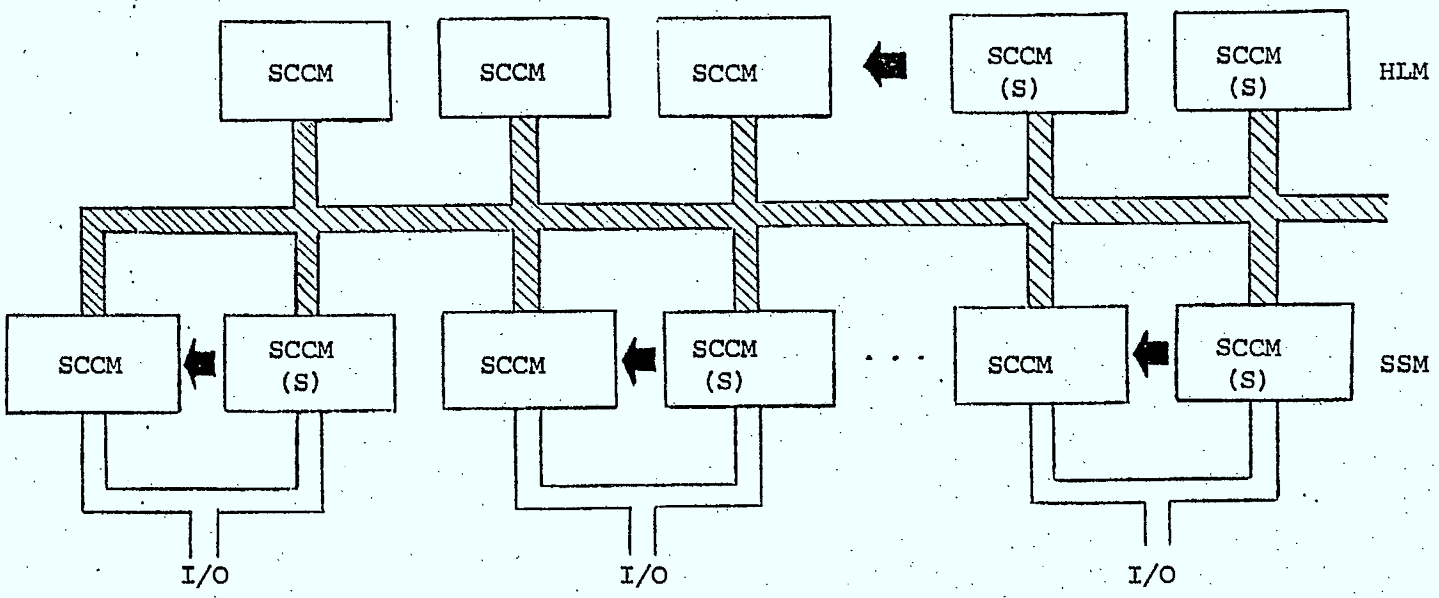
Each serial bus is connected to a Bus Adaptor (BA) in each of its associated SCCMs (HLM or TM). It is also assigned a primary BC in its controlling HLM. Control can pass to another HLM if the bus is not powered or the processor commands release to a lower priority BC for a specific time interval. Priority is established on a fixed basis and implemented through a daisy-chain structure for each bus which is shown in Figure 2. This, theoretically, enables spare modules to take over from a failed HLM or, in the event of bus failure, the BCs may share one bus. The buses are physically independent of each other and



a) Standby Redundant



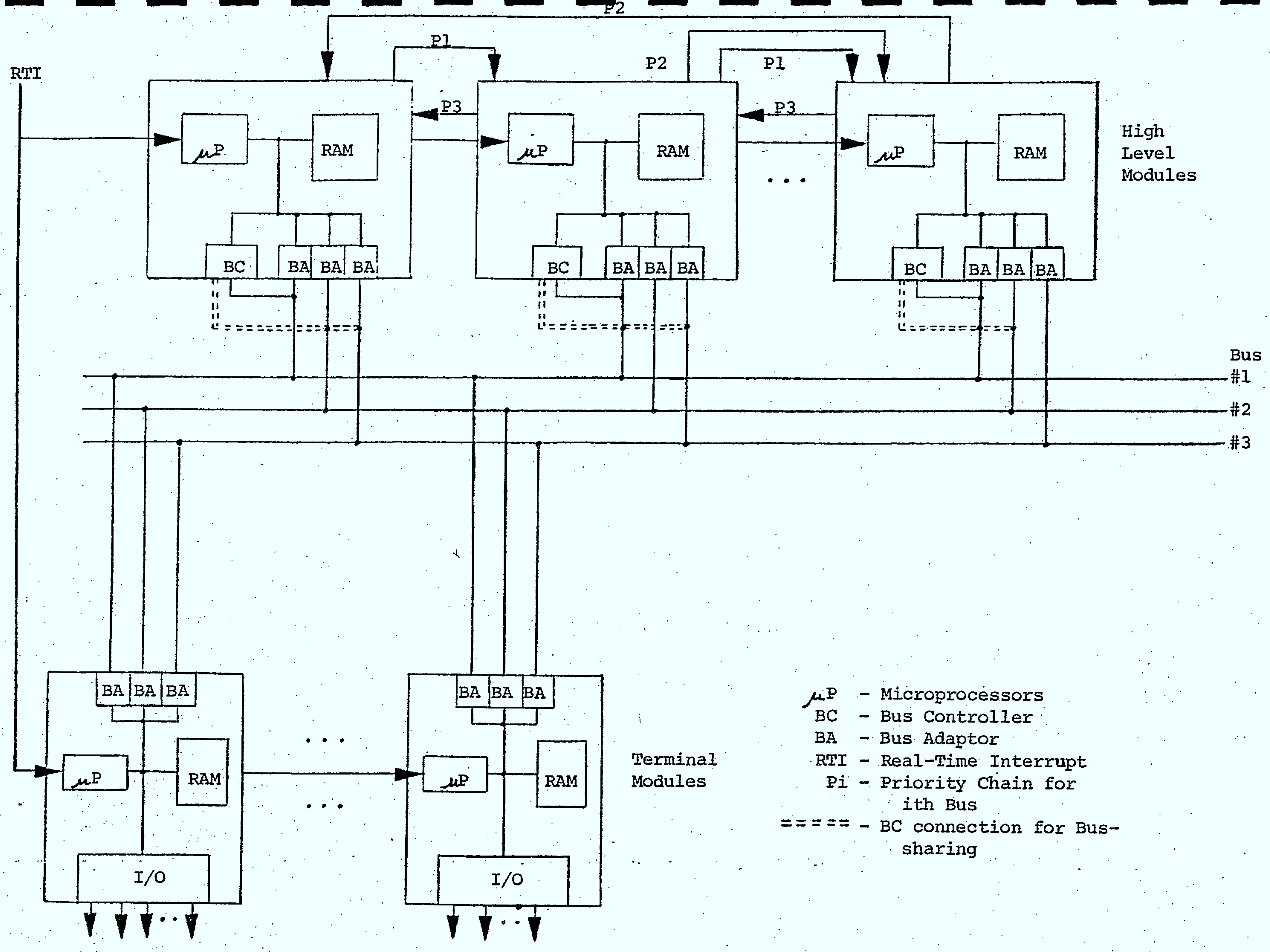
b) Voted Configuration



c) Distributed Computer Network

▨	1ll Intercommunications Bus
HLM	- High Level Module
SSM	- SCCMs in Subsystems
(S)	- Spare Module

Fig.1 Fault-Tolerant SCCM Configurations



μP	-	Microprocessors
BC	-	Bus Controller
BA	-	Bus Adaptor
RTI	-	Real-Time Interrupt
Pi	-	Priority Chain for ith Bus
=====	-	BC connection for Bus-sharing

Fig.2 RTI and Daisy-Chain Priority in SCCM Distributed Network.

there is no common clock. Each SCCM uses its own internal clock and the buses use the clock of whichever module is transmitting.

The number of buses is dependant on mission and redundancy requirements. In the event of failure a mission may be reconfigured throughout - even to the extent of a single remaining bus supporting essential functions.

One HLM is designated as the system executive and issues a common system-wide clock, the Real Time Interrupt, (RTI) of 2.5 ms. All modules receive and compare the RTI with their own internal clocks. On failure of an RTI generator, the computers will automatically switch to a backup. If an individual clock fails then damage will be confined to one module. Figure 2 shows the RTI line between modules.

Subordinate HLMs or TMs are commanded to start specific programs which the timing information from the executive HLM will synchronize with the rest of the system. Most programs are self-synchronizing and timing within them is tightly regulated, the time count of each event being precisely specified.

Each HLM/TM contains an identical local executive program for communication with user software. Software is run in a foreground/background partition with well-defined segmented programs running concurrently in the foreground, whilst more complex and lengthy programs are assigned to the background mode. The local executive is responsible for activating and de-activating foreground/background programs using a scheduling table. A UDS Program Design Language enables a program to communicate with the executive and request modifications to start, stop, suspend or delete itself.(Ref.5) On completing the execution of available foreground programs, control is returned to the background program for the remainder of the time slot.

I/O granularity is defined by sampling inputs and holding them for a fixed period. Several segments of concurrent foreground programs may be executing during a time period and their outputs collected and held for simultaneous execution at the end of the time period. Programs can be removed, added and re-ordered within a time interval without impacting others. This approach is intended to allow a great deal of visibility into the system and to simplify simulation, debugging and modification of software.

Fault-tolerant aspects of the software are mainly confined to increased reliability and testing. Timing has been simplified at the intercommunication bus interface to obviate incompatibilities between user-supplied subsystems. Bus access is restricted to localize the effects of software failure. Reasonableness checks at the HLM level can verify the proper functioning of low level dedicated software. The HLM has little actual protection against software faults. The UDS local executive has a mechanism for software fault detection which checks proper return of control after execution of a foreground program. In the event of a problem, a software fault is signalled and a user-supplied recovery routine is executed.

Fault recovery requirements for the UDS include a dedicated "hot" spare for the system executive HLM. Cross-checking by the two modules at each RTI will reveal failure of either one and trigger recovery of the extant module. It should also have the ability to activate a further "hot" spare if necessary. A primary HLM generates output and gathers data for the spare and itself. Additional "hot" spares may be utilized under critical conditions. Other HLMs are backed-up by non-dedicated spares. Controlling HLMs are responsible for polling TMs, recognizing failures, internal reconfiguration where possible or continuing operation in a degraded mode. Bus system failures are determined by rerouting suspect messages through a different HLM-bus configuration. Since TMs are dedicated, their spares must be also. Such spare TMs are assumed to be physically built into each subsystem. The fault-tolerant responsibility of a TM is detection within its associated subsystem and repairs where possible. Where repair is not possible the TM records error information, institutes a "safe" disabled state and notifies its HLM.

3.2 Features of the FTBBC

3.2.1 The Self-Checking Computer Module (SCCM)

The FTBBC itself arose from the desire to combine off-the-shelf components into an SCCM (Ref.3). The appeal of this approach lies in the following features:

- proven reliability of components
- shorter development time
- flexibility of configuration for different applications
- easy upgrading of facilities and standards.

The building blocks, four in number, are intended as VLSI fault-tolerant interfaces to the standard components. They consist of the Memory Interface Building Block (MIBB), Programmable Bus Interface Building Block (BIBB), Core Building Block (Core-BB), and the I/O Building Block (IOBB). Figure 3 shows these building blocks as they are combined to form a self-checking computer module (SCCM). The SCCM can be designated either as an HLM or TM. The building blocks should be able

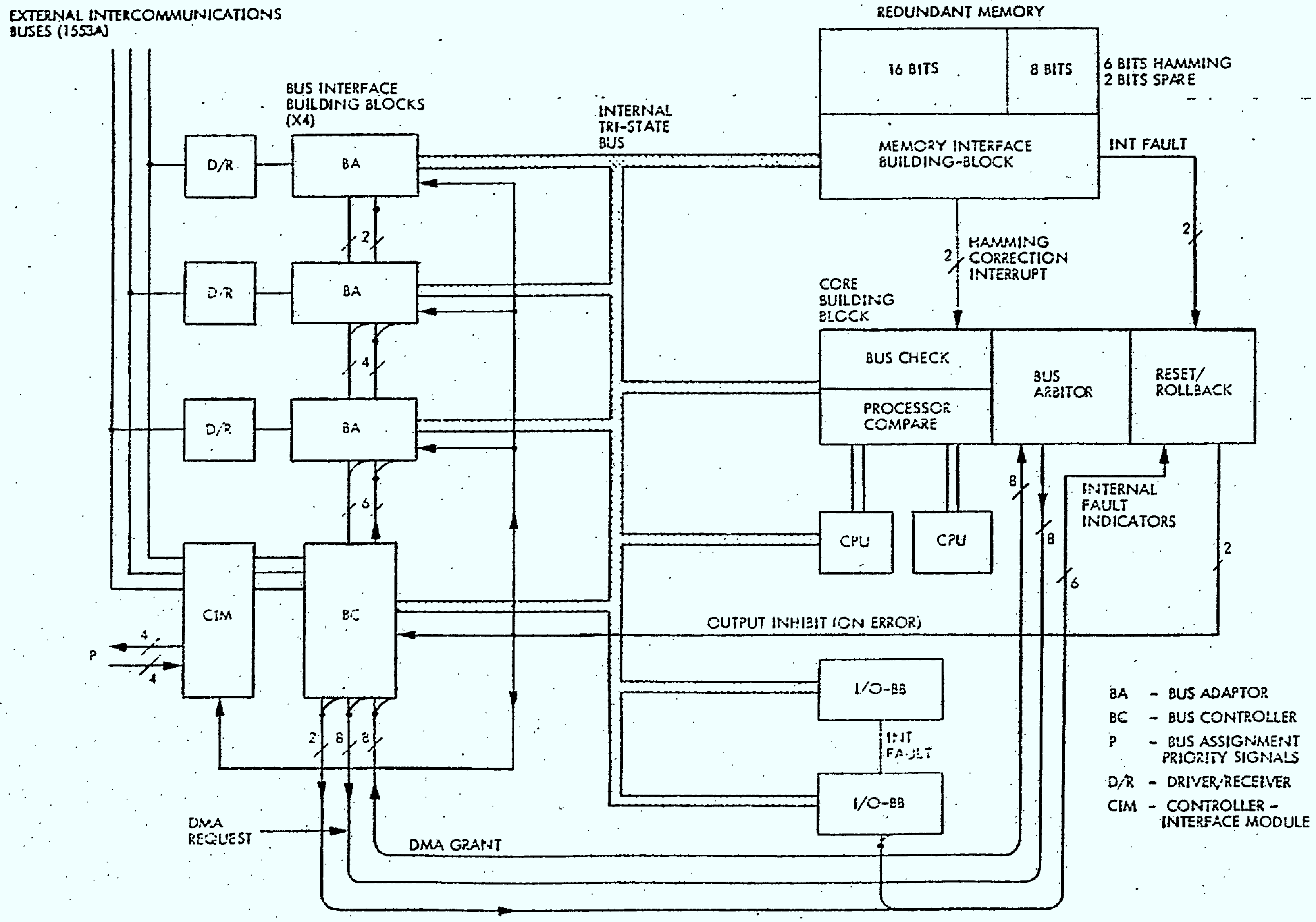


Fig.3 The Self-Checking Computer Module (SCCM)

to detect faults internally and in their associated circuitry and give a fault indication to the Core Building Block. The SCCM should, additionally, be able to signal its own malfunctions to other SCCMs and disable its outputs when necessary.

3.2.2. Memory Interface Building Block (MIBB)

The memory module consists of two parts; a set of RAM chips and the MIBB. Memory consists of 24 bits, separately packaged in order to contain circuit failure damage. Sixteen bits are used for storage, six for Single-error-correcting/ Double-error-detecting (SEC/DED) Hamming code and two are redundant bits in case of failure.

The BB itself is responsible for interfacing memory to the internal bus. It is capable of implementing fault detection and correction within memory using Hamming code and within its own circuits. Information transfer on the internal bus is safeguarded by parity generation and checking. Fault conditions are signalled to the Core-BB and, upon system command, up to two faulty bit planes in memory can be reconfigured.

The memory module uses "memory-mapped I/O" which avoids processor specific I/O operations and also allows access to building blocks by internal SCCM software and Direct Memory Access (DMA) by other SCCMs via the internal bus. This gives an external SCCM the ability to load and read memory via the bus, sample error status information, command internal reconfiguration, and even remotely control I/O in a faulty SCCM. Optional configurations available to the user include variations in memory size (8K, 16K, 32K) and a choice of Ham or non-Ham mode. Non-Ham mode, i.e. without Hamming code, is provided for applications requiring very low power, weight, and volume. In this mode, error detection, correction, and bit-plane replacement are performed under system control, but two parity bits for error detection are retained in the MIBB.

All circuits within the MIBB are either self-testing, fault-secure, or duplicated so that no single circuit failure will produce an undetected output error.

The MIBB consists of four sub-elements: Address Bus Interface (ABI), Error Control Section (ECS), Data Bus-Storage Array Interface (DBI), and Memory Control Section (MCS).

The ABI provides the address parity checking and decoding required to select a memory module. An incoming address is stored in the Memory Address Register (MAR) and validated before a read/write operation is performed. If no errors are detected the low-order 12 bits are sent to the Storage Array

Block for on-chip decoding. The high-order bits are detected and used for memory-mapping. An alternative is to associate a separate decoder with each bit-plane enabling the single error correction/double error detection SEC/DED data word error code to be used in the address decoding. The address which is currently being referenced is stored for future diagnosis in case a fault recurs.

The ECS is concerned with error detection, correction, and analysis within the MIBB. It generates and checks Hamming code check bits and syndromes, and byte parity for the internal bus. ECS circuits are self-testing. The Error Status Register/Memory Error Interrupts issue two fault-detection signals:

- a code-correction indicator, which is sent to the duplex processors as an interrupt indicating that a single memory bit is being corrected by Hamming code. This facilitates the processor's decision on bit replacement.
- internal fault indicator for faults which cannot be corrected within the memory system i.e., when:
    - a fault is detected within the MIBB itself
    - improperly coded information is received over the internal bus
    - a data error occurs within memory that cannot be corrected by Hamming code.

This internal fault indicator is sent to the Core-BB which may, in the case of a transient error, be able to resume correct operations using a rollback or reset/restart sequence.

All circuits are self-testing.

The DBI interfaces the Memory Data Register (MDR) with the internal bus. The MDR consists of two Data Bit Modules, a Check Bit Module for storing Hamming codes, a Replacement Control Section containing the spare bit planes, and interfacing networks. In the event of a faulty bit plane, error decoding is performed on chip but the replacement decision is made by the system.

The MCS generates control signals to implement operation and command algorithms. Read/write instructions with low-order addresses are treated as normal memory operations. High-order addresses are reserved for "memory-mapped I/O". Certain of these out-of-range commands are recognized as pertaining to the MIBB.

3.2.3. Core Building Block (Core-BB)

The Core-BB is responsible for:

- synchronizing and comparing duplex CPU outputs
- fault-handling through-out the SCCM.
- internal bus arbitration

One processor is designated as primary and the other as check processor. Processor faults are detected by the Processor Check Element (PCE) by running two synchronized units using the same data and executing the same programs in lock-step. Parity coding of incoming data is checked. Outputs to the internal bus are compared and outputs to address and data buses are parity encoded. The PCE contains self-checking parity checkers, a duplex command decoder and morphic (one-out-of-two) reduction trees. It also samples various fault indicators to provide information for external computer modules.

The Bus Arbitration Element (BAE) accepts bus request signals from the various DMA controllers throughout the SCCM and obtains release of the bus by the CPUs. Access is granted to requesting BB's on the basis of a fixed hardware priority. The BAE signals are duplicated and morphically compared.

Within the Fault Handler Element (FAE), the Fault Synchronizer (FS) accepts fault indicators from throughout the SCCM and morphically reduces them to produce a single clock synchronized morphic fault indicator, the Master Fault Indicator (MFI).

This MFI is sent to the Recovery Segment which disables SCCM output and resets the CPUs. A program rollback can optionally be caused and computation reinitialized. If no additional faults are detected the processors can reenable the module.

The FS and Recovery Sequencer (RS) are duplicated and paired and either pair can disable SCCM output. RS outputs are also compared and disagreements signalled to both Fault Sequencers.

There is a small circuit provided for manual or external module control. This facilitates program restart either by front panel switches or under program control via out-of-range commands.

3.2.4. Bus Interface Building Block (BIBB)

The BIBB enables information to be transferred between SCCMs via an intercommmunications bus system. It is programmable as either a Bus Controller, one per HLM, or a Bus Adaptor, one per bus in each HLM or TM. They utilize the MIL-STD-1553A bus format, with enhanced capabilities.

The BC can communicate with several redundant buses. A BA is connected to only one bus and serves as a remote terminal. The Controller and Adaptors operate in a relatively autonomous fashion. Their enhanced capability enables them to move data directly between SCCM memories attached to a given data bus with a minimum of software support.

When an HLM wishes to initiate a data transfer between modules on the bus it alerts the BC. This reads a Control Table in its host's memory, specifying the source and destination of information along with the length of transmission. It then specifies one BA as a data source and one or more remote BAs as data acceptors and names the data to be moved. As the 1553 format does not provide directly for multiple acceptors, additional modules must be commanded to "listen in" on a 1553 terminal-to-terminal transmission. The specified data is then extracted by the BA from its host's memory using cycle-stealing and is placed on the bus. It is removed from the bus simultaneously by the acceptor BA and loaded into its SCCM host's memory. The BC monitors this process and signals completion to the host. As many as three buses may communicate with an SCCM without conflict. A BA acts only in response to a Bus Controller, either remotely or within its own SCCM. A BC or BA can recognize several out-of-range addresses relevant to their own functions.

The internal structure of a BIBB is the same for a BC or BA and contains the following:

- the Mill; a small processor with ROM, RAM, internal registers, and an Arithmetic and Logic Unit (ALU). It has responsibility for generating addresses for DMA, word counting, testing control words, and buffering data in transit between the external bus and the SCCM.

- the External Bus Interface (EBI) which interfaces the Mill and the external bus. It encodes parallel commands and data words from the Mill for serial transmission over the bus. It also performs serial to parallel conversion on incoming Manchester encoded data words, making them available to the Mill, and signals their arrival to the Controller. Improperly coded commands and signal/parity errors are indicated to the Fault Handler (FH).

- a DMA interface between the Mill and SCCM provided by the Internal Bus Interface (IBI). Registers in the IBI are used to buffer incoming and outgoing data and DMA requests and acknowledge control logic. The IBI also contains a command decoder which is used to recognize and decode memory-mapped commands from the host SCCM to the BIBB.

- the Controller generates control signals for the other BIBB subelements from internal or external SCCM commands and conditions sampled within the BIBB. It is programmed using a ROM and a Programmed Logic Array (PLA).

The BIBB internal circuits use either duplication and self-checking comparison or error detection codes. The FH combines fault signals from these circuits into a single morphic Internal Fault Indicator. The FH terminates any ongoing transmission on detecting an internal fault. BC faults are signalled to the Core-BB which disables the host SCCM in order to prevent damaged information from being propagated throughout the system. A faulty BA merely disables its own communicating ability. As an SCCM contains several redundant BAs, messages can be rerouted via a different BA.

3.2.5 I/O Building Block (IOBB)

The design for the IOBB has not been completed. However, certain typical requirements and functions have been specified.

In order to retain consistency with FTBBC modules all building blocks must provide memory-mapped I/O. Fault tolerant requirements are that:

- the IOBB must check the coding of incoming addresses and data, and utilize duplication or coding checks to verify proper functioning of its internal logic.
- either data errors or internal faults must generate an error indicator to the Core-BB. Error indicators should be morphic to prevent a single point of failure.
- incoming data must be encoded for presentation to the host computer's bus.

One of the more important I/O functions which should be provided by IOBB modules is synchronization of inputs and outputs with the RTI. This:

- ensures synchronism in voting configurations
- decouples I/O timing from detailed instruction timing in the TM

- enables software to be changed without altering the I/O timing of unmodified programs
- prevents I/O timing being changed as a result of the use of stolen memory cycles during DMA activity on the IBS which may cause slight variations of processor speed.
- is expected to simplify verification and validation through restricted interaction with the host coupled with synchronous software.

The circuitry for I/O functions is not expected to be complex and fault-detection implementation should be straightforward. Parity checking can be utilized where the data structure is preserved and control functions can be duplicated with morphic comparison.

In order to achieve redundancy in TMs, two or more modules can be cross-strapped, i.e. their inputs and outputs hooked together. Only one module is powered; the others are cold standbys. In this usage short protection should be provided at all output connections to avoid deactivation of all the spares. IOBBs may be used redundantly within an SCCM.

3.3 The FTBBC Features Tree

This tree, which is shown in Table 1, is an aid to understanding the extent and placement of the aspects of fault-tolerance which are embodied in the FTBBC design. The tree is a means of representing a hierarchical structure and has been applied, in this case, to the fault-tolerance features distributed within the FTBBC. The level of each feature in the structure is indicated by its level of indentation within the tree. A table of functions has been included to show the purpose of each feature of the module.

The features have been classified into four areas of applicability: detection, containment, analysis, and recovery from faults. The types of fault-tolerant protection found within the FTBBC are enumerated and their location indicated by the right-most column.

A number in parentheses against a feature indicates the fixed number of units contained within the module. Flexibility in the quantity of units, as used for redundant spares for example, is indicated by "(n)".

Table 1: FTBBC FEATURE TREE

FEATURES	FUNCTIONS	F-T TYPES
SCCM - Self Checking Computer Module (n)	- configured as HLM or TM	C,R
(HLM - High-level Module (n)	- designated executive or lower HLM,	
or		
(TM - Terminal Module (n)	- dedicated to a specific subsystem	
MIBB - Memory Interface Building Block (1) (Ham mode or non-Ham mode)	- interfaces redundant memory chips to internal bus	
ABI - Address Bus Interface	- selects appropriate memory module	
MAR - Memory Address Register	- stores 16-bit memory address	
APC - Address Parity Checker	- validates address	1bm,6,12m,D
SNC - Soft Name Checker	- decodes high-order address bits	6,11m,12m,D,
EWAR - Error Word Address Register	- stores address currently referenced	10,R
ORC - Out-of-Range Command Detector	- detects out-of-range commands to the MIBB	11,D
ECS - Error Control Section	- generates Hamming code check bits, analyzes errors in bus and interface circuits	6,D
SGC - Syndrome Generators/ Checkers	- generates Hamming code check bits and syndromes (in Ham mode only)	2m,6,D
DPCG - Data Parity Checker/ Generator (Internal bus)	- byte-parity generation and checking	1am,1bm,6,D
SDA - Single and Double Error Analyzer	- analyzes inputs from error coding, parity checking & all self-testing circuits	1a,1b,5m,6,D,R,A
ESR/MEI - Error Status Register/ Memory Error Interrupts	- records error conditions from SDA, gives indication of correction, prevents write operation, checks SDA circuits	5m,6,10,R,C,
DBI - Data Bus-Storage Array Interface		
DBM - Data Bit Modules (2)	- 8-bit data storage	
CBM - Check Bit Modules	- stores Hamming code (Ham mode)	2,10D,R
Bit-Plane Interface	- interfaces 2 parity bits in non-Ham mode	
Position Decoder	- on-chip decoding	R
Spare Plane Interface	- interfaces MDR with spare bit plane	R
Bit Interface Module	- interfaces MDR with regular bit plane	R
RCS - Replacement Control Section		
Decoders (2)	- decodes replacement registers	R
Replacement Registers (2)	- used to replace faulty bits in DBM/CBM	4,R
MCS - Memory Control Section	- provides control signals for implementation of operation and	

Table 1: FTBBC FEATURE TREE

FEATURES	FUNCTIONS	F-T TYPES
	command algorithms	
CI - Control Interface	- SCCM/MIBB handshaking circuits	9,D
CPG - Clock Generator	- system synchronization	8,14a,14b
KG - Condition Generators (2)		11m,D
SS - State Sequencers (2)	- implements steps of operation/ control algorithms	8,11,D
CSG - Control Signal Generators (2)	- generate signals for register & selection networks	11m,D
CSR - Control Signal Comparator	- reduction of control signals	12m,5,D,
Core-BB - Core Building Block	- detects CPU/bus faults, collects faults from other BB's, self-disabling	11,D
CPU (i) Master (ii) Check	- operates synchronously with redundant processor	11,D
PCE - Processor Check Element	- compares output of processors, encodes/checks internal bus parity, decodes commands on internal bus	
MCMP - Morphic Comparators (i)	- compares address output of CPU's	12m,6,D,
(ii)	- compares CPU output with data bus	12m,6,D,
Isolator	- allows input data to be passed to Check CPU	15,D
MPC - Morphic Parity Check/ Generators (i) & (ii)	- check/generate parity (i) on address bus, (ii) on data bus	16m,6,D
CMD - Command Decoders (2)	- generates 2 out-of-range addresses	11m,D
Status Registers (2)	- samples fault indicators for access by external SCCM	5,10,D,A,R
BAE - Bus Arbitration Element	- arbitrates internal DMA requests from other BB circuits	5,12,D
Priority Resolvers (i) 'true' (ii) 'false'	- synchronously compared to detect faults, arbitrate bus requests	11,12m,D
FHE - Fault Handler Element	- responsible for overall fault detection in SCCM & limited recovery action	
Fault Synchronizers (2)	- examines signals from within SCCM & sends MFI to RS	8,11m 5,D,R
Control Signal Generator	- generation of internal control signals	
RS - Recovery Sequencers (2)	- disables outputs from SCCM/ resets CPU's	5,8,11,12m,13 D,C,R
Manual and External Module Control	- clears fault latches/initializes program restart	8,14a,14b,R
BIBB - Bus Interface Building Block	- programmed as either a or b	
a) Bus Controller or	- transfers information between SCCM's,	
b) Bus Adaptors (3)	- moves data into/out of SCCM memory	

Table I: FTBBC FEATURE TREE

FEATURES	FUNCTIONS	F-T TYPES
The Mill	- responsible for BIBB processing	1a,1b,1c,1d,m,5,12m
Memory	requirements; sampling incoming	1c,8,D,A,R
ALU a)	data, DMA address generation,	a) 1a
b)	word counts, testing control words	b) 5m,11,12m
Internal Registers (2)		
EBI - External Bus Interface A)	- interface between Mill/external bus	A) 11,D,
B)		B) 12m,D
MNT - Manchester/NRZ Translator	- translates incoming code from Manchester-NRZ	4,8,D
BAC - Buffer and Logic Control		
BAC Control		
Command Decoder	- detects improperly coded commands	5,D
State Control	- controls input/output mode	
M Counter	- counts incoming/outgoing data words	8,D
Controller Alert Logic	- alerts controller of throughput of data words, synchronizes A & B copies	
BAC Data Paths		
XFR - Transfer Register	- serial/parallel conversion	
CDR - Command Data Register	- single buffer for incoming/ outgoing words	
Manchester Encoder	- encodes outgoing data	3
Parity check/generate circuit	- checks incoming codes, encodes outgoing words	1a,1b,D
BAC Fault Detection Logic	- sends Master Fault Indicator to BIBB FH	5,12,D
IBI - Internal Bus Interface	- DMA interface between the Mill/ SCCM memory	1c,11,R
DC Reg - Direct Command Register	- contains fault handling circuits	2,D
Address Register		1b
Data Registers (2)	- one each, incoming/outgoing	1b
DMA Controller (2)	- request and acknowledge control logic	1bm,11,R
Command Decoder (2)	- detects memory-mapped commands	1b,11,12,D
Bus Assignment Latch	- stores number of external bus being requested	1b,D
Fault Handling Circuits	- generates single fault indicator	12m,5m,R
CONT - the Controller	- generates control signals for BIBB sub-elements	
CS - Control Sequencer a) & b)	- samples circuit conditions within BIBB outputs: a) data b) parity bits	a) 12m,D b) 1a,D
PLA - Programmed Logic Array	-	
MLC - Microprogram Location Counter	- generates addresses for CROM	8,D
Status Register	- contains status word for output	1a,5,7,D
ID compare	- compares hard/soft names with commands	12

Table 1: FTBBC FEATURE TREE

FEATURES	FUNCTIONS	F-T TYPES
Loop counter	- underflow signal sent to condition multiplexor	8,D 8,D
TOC1 - time out counter	- verifies completion of DMA	8,D
TOC2 - time out counter	- detects non-arrival of incoming/outgoing word	8,D
CROM - Control ROM	- receives addresses from CS, maps these into control signals to operate BIBB	1a,1b,2,D
FH - Fault Handler	- terminates ongoing transmission when internal BIBB fault detected stops clock to BIBB, resets pulse	5,11,12m,14b D,C,R
IBS - Intercommunication Bus System	- redundant bus system	4,R
IOBB - I/O Building Block (2)	- proposed attributes:	3,16,1b,5m,11

KEY to types of Fault-Tolerance:

1. a) Parity encoding (double)
   b) Parity checking "
   c) Parity encoding (single)
   d) Parity checking "
2. SEC/DED (Hamming) code
3. Manchester/NRZ coding
4. Redundant spares
5. Fault indications
6. Self-checking logic circuits
7. Status codes
8. Timers, counters
9. Handshaking
10. Error information storage for diagnostics
11. Duplication
12. Comparison/reduction of results
13. Restart, retry
14. Switched circuit a) manual
    b) automatic
15. Special circuits to monitor critical elements
16. Cross-strapping

m Morphic attributes

D - Detection
C - Containment
A - Analysis
R - Recovery

4. COMPARISONS WITH OTHER FAULT-TOLERANT ON-BOARD COMPUTERS

4.0 General

Comparisons were made among three significant fault-tolerant on-board computing facilities: the UDS/FTBBC, European Space Agency's On-Board Data Handling (OBDH) System (Refs.7-15), and the proposed Advanced Autonomous Spacecraft Computer (AASC) using Intel Corporation's new iAPX 432 computer and its advanced fault-tolerance extensions (the detail of iAPX 432 fault-tolerance is still restricted information).

The purpose of the comparisons is to place the FTBBC in perspective and to determine its strengths and weaknesses as a system. Every effort was taken to make the comparisons as impartial as possible. However, as is always the case with comparisons of objects from different origins, there may be a level of disposition unknown to the authors, in the form of a biased selection of features or topics against which comparisons were made, or the subjective evaluation of scoring by particular system(s) in such comparisons.

A brief outline of the OBDH and AASC follows.

4.1 Features of the OBDH

The OBDH is a modular, distributed system which comprises a Central Unit (CTU), a Data Bus, Remote Terminal Units (maximum 31), and a Command and Power Distribution Unit (CPDU) as shown in Figure 4.

The CTU is responsible for timing, internal bus traffic, time-shared access to the external bus, formatting data for telemetry purposes, command and housekeeping handling. Optionally, it can provide user facilities such as data collection from sub-systems, dedicated processing power with access to all OBDH data channels, and user specified pulses. (Ref.16)

The RTUs distribute operational commands and data to other subsystems and acquire data from these subsystems. There are three versions available, for single user, multiple users and multiple users with limited I/O. (Ref.12)

The Data Bus sends commands, data and an operating clock in a continuous, self-clocking, Litton-encoded form to the RTUs via the Interrogation Bus. The Response Bus is used for return messages either to the CTU or another RTU at the CTU's command. Messages are allocated according to predetermined time-slots under CTU control. In-flight programming can be performed via the Programming Bus which is controlled by the Smart-Controller

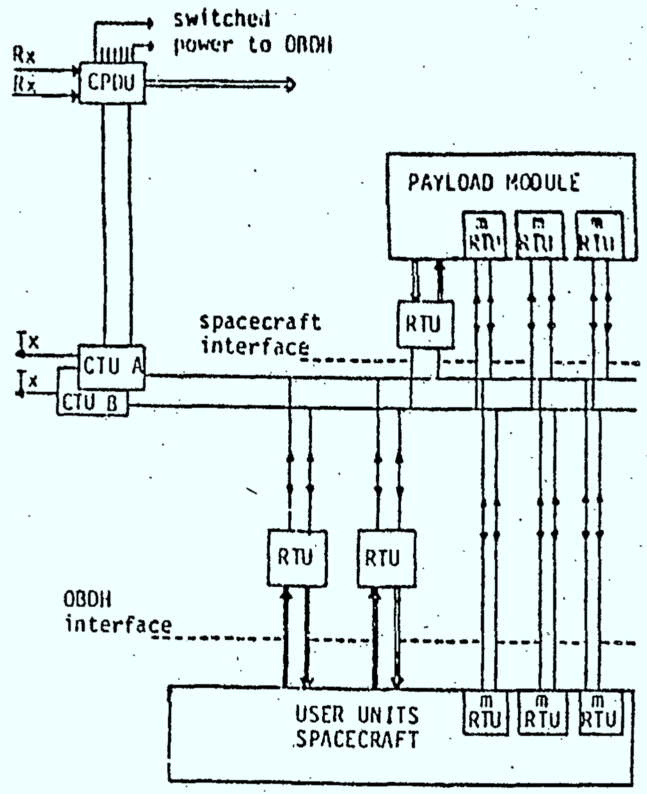


Fig.4 ESA-OBDH System

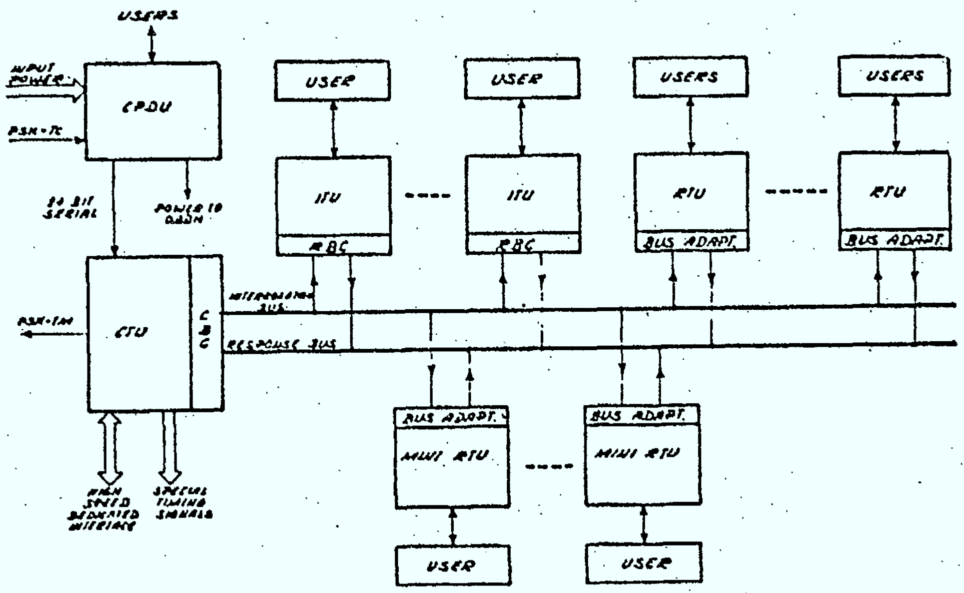


Fig.5 OBDH bus, CPDU, CTU, and RTU

within the CTU.

The CPDU has the following functions; demodulation and bit synchronization of up-link telecommand signals, frame verification, distribution of priority commands and serial load commands, and configuration control of the OBDH subsystem.

CTUs, RTUs and the Data Bus may be duplicated as required to provide a redundant configuration as shown in Figure 5. There is internal redundancy within the RTU and the CPDU. There is no internal redundancy within the CTU, which must rely on a standby spare.(Refs.8,12,16) Software fault detection features include monitoring terminal status, watchdog timers, and detection of abnormal conditions.

4.2 Features of the AASC

The AASC is an autonomous computer system consisting of clusters of highly fault-tolerant computer complexes distributed along inter-cluster buses. A cluster interfaces the inter-cluster buses at one end and, in general, i/o subsystem(s) at the other, as shown in Figure 6. The inter-cluster bus is very much like a local area network bus.

The hardware redundancy is provided at several levels within the system hierarchy: every element of the system may exist in multiple copies to provide non-dedicated redundancy at hardware level. These include the following: cluster, inter-cluster bus, inter-cluster bus to cluster interface, intra-cluster processor, intra-cluster bus, intra-cluster memory control unit, and local and global memory modules.

The inter-cluster bus is not specified in detail as its design is intended to be independent of available technology. However, a fully distributed architecture is assumed, i.e. no dedicated controllers on the bus and no polling scheme enforced over the bus, multi-drop, and a bandwidth of at least 10MBit per second.

A cluster consists of three sections; the network access section, the FTC complex, and the subsystem support section. An example of a typical cluster is shown in Figure 7.

The network access section establishes contacts between the cluster and the rest of the system. Each of the multi-drop connections to the intercluster bus is achieved by a Network Interface Unit (NIU). The NIU handles at least the first two layers of a multi-layer network protocol. The NIU, in turn, is connected to an Attached Processor (AP) which processes and deals with the issues associated with the network protocol in la-

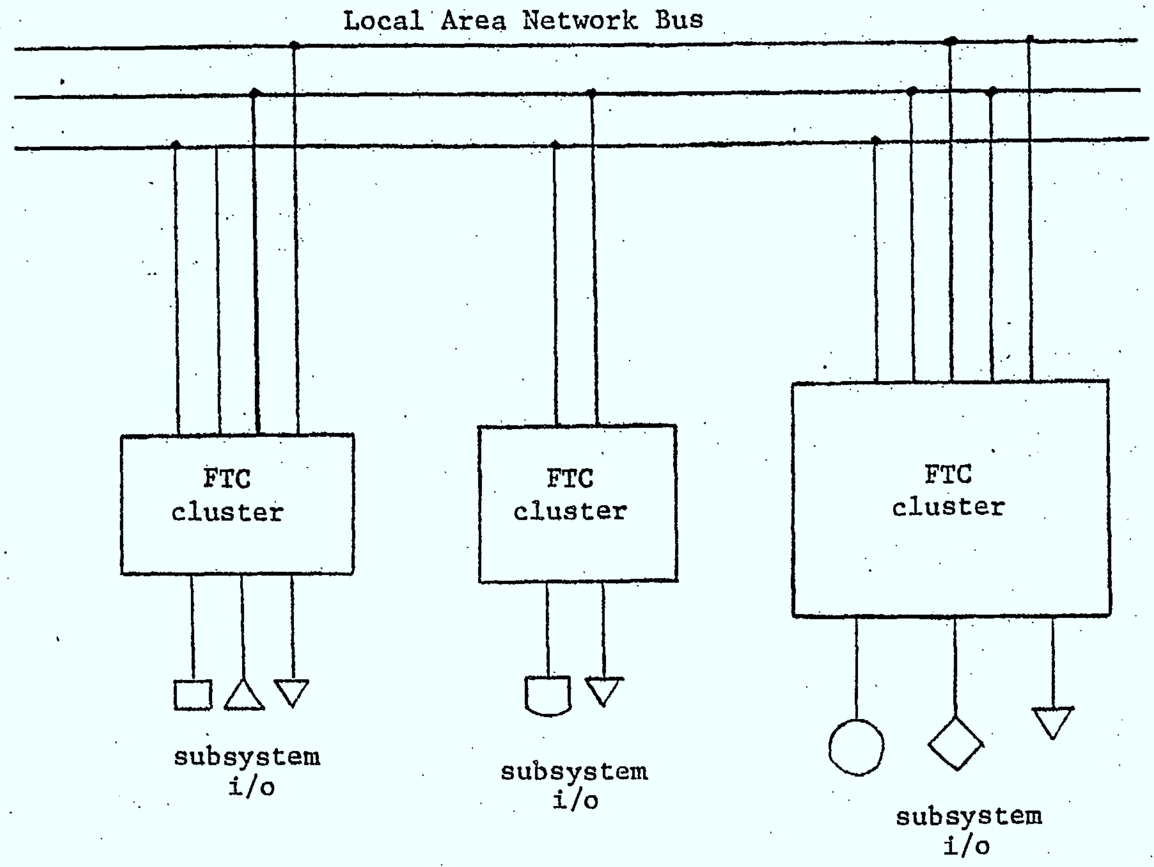


Figure 6. AASC On-Board System

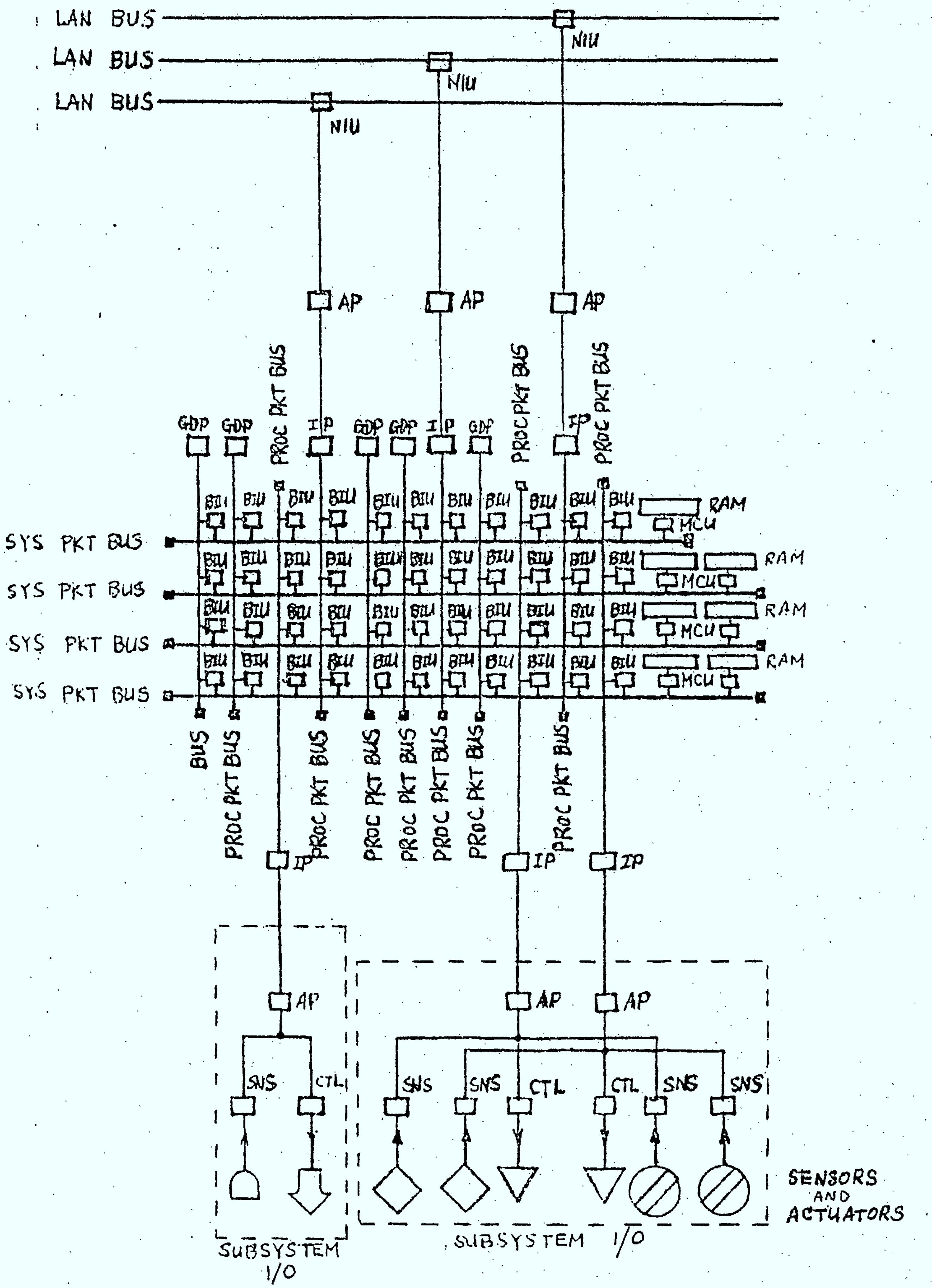


Fig.7. A 432FTC cluster and spacecraft subsystems

yer 3 and upwards. The other end of an AP is connected to an Interface Processor (IP). This processor is responsible for managing address windows into the FTC complex: it maps address spaces defined in the AP into those in the FTC complex, or vice versa.

The FTC complex is based on Intel's iAPX 432 fault tolerant computer. It houses the upper layers of the local intelligence required for the cluster and provides most of the numerical computing power of the system. It consists of a mesh of packet oriented buses flanked by processors and controllers. The System Packet bus handles inter-processor communications, while the Processor Packet Bus controls distribution of information between a processor and a number of System Packet Buses. The former is supported jointly by all processors in the complex, while the latter is sponsored by a single processor - either a General Data Processor (GDP) or an Interface Processor (IP). At the intersection of the processor packet buses is a Bus Interface Unit (BIU), which controls information transfer between them. The Memory Control Unit (MCU) attaches a memory module (RAM) to the System Packet Bus. The intelligent memory controller converts logical address spaces implied on the packet bus into physical addresses in the memory module.

The fault-tolerance embedded in the FTC complex hardware surpasses that of any known computer system. The current version of GDP provides a throughput of approximately 0.2 MIPS (Million Instructions Per Second), or the capability of a typical minicomputer. The multiprocessing is performed in a manner completely transparent to the overlaying software structure. The over-all computing power of the cluster is determined as a function mainly of the number of GDPs and system packet buses.

The subsystem support section handles all non-network i/o to and from the FTC complex. It employs the same IP-AP structure as in the Network Access Section. In place of NIUs, however, the AP is connected to a structure that controls an i/o subsystem. The i/o subsystem in this application would be a satellite subsystem such as an AOCS, a temperature control subsystem, an uplink/downlink channel, an on-board power management subsystem, etc. The fault-tolerance within the subsystems will be provided using known FTC techniques, including the use of multiple copies of such a subsystem.

The software will consist of three major module groups: the operating system and its support, the fault-tolerance management software, and the application software. They will be developed as Ada packages and will remain highly configurable to permit intense customization by the application system designer. Together, modules from these three groups will establish and support hierarchical process structures which may dynamically change shape during their life according to the needs of the system.

The execution of such software will be carried out under the protection of the data structure protection mechanism inherent in the FTC complex hardware. This implies the extension of the enforcement of Ada type-checking concepts into the execution environment (in Standard Ada, this is done only at compile time).

4.3 The Basis for Comparison

For the UDS/FTBBC system, a system with a typical UDS configuration that would employ three HLM-1553A bus pairs, each having several TMs is considered. For the OBDH system, a configuration with one CPDU, two CTUs, and two sets of OBDH buses, each supporting several RTUs, is assumed and is illustrated in Figures 4 and 5. For the AASC system a few clusters are envisaged, each containing several GDPs (Ref.17) and IPs, several packet buses and BIUs that inter-connect processors, and memory modules and MCUs. Each cluster is linked together by wide-band local network arrangements similar to Ethernet, as shown in Figures 6 and 7.

While the AASC is only a hypothetical system, its detailed design is well underway and supporting VLSI hardware and software components are the subject of the manufacturer's extensive development efforts. The system is considered here as a reference point in order to evaluate the other two systems against the latest technological developments in the area of fault-tolerant computing.

Comparisons are made on several aspects of these systems including over-all system design, software, hardware, and physical characteristics. In those comparisons which are less open to subjective evaluations, a score is given on the scale of 0 - 9 and a weighted score (0 - 81) is calculated in an attempt to establish the basis for a quantitative comparison. Such weighted scores are tabulated for each system to yield accumulated sums. Note that the weighting of features, individual system scores and hence the weighted scores themselves, may be subjective. The method is nevertheless preferrable to arbitrary, non-exhaustive comparisons.

As the result of the comparison is greatly in favour of the AASC, the authors would emphasize that they have made every effort to remain unbiased. The capabilities of the AASC can be fully demonstrated only after the details of the fault-tolerant extension to iAPX 432 are made public.

Explanatory notes to Table 2 are indicated under "#n", and are appended to the table.

Table 2: COMPARISON OF FAULT-TOLERANT ON-BOARD COMPUTER SYSTEMS.

FEATURES	Wt.	UDS/FTBBC	Sc.	ESA-OBDH	Sc.	AASC	Sc.
system hierarchy	8	sets of (HLM-bus-TM'S) #1	7 56	CTU-owned buses	5 40	fully distributed	9 72
system-wide synchronization		One HLM as clock master 2.5ms #2		CTU generates bus-clocks at 500KHz		tightly synchronized buses.	
estimated software/hardware effort ratio for typical application		30% : 70%		40% : 60%		80% : 20%	
multiple mission potential	9	yes	9 81	yes	9 81	yes	9 81
mission to mission adaptability	9	expected to be unstable in larger mission #4	5 45	limited to small to medium scientific missions #5	6 54	highly flexible #6	9 81
subsystem interface:							
- clarity #7	8	HLM involvement touchy #8	6 48	clean	9 72	clean	9 72
- load range	7	up to the capacity of MIL-STD-1553A bus	7 49	multiple RTU's, but limited #9	5 35	very wide #10	9 63
- system transparency #11	8	HLM-sets always visible #12	4 32	CPDU/CTU always visible #13	5 40	fully transparent #14	9 72
system test and validation estimates:							
- cost of setting up test procedure	5	average #15	8 40	low-average #16	9 45	major project #17	4 20
- cost of testing and validation	7	high #18	4 28	average #19	6 42	low - automated #20	9 63
- quality of testing and validation	9	average	6 54	average - high	7 63	very high - automated #21	9 81
- possibility of software validation	8	difficult	4 32	very difficult	3 24	possible #22	9 72
compliance to the FTC rules: #23							
-[1] single point of failure	9	HLM comprises a localized hardcore failure point #24	6 54	CPDU is a single point of failure. CTU, though limited, can create local paralysis if it fails #25	4 36	can remove all hardcore #26	9 81

FEATURES	Wt.	UDS/FTBBC	Sc.	ESA-OBDH	Sc.	AASC	Sc.
-[2] fixed master/ slave relationship	8	Yes, HLM-TM's	6 48	Yes, CTU-RTU's	5 40	none fixed #27	9 72
-[3] fixed fault arbiters	8	fully distributed into hardware gates hardware dependence #28	6 48	CTU does it all	3 24	fully software controlled #29	9 72
-[4] module decoupling	8	inter-bus daisy chaining ambiguous #30	6 48	multiplexed bus protected little	5 40	rigid capability-based isolation	9 72
-[5] module strength	8	good design language	7 49	by programmer discipline only	4 28	program, language, compiler and O/S, hardware discipline	9 72
-[6] subsystem layering (vertical decoupling)	8	by programmer discipline only	7 56	by programmer discipline	7 56	O/S, language and hardware encourage layering	9 72
ground/on-board communications		not discussed		1-mega bit/sec maximum		as much as technology can offer#31	
ASM features #32	8	autonomous lower level fault recovery	5 40	- entirely ground-controlled - block telemetry a move toward on-board processing	3 24	- hierarchical fault handling structure - hardware fully autonomous - large CPU throughput - most of ASM conditions can be met	9 72
building block type	5	non-dedicated but pre-assigned computer units (SCCM's) and their components	7 35	dedicated computers (xxU's) with some component options	5 25	non-dedicated and some dedicated redundant computers and their components	9 45
building block granularity #33		module and submodule levels		module level		submodule level	
data units/formats #34		MIL-STD-1553A frames, no application level formats		ESA standard telemetry		Ada datum - packet bus frames ISO 7-layer interconnect message model	
maturity of technology	9	breadboarded #35	6 54	in-orbit	9 81	under development #36	6 54

FEATURES	Wt.	UDS/FTBBC	Sc.	ESA-OBDH	Sc.	AASC	Sc.
space qualification	9	some component level difficulties	7 63	now	9 81	by 1986	5 45
processing power (minimum application)	8	1.2 MIPS approx.	6 48	0.8 MIPS approx.	4 32	0.2n~2.4 MIPS appr. n: # of clusters (0 < n < 256) #36a	9 72
Operating System:							
- name		customized O/S				iMAX432	
- distributed		some		no		yes	
- multitasking		limited - back ground/foreground		no		yes	
- layered structure		some		?		yes	
- hierarchical structure		some		not visible		yes	
- implementation language	7	UDS Design Language	7 49	assembler	3 21	Ada or Ada+Pascal	9 63
- inter-processor communication		restricted #37		controlled #38		fully asynchronous; capability based	
- inter-module protection	8	violation can occur #39	3 24	violation can occur #40	4 32	fully protected; capability based.	9 72
i/o processing:							
- response granularity	6	5 ~ 7 ms	4 24	500 microsec. approx.	6 36	10 microsec. approx	9 54
- timing	7	restricted	4 28	controlled	4 28	fully asynchronous	9 63
- mode	5	polling	5 25	polling	5 25	event driven	9 45
- realtime access		low level modules		low level modules, FORTRAN libraries		structured data access	
application software:							
- design language	6	UDS Design Language	7 42	flowchart	4 24	Ada	9 54
- coding language	6	UDS Design Language	7 42	assembler/FORTRAN	5 30	Ada	9 54
- modules		subroutines		subroutines/ functions		subprograms, tasks packages	
- structured programming	8	can be used	8 64	can be used at design level only	5 40	fully enforced	9 72
process-processor interdependence	9	dependent	4 63	dependent	4 63	independent #41	9 81
CPU type	5	TI9900, MC68000	6 18	HARRIS 6100	3 15	iAPX432	9 45
CPU bits	5	16	6 30	12	5 25	32 ~ 80	9 45
number of processors in a system	7	about dozen	7 49	about dozen	7 49	up to 31 per cluster #42	9 63

FEATURES	Wt.	UDS/FTBBC	Sc.	ESA-OBDH	Sc.	AASC	Sc.
inter-processor communications	6	HLM bus-controller initiated DMA	7 42	CTU controlled DMA	7 42	bus-controller controlled DMA with queueing	9 54
CPU unloading	6	yes - BC	8 48	no	2 12	yes - BIU	9 54
hardware fault confinement		yes		not clear		yes	
hardware reconfiguration	8	no - software initiated. Some cross-strapping	7 56	no	2 16	yes - fully automatic and transparent	9 72
redundancy management							
- defined	6	yes	9 54	discussion only	5 30	yes	9 54
- elements	7	CPU, bus, memory, i/o channel	9 63	CPU, bus	6 42	- CPU, bus, memory, i/o channel	9 63
- methodology	7	hot/blank spare	7 63	not defined yet	3 21	- "marriage" concept	9 63
VLSI implementation	9	being designed	7 63	not planned yet	3 27	prototyped	9 81
system bus:							
- type		MIL-STD-1553A		ESA Standard OBDH bus		iAPX432 packet bus and a LAN bus	
- format		serial		serial		parallel/serial(LAN)	
- clocking		Manchester code		baseband modulation Litton code			
- speed				1MHz		(5MHz)	
- capacity							
- mode				full duplex		full duplex	
- adaptor		BA/BC		Bus Adaptor/RBI		BIU	
- isolator				transformer		optical coupling	
- access	5	slotted	6 30	slotted	6 30	asynchronous	9 45
- transfer control	7	semi-distributed	7 49	centralized	3 21	distributed	9 63
- bus-CPU relation	8	semi-fixed #43	5 40	semi-fixed #44	4 32	software controlled	9 72
- max. length				20 m. approx.		5/1500m. approx.#45	
- media		twisted wire		twisted shielded pair		backplane/Coax cable	
- broadcasting	7	no	5 35	yes	9 63	yes	9 63
- priority control	6	preassigned and fixed	4 24	preassigned	9 36	software controlled	9 54
processor modules:							
- system executive module	8	executive HLM	6 48	CPDU	4 32	none	9 72
- high level module	8	HLM	7 56	CTU	6 48	GDP #46	9 72
- low level module		TM		RTU		IP,IPL,AP	

FEATURES	Wt.	UDS/FTBBC	Sc.	ESA-OBDH	Sc.	AASC	Sc.
- i/o module		TM-IOBB		RTU		AP	
- memory control module		TM-MIBB		RTU		MCU	
- multi-processing	7	centrally controlled	6 42	preassigned	5 35	fully transparent #47	9 63
boards		customized		EUROCARD (?)		Intel 432 standard	
minimum packaged volume				7500 cm cubed approx.			
weight (minimum mission)				8.2Kg approx.			
power consumption (minimum mission)				21 W			
WEIGHTED-SCORE TOTAL	3150		2179		1835		3062

EXPLANATORY NOTES TO TABLE 2.

#1 An HLM (high level module) "owns" an MIL-STD-1553 bus, onto which several TMs (terminal modules) are attached.

#2 The UDS has a 2.5 ms system-wide clocking system called RTI. The clock is generated by one of several high level modules (HLMs) designated as the command/control computer or system executive HLM. The clock is used by other HLMs and TMs to generate precisely timed results.

#4 The lack of system level consideration is evident through the absence of a structured and detailed discussion on software and system architecture in functional terms and will likely cause difficulty when designing a large scale and complex applications.

#5 The system can only have one or two system bus(es), each controlled by a control unit (CTU). The traffic capacity of the bus is low, and so is that of the satellite to ground link (Ref.13).

#6 The well-disciplined flexibility inherent in the design of the processor building blocks can be easily applied to give a wide range of mission adaptability.

#7 A measure of how easy and unified the method is of attaching the payload to the computer system.

#8 The HLM tightly controls the use of a bus. A number of TMs can then be attached to the bus to be put under the control of the HLM. The scheme involves careful preparation, taking into account various factors such as the bus priority, number of TMs, required response time, TM to TM traffic volume, HLM monitoring actions, etc. Futhermore, this HLM-bus-TM set-up must always exist in relation to other similar sets, as these sets are linked together by a hardware priority link.

#9 The capacity of the OBDH bus is fairly low (Ref.13).

#10 The processing, memory, bus and i/o channel capacities are all adjustable within a wide range to the demand imposed by the load.

#11 The measure of shielding the computer system's ideosyncracies (unnecessary details) from the user subsystem.

#12 A subsystem virtually belongs to an HLM through a TM. It has to be able to issue requests to the HLM supplying physical connection information (address, size of transfer) to achieve a subsystem to subsystem data transfer.

#13 A subsystem belongs to the CTU through the RTU(s). Every i/o a subsystem makes is controlled by it. CTU controls several subsystems. A CPDU controls all CTUs and RTUs. The subsystem must be aware of these facts (Refs.8,14).

#14 Packet-oriented inter-process (and NOT inter-processor) protocol and inter-cluster protocol will be the only significant interface between subsystems, between a subsystem and computer systems, or between an on-board function and ground functions. Such protocols are software defined and software controlled.

#15 This would be equivalent to the cost of setting up any other space qualification process involving VLSI components. Some effort has already been made in this area (Refs.11,18-24)

#16 ESA/ESTEC has started the basic hardware design with the anticipation of extensive validation and test requirements (of hardware) for each mission. (Refs.11,18,19,20)

#17 Because of the higher level of sophistication involved both in VLSI hardware and software, and the volume of processing capability expected on the system, the initial set up would be fairly expensive. However, the resulting test/validation facility would be highly automated.

#18 Most of the testing process will have to be set up anew each time a mission is defined because of the lack of designed-in testing concepts and facilities.

#19 The same testing procedure will be modified in accordance with the mission profile and applied using identical test facilities. However, most of the test will be done manually.

#20 The only way to rationally test sophisticated software/hardware as complex as this would require a highly sophisticated automated facility. A series of extensive tests would then be possible without much human intervention.

#21 By minimizing human intervention in the validation and test process, the chance of including operator and other environmental errors would be greatly reduced. By systematically executing a set of predefined test sequences geared to the mission profile, the reach of such validation and tests would be phenomenal.

#22 Highly-structured, rigidly-typed capability based languages (such as Ada) and process organization (Ada packages, tasks and procedures) supported and enforced by the iAPX 432 architecture should provide an opportunity for eventual automatic software validation.

#23 See Appendix B, Fault Tolerant Computing Design Rules.

#24 There will be several HLM-bus-TM(s) sets within a given UDS system. While distributed to a certain extent in this way, nevertheless, the HLM constitutes a single point of failure within a set.

#25 The CPDU is clearly a single point of failure (Ref.10). Under its control, the system can take up to two CTUs. Each CTU may in turn support several RTUs. Access to the bus is fully controlled by CTUs. Should one CTU fail, the chance of some RTUs failing is great, although in theory, the other CTU is supposed to take over the failed CTU.

#26 All five elements of the system (GDP, IPL/IP, BIU, MCU and AP) may be multiplied in an orderly fashion resulting in removal of any potential hardcore.

#27 From time to time the software may designate masters and slaves. However, such dynamic designations are not fixed as implemented in hardware.

#28 The FTBBC uses very deterministic error detection methods which should be sufficient for detecting simple faults. However, if there is an error in this detection logic, or if a fault involves interaction among more than one SCCM, this fixed detection mechanism may not always be effective.

#29 Any of the GDPs in the cluster can be designated as an arbiter from time to time, thus avoiding the establishment of a fixed arbiter-arbitee relationship.

#30 How are the priorities defined? What happens to a lower-priority HLM and its bus traffic if a high-priority HLM attempts to preempt them? What if a low-priority HLM loses its bus and attempts to use one which has a higher daisy-chain priority? The fact that buses are not non-dedicated spares may make the algorithm for granting the bus during reconfiguration extremely complicated.

#31 The external communication is separated from the inter-process, inter-processor, or inter-cluster communication. It can be designed to meet whatever the requirement.

#32 The USAF, in conjunction with NASA and JPL, is developing a concept called Autonomous Spacecraft Maintenance. (Ref.7) It is intended to be applied to all future satellites launched after 1990 by the US government. Their report, "Final Report of the Autonomous Spacecraft Maintenance Study Group"(Ref.25), lists conditions for a satellite to become fully autonomous. Here, three candidates are screened against this set of conditions.

#33 A measure of how flexible a configuration and reconfiguration can be.

#34 Assuming that in a layered data-processsing system, data formatting occurs at several levels.

#35 The system level proof-of-concept breadboarding for the UDS was completed in 1978. The FTBBC breadboarding to prove the hardware viability has been partially completed and the testing was started in the fall of 1981. Each building block must be created in VLSI form for actual in-space use.

#36 Most of the submodules are now being put into VLSI - there are two already in that form and others coming by December, 1982.

#36a A cluster may have up to 31 iAPX 432 processors, each having 0.2 MIPS CPU power. If we configure these into the most fault-tolerant configuration, the cluster will provide approximately 0.8 MIPS throughput in a virtually destruction-free set up.

#37 All inter-processor communications are governed by the HLM that owns the bus. There are three HLM-bus combinations in the system. Any TM on a bus must be cleared by the HLM before data could be transferred.

#38 There are two sets of system buses in the system, each governed by a CTU. The CTU dictates every transaction that an RTU would make over the bus either to other RTUs or to the CTU (Ref.10).

#39 The communication is not based on a rigid layered protocol. Hence the inter-layer protection virtually does not exist. That the actual transfer occurs in DMA mode even increases a chance of contaminating larger memory space in a short time should a protocol error occur.

#40 Same as #39, except that a relatively low traffic rate may lessen the damage in the event of a spillage.

#41 In iAPX 432, processes exist apart from processors as an independent abstract entity. This is the basis for the true user-transparent multiprocessing (for increased throughput) and the dynamic reconfiguration scheme (reliability feature) of that computer.

#42 An AASC cluster can contain up to 31 processor modules (GDP or IP), eight packet buses, and 63 total modules including MCUs and BIUs.

#43 As stated in note #39, a bus belongs to a control module (an HLM) and users (TMs) need its permission to access the bus. A TM typically is connected to more than one bus.

#44 A user module has a choice of selecting one of two buses. The selection is not transparent to the user or software.

#45 Within a cluster, parallel packet buses link processors and memory elements. Inter-cluster communication is achieved by multiple local area network buses.

#46 A system executive HLM virtually controls the mode and extent of multi-processing.

#47 Because of the successful separation of processes from processors, multiprocessing occurs in the most desirable form, namely, complete transparency to the user. Processes are dispensed from dispatch port(s) to physical processors. Because of this divorce of the two objects, the number of processors actually available in the system only affects the throughput and reliability of the process being executed and is completely transparent to the process as a logical entity.

5. DISCUSSION AND ANALYSIS

5.0 General

It is evident that the global development plan that includes both the UDS and FTBBC has been carried out in a proper top-down fashion, finding its inception in the TOPS (Thermoelectric Outer Planets Spacecraft) and other related projects conducted at the JPL in the early '70s. However, in spite of the great foresight in these projects, several major developments have taken place elsewhere, concurrently with the JPL activity, which have profoundly affected the evaluation and the implications of JPL's efforts. These external developments are:

- the flourishing of distributed computing and local area network technology in business, industrial and military sectors

- substantial refinements and proliferation of structured design and related software engineering techniques

- further diversification of computer architecture

- drastic improvements in the availability of VLSI components to system designers

- further study into fault-tolerant computing principles and practices.

The following subsections provide brief discussions and assessments of the impact which each of these developments might have made on the implications of the current FTBBC project.

5.1 Distributed systems and local area network.

Since effective fault-tolerance in the UDS/FTBBC depends largely on dedicated and non-dedicated redundancy in the form of distributed processor arrangements along multiplicated buses, any progress in the area of distributed processing would have the potential for a significant contribution to the design of the UDS/FTBBC. This is particularly the case with the emergence of local area network concepts such as Ethernet or IEEE802. In comparison, the MIL-STD-1553 buses (1553A and 1553B) are in several ways overly restrictive. In retrospect, the restrictiveness of the 1553 bus often resulted in an over-all system inflexibility in reconfiguration and a system level vulnerability in the presence of faults. Dr. Rennels acknowledges that the FTBBC bus structure is undergoing redesign at his laboratories to incorporate an enhanced Ethernet-like bus concept.

Another, more profound impact is felt in the emergence of the concept of fully non-dedicated distributed processing. There have been several efforts to implement this using the latest 16-bit mini and micro computers (SARGOS - Ref.15, C.Vmp - Ref.26, ARPANET IMP). In addition, movements towards establishing a distinction between the physical reality of computing and the manifested dynamism of program execution is becoming increasingly noticeable in modern computing facilities. Intel's iAPX 432 system, for example, completely separates processes from processors, implementing in VLSI what the SARGOS ground station computer system attempted to do using conventional 16-bit microprocessors. The significance of this in the light of implementing truly user-transparent multiprocessing, as well as providing a highly flexible reconfiguration mechanism, is obvious. When combined with hardware supported, capability-based data protection mechanisms (as compared to conventional memory-protection mechanisms), the reliability of systems constructed using such components is expected to be phenomenal. The HLM-controlled data transfer mechanism of the FTBBC can be said, in comparison, to be less distributed, less specialized and more restrictive.

5.2 Structured design and software engineering techniques.

After a decade of often turbulent experiments with top-down/structured system techniques by the data processing and software development industries and academia, it is now obvious that the methodology has gained a considerable acceptance. As a side effect, the distinction between hardware and software became of decreasing importance in system design. This leaves the definition of principal system function and its systematic decomposition as the major concerns of the design process.

Developments in semiconductor technology permitted the implementation as hardware components of many increasingly higher level functions traditionally supported by software. This further blurred the boundary between hardware and software as seen by the system designer. The trend continued until the distinction became a convenience or an economic issue rather than an issue related to design principles.

Both the UDS and FTBBC design phases were completed well before this revolutionary change in system design concepts hit the world. As a result, they did not receive some of its benefits. This can be seen in the FTBBC, for example, in the absence of continuous high level to low level support for fault-handling. In general, a given system is assumed to have fault potentials that form a hierarchy in terms of "levels of abstraction" of such faults; there are faults that may occur at higher as well as at lower levels in the system hierarchy. The difference in the diagnosis and remedy of system faults at both extremes could be as disparate as those required for curing insomnia and removing a sliver from a fingertip, to give a human analogy. Most error handling in the UDS/FTBBC appears to be concentrated in the lower layers of the hierarchy, or in the FTBBC hardware. Concerns over fault-handling in the higher or software levels are mentioned in the study but are not as obvious or detailed.

Other developments in software engineering include the discovery of elaborate module decomposition rules, namely G. Myer's "module strength" and "module decoupling" rules (Refs.27,28). Seen in this light, some of the ways in which data and control exchanges occur between HLMs and TMs may be considered questionable, although they are very much dictated by the MIL-STD-1553A bus protocol. For example, the extreme authority given to an HLM in such a situation permits it to step into the data space of a subordinate TM to set up a data block transfer. This may result in reduced modular strength of the TM (in this case, a TM is a combined hardware/software entity) and increased modular coupling between the HLM and TM which might have subtle yet potentially drastic side effects.

5.3 Advances in Computer Architecture and VLSI components

The divergence of computer architecture has started and its rate has accelerated since microprocessors came into wider use. Several interesting computer architectures have been attempted to suit specific application needs (minicomputer arrays such as C.Vmp of Carnegie-Melon University, bit-sliced microcomputers for applications that require an extreme real-time response, processor arrays and pipe-lining for radar image processing, to pick a few). Computers are no longer limited to von Neumann architecture. Over and above the ordinary demands for increased throughput and reliability, there is a general trend for multiprocessing to support a higher degree of process concurrency and other dynamic requirements. Also of importance are the sophisticated memory access and protection schemes which are needed to maintain the increasingly complex data structures being utilized in sophisticated execution environment.

Similar philosophical changes are taking place in the software world, as is seen in the development of the Ada programming language and Ada program development and execution environment issues. Tighter data structures and procedure controls, and a need for clearer module encapsulation are demanded by Ada in exchange for an added fluency in establishing process concurrency.

While the FTBBC's architecture is the result of a highly innovative design concept, it has not successfully accommodated many of these external conceptual developments, most of which became known after the UDS/FTBBC was well underway. In particular, strict data protection measures and the "liberation" of concurrent processes from tight synchronization requirements are not evident in the design.

The FTC expansion of the above mentioned iAPX 432 architecture will witness the implementation of building block concepts, similar to those developed for the FTBBC, as commercially available VLSI components. There are differences between the two. However, the degree of similarity between the components in the two computers is of more importance. The BIBB, MIBB, IOBB, HLM and TM each has a functional counterpart in the BIU, MCU, AP, GDP and IPL, respectively. This can be viewed as indirect but strong support for the veracity of the original UDS/FTBBC approach chosen by Dr. Rennels and his group.

The advantage the designer of any on-board computing facility would have over the FTBBC is that all the theoretical and technological developments since the start of the UDS and FTBBC projects can be now evaluated and systematically incorporated into a new design.

5.4 Post UDS/FTBBC fault-tolerant computing developments

The fault-tolerant computing issues caught the attention of the general computing community in the early 1970s. Since then interest has been steadily growing mostly among researchers, designers, and project managers involved in aviation or space-related projects. The trend will not be limited to general applications but will be shared by any field where system reliability is of importance. It is anticipated that sophisticated fault-tolerant features will become the norm rather than the exception in many such computing facilities.

The Annual International Conference of Fault-Tolerant Computing (FTCS) is one forum where results from on-going studies and experiments are exchanged. At its last meeting (FTCS-11, June 1981, Portland, Maine) a trend towards clearer classification and axiomization of fault-tolerant system theories and design methodologies was noticeable. The following is a summary of a report, compiled by Eidetic Systems Corporation on the significant activities at the conference:

- J. Kuhl and S. Reddy of the University of Iowa proposed a revised diagnosis model that would be useful in establishing a system level fault profile in a fully distributed system (Ref. 29). It is stressed in the study that even to identify a faulty unit in such an environment is not a trivial task. The theory also implied the uselessness of facilitating a centralized fault-arbiter, supporting the third rule of the FTC Rules (see Appendix B).

- A similar theoretical study was made by J. McPherson et al of the University of Wisconsin, Madison, using the UDS as a model (Ref.30). They have developed a set of theorems that describes the correctness of the system in the presence of faults. The theorems are developed for various control structures realizable within the global UDS scheme. The process-oriented guardian-ward relationship, as they call it, also implies a support for the second rule of the FTC Rules, as the relationship takes processes rather than fixed (hardware) processing units as the components of the control structure.

- A remarkable development in the area of fault-tolerant data structure was presented by J. Black of the University of Waterloo (Ref.31). This unique study not only classified various data structures in terms of robustness against faults, but went on to suggest algorithms for the detection and correction of faults applicable to some of the data structures.

- W.G. Wood of the University of Newcastle upon Tyne reports an inter-process protocol model that aids the recovery process in a distributed system (Ref.32). His theory is one of the first such studies that address issues of recovery from higher level faults (that occur within processes rather than processors or other hardware elements).

- A theoretical study by C.L. Kan and S. Toida of the University of Waterloo also demonstrates the hierarchical nature of the fault and fault-tolerance within a system by describing the application of "fault-tolerant graphs" (Ref.33).

- Representing the French effort to establish a country-wide distributed fault-tolerant database, P. Azema of Laboratoire d'Automatique et d'Analyse des Systemes du C.N.R.S. of Toulouse and his group are working on the application of computer networking technology to establish system level fault-tolerance (Ref.34). In particular, he described how they would establish a system-wide fault-tolerance using the Transport layer of the multiple-layer interconnect protocol model (such as ISO's 7-layer interconnect model). The study represents the uniqueness of several distributed system studies in Europe, many of which, like this one, utilize a Petri-net model as a study tool. The application of system wide fault-tolerance, as represented by this study, is essential to a successful fault-tolerant spacecraft system in its entirety.

- A study of the topology of a distributed system to determine a highly fault-tolerant computer architecture was conducted by D.Pradhan of the Oakland University of Rochester, Michigan and S. Reddy of the University of Iowa (Ref.35). A topology that permits efficient routing and readily distributed fault-diagnosis is proposed.

6. CONCLUSION

The extent of the fault-tolerant features of the FTBBC was revealed in this study. The FTBBC was then compared with two other on-board computing facilities which have similar objectives; an existing European system developed for unmanned applications (ESA/OBDH), and the preliminary design of a proposed system, the AASC, which would be implemented using the latest available technology with potential for utilizing future developments.

The VLSI implementation of the FTBBC study, begun in 1972, has been a pioneering venture in providing fault-tolerance combined with versatility in a spacecraft on-board environment and has been instrumental in pushing the state of the art a stage further. It has, however, been subject to various constraints; historic and economic considerations amongst them. The hardware experience of the design team has resulted in an excellent level of hardware fault-detection and confinement but there are ambiguities in its diagnosis and recovery capabilities. The enforced use of the MIL-STD-1553A bus has entailed restrictions in performance and expandability. Its custom-built operating system also has limited future capability and there would be structural restrictions placed on applications. The software interface is not clear and attention to the design of fault-tolerant software is lacking. Space-qualification is still in progress and, at the time of writing this report, there is some doubt as to the probability of the project being completed because of economic constraints.

The OBDH, on the other hand, has been space-qualified and has good test procedures in place both for hardware and software. The project has been completed, although there are some on-going modifications. There would, however, appear to be limits to its operating system capabilities and in its adaption to future missions. Its design, too, is hardware oriented. The level of its hardware fault-tolerance does not have the same sophistication as the FTBBC and its design violates the FTC rules in some critical areas. The micro-electronics technology used in the OBDH is now becoming partially obsolete.

Since the inception of these two computers, there has been rapid progress in many pertinent areas such as system design methodology, advanced high-level language concepts, software engineering techniques, computer communications, VLSI technology, computer architecture study, and fault-tolerant computing theory and practices. It is felt that, in view of the current trend towards greater fault-tolerance and increased on-board processing power, advantage must be taken of this progress in order to achieve a computer system capable of meeting these goals. As the design of the AASC shows, such a system is a viable concept. Furthermore, it would be capable of surpassing in performance, reliability and cost-effectiveness, any existing microcomputer facilities developed for on-board use.

REFERENCES

1. George Gilley, "Digital Hardware for Use in Spacecraft Control Applications" AAS 80-031.

2. David A. Rennels, JPL, "Fault-Tolerant Building Block Computer Study" JPL Publication 78-67, July 15, 1978.

3. David A. Rennels, Algirdas A. Avizienis, Milos D. Ercegovac, "Fault-Tolerant Computer Study - Final Report" JPL Publication 80-73

4. David A. Rennels, B. Riis-Vestergaard, Tyree, "The Unified Data System: A Distributed Processing Network for Control and Data Handling on a Spacecraft" NAECON '76 Record. p.283-289.

5. Fred Lesh, Paul Lecoq, JPL, "Software Techniques for a Distributed Real- Time Processing System", Proc. IEEE National Aerospace Electronics Conference, NAECON '76 Record.

6. Eidetic Systems Corp. "Towards Autonomy in Spacecraft Computer Systems", ESC-SP-001, July 1981.

7. G.C. Gilley, " Fault-Tolerant Design and Autonomous Spacecraft", Aerospace Corporation, Los Angeles, Calif.

8. P. Lo Galbo, "ESA On-Board Data Handling System Concept." Proc. of Internat. Conf. on Spacecraft OBDM, Oct. 1978.

9. E. Mattson, "Signal Processing on Board Scientific Satellites" Proc. of Internat. Conference on Spacecraft OBDM, Oct. 1978.

10. P. May,(ESA/ESTEC), Noordwijk, "ESA Approach to OBDH Technology" Proc. of Internat. Conf. on Spacecraft OBDM, Oct. 1978.

11. Des Deaney (ESTEC) "Major Influences in the Development of Overall Checkout Equipment (OCOE)" Proc. of Internat. Conference on Spacecraft OBDM, Oct. 1978

12. A. Beretta, "The ESA Standard Remote Terminal Units (Part 1)" Proc. of Internat. Conf. on Spacecraft OBDM, Oct. 1978

13. B. Fritsch, R. Gunzenhauser, "The ESA Standard Data Bus", Proc. of Internat. Conf. on Spacecraft OBDM, Oct. 1978.

14. D.C. Chaturvedi, "Typical Applications of the ESA OBDH Subsystem" Proc. of Internat. Conf. on Spacecraft OBDM, Oct. 1978.

15. Y.Deswarte et al. "A Fault-Tolerant Multi-microprocessor Architecture for SARGOS" Fault-Tolerant Computing Symposium (FTCS-11) June 1981.

16. A. Lucas, "The ESA Standard Central Terminal Unit" Proc. of Internat. Conf. on Spacecraft OBDM, Oct. 1978.

17. Introduction to 432 Architecture - Intel 1981 171821-001

18. J.B. Rowles, "ESA Satellite Checkout and Test Philosophy" ESA/ESTEC, Noordwijk, ESA SP-141.

19. G. Buroni & M. Pascucci, "The OBDH Test Equipment and its Goals", Laben, Milan, ESA SP-141.

20. B.E. Melton, "Software for Checkout Equipment", ESA/ESTEC, Noorwijk, ESA SP-141.

21. C.Kurvin & J.Luther, "Use of Checkout Equipment for Scientific Satellite Testing", MBB, Ottobrunn, ESA SP-141.

22. S.P. Chellingsworth, "Spacelab Integration & Checkout", Bell Telephone, Antwerp, ESA SP-141.

23. G.W.Holmes & V.Stenning, "The Impact of New Technology on Checkout",. Systems Designers, Camberley, ESA SP-141.

24. J.B. Rowles & al, "Appendix: Trends in the Development of Satellite Checkout Equipment", ESA/ESTEC, Noordwijk, ESA SP-141

25. Michael H. Marshall, G. David Low, "Final Report of the Autonomous Spacecraft Maintenance Study Group" JPL Publication 80-88

26. Daniel P. Siewiorek & Stephen McConnel, "C.Vmp: The Implementation, Performance and Reliability of a Fault Tolerant Multiprocessor" 3rd USA-Japan Computer Conference, 1978.

27. Glenford Myers, "Software Development by Composite Design", 1975

28. Glenford Myers, "Software Reliability", 1978.

29. J.G. Kuhl and S.M. Reddy, Division of Information Engineering, Univ. of Iowa, "Fault-Diagnosis in Fully Distributed Systems", proceedings of The 11th Annual Symposium on Fault-Tolerant Computing.

30. J.A. McPherson and Charles R. Kime, Dept. of Electrical and Computer Engineering, Univ. of Wisconsin, Madison, "A Model for Fault-Tolerant Process Maintenance", procs. of The 11th Annual Symposium on Fault-Tolerant Computing.

31. J.P. Black and D.J.Taylor, Dept. of Computer Science and Computer Communications Network Group, Univ. of Waterloo, Ont. "A Compendium of Robust Data Structures", procs. of The 11th Annual Symposium on Fault-Tolerant Computing.

32. W. Graham Wood, Computing Laboratory, Univ. of Newcastle upon Tyne, England, "A Decentralised Recovery Control Protocol", procs. of The 11th Annual Symposium on Fault-Tolerant Computing.

33. C.L. Kwan and S. Toida, Dept. of Systems Design, Univ. of Waterloo, Ont. "Optimal Fault-Tolerant Realizations of Some Classes of Hierarchical Tree Systems", procs. of The 11th Annual Symposium on Fault-Tolerant Computing.

34. Pierre Azema, et al, Labratoire d'Automatique at d"Analyse des Systemes du C.N.R.S., Toulouse, France, "Virtual Ring Protection in Distributed Systems", procs. of The 11th Annual Symposium on Fault-Tolerant Computing.

35. D.K. Pradhan & S.M. Reddy, "A Fault-Tolerant Communication Architecture for Distributed Systems", procs. of The 11th Annual Symposium on Fault-Tolerant Computing.

APPENDIX A

Question and Answer Session with Dr. D.A. Rennels, of the JPL.

A number of questions were generated during the study of the FTBBC and UDS systems. These questions were compiled into a list and brought up in two question and answer sessions held with Dr. Rennels of JPL and both at JPL and UCLA in October, 1981. Several hardware-related questions were asked of JPL engineer Dwight Geer during the visit to the laboratory. "Q." in the following pages implies questions asked by EIDETIC and "A." answers given by Dr. Rennels or D. Geer. Unless otherwise marked, all answers were given or implied by Dr. Rennels.

There were about 120 questions with a slight overlap among some of them. The sessions lasted five hours in total. Answers were obtained for some eighty questions. Thus, some of the questions were left unanswered due to the limit of available time. However, this is not likely to cause a serious impact on the evaluation process as questions were prioritized beforehand in order of significance with regard to understanding the systems.

Answers which are enclosed in parentheses are those which were implied in related conversations during the meetings.

In addition to these sessions, there were a few telephone conversations with Dr. Rennels which included a few system level questions.

Q. - "classes of faults", what are D. Rennel's definitions? (2-50)
TRANSIENT/PERMANENT basis of classification?

Q. - "advanced degradation techniques" (2-52)

Q. - Asynchronous implied in FTBBC?

Q. - Discuss HLM/TM separation in relation to ESA system

Q. - Prearranged message area - why not message protocol handling

A. - (Received the impression Dr. Rennels, being hardware oriented, has not been exposed to the layered software approach.)

Q. - Traffic volume between TMs

A. - (Applications design to do traffic volume calculation)

Q. - Does a HLM own a bus? If so, why does it have to? (3-14)

A. - MIL-STD-1553A protocol demands this.

Q. - Commander-soldier relationship analogy.

Q. - Inter-TM transfers as simple as described in 3-10?

Q. - Inter-TM traffic so high as to warrant DMA?

Q. - HLM-controlled TM reconfiguration, reality of....? (3-12)

Q. - Why does instantaneous detection and signaling of internal faults guarantee straightforward implementation of automated recovery? There doesn't seem to be any such guarantee. (1)

A. - More work needed in software. However, it is a general rule that if fault notifications are made instanteously, there is greater chance of confining them and hence a greater chance of analyzing them properly.

Q. - How much of original architectural freedom is still there? (1)

A. - The Core-BB, for example, is prototyped to take MC68000 processor, as well as TI9900 at CPU subsystem level. Also there is a certain range of freedom in the selection of RAM chips for MIBB. (D. Geer)

Q. - Degraded mode of operation really defined? (2)
- "degraded system state"

Q. - Design methodology used in the 1978 study:
were the four blocks introduced systematically or by guts feelings?

A. - The UDS was a highly systematic study.

Q. - What exactly does he mean by "transient mistakes" (3L) "erroneous bits in memory" accidental - fault containment issue?

Q. - The way backup spares are provided on the bus (3L)

A. - Dedicated "hot spare" for executive HLM, non-dedicated "blank spares" for other HLMs, and dedicated SCCMs for TMs subsystem proper.

Q. - Spare fire-up sequence (3L)

A. - Not precisely defined - application proper elements.

Q. - When everything is packaged in a chip, geographical closeness may nullify the effectiveness of providing redundancy. How does he feel about this. This is a question with the 432/670-type system too.

A. - Yes, a more global level of redundancy must be considered. Strategical distribution of components will become necessary.

Q. - Any conceptual change between 78 and 80? (4L)

A. - A few minor ones, but nothing essential.

Q. - A source per BC, for simplicity?
Then BC should be part of Source Terminal?

A. - This is a restriction placed upon the design because of the MIL-STD-1553A bus concept:

Q. - Hence shall controlling SCCM become part of the Terminal?
And if so, isn't Ethernet-type architecture the natural progression?

- fewer modules
- more autonomous
- more flexible

A. - Yes. No arguments. (He was already contemplating an Ethernet-like bus structure to replace current MIL-STD-1553A bus)

- less exposed protocol (fewer connections)

Q. - Don't need mediator if one (source BA) can manage himself

- also NO ARBITER as in Rule Number 3 of FTC rules.
- also NO SINGLE POINT OF FAILURE as in Rule Number 1.

A. - Yes, if we adopt Ethernet-like bus architecture.

Q. - I/O mapped id address: again, isn't local net-like message, imbedded-id more general?

A. - (He agrees. He explained a modified Ethernet-like arrangement in which units have multiple access to common channels.)

Q. - Examples of "candidate i/o functions" 1) thro' 8):

1) 16 bit parallel i/o
2) 16 bit serial i/o
3) pulse sampling
4) pulse counter
5) pulse generator
6) adjustable frequency generator
7) analog multiplexor with a/d converter
8) high rate DMA channel.

Q. - Recovery options 1), 2) & 3) - how are they derived?

Q. - "external commands" to Core-BB

Are there scenarios concerning how to use them? Just provided for potential future use?

A. - Yes.

Q. - Isn't "halt computation on recurring faults" too simplistic?

A. - Don't think so. Chances are such errors are permanent errors, although debouncing problem must be handled somehow.

Q. - Whole roll-back issue - could it be that simple? (4-58)

A. - There is a scientist in his group working on that issue. He found a rule for setting proper checkpoints.

Q. - Under the assumption that VLSI technology will advance at present rate, is the "BUILDING BLOCK" really a useful approach?

If a "unit computer" with "full" capabilities can be produced using advanced VLSI techniques, and if such a unit can be highly (software) configurable, would people bother "building" an SCCM picking necessary building blocks?

If an SCCM-like chip speaks Ada, or similar standard language; provides enough facilities for the payload, and has all the self-checking capability of the SCCM, would they need a finer breakdown?

A. - Such development is possible. In the FTBBC, SCCM is the unit of redundancy.

Q. - High-level processing, or number crunching, is no more than a specific process type and does not seem to justify division of HLM/TM.

A. - TM normally belongs to a specific subsystem. The distinction will continue to exist.

Q. - Why would the primary unit collect input for a "hot spare" unit?

A. - To aid in the process of potential reconfiguration - although such reconfiguration is not clearly defined.

Q. - Algorithm to determine "good" one is not simple. The mutual checking has been an important issue in fault detection theory.

A. - Cross-checking between HLMs is done in UDS. More study is surely needed.

Q. - BC getting into other's memory space is a violation of the decoupling rule

A. - Dr. Rennels was unaware of the decoupling rule. Also, MIL-STD-1553A protocol almost force this mode of operation.

Q. - Controversial bus-ownership convention:

Temporary relinquishing (loaning) of bus to other an

HLM - what algorithm is used?

A. - Hardware level considerations only. Buses are daisy chained providing a priority structure.

Q. - Any priority should be made software controllable. Or, how is the priority assigned and on what basis? And what happens if contention over the access to a bus occurs?

A. - Priorities are given to the HLMs and they are fixed. No serious considerations were were given to the implications for system or hardware level activities.

Q. - Is the difference between "hot spare" and "blank spare" in distributed UDS configuration all that great?

A. - The hot spare concurrently executes same instruction sequence as the SCCM (in this case the system executive HLM) it backs up. Blank spares are powered up but no execution of instruction takes place until one of them takes over an HLM.

Q. - TM need not be restricted from having:

- the capability to initiate intercommunication

- self-check and localized reconfiguration

A. - TM is a slave process in the UDS design. This comes from the historical background of developing subsystems in which TMs reside, and the control structure adopted at JPL in the past.

Q. - Maintaining "inventory" of available paths, and the system level ability to recognise the remaining topology may not be trivial. Again, the design should have started from the top - including this and other recovery algorithms.

(We confirmed that the planned recovery capability in the SCCM ends at a very localized level such as duplicated CPUs and memory bit correction. The SCCM design is mostly aimed at effective and accurate error detection and confinement of detected errors)

Q. - "HLM-owned" bus necessary? Can one adopt MULTIMASTER type arrangements whereby bus belongs to the system as a non-dedicated redundancy and is up for grabs by anybody who needs it momentarily?

A. - Again, MIL-STD-1553A won't allow such arrangements.

Q. - At least, can we not avoid "ownership" which creates a MASTER-SLAVE relationship?

A. - No, unfortunately.

Q. - Broadcast methodology (4-73)
  - 7 layer issue
  - optimization obvious

A. - Broadcasting in Ethernet is attractive.

Q. - DMA direct to some other SCCM (4-73). Isn't it too bold if BC controlling?

A. - BC's access is limited to a predefined data area and it was assumed safe. (If new bus becomes a reality, this will be all right.)

Q. - "Master-Slave" is the issue here. We'd like to eliminate this single point of failure (Master) since redundancy in sync circuit would be costly and tricky.

A. - Agreed.

Q. - (4-56 1st paragraph) ".... which is a combination of all clock-synchronized morphic fault indicators"

Is combining them all effective in planning recovery strategy which can be hierarchical?

Q. - Master Fault Indicator (4-66)

Q. - rollback - linkage to a software recovery manager, how?

Q. - Why is "listen in" by host seen as necessary? How is it strategically needed in the overall system? (4-78 (2))

A. - When reconfiguration involving TMs may be needed, an HLM which understands the nature of the transaction before the fault would be in a better position to do it.

Q. - LSI-11 original target machine? (2)

A. - No. TI9900 is the one being used to build prototype SCCM. LSI-5 was not even the machine used for the UDS system.

Q. - What do you think about recent redundant memory technology? (3)

Q. - 1.5 times as expensive - a close figure still? (3L)

Q. - Why are self-checking chips more easily tested by the manufacturer? Would seem to be the opposite.

Q. - Out-of-range approach to internal bus access: how does the memory mapped i/o work in segmented machines such as Intel's 8086? (4L)

A. - Don't have experience with such machines. Could be difficult. (D. Geer)

Q. - BA/BC distinction

Q. - SCCM/BC division ambiguous - isn't BC a part of SCCM?

A. - Definition of SCCM experienced a few changes throughout the life of the project.

Q. - Discrete commands:
- how POWER ON would work?
- how INTERRUPT used?
- isn't RECONFIGURE same as INIT?

Q. - Global BIBB structure - were there evolutionary changes?

MANCHESTER NRZ translator	->	EBI
Microprocessor Control Unit	->	MILL or CONT?
CONTROL ROM	->	CONT
DMA control	->	IBI
Data Path Element	->	MILL
??	->	FH

A. - (Could not ask question specifically, but Dr. Rennels suggested there were revisions of structure that resulted in naming confusions.)

Q. - Core-BB block level design: did it change since 1978 design?

A. - No.

Q. - Is Recovery Sequencer still an option? If so, used in what type of circumstances?

A. - Most errors assumed to be transitory. One attempt would be sufficient.

Q. - Morphic, Hamming, parity true (4-39) Morphic pairs (4-55)

Q. - 100 pin package

Q. - Core-BB, what will its position be after disabling its own CPUs?

Q. - Core-BB recurring faults, recurrence defined how? (4-52)

A. - Only once more.

Q. - Reality of VLSI building - how to proceed from the design

A. - (Did not ask the question but an indication was given that VLSI mask design is underway on some of the building blocks)

Q. - Duplex CPU synchronization?

A. - In precise step.

Q. - Stop clock, for what? (4-58)

A. - For the single step sequencing.

Q. - On the headings "Input to DCE/Outputs from DCE." What is DCE?

A. - (Did not ask but apparently a typographical error for PCE, or Processor Check Element.)

Q. - For example, is PCE going to be a portion of a VSLI? Items shown in Fig 4-23 (P4-60) talk about chips, but are they going to be included in a VLSI chip? Are all chip designations for current or past breadboard?

Q. - PLA: Programming Logic Array? (4-63)

A. - Yes.

Q. - Extensive use of Morphic-AND common in space applications? (4-65)

Q. - If HLM/TMs have their own internal clock, what is the use of RTI?

A. - (I pointed out and we eventually agreed, that if the local clock can maintain a reasonable clock accuracy, the system wide synchronization may have to occur only to compensate the interprocessor timing skew. Hence the frequency of RTI signal can be as low as once every second or hour instead of every 2.5 ms.)

Q. - Hardware priority assignment between BCs - not very clear what this means. Each BC is supposed to own only one bus.

A. - This is the arrangement multiple usage of MIL-STD-1553A bus would force on the design.

Q. - Four clock pulse recovery sequence - how common is the design?
Why is this clocking considered effective, if not?

Q. - Notation in Fig. 4-27 (4-67)

Q. - What kind of faults can be corrected and system recovered from by the recovery method indicated in the Recovery Sequencer approach (4-70)

Q. - Control Signal Generator in FHE (4-70), which of FS and RS has it?

Q. - How precise and close are synchronization requirements?

A. - To offset the interprocessor skew. Once a second would be sufficient. (As mentioned above, we discussed this issue a lot.)

Q. - Can we maintain accuracy of local timer higher than the combination of the propagation delay (skew) of the communication bus and the variance at local oscillators?

A. - Yes, it would be possible.

Q. - SCCM memory size 61,440: isn't it too small for some applications?

A. - Used to be sufficient. Admit situation is changing. It actually is only 32KB. The Galileo project is giving us pressure. (D. Geer).

Q. - BIBB/bus control table format - Is the use of Word 1, assigning 0 or 1 as a value, aimed at multiple linking of the control tables? Wouldn't a more formal linked-list convention be more helpful? (4-77)

A. - (Did not ask the question but it became obvious Dr. Rennels did not give that much consideration to the data structure.)

Q. - distinction between Controller/Terminal and Terminal/Terminal significant enough to warrant separate table formats? (4-77,78)

Q. - Any industrial or other applications of FTBBC discussed?

A. - A company in the space industry has approached us.

Q. - Are there fault statistics in aero-space applications? (4-62)

Q. - Is the UDS on Galileo spacecraft?

A. - No. Time constraints forced them to build their own based on RCA1802 processor.

Q. - Still consider fault tolerant computer design a mature discipline? (2)

A. - It depends on one's point of view. Fault detection and containment seem to have well matured.

Q. - Extra copy of the report?

A. - (We obtained one.)

Q. - Which paper at FTCS-11 most impressive?

Q. - Future of UDS?

A. - The project was for the system level study. The concept is still alive and calls for elements to build UDS-like systems. The FTBBC is an answer to the hardware aspect of the UDS.

Q. - VLSI modules coming? (4-51)

A. - Prototype being completed for Core-BB and MIBB. More steps needed. (Shown breadboard of each by Dwight Geer, Engineer at JPL)

Q. - AI in space? (2-52)

A. - May be use of heuristic methods in fault diagnosis in the near future.

Q. - "UDS" a generic name for a group of SCCM, or almost a synonym for SCCM?

A. - No, although there is that overtone, UDS is a system level exercise.

Q. - Was the result of breadboard experiment good enough to proceed to VLSI design?

A. - It is going well but not quite ready for VLSI yet. (When I visited there were two large scale - approximately 16" square breadboards: an MIBB and a Core-BB. They were both being checked out. The MIBB had about 375 ICs excluding RAM chips and Core-BB, 170, excluding a removable CPU subsystem. The current CPU subsystem is based on two TI9900 chips and a few dozen "interface chips" to make the subsystem adaptable to the Core BB main. Their next plan is to create a CPU subsystem based on Motorola's MC68000 CPUs.)

Q. - After automatic replacement, how is it indicated to the external world for easy maintenance access?

A. - (There appeared to be no detailed considerations given to this issue.)

Q. - Modified Ethernet issue again

- MIT C-cube: does it have any relation to what you are thinking about concerning bus redesign?

A. - No.

Q. - FT layer as the transport layer in the 7-layer interconnect protocol hierarchy.

A. - Interesting: sequencing and Hamming code at layer 2, multiple message transfer and CRC, status message capability at layer 3, retransmission potential at layer 4.

Q. - Inter-layer clearance

A. - (Dr. Rennels was not aware of the details of the ISO interconnect standard.)

Q. - 1982 International Symposium on Fault-Tolerant Computing Systems (FTCS-12) - where and when?

A. - 22-24 June, 1982, Santa Monica, California

Q. - Who will be presenting papers?

A. - Half theoretical, half practical.

Q. - How will the contents be chosen?

A. - Mixture of theoretical studies and real project reports. Very carefully refereed papers only. Short papers and special sessions considered.

Q. - What is Dr. Rennel's role?

A. - General Chairman. Dr. G. Gilley of the Aerospace Corporation will be the Program Chairman.

Q. - Recovery machine - AI application?

A. - (Dr. Rennels thinks most likely initial application of AI technology to the FTC will occur in diagnostics then in recovery. After all the deterministic searches are exhausted, heuristic steps might be used to guess faults. The approach seems rather conservative and uses AI only as supplementary means.)

Q. - Top-down - software heavy design of on-board processing applicable seems to be coming. What is your opinion on this with regard to the UDS system?

A. - Local executive software design will be like that. The recovery issue is equated to the issues associated with the scheduling of processes.

Q. - FTC rules (see note #23 of Section 5)

A. - (Dr. Rennels agreed with Rules 1,3,4,5 and 6. "Must be agreed." He objected to Rule 2. He especially liked Rule 6, or rule concerning levels of abstraction.)

Q. - History of UDS/FTBBC

A. - UDS ran from 1972 through 1977 and completed when software demonstration of the feasibility was made. The need for a highly fault-tolerant computer system was determined early in 1970s. System level study was conducted with the UDS project. FTBBC, supported by U.S. Navy and NASA, is solely to develop hardware mechanisms that support fault-tolerant computers made up of VLSI components.

Q. - Hot-spare/blank-spare usage distinction.

A. - Hot-spare is for system executive HLM only. Blank-spare applies to all other HLMs.

Q. - Topology control - wouldn't it be an involved process?

A. - Yes, it would be a very hard task.

Q. - Is the USAF interested in the FTBBC?

A. - Yes.

Q. - Algorithm to determine good from bad SCCMs may not be simple.

A. - Yes, but cross-checking should be sufficient along with the triplicated communication channel.

Q. - Once more about concentration of authority in the system.

A. - Tree-type control hierarchy assumed. System control is application oriented.

Q. - Industry again (Aerospace Corporation only?)

A. - The USAF is showing some interest in supporting future of the FTBBC. Rockwell International is interested in creating MIL-STD-1553A bus interface chips.

Q. - Once more about HLM/TM dependence?

A. - (Dr. Rennels believes there will still be hardware master-slave relationship because of highly specific nature of subsystems.)

Q. - Software capability at JPL. In project teams only?

A. - Matrix organization - FTBBC project does not draw on a software development capability, while such capability is always accessible. However, I am considering involving a graduate student at UCLA on the software aspect of the project. Another graduate student was commissioned before but he left for a computer language project.

Q. - Hardware Recovery Sequencer - how effective?(4-70)

A. - Strictly roll back.

Q. - Distinction between Controller/Terminal and Terminal/Terminal tables significant enough? (4-77,78)

A. - Yes, as long as sticking to the MIL-STD-1553A (so logically not important now that Dr. Rennels is considering a switch to Ethernet-like bus. In the latter case (Terminal/Terminal) controlling HLMs "listen in" and provide a limited form of broadcasting.)

Q. - Use of simulation in FTC study. Any activities?

A. - System architecture simulation studies were done using Multics System through ARPA network.

Q. - HLM relinquishes bus - then what will happens to it?

A. - The sequence of regaining the control of the bus may be involved. When and how are left for software to solve.

Q. - EXEC need not be identical, but protocol needs to be. "all executive HLMs and TMs need to be same".

A. - (Dr. Rennels eventually agreed that it is the protocol between processors that needs to be common, but not the software that generates such protocol on individual processors.)

Q. - Recovery transients considered?

A. - Yes at the hardware level, but not yet at software level.

Q. - What kind of transients possible?

Q. - Current revision

Q. - What happens when an HLM relinquishes his bus after the failure?

- Where does it go within the system architecture?

- What does the other host HLM do if the bus-less HLM tries to use his bus?

- At this point the priority structure becomes fuzzy. What priority will the bus-less HLM assume? How others reconfigured?

- Transient issue - an HLM notices bus failure while he was processing a task. Will he go back to the check point in the task?
  Will he grab new bus before going back?
  Will he transfer the task to a new HLM while he's trying to reorganise?

- Memory code-correction; which processor handles it? Core-BB? If Core-BB, this can mean inter-BB recovery and can get tricky.

APPENDIX B.

The Fault-Tolerant Computing Rules (FTC Rules)

Eidetic has compiled a tentative list of rules which a good fault-tolerant computer system should comply with. Such rules were proposed from time to time, by several groups and individuals in the fault-tolerant and space computing communities, mainly on an empirical basis. Added to this existing set of findings are three further constraints (Rules [4], [5] & [6]) which are brought up anew by Eidetic, and which have been accepted among researchers and practitioners in the field of software sciences as system design principles considered essential in order to increase reliability of complicated software systems. Here, the distinction between the software rules and system or hardware rules is considered insignificant as trends towards acceptance of functional decomposition as the fundamental methodology of system design are increasing among planners and designers.

The rules were explained to Drs. D. Rennels and G. Gilley on separate occasions and received approval, except for Dr. Rennels' objection to Rule [2].

The following are the proposed fault-tolerant computing design rules (FTC design rules):

[1] There shall be no, or as few as possible, single points of failure in the system (the hardware rule).

[2] There shall be no fixed master-slave relationships among processing units (the democracy rule).

[3] There shall be no permanent fault arbiters or judges in the system (the modesty rule).

[4] Whenever a function is supported by processors, processes, tasks, subprograms, or other form of sub-functional modules, the method of inter-connecting them shall obey the module decoupling rules proposed by Glenford Myers (the module decoupling rule).

[5] Similarly, every subfunctional module must follow Myer's module strength rules (the module strength rule).

[6] As well as the horizontal breakdown, a function must be

broken down vertically into layers. Levels of abstraction must be defined for each layer and independence between the layers must be observed (the layer rule).

GOMI, T.
A review of spacecraft fault-tolerant computer design concepts

P
91
C655
G64
1982