



## **PROCESSOR OBSOLESCENCE IN AVIONICS** \*

### **0 EXECUTIVE SUMMARY**

This paper was written in response to a request to prepare a discussion paper on processor obsolescence issues. It reviews the reasons for processor obsolescence and the subsequent consequences.

The paper considers briefly guidelines, which already exist, and how these fall short of the ideal. It then considers some of the techniques, which could be used to assist in mitigating against processor obsolescence.

The report concludes that guidance is needed in two contexts, one associated with existing systems where processor obsolescence is likely to become an issue and the other in the context of making systems more proof against future changes.

---

\* This report is published by the Avionic Systems Standardisation Committee (ASSC) to advance the role of standardisation in avionics. The use of this is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom is the sole responsibility of the user. Copies of this paper are obtainable from the ASSC Agency as an Avionic Systems Standardisation Committee publication.

<b>Table of Contents</b>	<b>Page no.</b>
<b>0 EXECUTIVE SUMMARY</b>	<b>1</b>
<b>1 INTRODUCTION</b>	<b>4</b>
<b>2 OBJECTIVE OF THIS PAPER</b>	<b>5</b>
<b>3 REASONS FOR PROCESSOR OBSOLESCENCE</b>	<b>6</b>
3.1 Introduction	6
3.2 Market Forces	6
3.3 Changing Requirements	6
<b>4 IMPACT OF PROCESSOR OBSOLESCENCE</b>	<b>7</b>
4.1 System Dependencies	7
4.2 Levels of Re-design	9
4.3 Conversion Skills	9
4.4 Re-qualification of the System and Software	10
4.5 Software Maintenance	10
<b>5 RELEVANT STANDARDS</b>	<b>12</b>
5.1 General	12
5.2 Def Stan 00-71	12
5.3 Def Stan 00-60	12
<b>6 PROCESSOR OBSOLESCENCE STRATEGIES</b>	<b>13</b>
<b>7 TECHNICAL SOLUTIONS</b>	<b>14</b>
7.1 General	14
7.2 Complete Software Re-development	14
7.3 Code to Code Translation	14
7.4 Assembler to HLL Translation	15
7.5 Processor Emulation	15
<b>8 CONCLUSIONS</b>	<b>16</b>

## Abbreviations List

HLL	High Level Language
WG	Working Group

## 1 INTRODUCTION

In recent years the computer industry has responded to demands for faster and more versatile systems by producing ranges of faster and more versatile processors. This has the benefit, to most users, that it is now possible to realise systems having functionality and performance that would have been inconceivable, only a few years ago.

An unfortunate consequence of this development is that there now exist many systems, which contain obsolete processors. In most cases, it is not possible to make a direct replacement of an obsolescent processor with a more up-to-date version, should a replacement become necessary. In the avionics industry, the problem is particularly acute since the life expectancy of an aircraft is typically of an order of magnitude greater than that of the processors, which may be used in the aircraft's control systems. Large costs may arise from the replacement of a processor by one of a different type, due to the redesign and re-qualification of the system as complex software may be involved.

## **2 OBJECTIVE OF THIS PAPER**

This paper has been prepared for the ASSC Systems Engineering Subcommittee (previously the Processing Working Group) to stimulate discussion on the topic of processor obsolescence and its effect on the avionics industry. In order to encourage creative thinking about ways of overcoming the problem, it is necessary to understand something of the reasons for processor obsolescence and the consequences of it. It is also necessary to understand the steps that have already been taken to tackle the problem and to understand their limitations.

Later sections of this paper give background material on the reasons for processor obsolescence and its impact on systems. There is discussion of some relevant standards and the contribution, which they currently make to achieving a solution. A discussion of strategy, planning and technical aspects of the solution then follows.

The overall objective of this paper is to provide sufficient background to readers to ensure that the general awareness of the issues is raised and to provoke thought as to how better solutions might be achieved. A number of possibilities is discussed, none of which provides a complete solution to the problem. The ideal outcome from future WG discussions would be to produce a set of guidelines as to how a cost-effective total solution might be achieved. Any progress towards this goal would be a useful contribution.

### **3 REASONS FOR PROCESSOR OBSOLESCENCE**

#### **3.1 Introduction**

There are two main causes of a particular system processor becoming obsolescent. The first of these relates to the fact that market forces push manufacturers into producing processors having ever more functionality and performance. The second cause relates to the fact that the requirements, which a complex system is designed to satisfy, are seldom static. They tend to increase and frequently require more processor power to satisfy them.

Some of the techniques, which have been discussed, help to achieve a solution in one or other of these scenarios. However, it is necessary to appreciate that a good solution will be effective in both. It is unsatisfactory to provide a solution, which assumes that all requirements are frozen with the first version of a system and will never change in the future.

#### **3.2 Market Forces**

There is great economic pressure to use industry standard processors and cards containing them. The cost of producing a special design is expensive and adds significantly to the cost of a system, unless it can be spread across a high volume production run. The corollary of this is that most systems use processors and cards, which are generic and whose features are tailored to their most common application. This may not be the application under consideration. If a large new application area emerges, manufacturers will normally bias the design of their products in order to meet the expectations of that market.

Manufacturers are aware of the problems of maintaining compatibility with their earlier products. However they have other demands placed on them, which are often stronger, associated with delivering increased functionality and performance at a lower cost. Backward compatibility is often sacrificed in the interests of cost-effectiveness.

#### **3.3 Changing Requirements**

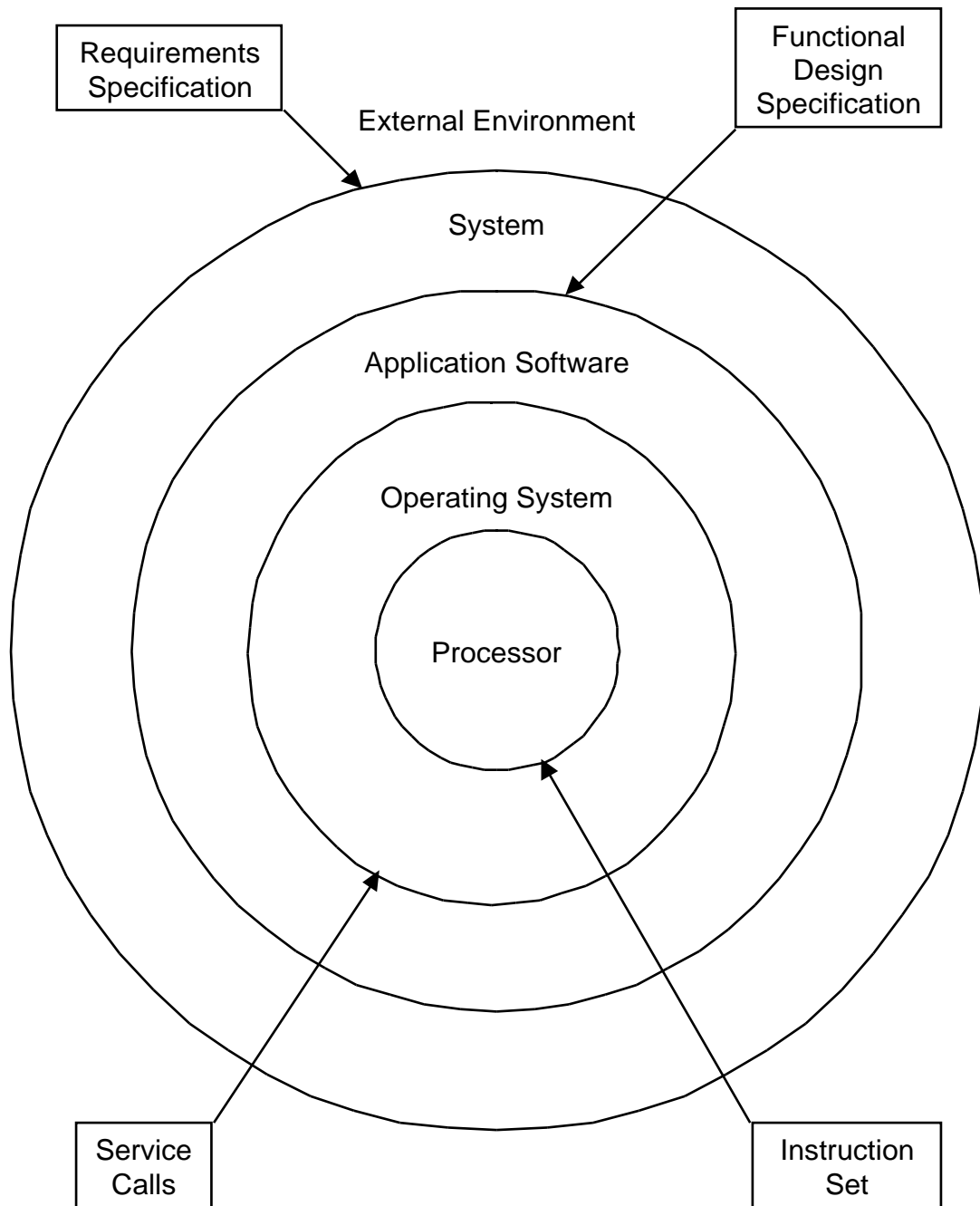
It is an illusion to believe that the requirements of a system will remain frozen for all time. Over an extended period it is likely that the environment in which the system operates will change and a solution to the new demands, which it places upon the system, will become necessary. This leads to a situation where the system must either be capable of being adapted to the changed environment or must be replaced.

The effective management of obsolescence has a contribution to make in either of these scenarios. Even if it is necessary to replace the whole system, because enhancements to it become non-viable, the systems design will almost certainly encapsulate a degree of thought as to how the various requirements inter-relate and as to how they may be satisfied (e.g. interface definitions, design philosophy, implementation detail). It is essential that this valuable information is not discarded without considerable thought.

## **4 IMPACT OF PROCESSOR OBSOLESCENCE**

### **4.1 System Dependencies**

In order to understand the impact of processor obsolescence, it is necessary to understand the main dependencies, which exist within a typical system. A typical structure is shown in figure 1.



**Figure 1 System Dependencies and Boundaries**

- A system exists to fulfil a need in the environment in which it operates. The system requirements specification may be considered as defining the boundary between the environment and the system.
- The system relies on a set of application software. The functional design specification may be considered as defining the boundary between the system and its application software.

- The application software relies on the services of an operating system. The set of operating system service calls may be considered as defining the boundary between the application software and the operating system.
- The operating system interacts with the processor. The processor instruction set may be considered as defining the boundary between the operating system and the processor.

This simple model will serve to illustrate some of the problems, which may result from processor obsolescence.

## **4.2 Levels of Re-design**

If it becomes necessary to replace a processor with one of a different type, the simplest action is to replace it with one having the same instruction set. This may involve some hardware re-design if the pin configuration has changed or different supply voltages are used, but there is a possibility that much of the original software could be re-used. Caution is needed, however, since the replacement processor is likely to operate at a different clock rate and the timing of asynchronous processes within the system may be altered.

If the new processor has a different instruction set, then re-compilation of all the software will be necessary. Furthermore a new operating system, which will operate with the new processor, must be chosen. If the new operating system provides the same or similar service calls to the old, it may be possible to avoid a major update to the application software, though each point where the application software makes an operating system service call will need to be checked for compatibility.

If a completely different operating system must be chosen, the system structure must be considered in detail. It may well be possible to preserve the structure when changing from one proprietary operating system to another, since most of them provide a similar range of service calls. This may not be possible, however, where a specific operating system has been designed to provide service calls relating to special hardware.

Finally, if the system structure cannot be preserved, it may be possible to design a new system to meet the requirements, if these have been captured in a consistent manner and maintained up-to-date.

## **4.3 Conversion Skills**

A major difficulty with updating a system, as a consequence of processor obsolescence, is that a considerable period of time may elapse between the original design and the update. During this period, it is possible that many of the skills and techniques used to implement the original design may have been forgotten. In order to perform the update, it will be necessary for personnel to be equipped with skills associated with both the new and the old environment. It may be difficult to acquire the old skills since it is quite possible that suitable training courses will no longer be available and the design documentation may be limited. The difficulty may be accentuated by the natural reluctance of companies and personnel to

spend time and effort acquiring skills, related to obsolete technology, which are unlikely to be used in the future.

#### **4.4 Re-qualification of the System and Software**

The ultimate aim in designing a system is to ensure that it will deliver all its functionality, correctly, whenever it is needed. In order to achieve this aim totally, it would be necessary to write perfect software and to build fault free hardware. Neither of these options is possible. However, for a system of any complexity, much effort is expended in eliminating potential problems with the aim of approaching the ideal asymptotically.

The common approach to this process is to test the hardware rigorously and to design the software so that it may be partitioned into a number of components having a few well-defined interactions between them. The components are tested in isolation, then assembled together and the complete system is tested. A number of methodologies are available for the systematic control of the process.

The process of eliminating errors can be very time-consuming and expensive due to its very nature. It is impossible to know, in advance, where errors will occur. As a consequence, it is necessary to investigate the behaviour of large amounts of software, most of which is probably correct, if good design principles have been followed, in order to detect a relatively small number of errors.

When a processor or part of a system is replaced, it is difficult to decide how much testing is necessary for re-qualification. If a change of processor is the sole modification, it may be sufficient to repeat all previous testing. If other system changes have been necessary, it could be that changes or additions to the tests will be necessary.

Because of the expensive nature of the testing process, there is always pressure to find more cost-effective alternatives. The problem of deciding how much testing is necessary after a change of processor is similar to the decision needed when enhancements are made to a previously qualified system. Decisions of this nature appear to be reached on an ad hoc basis, and it would contribute towards consistency if suitable guidelines could be prepared for use in such an eventuality.

#### **4.5 Software Maintenance**

Software requires maintenance. The maintenance is of a different nature to hardware maintenance and its need arises either because errors are discovered, because the requirements have changed or because practical experience of using the system suggests ways in which its behaviour could be optimised.

There are two possible consequences of maintenance for which it is necessary to introduce safeguards. Firstly, the new functionality, which it was the intention to introduce, may not behave correctly; secondly, the introduction of the new functionality may have caused

unwanted side effects. It is relatively straightforward to test for the first of these consequences but may be extremely difficult to test for the second.

If a change of processor has taken place, between the time of the original system design and the time that maintenance becomes necessary, the detection of unwanted side effects may become even more difficult. The insertion of a faster processor or more local memory will not eliminate latent problems associated with common resources; it may well just bury them deeper.

To compound the problem, much of the original design information may have been captured on an obsolescent system and personnel with the skills to manipulate the tools used on the original design may no longer be available. Even if the correct design environment can be re-created, there may be a future configuration control problem in combining information originating from two or more design environments.

## **5 RELEVANT STANDARDS**

### **5.1 General**

Def Stan 00-71 and Def Stan 00-60 both have a contribution to make towards dealing with the problem of processor obsolescence. Both mandate that early planning is necessary to avoid problems associated with obsolescence but restrict themselves to dealing with the generalities of the situation.

### **5.2 Def Stan 00-71**

Def Stan 00-71 provides a guide to managing obsolescence in general but specifically excludes the management of software obsolescence. It describes two main strategies for dealing with the problem, re-active and pro-active and defines circumstances where each is appropriate in terms of a triplet.

- The impact of equipment being unavailable due to a lack of spares or of performance degradation due to substituted parts.
- The likely cost of premature replacement or of other measures to circumvent obsolescence.
- The probability of obsolescence occurring.

Most avionics systems probably fall into the category where a pro-active strategy is recommended. Def Stan 00-71 makes recommendations as to what such a strategy should comprise.

Because of the intimate association of the processor with software, although the provisions of Def Stan 00-71 are relevant, they are not sufficient and further guidance is needed.

### **5.3 Def Stan 00-60**

Part 3 of Def Stan 00-60 provides guidance for application software support and describes in some details those elements of planning which are necessary for this activity. It contains much useful advice. However, the standard confines itself to dealing with application software, with the unstated implication that the processor and operating system will remain reasonably constant. When the processor/operating system changes, there are additional factors to be taken into account and guidance on the management of these aspects is needed.

## 6 PROCESSOR OBSOLESCENCE STRATEGIES

In contrast to the perceived actions of some manufacturers, who appear to have a policy of encouraging processor obsolescence in order to enhance the market available for their latest products, most users require a strategy to mitigate against the effects of obsolescence. In terms of the Def Stan 00-71 options, a pro-active strategy is required, since the alternative approach of waiting until a processor becomes obsolete before taking action is likely to lead to a long period, when the system containing the processor cannot be used, while the problem is solved.

Because of the extensive repercussions of replacing a processor with one of a different type, it is prudent to ensure that a stock of processors, sufficient to last the lifetime of the product, will be available (lifetime purchase). In addition to the processor, consideration must be given to other associated components such as memory and the integrated circuits associated with bus handling and timing.

In a static environment, it should be possible to make statistical predictions of the quantities of components that will be needed throughout the life of the system and to make provision for their acquisition. In avionics systems, there is a reasonable likelihood that the requirements will change to the extent that at least a partial system re-design will become necessary. In such circumstances, it may not be possible to use the types of components, including the processor, that were specified originally.

It is extremely difficult to forecast how the requirements placed on a system may change in future. As a consequence, any such forecast is not likely to be very accurate. Any strategy for dealing with the consequences of obsolescence must take this fact into account.

Plans for dealing with processor obsolescence need to be prepared in the early stages of the lifecycle of a system. One of the essential requirements of the system must be its operational life expectancy. Given the problems associated with changes of processors, the first aim must be to consider whether the environment in which the processor is used makes it practicable to avoid such changes altogether. As stated previously, the acquisition of a sufficient stock of processors and other components will be sufficient to maintain the system throughout its lifecycle in a static environment.

If changes to requirements cannot be avoided altogether, it is prudent to use processors in the system, which have considerable amounts of spare capacity at the design stage. Of course, there is a price to be paid for over-specification of the processor but this is usually small compared to the costs associated with changing the processor later in the system lifecycle. If it proves impossible to ensure that a particular processor has adequate spare capacity, consideration should be given to partitioning the tasks, which are run on highly loaded processors so that they may be run on lesser-loaded processors. There is a limit to how far it is possible to follow this path since, once a task is split between two or more processors, problems associated with task co-ordination are introduced. If many processors are involved, such problems may be difficult to solve.

## **7 TECHNICAL SOLUTIONS**

### **7.1 General**

Having taken such steps as were possible to avoid the need for a processor replacement and failed, what techniques may be brought to bear to ease the transition to the new processing environment? In view of the potential problems associated with processor replacement, every attempt should be made to minimise the extent of the changes. There is no known single technique, which will accomplish a pain-free transition alone. However, a number of possibilities exist which may have greater or lesser applicability depending on the particular circumstances. Some of the possibilities are

- Complete software redevelopment
- Code to code translation
- Assembler to HLL translation
- Processor Emulation

### **7.2 Complete Software Re-development**

One possibility for dealing with a processor change is a complete software re-development. If the original system has been subjected to numerous changes since its inception and the end product still has a reasonable life expectancy, this may be a viable option. Complete re-development is likely to be very expensive, of the order of the original development in real terms, though some economies may be possible, due to the fact that the existing requirements are captured in a single place, i.e. the original system. One of the benefits of total re-development is that the opportunity may be taken to produce a design using modern tools and to improve the standard of design documentation, easing the path for future enhancements. In this scenario, although a long development time may be necessary, it should be possible to plan and forecast it reasonably accurately.

### **7.3 Code to Code Translation**

If it is possible to select a new processor with a similar instruction set to the old, code to code translation of the software may be considered. If the instruction sets are grossly dissimilar, code to code translation may be extremely difficult, verging on the impossible, since the translation process takes no heed of the context in which the instructions are used.

In order to accomplish the process, a set of translator software will be needed. It is possible but unlikely that such software will be available from the new processor manufacturer, if demand is high enough, or from a third party, if the particular conversion becomes common throughout the industry. As a consequence, even though every opportunity to obtain a suitable translator should be explored, it is likely that it will be necessary to design a translator from scratch.

The design of a translator is not particularly difficult compared to some other software tasks but care must be taken to ensure that the code which is output from it behaves correctly in the new environment. Qualification may be an expensive and time-consuming process.

#### **7.4 Assembler to HLL Translation**

One of the problems with software, which was written some time ago, particularly if efficiency was a prime consideration in its design, is that it may have been written in an assembly language. Assembly languages tend to be very specific to the target processor on which the software is to run. It is possible to write a translator from one assembly language to another, much as it is possible to translate code, but problems may occur because the context is lost.

An alternative approach is to translate the assembler to a suitable high level language. This approach is not trouble free, since it is difficult to allocate meaningful variable names in the absence of the original context and the constructs implied by the assembler source may not be those which would be chosen according to good modern software engineering practice. Nevertheless a combination of automatic translation followed by editing, using modern tools, may produce source code which can be compiled to generate code suitable for running in the new processor environment.

#### **7.5 Processor Emulation**

Another possible technique, which may be useful, is to take a modern high-powered processor and use it to emulate an older lower powered processor. This approach ensures that the new processor uses up much of its available power in performing the emulation. This fact alone may not discount emulation, however the underlying system will still be expressed in obsolete code and future enhancements must be implemented similarly.

## 8 CONCLUSIONS

The problem of processor obsolescence merits consideration from two differing viewpoints. Firstly, a number of systems already exist which rely on a processor, which may become obsolescent. Secondly, the problem of processor obsolescence is likely to increase in future as the production span of processors decreases.

Standards already exist, which shed some light on the problems of managing for obsolescence. Additional provisions are needed to handle the problems of processor obsolescence due to the close connection between the processor and the software, which controls it.