A MULTIPURPOSE COMPUTER FOR MARS SPACE MISSIONS

A. Balazs, S. Szalai, L. Varhalmi
Central Research Institute for Physics
Hungary Budapest 1525 Konkoly T. 29-33
P.O. Box. 49.

Our department has been dealing with developing different scientific instruments and computers used in space for years. Some instruments were made for VEGA and PHOBOS projects. Nowadays a modular on-board computer (MOBC) is under development.

The basic elements of the computer are the so-called processor modules. They are the real copy of each other, and contain all of the tasks to be performed by the computer in their own local program memory. The processor modules are connected together by two bidirectional serial buses, and further buses are used for controlling external instruments.

An error in any one of the modules has no catastrophic effect on the system, due to the loose connection between the modules. Except the memories, the modules themselves are not fault tolerant, the expected reliability is ensured on system level by having more modules in it. As far as reliability concerned, the computer itself is able to sense faults and to avert them allowing minimal interruption in operation. It is made by disqualifying the faulty module, either as a result of self-isolation or by majority voting of the correct ones. After dismissing of the faulty module, the system software reschedules the tasks in a flexible way among the other ones left. During operation the tasks save
their status into the other modules, so that the MOBC should continue correct operation even if a module fails. The ability of rescheduling makes possible to choose the most appropriate mode of operation (hot-, cold redundancy or both).

1. INTRODUCTION

As for a space computer, particular attention has to be paid to reliability, throughput, low power consumption, light weight, simple construction, modularity and price. Running through these items shows, that an actual realization of a computer for space application is a result of a compromise in most cases. This is because it is not easy to satisfy numerous requirements, being often contradictory, at the same time.

The MOBC is not a multiprocessor system in a "classical sense", where the major goal was to improve throughput. Our major intention is to develop a computer, of which reliability and throughput can be changed according to different demands, due to its modular construction.

2. THE MAJOR ELEMENTS OF MODULAR COMPUTER

a. Processor modules PM1,n
Apart from a jumper-set by which the individual module addresses are to be set, the processor modules are a real copy of each other. This statement refers both to the electrical circuits and to the system and user programs.

b. Processor Unit Buses PUB1,2
The data exchange between processor modules takes place on these serial, bidirectional, Manchester coded, redundant data links.

c. Serial Interface Buses SIB1,n
Data links from MOBC to the external world (experiments, other computers etc.). All of the processor modules are able to communicate on these serial, bidirectional, Manchester coded buses.

d. Isolation Circuit
The isolation circuit can isolate any one of the processor modules from the system both
temporarily and finally. The isolation is accomplished on the basis of majority voting.

![Diagram](image)

**Fig. 1** Modular on-Board computer

3. PROCESSOR MODULES

A processor module itself is not redundant, though in some functions it has redundant components as well (i.e. bus interface, memory error corrector). The reliability required is ensured on system level by having more processor modules in it, which are able to replace each other, and to reschedule the tasks among themselves. This of course does not mean, that the processor modules do not have to be reliable, otherwise losing of system-sources might be speeded up. The appropriate reliability can be achieved by applying qualified components (military and radiation hardened IC-s).

The memories (PROM, RAM) are LSI components and their proper operation is indispensible. At the same time, they are the most sensitive to the radiation in the outer space. Thus it is very desirable to take care of program and data protection. Triplication of memory chips needs considerably more components and higher power consumption, this is why a memory-error/detector corrector is used.
Error detection during operation is expected to be efficient and quick. To meet this expectation is supported by the error detection block on processor module level. Its task is to qualify the processor module itself. In principle it could be doubtful if the processor module itself is able to make such a decision, which regards to its own operation. Fortunately, a large number of faults can be recognized with the help of some simple hardware and/or software components. The error detector block must not only be efficient but also as simple as possible as well, otherwise it could badly influence the reliability of the processor module. Using more, simple methods to be principally different from each other, increases the range of recognizable faults and
the probability of recognition as well (i.e. memory error-detector -corrector, memory write attempt into the PROM field, hardware watchdog etc.). As a rule, there are some possible faults, which can not be recognized by the error detector block. See par.5., how to defend the system against such faults.

4. PROCESSOR UNIT BUSSES, BUS INTERFACES

The processor modules are connected together on PUB1,2 redundant buses. There is no priority between the buses, any one of the modules is allowed to transmit data (messages) to another module on both of them. The arbitration (how to share the access to the buses) is an important question.

To avoid different problems, the token-ring method is used in MOBC. The right for bus access (transmission) is rotating in the system module by module with the help of a simple, special Manchester word (token). Whenever a module needs the bus for transmission, it sets an internal hardware flag to signal this demand. As soon as the token arrives at, the module in question catches it. The circulation of token is stopped, until a message has been transmitted to another module. Immediately after the message having been transmitted, the token is issued again to the next module. The token can be issued just the same way as a normal message, so there is no need for extra wires between modules. An incoming token is passed to the next module immediately - without interrupting the local software -, if the hardware flag for bus request has not been set. The address field of token word contains the address of module which the right for usage of bus is just offered to. The token to be sent to the next module is kept in a hardware register in every module, so the address of the next module can be changed by the local program if necessary (i.e. reconfiguration of MOBC in case of malfunction or system initialization during cold-start).

4.1 Transmission on PUB

For higher reliability and maximal efficiency, there are two independent token-ring buses according to PUB1,2. A module does not necessarily have to be involved in the token
ring on both buses. In degraded state a module communicates only on one of the PUBs, because its other bus-interface has gone wrong.

The processor modules communicate with each other with messages. Once the demand for a PUB has been signalled and the token has arrived, the message must be sent with the maximal speed possible in order to increase bus availability for the other members (processor modules) of MOBC. This can easily be satisfied by a DMA circuit. Choosing of a PUB, which a message has actually to be transmitted to, is made by software. By raising the flag to acquire a PUB, the required bus has to be designated too. Three choices are possible:
- PUB1 if PUB2 has failed
- PUB2 if PUB1 has failed
- "ANY" the message is sent to the bus, the token has earlier arrived at.

4.2 Reception of messages

The first word of the messages must contain the address of the destination module, so that the unnecessary software administration and memory occupation in vain should be avoided. Thus a message is received by the only module addressed. The messages can easily be separated by simple hardware – that is, the beginning of the messages can be recognized – because the Manchester coder-decoder chip offers the possibility of distinguishing two types of words. The messages received are written directly into the local memory by means of DMA.

4.3 The length of messages

Because the messages contain their length as well, they may have different length, it is always known where to look up the next one in the memory.

Nevertheless care must be taken on choosing the maximal length of messages. The time gap, during which a module can not use the PUB, depends immediately on the maximal length of messages. Allowing too long messages "in one go" to transmit could badly influence the real-time behavior of MOBC. At the same time, an important condition of the correct operation of MOBC is that the token should arrive to a module within a maximal time limit, otherwise an alarm signal is generated.
5. THE SERIAL INTERFACE BUSES

The data exchange between the "outer world" and the MOBC takes place on the SIBs. The SIBs are serial bidirectional Manchester coded buses with the protocol according to the MIL-1553B standard. The number of redundant SIBs depends on the actual reliability requirements. All the processor modules are connected to all the redundant SIBs. Two features help the modules in using the SIBs with no conflict.

In one hand a special, very short message can be sent from a module to another one on PUB. It causes an immediate interrupt in the destination module, without entering into the queue of receiver buffer.

On the other hand a simple hardware monitors the SIBs in every module all the time, and an interrupt request is generated if the SIBs are quiet (i.e. the previous bus action of an external experiment has been completed).

6. FAULT DETECTION, RECONFIGURATION

6.1 Fault detection

The qualification of the MOBC is accomplished through different built-in tests. Two types of tests are implemented:
- self-test
- cross-test

The duty of self-test is to qualify every single processor module. It is performed by both software and hardware means. Immediately after a fault has been detected, the hardware error detector block of the processor modules (see par. 3.) generates an alarm signal. In addition, a special test task runs in each module, which keeps checking the status of the MOBC periodically. The test task has self- and cross-test functions as well. The self-test function checks those components of the local module, which the hardware error detector circuits are unable to check. For higher security they even overlap each other in some functions.

The cross-test is implemented in software. As it was written in par. 3., some faults can escape the self test or they are simply out of
its range. In addition, some components can reasonably be tested by more processor modules cooperating with each other (i.e. bus interfaces, PUBs). These types of faults can only be recognized by the cross-test, because the duty of cross test is to qualify other modules.

Taking into account that the self-test offers the fastest recognition of a fault, it is important to find the most reasonable division of labour between self- and cross-test. There is no need to force the cross-test to do such checks, which can be performed by the self-test effectively. So, the PUB bus-controllers and the components to be responsible for the self test are only to be tested in the other modules by the cross-test. These are:
- microprocessor
- the memory segments containing the self test programs
- to make sure, that the self-test program in the other modules runs properly. In this way, the two types of tests give a good coverage of possible faults.

6.2 Alarm

In case of malfunction an alarm has to be produced by one of the tests to all the members of the MOBC, so that a joint decision can be made (is any one of the modules to be isolated or not?). This can be performed with the help of the PUB. The module, which detects a fault first, provokes collision by continuous transmission to the PUBs deliberately. The collision is sensed in every module, at least in the correct ones.

6.3 Isolation methods

Owing to the effects of a faulty module on the system are unpredictable, it is necessary to ensure, that such a module shall not remain in the MOBC. Because a faulty module may disturb even the buses (PUBs, SIBs) it must be isolated from the correct ones (i.e. disabling its transmitters or using relay contacts).

There are two ways of isolation according to the nature of faults. The first one -the self-isolation- occurs whenever a fault can be
recognized by the self-test. Faults, which escape the self-test, must be detected by the cross-test. This time, the module broken down is unable to isolate itself. At the same time the MOBC could still have enough correct modules, which can.

6.4 Process of isolation

The first obvious solution for isolation could be, that every module (disregarding if it is still correct or not) sends the identifier of module supposed to be faulty to the isolation circuit, then it carries out the isolation. However, this method can not be realized in practice, simply because there is no guarantee, that the correct modules are able to determine which one of them has broken down, especially in the most serious situation, when a module ruins the PUBs suddenly.

The principle of another method, which eliminates the difficulties arisen by the first one, is the following: the modules are always able to judge, whether a faulty module exists in the MOBC in a certain moment or not. The isolation procedure begins with isolating a module temporarily. After doing it, the modules left accomplish a complete test. If these tests do not find erroneous component being in the MOBC, then the first step of the isolation process has been successful, in other words the faulty module has been found. After confirming the temporary isolation and rescheduling the tasks between the new members, the MOBC can go on its normal operation. If the modules have not succeeded in finding the faulty one, then the isolation process has to be continued by switching on the formerly dismissed module again and then by isolating the next one. This procedure is to be done step by step, until the erroneous module has been found. In order to prevent a correct module from being isolated by a faulty one, all of the decisions have to be made by majority voting (stepping of isolation scheme, temporary and final isolation). This method has disadvantages too. In one hand, the isolation process could be a time wasting procedure. On the other hand, the method is operational when there is only one erroneous module in MOBC at a moment. If more, then it can already be relied only on self-isolation.
6.5 Reconfiguration and cold-start

As it could be seen, after sensing any discrepancy in the normal operation, a collision on PUBs is provoked. In response to it, all of the members of MOBC enter into a special routine (called CONFIG, see fig. 3.), in which every module makes decisions for further actions. Until there is no possibility for collaboration between the modules, individual decisions are to be made (i.e. as far as the necessity of isolation concerned).

The special function of collision can be interpreted more general: whenever collision occurs, then there has been something change in the status of MOBC. The task of the CONFIG programs is to analyze this change individually and to decide what measures are to be taken. The following changes are possible:

a. A member of MOBC has gone wrong and it has isolated itself.
Accepting this fact, the modules left reschedule the tasks of MOBC among themselves again, then the operation is going on.

b. Same as a., however it can not isolate itself.
The isolation of the faulty module must be fulfilled by the correct ones.

c. Transient error has occurred.
Returning from CONFIGs, the modules continue their operation.

d. One (or more) new module has entered into the system.
A module, entering into the MOBC as a "newcomer" (i.e. power on), signals its presence to the others by provoking collision. In this way, the CONFIG programs can be used for managing the system cold-start, because as a result of general interpretation of collision, they cover practically those functions as well. If there is no collision detected within a watchdog time during cold-start, then the module in question is the only one being operated in the MOBC.

In order to avoid isolation of a module due to a transient error and to make possible entering "newcomers" into the MOBC (cold-start), the CONFIG begins with no immediate isolation. The isolation procedure can only be initiated after
any kind of irregularity has been found by the tests inside CONFIG too. If any discrepancy is sensed by these tests, then a collision is generated again and all of the modules go back to the beginning of CONFIG, but this time it starts with isolating the next module. Care must be taken, that any error, which was detected by the outer test task, shall not remain undetected inside CONFIG, otherwise the whole MOBC falls into an endless loop.

![Diagram of CONFIG program](image-url)
7. SUMMARY

As regards the number of modules being operated in a moment, the MOBC is flexible enough. It can operate with both hot and cold redundant modules in it at the same time. Particularly in a long term space mission, it has a special importance due to the fact, that the lifetime of a cold redundant component is considerably higher than that of a hot redundant one. In case of fault the MOBC can operate with or without degradation depending on the number of processor modules. The ability of resceduling gives the opportunity to choose the most appropriate mode of operation (hot-, cold redundacy or both).

REFERENCES

[1] B. Thompson e.a.: CRMMFCS and beyond, NAECOn IEEE 1986, 376-382
[3] I. Horvath e.a.: Highly reliable data acquisition system for space research, XIII. Symposium on Nuclear Electronics, Varna 1988