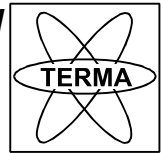


# ***Software System Development for Spacecraft Data Handling & Control Final Report***

---

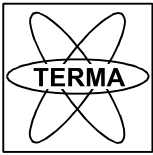
**Document No.:** *TERMA/OBOSS-2/TN/014*  
**Date:** *26.11.99*  
**Issue:** *1*  
**Revision:** *-*  
**Distribution:** *Public*  
**Prepared by:** *Ulrich Denskat, Dornier Satellitensysteme*  
*Håkan Svanberg, SAAB Ericsson Space*  
*Gert Caspersen, TERMA Elektronik*  
**Authorised by:** *Carsten Jørgensen, TERMA Elektronik*

The intellectual property of this document is vested in TERMA Elektronik AS.



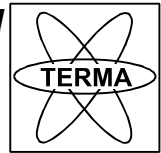
## ***Document Change Record***

<i>Issue</i>	<i>Date</i>	<i>Change</i>
1	26.11.99	Initial Issue

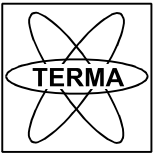


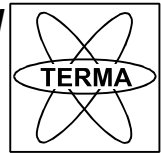
## **Table of Contents**

1	Introduction .....	1
1.1	Scope .....	1
1.2	Abbreviations and Acronyms .....	1
1.3	Document Outline .....	2
2	Bibliography .....	3
3	Definition of Proposed Data Handling System Architecture .....	4
3.1	Objectives .....	4
3.2	Findings .....	4
3.3	Documents .....	5
4	Analysis of OBOSS Software and Identification of Desirable Enhancements ...	6
4.1	Objectives .....	6
4.2	Findings .....	6
4.3	Documents .....	7
5	Design and Implementation of Proposed Data Handling System .....	8
5.1	Objectives .....	8
5.2	Findings .....	8
5.3	Documents .....	9
6	Integration and Demonstration of Proposed Data Handling System .....	10
6.1	Objectives .....	10
6.2	Findings .....	10
6.3	Documents .....	10
7	Clean-Up, Packaging, and Dissemination of Demonstration Package .....	11
7.1	Objectives .....	11
7.2	Findings .....	12



7.3 Documents .....	14
8 Analysis of OBOSS Concept Applicability to DSP-Based Instrument Control Units .....	15
8.1 Objectives .....	15
8.2 Findings .....	16
8.3 Documents .....	17
Appendix A	
Hard Real-Time Tool-set Evaluation .....	18





# **1 Introduction**

## **1.1 Scope**

This document is the final report for the ESA project ‘Software System Development for Spacecraft Data Handling & Control’ (OBOSS-II), contract number 12797/98/NL/PA.

The primary objectives of the project may be summarised as follows:

- *Re-engineering of OBOSS Baseline.* The existing baseline is not fully compliant with the ERC32 HRT development approach. Some restructuring is required.
- *Extension of set of Supported Packet Utilisation Standard Services.* New services and sub-services shall be added moving towards a complete Packet Utilisation Standard [PUS] implementation.
- *Use and Evaluation of ERC32 Tool-Set.* To increase the maturity of this tool-set, it shall be used for development, and response shall be provided as an evaluation.
- *Moving OBOSS Concept to Instrument Control Units.* The notion of application processes may be moved on to instrument control units, making these more autonomous supporting several Packet Utilisation Standard services.
- *Producing Operational Demonstration Scenario.* The entire feasibility of the approach has to be shown through production of one or more demonstration scenarios.

Although some redirection has taken place throughout the project lifetime, the overall objectives have been met.

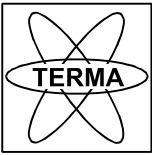
All outcomes of the project have been documented at the project home page situated at the following uniform resource locator (URL):

[http://spd-web.terma.com/Projects/OBOSS/Home\\_Page](http://spd-web.terma.com/Projects/OBOSS/Home_Page)

The remainder of this document will go through the work packages making up the project and outline the major findings and deliverables.

## **1.2 Abbreviations and Acronyms**

<i>COBS</i>	Central On-Board Software
<i>DSP</i>	Digital Signal Processor



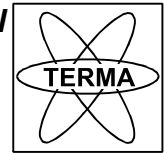
<i>ESA</i>	European Space Agency
<i>HOOD</i>	Hierarchical object-oriented Design
<i>HRT</i>	Hard Real-Time
<i>OBOSS</i>	Onboard Operations Support Software
<i>OBOSS-II</i>	Software System Development for Spacecraft Data Handling & Control
<i>PUS</i>	Packet Utilisation Standard
<i>SVF</i>	Software Validation Facility

### **1.3 Document Outline**

The document consists of one chapter for each work package carried out in the project<sup>1</sup>. Each chapter summarises the objectives and major outcome of the given work package and lists any documentation produced.

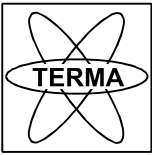
---

<sup>1</sup>: Work Package 7000: 'Management' has been left out as there is not much to summarise here.



## **2 Bibliography**

- [ADD]     *Data Handling System Architectural Design Document*  
TERMA/OBOSS–2/TN/011, Issue 1.0  
[http://sped-web.terma.com/Projects/OBOSS/Data\\_Handling\\_System\\_ADD.pdf](http://sped-web.terma.com/Projects/OBOSS/Data_Handling_System_ADD.pdf)
- [DCS]     *Data Handling System Concepts and Structure*  
TERMA/OBOSS–2/TN/013, Issue 1.-  
[http://sped-web.terma.com/Projects/OBOSS/Data\\_Handling\\_System\\_Concepts\\_and\\_Structure.pdf](http://sped-web.terma.com/Projects/OBOSS/Data_Handling_System_Concepts_and_Structure.pdf)
- [DOM]     *Domain Analysis*  
DO-OBOII-98-0001, Issue C  
[http://sped-web.terma.com/Projects/OBOSS/ObossII\\_domain\\_analysisB.pdf](http://sped-web.terma.com/Projects/OBOSS/ObossII_domain_analysisB.pdf)
- [DSRD]    *Data Handling System Software Requirements Document*  
TERMA/OBOSS–2/TN/010, Issue 1.C  
[http://sped-web.terma.com/Projects/OBOSS/Data\\_Handling\\_System\\_SRD.pdf](http://sped-web.terma.com/Projects/OBOSS/Data_Handling_System_SRD.pdf)
- [IADD]    *OBOSS Architectural Deviations for the ICU Software*  
DO-OBOII-98-0002, Issue A  
[http://sped-web.terma.com/Projects/OBOSS/Instrument\\_Control\\_Unit\\_ADD.pdf](http://sped-web.terma.com/Projects/OBOSS/Instrument_Control_Unit_ADD.pdf)
- [ISRD]    *OBOSS ICU S/W SRD*  
DO-OBOII-99-0003, Issue A  
[http://sped-web.terma.com/Projects/OBOSS/Instrument\\_Control\\_Unit\\_SRD.pdf](http://sped-web.terma.com/Projects/OBOSS/Instrument_Control_Unit_SRD.pdf)
- [MFR]     *Manual for Reuse*  
TERMA/OBOSS–2/TN/012, Issue 1.1  
[http://sped-web.terma.com/Projects/OBOSS/Manual\\_For\\_Reuse.pdf](http://sped-web.terma.com/Projects/OBOSS/Manual_For_Reuse.pdf)
- [PUS]     *Packet Utilisation Standard*  
ESA, PSS-07-101  
Issue 1, 1994



## **3 Definition of Proposed Data Handling System Architecture**

### **3.1 Objectives**

#### ***WP1100 : Analysis of Domain for Data Handling Systems***

This work package will extend the Domain Analysis done for OBOSS with respect to the inclusions of data handling information of additional missions. Furthermore some of the latest instrument developments shall be scrutinised, investigating the ICU system architectures and software features.

The impact of the use of the ERC 32 platform for data handling system and the 21020 DSP for instrument control units shall be analysed as well.

#### ***WP 1200: Definition of Data Handling System Architecture***

After an assessment of the OBOSS system architecture, a new system architectural scenario for data handling system, instrument control units and onboard communication shall be established.

The system requirements for the data handling system and instrument control unit scenario shall be established.

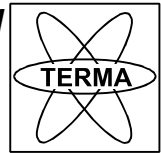
Another outcome of this work package shall be a draft for software requirements for a data handling system and an instrument control unit.

By refining the OBOSS architecture and regarding the system requirements, reference architectures for the data handling system and instrument control unit shall be specified.

### **3.2 Findings**

For WP 1100 eight parts of reference missions have been analysed, investigating system architectures and software features. Each part of the reference missions gives a description of the tasks and the environment of the data handling system under scrutiny for DHS and ICU systems. A functional description of the reference mission is translated into a 'PUS instantiation'. An assessment regarding a comparison of the mission functionality with the PUS was also done.

The information has been summarised in a table showing the number of occurrences of features in the reference missions. This analysis has shown that some reference missions - with respect to equivalent PUS functions - support less functionality than



the set of services provided by the present version of the OBOSS software. As a generic set of reusable software where single services can be selected in a 'shopping list' manner following the philosophy of the PUS, as much services as possible should be supported that can be selected in mission specific instantiations. The criteria for inclusion into the set of OBOSS modules should not be the necessity of the function in an onboard system, but the suitability for a generic, reusable implementation.

For WP 1200 are some subchapters in the Domain Analysis included which provide example architectural classes for the application of OBOSS in data handling and ICU domain.

Obviously, the application of one or the other architecture is dependent on the processor capabilities of a specific mission and the amount of data handling to be done within. All classes reflect the application of the generic software sets for one OBOSS data handling and one ICU instantiation. Other subsystems are added merely for illustration.

Another chapter contains the software requirements for data handling and ICU software following the architectural choice having been made. Being based on a PUS level discussion, the requirements are high-level demands that have to be refined in following work packages.

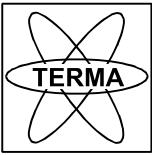
### **3.3 Documents**

*Domain Analysis*

DO-OBOII-98-0001

Issue C

[http://sped-web.terma.com/Projects/OBOSS/ObossII\\_domain\\_analysisB.pdf](http://sped-web.terma.com/Projects/OBOSS/ObossII_domain_analysisB.pdf)



## **4 Analysis of OBOSS Software and Identification of Desirable Enhancements**

### **4.1 Objectives**

This work package was to meet the following objectives:

- Identify required modifications to be carried out when moving the existing baseline to the ERC32 HRT tool-set.
- Define additions to existing collection of supported Packet Utilisation Standard services.
- Establishment of Software Validation Facility to be used for verification and presentation of the demonstration scenario.

The following section will elaborate on the actual outcome for each of these objectives.

### **4.2 Findings**

Applying the method inherent in the ERC32 HRT tool-set required changes to the existing architectural design and to the existing source code. Design and implementation had to be restructured as follows:

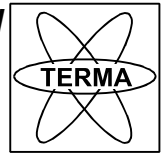
- Decomposition of objects shall follow the rules implied by the HRT HOOD method.
- Implementation of the objects shall follow the templates implied by the ERC32 HRT tool-set.

The above transformations were applied to a small subset of the system, and a structured approach was defined.

Packet Utilisation Standard services not yet supported by the implementation was scrutinised taking the outcome of the domain analysis (re. Chapter 3 'Definition of Proposed Data Handling System Architecture') into account. The following extensions to the baseline were selected for implementation<sup>2</sup>:

---

<sup>2</sup>: Support for interlocking was originally planned but eventually dropped due to effort re-distribution.



- Memory Management Service
- Invocation of onboard contingency procedures as response to out-of-limit events detected by the Onboard Monitoring service.

Design and implementation of these extensions were performed as part of work package 3000 'Design and Implementation of Proposed Data Handling System', see chapter 5.

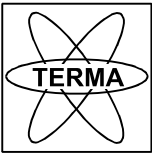
A combined scenario including data handling software and instrument control unit software was to be set up to show feasibility of approach (see chapter 6 'Integration and Demonstration of Proposed Data Handling System'). It was decided to establish a test set-up consisting of simulators for the ERC32 processor and the AD21020 digital signal processor. Provision of facilities for injection of telecommands and reception of telemetry was based on another ESTEC project named 'Reference SVF'. This provided a general software validation facility that allowed for the user to validate data handling software based on the Packet Utilisation Standard.

## **4.3 Documents**

*Data Handling System Software Requirements Document*

TERMA/OBOSS-2/TN/010, Issue 1.C

[http://sped-web.terma.com/Projects/OBOSS/Data\\_Handling\\_System\\_SRD.pdf](http://sped-web.terma.com/Projects/OBOSS/Data_Handling_System_SRD.pdf)



## **5 Design and Implementation of Proposed Data Handling System**

### **5.1 Objectives**

Three major tasks were performed related to the reusable data handling software baseline inherited from the 'Onboard Operations Support Software' project:

- Making the system HRT-HOOD compliant
- Porting existing 1750A implementation to ERC32
- Optimizing ERC32 implementation

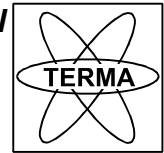
Results for each of these objectives are outlined below.

### **5.2 Findings**

Transforming the existing baseline into an HRT-HOOD compliant system required only a few, localized, changes to transform "hybrid" objects (and the corresponding Ada tasks) into pure cyclic and sporadic ones. The requirement for unique task priorities required changes due to non-staticness of generic parameters. An approach was taken in which priorities associated to tasks contained in generic packages are given as actual generic parameters. This is possible for 'normal' Ada tasks - through use of an Alsys specific library package - but poses a problem for protected objects implemented as 'passive tasks'.

Moving from the existing 16-bit 1750A platform to a 32-bit ERC32 platform hardly needed to be reflected in the code. Only very few compiler-related changes to Ada code had to be introduced. The total amount of source code being updated is estimated to be less than 200. The direct use of generics turned out to be very advantageous as no 'manual' source code expansion was needed.

Reusing the software on a real mission revealed performance problems with the existing implementation. Effort was put into optimizing the reusable software with respect to time. Identifying the primary bottlenecks in the system, these were eliminated one by one. Applying only a limited amount of resources (approximately 1-2 person months), the performance of selected software components was increased by a factor of five to ten. The statement that 'it is easier to optimize functionally correct code than it is to make optimized code functionally correct' was again confirmed.



## **5.3 Documents**

*Data Handling System Architectural Design Document*

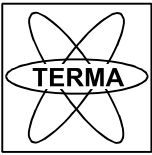
TERMA/OBOSS-2/TN/011, Issue 1.0

[http://sped-web.terma.com/Projects/OBOSS/Data\\_Handling\\_System\\_ADD.pdf](http://sped-web.terma.com/Projects/OBOSS/Data_Handling_System_ADD.pdf)

*Data Handling System Concepts and Structure*

TERMA/OBOSS-2/TN/013, Issue 1.-

[http://sped-web.terma.com/Projects/OBOSS/Data\\_Handling\\_System\\_Concepts\\_and\\_Structure.pdf](http://sped-web.terma.com/Projects/OBOSS/Data_Handling_System_Concepts_and_Structure.pdf)



## **6 Integration and Demonstration of Proposed Data Handling System**

### **6.1 Objectives**

This work package was to establish the infrastructure required to demonstrate a data handling system developed based on the reusable software components. This involved the following major tasks:

- Adaptation of existing 'Reference SVF' to support the selected demonstration scenario.
- Validation of OBOSS reuse concept through execution of the demonstration scenario.

The overall criterion for success was to be able to show a full functional data handling software system at the final presentation.

### **6.2 Findings**

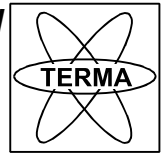
Establishment of the software validation facility posed some problems as the current project was the initial user of the system. The reference software validation facility supports validation of software adhering to the packet utilisation standard inasmuch as it allows for definitions of source packet formats based on the packet parameter types defined in the standard. There is however no baseline to start from implying that the complete source packet definitions had to be constructed from scratch.

The data handling software had already been subjected to test using a simple test harness and the ERC32 target simulator. As the reference software validation facility is based on this simulator as well, the transfer of the software to the new test environment only required a very limited amount of resources.

### **6.3 Documents**

The demonstration scenario is documented at the project home page at:

[http://sped-web.terma.com/Projects/OBOSS/Home\\_Page](http://sped-web.terma.com/Projects/OBOSS/Home_Page)



## **7 Clean-Up, Packaging, and Dissemination of Demonstration Package**

### **7.1 Objectives**

The work package falls into two major parts:

- Demonstration of combined Data Handling System and Instrument Control Unit concepts.
- Evaluation of ERC32 hard real-time tool-set.
- Third party evaluation of approach for reuse.

Each of these is elaborated below.

#### ***Demonstration of DHS/ICU Concepts***

Coming to the end of the project, this work package is concerned with evaluation and demonstration of the applied approach:

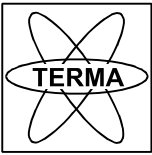
- Development of Manual for Reuse and project homepage for distribution of results.
- Adaptation of existing software validation facility to support combined scenario including data handling software *and* instrument control unit software.
- Integration of instrument control unit software and data handling system software.
- Validation of concept through execution of the demonstration scenario.

#### ***Evaluation of ERC32 Hard Real-Time Tool-Set***

Transfer of the reusable data handling software components to the new platform was based on the ERC32 hard real-time tool-set. A summary of the experiences gained from use of these tools shall be provided.

#### ***Third Party Evaluation of Approach for Reuse***

The work package WP 5300: Third Party Evaluation of Approach for Reuse comprises the following main tasks:



- To address the economic implications of reusing the OBOSS (architecture and components) compared with an in-house development of a data handling system (from scratch) based on the Packet Utilisation Standard (PUS).
- To investigate the effect OBOSS would have had on the software development in a data handling system (COBS) already developed by Saab Ericsson Space.

## **7.2 Findings**

### **Demonstration of DHS/ICU Concepts**

A 'Manual for Reuse' has been written for the reusable data handling software components. Taking the re-user through the process step by step, it provides exact indications of the source code modifications required to adapt the software for a specific mission. The electronic version found at the project home page includes hyper-text links to Ada 83 source code.

All documentation produced during the project is available at the project home page having the following uniform resource locator (URL):

[http://sped-web.terma.com/Projects/OBOSS/Home\\_Page](http://sped-web.terma.com/Projects/OBOSS/Home_Page)

The home page has been developed and hosted by TERMA Elektronik AS. It provides a structured introduction to the project and provides a starting point for exploration of the project and its results.

Integration of the instrument control unit software and the data handling system software had to be based on shared files as the AD21020 simulator was running on a personal computer and the ERC32 target simulator was running on a workstation. Except for a very poor performance, simulation of a serial bus was achieved.

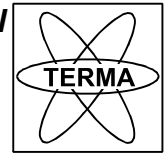
Validity of the overall concept was shown through execution of a demonstration session in which packet utilisation standard services were requested in the data handling software and in the instrument control unit software. A similar demonstration was given at the final presentation.

### **Evaluation of ERC32 Hard Real-Time Tool-Set**

The overall conclusion is that the ERC32 hard real-time tool-set does the job. Porting of the existing baseline of data handling software was completed within the amount of resources foreseen in the contract. However, there is a difference between the scope of the current project and a 'real' operation onboard software development. It is the belief of the software development team, that the tool-set could support a 'real' development as well, but no solid evidence can be given to support this.

There are two cardinal points in the evaluation:

- Some of the tools – especially the HRT-HOOD Nice tool – are not as mature as could be expected for an operational onboard software development. This is due to the limited amount of users and is likely to improve over time.
- Cost of the tools – and especially of the associated maintenance – is excessive compared with what is seen in the domain of hard real-time tools and Ada. A reduction of 50% on the maintenance price would be more



fair and desirable..

A detailed evaluation of the tools is enclosed in Appendix A: 'Hard Real-Time Tool-set Evaluation'.

### ***Third Party Evaluation of Approach for Reuse***

Saab Ericsson Space has performed a "Third Party Evaluation of Approach for Reuse" comprising the following main tasks:

- To address the economic implications of reusing the OBOSS (architecture and components) compared with an in-house development of a data handling system (from scratch) based on the Packet Utilisation Standard (PUS).
- To investigate the effect OBOSS would have had on the software development in a data handling system (COBS) already developed by Saab Ericsson Space

Cost savings are dependent on many factors: how well OBOSS covers the needed PUS functionality, the amount of new units to be developed, how much of OBOSS to be modified or adapted, availability of test software, the quality and availability of documentation, etc.

Some risks are identified when reusing OBOSS. Performance objectives (size and time) could be difficult to achieve without any modifications of OBOSS.

For the best benefit from OBOSS, the required PUS capabilities should be close to what is supported by OBOSS. This should be considered early in the system definition. Thus, to "maximise" cost savings, the following is valid:

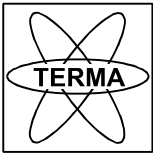
- The customer must "harmonise" his specification to PUS and OBOSS.
- The activities required for the reused software must be clearly defined before the start of the project.

Design and Coding are probably those activities that contribute most to the cost savings when reusing software. Integration testing is also a candidate of significant cost savings, given certain prerequisites (e.g. documentation).

With respect to the hypothesis of building new COBS complying with PUS capabilities and harmonising with OBOSS, the study has determined the following:

In short, OBOSS would be expected to reduce the amount of own developed software by 35-40% of total software in our example project, but also itself contributes with somewhat more reused software such as buffer management and templates for cyclic and sporadic control flows. Reusing these into the development as well would result in a further saving that has not been quantified within the scope of the current project.

All together, using OBOSS would allow for a 20% cost saving in the example COBS/PUS project, disregarding the cost of OBOSS itself. This is to be considered a ceiling for the COBS/PUS-system, as the estimated saving would be eroded by less suited project conditions.



## **7.3 Documents**

*Manual for Reuse*

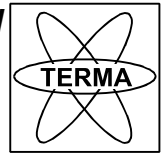
TERMA/OBOSS-2/TN/012, Issue 1.1

[http://spd-web.terma.com/Projects/OBOSS/Manual\\_For\\_Reuse.pdf](http://spd-web.terma.com/Projects/OBOSS/Manual_For_Reuse.pdf)

*Third Party Evaluation of Approach for Reuse*

D-D-REP-0006-SE, Issue 1

[http://spd-web.terma.com/Projects/OBOSS/Third\\_Party\\_Evaluation.pdf](http://spd-web.terma.com/Projects/OBOSS/Third_Party_Evaluation.pdf)



## **8 Analysis of OBOSS Concept Applicability to DSP-Based Instrument Control Units**

### **8.1 Objectives**

#### **WP 6100: Selection of PUS Services Applicable to ICUs**

The PUS services implemented for the OBOSS instantiation shall be assessed regarding their applicability for the Instrument Control Unit Domain.

A subset of the services available from OBOSS shall be identified which shall form the instantiation baseline on the ICU demonstration system. For the chosen set of PUS services, software requirements and subsystem test cases shall be identified.

#### **WP 6200: Definition of Architectures for ICU software**

The architecture of the command and data handling system of OBOSS shall be reviewed regarding its applicability for the ICU domain. Considering also the specification of the ICU architecture from WP 1200 the final architecture specifying the hardware and software demonstration system to support the PUS on ICUs based on a TSC 21020E DSP shall be established.

#### **WP 6300: Design and Implementation of ICU Software Validation Facility**

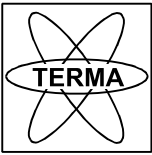
Specification and implementation of a simple software validation facility based on a TSC 21020 E DSP.

#### **WP 6400: Detailed Design and Implementation of ICU Software Architecture**

The architecture in WP 6200 shall be implemented by adapting the OBOSS architectural design to fulfil the requirements identified in the earlier work packages. The OBOSS implementation has to be ported from Ada83 to C and adapted to obtain the architecture defined in WP 6200. The resulting adapted and ported software shall be subsystem tested.

#### **WP 6500 : Porting of Selected PUS Services to ICU Platform**

The OBOSS generic software has to be instantiated covering the PUS services selected as candidates for the ICU implementation.



The (ICU mission representative) demonstration scenario parameters have to be incorporated in the OBOSS instantiation.

The instantiated software shall be ported to C.

The platform specific OBOSS components must be adapted to be able to run on a TSC 21020E DSP.

Subsystem tests shall be executed.

### ***WP 6600: Execution of Acceptance Test on ICU SVF***

Execution and documentation of the acceptance test for the OBOSS ICU demonstration instantiation.

## **8.2 Findings**

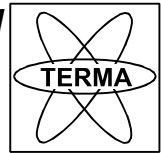
As for the data handling part, the identification of the set of supported services to be included into the OBOSS ICU software derivative should also consider the provision of the join of sets of PUS similar functionality available in the reference missions. The ideal objective for an OBOSS derivative ported to the ICU domain should be to achieve the coverage of the following functions:

- Telecommand Verification Service
- Housekeeping and Diagnostic Data Reporting Service
- On-Board Monitoring Service
- Event Reporting Service
- Function Management Service
- On-Board Scheduling Service
- Device Level Commanding Service
- Time Reporting Service
- On-Board Traffic Management Service

Task Management, Memory Management and Test services are considered too mission specific to be included in the generic set. The inclusion of On-Board Traffic Management and Time Reporting in conjunction with feasibility and the fact of basing on the present OBOSS data handling instantiation votes for the application of the OBOSS Packet Router concept for the ICU domain as well.

Software requirements for the set of supported services have been made in the SRD.

A major aspect for the assessment is the fact that instrument control is often less complex in comparison to a data handling software. Usually fewer functions are available and the amount of data to be handled is smaller as well. Therefore the general statement was that for most of the real projects a division of the control services into different allocated Application Processes would not be necessary.



Thus for large ICU systems having many parameters to be controlled, a structuring concept like allocating APs with instantiated onboard services for each APID could be an applicable and useful design item.

For the project demonstration a DSP simulator will reside as a process on a PC. The interaction with the data handling system will be simulated by memory mapped I/O ports. The memory mapped ports will be physically represented by files on the host for the SVF testing the data handling software. The simulator will access the I/O files by mapping the hard disk of the SVF host as a network drive.

For the WP 6300-6600 we have used the operating system Virtuoso as the basis for the OBOSS modules ported to the DSP. Virtuoso provides valuable features like libraries for tasks and semaphores that can be used in the implementation. This is the only way to port the OBOSS cyclic and sporadic tasks without obvious changes to the present implementation concept.

With respect to the feasibility of a combined demonstration with the SPARC SVF I we have not seen any alternatives to the use of a DSP simulator, given the resources for the establishment of the interface. A MOSAIC board was no option at all even if it would be the most impressive regarding the demonstration. The purchase of such a board would have eaten up a major part of our resources available for OBOSSII even when being manufactured on the lowest quality. Additionally the board only comprises IEEE 1355 high speed serial communication links which we cannot use for the establishment of a communication with the SPARC SVF.

## **8.3 Documents**

### *OBOSS Architectural Deviations for the ICU Software*

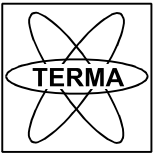
DO-OBOII-98-0002, Issue A

[http://sped-web.terma.com/Projects/OBOSS/Instrument\\_Control\\_Unit\\_ADD.pdf](http://sped-web.terma.com/Projects/OBOSS/Instrument_Control_Unit_ADD.pdf)

### *OBOSS ICU S/W SRD*

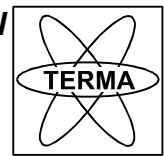
DO-OBOII-99-0003, Issue A

[http://sped-web.terma.com/Projects/OBOSS/Instrument\\_Control\\_Unit\\_SRD.pdf](http://sped-web.terma.com/Projects/OBOSS/Instrument_Control_Unit_SRD.pdf)



## ***Appendix A***

# ***Hard Real-Time Tool-set Evaluation***



This appendix is to provide an evaluation of the ERC32 hard real-time tool-set used for the transfer of the reusable data handling software components to the ERC32 platform.

### ***HRT-HOOD Tool (HRT-HoodNICE)***

Since the HRT-HoodNICE tool is developed by Intecs Sistemi (Italy), which has been supplying HOOD tools for several years, the tool facilities supporting the basic HOOD concepts are adequate and stable, as one would expect.

The tool allows the user to specify a document skeleton through their own specification language (DSDL). This gives the user great flexibility in choosing the parts of a design that should be included in the document, e.g. based on selection criteria. Attributes for objects can be added and is supported by the document generator. Both latex TPS and framemaker document generation are supported (unfortunately not working right though).

The concept of "shared objects" is quite useful for a system such as OBOSS, but unfortunately the mechanism is very inflexible, in particular the fact that a shared object referenced by other objects cannot be changed renders the mechanism virtually useless during development, where objects typically change (also the shared ones).

The internal database used by the tool may get corrupted meaning that export and print of a project is not possible. Making a copy of the complete project eliminated the problem in the new copy.

Since the goal of the HOOD SIF (Standard Interchange Format) is to exchange HOOD designs, even among potentially different HOOD tools, it was disappointing that sometimes exporting a project in SIF format and later importing it, resulted in an error, which meant that the import failed.

When the user violates certain rules, the tool simply crashes with an "internal error".

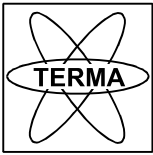
The maintenance quality was quite low. The reaction to reported errors was slow. We never got any bug fixes during the entire project even though we had a maintenance agreement, and moreover no usable work-arounds were suggested.

### ***Ada Compiler (Aonix)***

Compared to experiences with compiling similar code using a TLD cross compiler for a 1750A platform, the Aonix Ada compiler had no special limitations or errors that would require work-arounds at the Ada source code level, which was the case with the TLD compiler when for example non-trivial generics was used.

The Aonix Ada compilation system includes several utility packages that are not required by the Ada standard, which are quite useful in developing onboard software that uses tasks and generics. In particular the Real\_Time, Dynamic\_Priorities and Task\_IDs packages were utilised by the OBOSS code.

When using the compiler without optimisations, the code quality was not impressive, in particular many checks were generated that needed not be performed due to the



statically determinable context in which an operation was to be performed, e.g. array indexing within an attribute-based (Range) for-loop.

Explicitly requested extensive optimisations (by compiler options) often result in failing compilation when nested generic instantiations were involved.

Lack of an Exception\_Name construct (a function returning the name of the current exception in an "others"-handler) meant that tasks dying of an exception could not feasibly report the causing exception.

The price for maintenance is quite high. A reduction of maintenance cost by 50% would place it in the price range normally seen in the Ada 83 compiler market.

### ***Ada Debugger (Aonix)***

The debugger provided the functionality that one would expect from an Ada cross debugger.

Combining the Ada debugger with the ERC32 target simulator was relatively easy, and the resulting combination provided an effective development platform.

The OBOSS software in several places contains instances of generic units within other generic units. Setting breakpoints in such inner instances was quite cumbersome, since prior to execution start, the source code is only available for a library-level instance. Therefore, one must dynamically "step into" an operation of the inner instance to be able to set a breakpoint somewhere in that unit.

Sometimes the debugger refused to "continue" or "step" after a breakpoint was hit<sup>3</sup>, which was particularly annoying when one had finally managed to set a breakpoint in a place that was difficult to reach (cf. the preceding paragraph).

### ***ERC32 Target Simulator***

Basing the simulator on the Tcl/Tk command language provides a very good baseline for user adaptations and extensions. Environment simulations based on Tcl/Tk scripts and markers in the code worked out well. The set of simulator commands available for Tcl/Tk scripts are however limited to a subset of the simulator commands. Providing the entire set would have allowed for complete control of the simulations.

Using coverage bits is cumbersome. Summarising the areas being covered should be possible. Support for establishing the relationship to the compilation units would also be helpful.

The price is quite high compared with state-of-the-art simulators for other processors. Again a 50% price reduction would be in place. However, maintenance is provided at no cost.

Performance is acceptable, but it should be improved, since the simulated time is still approximately 10 times that of the real elapsed time, when executing on a Sun Ultra 4.

---

<sup>3</sup>: This is a documented error in the integration between the Ada Probe debugger and the ERC32 target simulator.