Contract No: 9848/92/NL/FM

PROJECT

32-BIT MICROPROCESSOR AND COMPUTER DEVELOPMENT PROGRAMME

TITLE

FINAL REPORT

	Name	Function	Date	Signature
Prepared :	Mikael Ramström	ERC32 Project Engineer		
	Jonas Höglund	ERC32 Product Specialist		
	Björn Enoksson	ERC32 Product Specialist		
	Ritva Svenningsson	ERC32 Project Manager		
Checked :	Mats Steinert	Head of Computer Software Section		
	Torbjörn Hult	Chief Engineer Computer Systems		
Authorized :	Ritva Svenningsson	ERC32 Project Manager		

Distribution
Complete : ESTEC B2B B2L B2O B2P B2S B5B T/K T/KP
T/C T/CK T/CK-MR T/CL T/CM T/CP T/CS T/CS-AC T/CS-RJ T/CV
T/D T/DP T/DP-BE T/DS T/DS-RJ T/DS-ÅJ T/DS-IM T/DS-OM T/DS-MN T/DV

Summary :

Reg. Office:
Saab Ericsson Space AB
S-405 15 Göteborg
Sweden
Reg. No: 556134-2204

Telephone: +46 31 35 00 00
Telefax: +46 31 35 95 20

Mölndal Office:
Telephone: +46 31 67 10 00
Telefax: +46 31 67 38 66

Linköping Office:
Saab Ericsson Space AB
S-581 88 Linköping
Sweden
Telephone: +46 13 28 64 00
Telefax: +46 13 13 16 28

Class : Host System : Word 2.0c for Windows, SE Macro Rev 2.0
Contract No : Host File : \\NTSRV1\ARKIV1\MCD\MNT0015_01.DOC

SUMMARY

This 32-bit Microprocessor and Computer Development programme has developed an infrastructure of sophisticated hardware components and software tools to support the development of 32-bit on-board computers for future space applications.

The hardware development resulted in a computer core (ERC32) which was built of three chips, i.e an Integer Unit, a Floating Point Unit and a Memory Controller. The cor is based on SPARC V7 architecture, including all functionality required to form an embedded 32-bit on-board computer. By adding only external memory and required I/C devices a complete system is configured. I.e it is easy to design systems with the ERC3 core.

The software development has resulted in a suite of tools, the “32-bit toolset”, which support the user in the development and verification of hard real-time software written i the Ada programming language. The toolset supports the complete Hard Real-Time software development lifecycle and consequently, can be used from the start of a programme, for example in the prototyping and conceptual design, through to the final verification and validation of the application.

DOCUMENT CHANGE RECORD

Changes between issues are marked with a left-bar.

Issue	Date	Paragraphs affected	Change information
Draft 1	14 Feb 1997	All	New document
Draft 2	14 Mar 1997	All	Updated draft
1	27 May 1997	All	First issue

TABLE OF CONTENTS Page

1. EXECUTIVE SUMMARY 7
1.1. Hardware Development 7
1.2. Software Development 8

2. INTRODUCTION 10
2.1. Scope 10
2.2. Background 10
2.3. Objective 11
2.4. Project Organization 13
2.5. Project History 15
2.5.1. Hardware Development 15
2.5.2. Software Development 16
2.6. Acknowledgements 18
2.7. Documents 19
2.7.1. Applicable Documents 19
2.7.2. Reference Documents 20
2.8. Abbreviations 20

3. DEVELOPMENT OF THE ERC32 HARDWARE 22
3.1. Description of the work performed 22
3.1.1. Work Breakdown Structure 22
3.1.1.1. Phase 1 22
3.1.1.1.1. Technology Selection 22
3.1.1.1.2. System Definition 23
3.1.1.1.3. System Detail Design 23
3.1.1.1.4. System and Component Level Simulation 24
3.1.1.1.5. Device Development Plan 24
3.1.1.2. Phase 2 25
3.1.1.2.1. Device Detail Design 25
3.1.1.2.2. Device Manufacturing 26
3.1.1.2.3. Demonstration Breadboard Definition and Manufacturing 26
3.1.1.3. Development Finalisation Phase 27
3.1.1.3.1. Electrical and Radiation Characterization 27
3.1.1.3.2. System Testing and Benchmarking 27
3.2. Design, Verification and Implementation 28
3.2.1. Design 28
3.2.1.1. Design Methodology 28
3.2.1.2. IU Design and Layout 28
3.2.1.3. FPU Design and Layout 29
3.2.1.4. MEC Design and Layout 30
3.2.1.5. Design Tools 31
3.2.2. Verification 31
3.2.2.1. Verification Methodology 31
3.2.2.2. VHDL Model Verification 31
3.2.2.3. Design Database Verification 33

3.2.2.3.1. IU and FPU Circuits 33
3.2.2.3.2. MEC Circuit 33
3.2.2.4. Characterisation 34
3.2.2.5. Verification Tools 34
3.2.3. Implementation 34
3.3. Problems Encountered 35
3.3.1. IU Circuit 35
3.3.2. FPU Circuit 35
3.3.3. MEC Circuit 36
3.3.4. DEM32 Board 37

4. ACHIEVED RESULTS FOR THE ERC32 HARDWARE 38
4.1. Description of the ERC32 Chipset 38
4.1.1. Functionalities 38
4.1.1.1. Integer Unit (IU) 39
4.1.1.1.1. Basic Functions 40
4.1.1.1.2. Concurrent Error Detection Capabilities 40
4.1.1.1.3. Concurrent Error Handling Capabilities 41
4.1.1.2. Floating-Point Unit (FPU) 41
4.1.1.2.1. Basic Functions 41
4.1.1.2.2. Concurrent Error Detection Capabilities 41
4.1.1.2.3. Concurrent Error Handling Capabilities 42
4.1.1.3. Memory Controller (MEC) 42
4.1.1.3.1. Basic Functions 42
4.1.1.3.2. Concurrent Error Detection Capabilities 43
4.1.1.3.3. Concurrent Error Handling Capabilities 43
4.1.2. Performance 43
4.1.3. Electrical Characteristics 44
4.1.4. Radiation Harness 45
4.1.4.1. Total Dose 45
4.1.4.2. Electrostatic Discharge (ESD) 46
4.1.4.3. Latch Up 47
4.1.4.4. Single Event Upset (SEU) 47
4.1.5. Known Limitations and Bugs 49
4.1.5.1. IU Circuit 49
4.1.5.2. FPU Circuit 49
4.1.5.3. MEC Circuit 50
4.1.5.4. ERC32 System 50
4.2. Use of the ERC32 Chipset 51
4.3. Availability of the ERC32 Chipset 52
4.4. Future improvements for the ERC32 Chipset 53

5. DEVELOPMENT OF THE ERC32 SOFTWARE 54
5.1. Description of the work performed 54
5.1.1. Work Breakdown Structure 54
5.1.1.1. Phase 1 54
5.1.1.1.1. Overall Co-ordination of Tool Specification 54
5.1.1.1.2. Specification of Software Tools 55

5.1.1.1.3. Design and Development Plan for Tools 55
5.1.1.1.4. Test Plan for Tools 55
5.1.1.2. Phase 2 and Development Finalisation Phase 55
5.1.1.2.1. Ada Compiler System and Debugger 57
5.1.1.2.2. HRT Support Tools 57
5.1.1.2.2.1. ERC32 Target Simulator 58
5.1.1.2.2.2. Schedulability Analyser 59
5.1.1.2.2.3. Scheduler Simulator 59
5.1.1.2.3. SW Support Environment User Guide 60
5.1.1.2.4. Overall Co-ordination and Management 60
5.1.1.2.4.1. Software Management 60
5.1.1.2.4.2. Target Simulator Consultancy 61
5.1.1.2.5. Acceptance of Tools 61
5.1.1.2.5.1. Tool Acceptance 61
5.2. Design, Implementation and Verification 61
5.2.1. Design and Implementation 61
5.2.1.1. Design methodology 61
5.2.1.2. Design Decisions 62
5.2.1.2.1. ACS 62
5.2.1.2.2. ERC32 Target Simulator 63
5.2.1.3. Schedulability Analyser and Scheduler Simulator 65
5.2.2. Verification 66
5.2.2.1. Ada Compilation System 66
5.2.2.2. ERC32 Target Simulator 66
5.2.2.3. Schedulability Analyser and Scheduler Simulator 67
5.3. Problems Encountered 68

6. ACHIEVED RESULTS FOR THE ERC32 SOFTWARE 70
6.1. Description of the ERC32 Toolset 70
6.1.1. Functionalities 71
6.1.1.1. Ada Compilation System 71
6.1.1.2. Hard Real-Time Tools 73
6.1.1.2.1. Schedulability Analyser 75
6.1.1.2.2. Scheduler Simulator 76
6.1.1.3. ERC32 Target Simulator 76
6.1.2. Performance 77
6.1.2.1. ACS benchmarks 77
6.1.2.1.1. PIWG 78
6.1.2.1.2. JIAWG 80
6.1.2.1.3. ESTEC-B 80
6.1.2.1.4. ERC32 Target Simulator 81
6.1.3. Known Limitations and Bugs 81
6.1.3.1. ACS 81
6.1.3.1.1. AlsyMonitor 82
6.1.3.1.2. User Interface 82
6.1.3.1.3. IEEE 754 Exceptions 82
6.1.3.1.4. AdaProbe 82
6.1.3.2. ERC32 Target Simulator 83

6.1.3.3. Schedulability Analyser 83
6.1.3.4. Scheduler Simulator 83
6.2. Use of the ERC32 Toolset 84
6.3. Availability of the ERC32 Toolset 86
6.4. Future Improvements for the ERC32 Toolset 87

7. ERC32 SYSTEM 89
7.1. ERC32 System Overview 89
7.2. Foreseen Applications 90
7.2.1. ERC32 Chipset in Fault Tolerant Systems 91
7.2.2. ERC32 Toolset in Hard Real-Time Applications 92

8. CONCLUSIONS 94

APPENDIX 1 - Generated documents 95

1. EXECUTIVE SUMMARY

The main objective of this 32-bit Microprocessor and Computer Development Programme was to develop an infrastructure of sophisticated hardware components and software tools to support the development of 32-bit on-board computers for future space applications.

The programme was managed by Saab Ericsson Space.

1.1. Hardware Development

The hardware was to be based on a widely supported, open architecture available without license requirements. The key characteristics were focused on performance, concurrent error detection and handling capabilities as well as on radiation hardness technology and power consumption.

The hardware development resulted in a 32-bit Embedded Real-time Computing core (ERC32) which is based on SPARC V7 architecture amended with built-in error detection and handling capabilities. The amendments are implemented mainly by parity checking logic for internal registers and for external data, address and control buses.

The ERC32 is built of the following three chips:

- Integer Unit (IU), the IU is the main computing engine of the ERC32 chipset. The IU is based on 32-bit RISC architecture, defining execution at a rate of approaching one instruction per clock cycle.

- Floating Point Unit (FPU), the FPU is a high-performance, single-chip implementation of the SPARC reference floating-point unit. The FPU executes single and double-precision floating-point instructions concurrent with execution of integer instructions by the IU. The FPU is compliant with the ANSI/IEEE-754 floating-point standard.

- Memory Controller (MEC), the MEC interfaces the IU and the FPU to external memory and I/O units thus forming a system, with which computers for on-board embedded real-time applications can be built. In order to achieve this the MEC constitutes all necessary support and on-chip resources.

The core includes all functionality required to form an embedded 32-bit on-board computer. Only adding external memory and required I/O devices a complete system is configured. I.e. it is easy to design systems with the ERC32 core.

The ERC32 chipset is characterized of:

- Performance up to 10 MIPS at 14 MHz clock frequenzy
- Typically more than 95% error detection of ERC32 including memory
- Rad hard latchup free technology up to 70 kRad
- Power consumption below 2,25 W at 5,5 V.

The chipset has passed extensive simulations in VHDL and characterisation tests of procured prototypes on-site. The functionality has also been verified within frame of 32-bit Demonstration Breadboard (DEM32) used by both hardware and software co-ordinators and by software tools developers.

The ERC32 chipset is now available for procurement for flight application purposes.

1.2. Software Development

The software development has focused on ensuring that requirements for future space missions are addressed. In particular, effort has been spent to ensure that support is provided for producing efficient and effective code as well as for the implementation and verification of the Hard Real-Time (HRT) requirements that are prevalent to on-board software applications.

These requirements specify that the application must not only operate correctly, but that the operation is performed within specific time constraints, such that failure to meet the timing constraints is considered as an operational failure.

The result of the development has been a suite of tools (ERC32 software toolset) which support the user in the development and verification of hard real-time software written in the Ada programming language. The tools can be used from the start of a programme, for example in the prototyping and conceptual design, through to the final verification and validation of the application.

The ERC32 toolset is hosted on a Sun SPARC platform running Solaris 2.4 and consists of:

- An Ada Compilation System,which provides for the generation and cross-debugging of Ada applications, which utilise tasking mechanisms in an efficient and effective manner that allows verification of the execution times;

- A Schedulability Analyser to provide for the analysis and verification of the timing requirements of an application;

- A Scheduler Simulator to provide a visualisation of the execution of the application, including highlighting the Run-Time operations involved in executing the application;

- An ERC32 Target Simulator, which provides a highly configurable and detailed simulation of the ERC32 hardware and its environment, thus allowing comprehensive testing of an application prior without recourse to the actual hardware.

The tools have undergone a verification and validation process, which includes both a limited evaluation performed as part of the programme and a formal validation of each tool.

The ERC32 toolset is now ready to be used for flight application purposes.

2. INTRODUCTION

2.1. Scope

This document describes the work which has been carried out under the ESTEC Contract no 9848/92/NL/FM for 32-bit Microprocessor and Computer Development programme.

The first chapter will give the background and objectives of the programme as well as the project history, while subsequent chapters will outline the work which has been performed and the results received.

Major technical specifications, descriptions and other relevant documents have been provided in the list of documents and abbreviations used in this report are listed separately.

2.2. Background

At the time the 32-bit Microprocessor and Computer Development programme started the availability of ESA supported processors was limited to 16-bit processors.

However, the requirements for computational throughput in on-board computers had risen almost constantly with each new project. Although the 16-bit processors was capable of up to 3 MIPS (DAIS), higher demands had been expressed from projects like Hermes and Columbus. With this background, ESA took a decision to develop a 32-bit microprocessor together with the necessary support circuits and software development tools, to be used in high performance on-board computers.

Furthermore, it was requested to "reuse" an existing processor architecture in order to minimize both software and hardware development cost. Performed ESA and industrial studies resulted, at that time, in the selection of SPARC instruction set architecture as the baseline. This was in order to e.g. simplify bread-boarding and software development.

It was requested that the developed processor, support circuitry and software tools must support different type of applications with minimum performance penalties, even though the applications can call for completely different system architectures. This was for cost reasons for development, qualification and long term support.

The first type of application identified was characterized by strong requirements on system availability, safety, fault-tolerance and real-time behaviour. High computational throughput was also a requirement, but the dependability requirements were prioritized.

In the second type of application, computing throughput was the driving requirement, and requirements for system availability, fault-tolerance and real-time behaviour were less emphasized.

On the software side, requirements for an industrial-quality Ada cross compilation system had been identified. It was requested that the system should be able to generate efficient code complying with the levels of predictability and efficiency demanded by the application. Furthermore, the compilation system should interface with the ESA-funded Ada tasking coprocessor (ATAC).

The need for more sophisticated tools such as symbolic cross-debuggers, target simulators and real-time oriented tools had also been identified. These requirements were based on the experiences of previous ESA space programmes and also the academic research into HRT software development.

The two predecessor projects, the Hard Real Time Operating System Kernel (HRTOSK) and the Hard Real-Time Embedded Software Support Environment (HESSE) built on this research. These programmes resulted in a framework for the development of analysable hard real-time software written in Ada and the identification and prototyping of tools to perform this analysis.

This project was to be built upon this foundation, incorporating further developments in the theory, with the intention of moving towards industrialised tools. Thus these tools were to allow the ERC32 users to immediately begin developing applications.

2.3. Objective

The main objective with this programme was to develop an infrastructure of hardware and software components to support the development of 32-bit onboard computers for future space applications.

This was to be done by developing a complete 32-bit Embedded Real-time Computing core (ERC32) based on a SPARC processor and a companion ASIC targeted for embedded real-time applications, with built-in concurrent error detection and handling capabilities. By adding buffers for the address and data bus to the core, ERC32 was to be capable to interface to different types of memories and peripherals without redesign or additional hardware.

Three individual chips were foreseen to be developed within the programme; an Integer Unit (IU), a Floating Point unit (FPU) and a Memory Controller (MEC). An ADA Tasking Coprocessor (ATAC) chip was under development in a separate contract, and was to be used as a special purpose peripheral.

To demonstrate the capability of the developed computing core, a 32-bit Demonstration Breadboard (DEM32) was to be built around the ERC32, configured as a typical on-board embedded real-time computer.

Due to the high cost of this programme, strong emphasis had been put on modelling and simulation of the complete system with all its components before starting the design phase of the individual devices. The need for precise and unambiguous specifications, had led to the decision to use VHDL as the primary specification language and, consequently, for all modelling and simulation. The VHDL models would also ease system design, by providing a simulation model for board level simulation, and simplifying any modifications or enhancements of the developed components.

Furthermore, it was requested that in order to avoid any possible export licensing problems, all developed components should be manufactured and made available by a European source without any restriction of use within the European space community. The components were requested to be non-proprietary, i.e. they shall be actively marketed and supported by the manufacturer, insuring long-term availability of the components to the European space industry.

The software part of the ERC32 programme had been undertaken to ensure that a modern development environment is available for flight computer designs using the ERC32 chipset. Thus, a software toolset was to be developed focusing on hard real-time software design typical for onboard space applications. The toolset should provide all the necessary support for the analysis, coding and testing of a real-time software.

The toolset was to include an Ada cross compiler environment, based on Ada83 but with Ada95-like additions to provide a deterministic solution which could be analysed by means of the deadline monotonic theory.

To support software development, tools were also needed to be able to analyse the real-time behaviour of an application with respect to this theory. These tools should be used during all phases of software development and be able to interface in the coding phase with the cross compiler, the cross compiler providing input derived from the actual code to be used for analysis.

Finally, a performant simulator was needed to support embedded software development before hardware is available. The simulator should interface with the debugger of the cross compiler system, but also be able to be used as a stand-alone tool.

As with the hardware components, the software tools were requested to be provided to the European industry without any export or other restrictions.

2.4. Project Organization

The work has been performed by the following contractors with ESTEC as customer:

* **Saab Ericsson Space**
  405 15 Göteborg
  Sweden

* **Logica UK Ltd**
  Space Division
  Wyndham Court
  94 Portsmouth Road
  Cobham
  Surrey KT11 3LG
  England

* **Alsys/Thomson Software Products**
  Partridge House
  Newton Road
  Henley-on-Thames
  Oxon, RG9 1EN
  England

* **Spacebel Informatique**
  111 Rue Colonel Bourg
  B-1140 Bruxelles
  Belgium

* **Matra Marconi Space**
  37, avenue Louis Breguet - B.P. 1
  78146 Vélizy-Villacoublay Cedex
  France

* **Matra MHS/Temic**
  La Chantrerie - Route de Gachet
  CP 3008
  44087 Nantes Cedex 03
  France

The project organization is as follows:

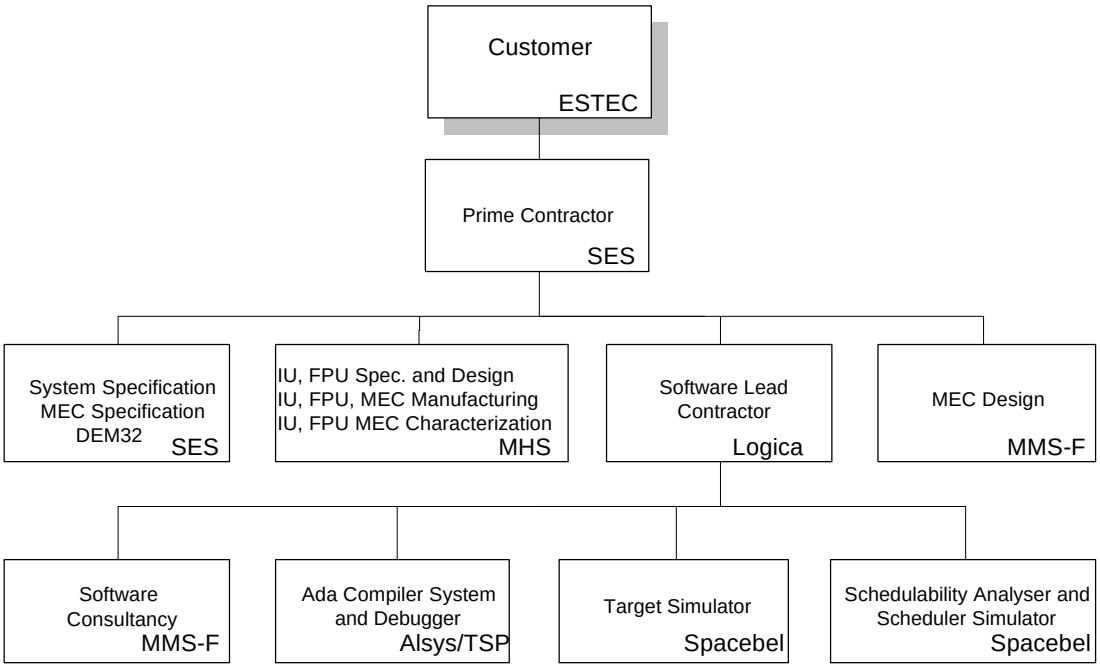

Figure 1 - The project organization

Saab Ericsson Space have been prime contractor for the 32-bit Microprocessor and Computer Development Program. The company has co-ordinated, supervised and reviewed hardware and software development including, among others, activities required to develop and release an ERC32 based Demonstration Breadboard (DEM32)

The hardware development consisted basically of three ASICs, Integer Unit, Floating Point Unit and Memory Controller.

MHS modified the design of existing Integer and Floating Point Units.

Saab Ericsson Space specified a Memory Controller and MMS-F made the design of it based on the VHDL simulation model released during specification phase.

MHS manufactured and performed the characterisation of all ASICs.

Logica have been software lead contractor, providing the overall co-ordination and management of the tools development as well as the definition of overall models for the tools and a toolset user manual.

MMS-F performed the functional analysis of the initial overall requirements for the toolset as well as the compilation of the requirements documents for the individual tools.

Alsys/TSP have developed the Ada Compilation System, providing also significant input to the compilation requirements specification and to the analysis for the Worst Case Execution Time Estimator tool.

Spacebel have developed the Schedulability Analyser, Scheduler Simulator and the ERC32 Target Simulator, including also the consolidation of the requirements initially specified for the tools.

2.5. Project History

The 32-bit Microprocessor and Computer Development programme has been basicly run in two different parallel lines, one for the hardware development and the other one for software development.

Different inputs from hardware development have been generated for the software development and vice versa in order to control the correct interfaces between the development lines.

2.5.1. Hardware Development

The hardware parts of the project has been run during three phases.

In phase 1 an analysis of an ERC32 based computer was made to define key system characteristics such as performance, concurrent error detection and handling capabilities, radiation hardness and power consumption.

The analysis resulted in a computer core (ERC32) which was built of three chips, i.e an Integer Unit (IU), a Floating Point Unit (FPU) and a Memory Controller (MEC).

Device level specifications were established. The ERC32 system was modelled and simulated in VHDL to consolidate the concept and to verify performance and behaviour.

In phase 2 detailed design and synthesis of the IU, the FPU and the MEC was performed, and prototypes were manufactured. The 32-bit Demonstration Breadboard, DEM32, was also defined, manufactured and tested, and integrated with a stand-alone monitor.

However, some revealed bugs inhibited the full use of this board for software developers. Thus, a rerun of the chipset were initiated and new versions of the prototypes were manufactured.

In phase 3, called Development Finalisation Phase, all developed prototypes have been characterized accordingly on-site. The functionality of the devices was also verified within frame of 32-bit Demonstration Breadboard. This board was also delivered to software developers which could fully utilize all capabilities of it for the software verification purposes.

2.5.2. Software Development

The software part of the project was initially split into three phases, 1, 2 and 3. Phase 1 was the requirements definition phase and was to result in the tool requirements. Phase 2 was the implementation of the tools and phase 3 was for the characterisation and validation of the tools, i.e. benchmarking and Ada Compilation Validation Certificate (ACVC).

The tools initially identified by the statement of work [AD1] were the Ada Compilation System, the Schedulability Analyser, the Scheduler Simulator and the ERC32 Target Simulator.

During phase 1, it was identified that a Worst Case Execution Time Estimator tool (WCETE) should be included in the set to provide for the semi-automatic generation of worst case timings for an application's constituent tasks. A Contract Change Notice, CCN, was produced and agreed to include such a tool in the toolset.

Completion of phase 1 was delayed due to the addition of the WCETE tool and due to the major deficiencies being identified in the initial requirements specifications. Following further work on the specifications and a Preliminary Software Requirements Review, closer communication links were established to consolidate the requirements process and phase 1 completed with a Software Requirements Review.

The phase 1 activities were undertaken by Saab Ericsson Space, Logica, Alsys/TSP and MMS-F. The initial consortium had also included Tecsidel, who were to have been responsible for the Schedulability Analyser and scheduler simulator. However, following a response to an Invitation To Tender and discussion between ESTEC and the consortium, Spacebel were introduced to the consortium for phase 2. In addition, due to the financial limitations of the programme it was not possible to include the WCETE tool implementation within the ERC32 programme. This activity was subsequently addressed through an external programme under the direct control of ESTEC.

The introduction of Spacebel and the delays in both the completion of phase 1 and start of phase 2, meant that it was necessary to split phase 2 development. Ada Compilation System was started ahead of official phase 2 kick-off, while the Schedulability Analyser, Scheduler Simulator and ERC32 Target Simulator development was split into two sub-phases, 2a and 2b. Phase 2a was introduced to provide for the consolidation of the requirements documents and to allow Spacebel better understand the development task. While phase 2b consisted of the development.

The requirements consolidation, phase 2a, was completed and lead to a proposal for phase 2b for the implementation of the tools.

However, due to the delays in the re-organisation of phase 2 the timescales no longer made it possible to maintain the original programme timescales and so advanced deliveries of the tools were introduced to provide early visibility of the tools. Refinements were agreed to be made as a result of ESTEC evaluation .

Throughout the period of phase 2, MMS-F have provided consultancy on the ERC32 Target Simulator development.

The ACS development progressed independently of the other tools. However, bugs in the first version of the ERC32 chipset inhibited full utilization of the DEM32 breadboard, which was to provide support for the ACS targeting. As a result a CCN was raised to extend the target testing of the ACS. However, due to programme financial constraints this caused the formal benchmarking activities, which were to have taken place at the end of the development, to be removed. Finally it was agreed that the ACS had to be accepted during phase 2 using a Force SPARC breadboard which used a commercial chipset and the ESTEC produced simulator *SIS*.

The phase 3 was redefined to Development Finalisation Phase and the work was performed in form of individual call-of orders for the ACS, the target simulator and the HRT tools.

The call-of order for the ACS development was agreed to allow the ACS to be modified for the new revised ERC32 chipset and to allow testing to be performed on an updated DEM32 breadboard. At the same time additional ACS functionality was requested by ESTEC.

The development was completed successfully using the DEM32 V2 board, thus ensuring that the compiler had been accepted on an actual target. I.e. complete qualification was reached including running the ACVC and benchmark tests.

The call-of order for the development of HRT tools was to provide for the refinements, agreed during phase 2, of the Schedulability Analyser and Scheduler Simulator and to incorporate the new functionalities in the revised ERC32 chipset into the ERC32 Target Simulator. This led to the final acceptance of the tools.

Logica's role was re-directed as a result of CCNs and call-off orders for the tools agreed during phase 2 and Development Finalisation Phase.

The work was re-directed from toolset integration to include the acceptance of the individual tools and production of the overall toolset user manual as well as to overall management of tool development.

2.6. Acknowledgements

Many thanks to

* Mr Andy Walter	Logica
* Mrs Françoise Beghin	MHS
* Mr Vincent Stachetti	MHS
* Mr Rémi Cissou	MMS-F

for their contribution, including their project team contribution, to this Final Report

and to

* Mr Jiri Gaisler	ESTEC
* Mr Tullio Vardanega	ESTEC
* Mr Andy Walter	Logica

for their great engagement in the program.

2.7. Documents

2.7.1. Applicable Documents

This section contains the list of the applicable documents for the development.

[AD1] ESA 32-bit Microprocessor and Computer Development Programme Statement of Work
WDI/JG/1317/NL, Issue 2.1, 28 May 1991

[AD2] ESA Software Engineering Standards
ESA PSS-05-0, Issue 2, Feb 1991

[AD3] Specification for a 32-bit Embedded Computing Core (ERC32)
WDI/JG/1334/NL, Issue 3, 29 May 1991

[AD4] 32-bit Microprocessor ATAC 2.0 Preliminary Interface Definition
Applicable Document 3, Aug 1991

[AD5] Requirements for Evaluation of a Manufacturer for the Manufacture and Supply of Standard Electronic Components for Space Applications
ESA/SCC Basic Specification No. 20200, Issue 2 Rev B, Jun 1991

[AD6] 32-bit Microprocessor Software Tools Technical Requirements
WDI/1339/FGM/NL, 5 Jun 1991

[AD7] Integrated Circuits, Monolithic
ESA/SCC Generic Specification No. 9000, Issue 7 Rev B, Nov 1991

[AD8] Component Selection, Procurement and Control for ESA Space Systems
ESA PSS-01-60, Issue 2, Nov 1988

[AD9] ESA Preferred Parts List
ESA PSS-01-603, Issue 2 Rev 0, Jun 1985

[AD10] ESA/SCC Qualified Parts List
Issue 91/2, Amendment A, Dec 1991

[AD11] Derating Requirements, Worst Case Analysis and Application Rules for Electronic Components
ESA PSS-01-301, Issue 2, May 1991

[AD12] Statement of Work for Participation in the ESA 32-bit Microprocessor and Computer System Development Phase 2 and 3, S/W Part
MCD/SOW/0007/SE, Issue 1, 16 Feb 1994

2.7.2. Reference Documents

[RD1] Hard Real-Time Operating System Kernel - Volume 1
TP1079 - Volume 1, Feb 1993
ESTEC Contract 9198/90/NL/SF

[RD2] Hard Real-Time Operating System Kernel - Volume 2
TP1079 - Volume 2, Feb 1993
ESTEC Contract 9198/90/NL/SF

[RD3] HMA31750 HESSE Implementation - Final Report
31750-LOG-IMP-TN-011, Nov 1993

[RD4] ERC32 single-event upset test results
WSD/JG/320/NL, Issue 1, 22 Oct 1996

[RD5] Benchmarking of 32-bit processors for space applications
WDI/JG/2105/NL, Issue 4, 20 Nov 1995

2.8. Abbreviations

ACS	*Ada Compilation System*
ACVC	*Ada Compiler Validation Certificate*
ADR	*Architectural Design Review*
ADS	*Arbitary Deadline Scheduling*
AOCS	*Attitude and Orbit Control System*
AR	*Acceptance Review*
ATAC	*ADA Tasking Accelerator*
BCR	*Baseline Consolidation Review*
CCN	*Contract Change Notice*
CUT/KP	*Code and Unit Test Key Point*
CWP	*Current Window Pointer*
DAIS	*Digital Avionics Instruction Set*
DEM32	*32-bit Demonstration Breadboard*
DMA	*Direct Memory Access interface*
DMS	*Deadline Monotonic Scheduling*
DRC	*Design Rule Check*
EDAC	*Error Detection And Correction unit*
ERC	*Electrical rule Check*

ERC32	*32-bit Embedded Real-time Computing Core*
ESD	*Electrostatic Discharge*
ESM	*Embedded Signature-monitor Technique*
FPU	*Floating Point Unit*
HESSE	*Hard Real-Time Embedded Software Support Environment*
HOOD	*Hierarchical Object Oriented Design*
HRT	*Hard Real-Time*
HRTOSK	*Hard Real-Time Operating System Kernel*
IR	*Integration Review*
IU	*Integer Unit*
LVS	*Layout versus Schematic*
MEC	*Memory Controller*
MIPS	*Mega instructions per second*
MMU	*Memory Management Unit*
NRL	*Naval Research Laboratory*
OBDH	*On-Board Data Handling*
PADR	*Preliminary Architectural Design Review*
PBPS	*Priority Based Preemptive Scheduling*
PCE	*Priority Ceiling Emulation*
PSR	*Processor Status Register*
RMS	*Rate Monotonic Scheduling*
SEU	*Single Event Upset*
SPARC	*Scalable Processor Architecture*
WCETE	*Worst Case Execution Time Estimator*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuits*

3. DEVELOPMENT OF THE ERC32 HARDWARE

This chapter describes the work performed to develop the ERC32 hardware. The design, verification and implementation aspects as well as problems encountered are described separately.

3.1. Description of the work performed

This section summarizes the activities performed for the ERC32 chipset development. By way of introduction the work package breakdown is presented for phase 1, 2 and Development Finalisation Phase. This is followed by subsections addressing specific aspects of the development.

3.1.1. Work Breakdown Structure

3.1.1.1. Phase 1

This phase consisted of five work packages:

- technology selection
- system definition
- system detail design
- system and component level simulation
- device development plan

3.1.1.1.1. Technology Selection

This work package dealt with the task to procure and analyze documentation describing the SPARC standard and design database.

The work package also dealt with the task to define all components and technologies to be used in an ERC32 based computer.

Previous ESA studies of various processor architectures, both CISC and RISC, drew as input to this work package. During these two independent studies, the SPARC architecture was chosen, mainly because it is a widely supported, open architecture available without license requirements.

The result of this work package was the selection of the SPARC V.7 architecture , the processor validation plan [GD55] and the preferred technology list for the ERC32 based computer [GD56].

3.1.1.1.2. System Definition

This work package dealt with the task to analyze and review the baseline ERC32 specification [AD3]. The work package also dealt with the task to study and select an implementation of a high degree of concurrent error detection and handling in the ERC32.

The baseline ERC32 specification was analyzed and an updated ERC32 specification was issued. An implementation of the concurrent error detection and handling scheme was selected and a report of error detection and reliability methods was issued. The concurrent error detection and handling scheme was based on master-checker circuits, program flow control, internal register parity, external bus parity and EDAC on memory.

The result of this work package was the updated ERC32 specification [GD58] and the error detection and reliability methods report [GD57].

3.1.1.1.3. System Detail Design

This work package dealt with the task to define the ERC32 bus architecture and all the interfaces. A detailed timing of all the interfaces was to be defined and the target was to achieve maximum computational throughput. The work package also dealt with the task to define an ERC32 based computer using only components from the ERC32 preferred parts list. Finally the work package dealt with the task to generate detailed specifications for all the devices to be developed in phase 2.

The ERC32 bus architecture was analyzed in many aspects; clocking, waitstates, Memory and I/O bus configuration, bus loading, DMA operation and power consumption etc. A clocking scheme of one system clock and one double clock was chosen. A bus architecture of separate Memory and I/O bus was chosen.

The first ERC32 based computer was designed. A detailed memory map was defined and a signal description with timing was generated. The design consisted of IU, FPU, MEC, PROM, SRAM, Ada Tasking Co-processor (ATAC) and 2 serial ports.

Finally the preliminary specifications, at a functional level with critical timing of the circuits was issued. The purpose of these first specifications was to serve as input to the development of VHDL models of all circuits.

The result of this work package was a specification over the ERC32 bus architecture and interfaces [GD59], the ERC32 system design document [GD60] and the preliminary specifications of the IU, FPU and MEC circuits [GD61], [GD62], [GD63].

3.1.1.1.4. System and Component Level Simulation

This work package dealt with the task to generate VHDL models at behaviour level of the devices to be developed, and to generate regression test suites for the validation of the models. The work package also dealt with the task to add timing properties for the VHDL models. Finally the work package dealt with the development and acceptance of the system level VHDL model of the ERC32.

An exhaustive amount of work was put into the development of the device VHDL models and it's regression test suits. During this task most of the ERC32 system design work performed in previous work packages were ennobled. Timing properties was added on all output pins and setup and hold time checks on all input pins. The models were tested against developed regression test suits that were to serve as input to the development of manufacturing test suites. At the end of the work package ERC32 system level simulations was performed and the individual models were delivered after a formal acceptance of the VHDL models.

The result of this work package was VHDL models of the devices including regression test suites, VHDL models test reports [GD64] and an integrated ERC32 system level model [GD65].

The ERC32 system level VHDL model was after delivery managed by ESTEC [GD66].

3.1.1.1.5. Device Development Plan

This work package dealt with the preparation for the phase 2 activities, generation of a device development plan and a device verification plan. This work package also dealt with the task to propose a manufacturer's organization, facilities, production control and inspection system.

The result of this work package was the device development and verification plan [GD67] and the manufacturer's organization for phase 2, see figure below.

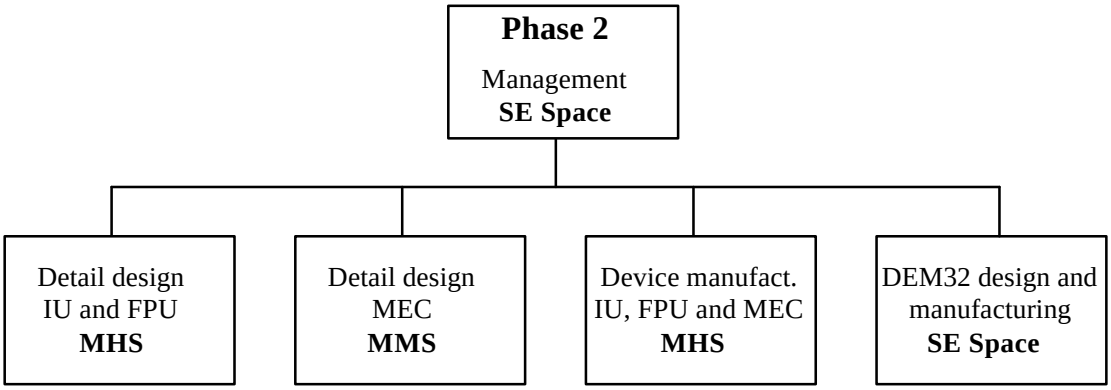

Figure 2 - Phase 2 organization

3.1.1.2. Phase 2

This phase consisted of three work packages:

- device detail design
- device manufacturing
- demonstration breadboard definition and manufacturing.

3.1.1.2.1. Device Detail Design

This work package dealt with the task to design the IU, FPU and MEC and verify the designs against the VHDL models functionally. This work package also dealt with the task to develop a complete set of test vectors for the devices to be used by the tester during manufacturing test.

The IU circuit was designed using GDT/Mentor for schematic capture and Cypress test vectors for database validation. After validation of IU basic functions introduction of fault tolerant mechanism was done and the RT database were validated.

The FPU circuit was designed using Verilog hardware description language and a modified set of Meiko test vectors for database validation. After validation of FPU basic functions introduction of fault tolerant mechanism was done and the RT database was validated.

The MEC circuit was designed using VHDL hardware description language and VHDL test vectors generated during system level simulations in phase 1 for database validation.

The result of this work package was the design data package including netlists of each device, updated VHDL models and manufacturing test vectors [GD68], [GD69], [GD70]. The specifications of the IU and FPU devices was issued in updated versions and for the MEC a new specification was issued [GD71].

3.1.1.2.2. Device Manufacturing

This work package dealt with the task to manufacture the devices in a number enough to start initial testing and characterization.

The result of this work package was 25 manufactured samples of each device tested and delivered of which 5 chipsets where delivered after electrical characterization in Development Finalisation Phase.

The validation of the devices was reported in the product validation reports [GD72], [GD73].

3.1.1.2.3. Demonstration Breadboard Definition and Manufacturing

This work package dealt with the task to define and manufacture a 32-bit Demonstration Breadboard, DEM32, based on the developed components and additional memory and peripherals from the preferred parts list.

A DEM32 for use in laboratory environment was developed based on :

- ERC32 computing core,
- 2 x 2 Mbyte RAM (EDAC protected) with 1 x 2 Mbyte redundant RAM
- 512 kByte EEPROM
- ATAC
- Two serial ports (RS232C)

The DEM32 was equipped with an adapted version of the SPARCmon monitor from Sun.
Complete schematics and timing analysis was produced. The computer was tested and was later used in the development of the ERC32 software tools.

The result of this work package was the ERC32 system overview [GD74] including DEM32 schematics and timing analysis. A DEM32 board including user's manual [GD75] and test report [GD76] delivered.

3.1.1.3. Development Finalisation Phase

This phase consisted of two work packages:

- the electrical and radiation characterization of the devices
- system testing and benchmarking.

3.1.1.3.1. Electrical and Radiation Characterization

In this work package, all manufactured device types were to be characterized. Especially all timing parameters were to be tested against the specifications and all DC parameters were to be measured.

The chipsets were tested in functional tests, electrical tests, applicative tests, SEU tests and total dose radiation tests. During functional and electrical evaluation test vectors developed during phase 2 was used and a device iteration was needed to correct some detected bugs.

Heavy ion induced single event upset tests (SEU) show an expected high system detection level of SEU´s in the chipset.
Total dose evaluation shows that the expected most sensitive parameter is leaking current.

The result of this work package was the ERC32 SEU test report [GD77], Total dose reports [GD78], [GD79], [GD80], Electrical Characterization reports [GD81], [GD82], [GD83] and Life test reports [GD84], [GD85], [GD86].

3.1.1.3.2. System Testing and Benchmarking

This work package dealt with the task to benchmark the complete system, DEM32 on the hardware side and ADA compiler on the software side.

The benchmarking was done at 10 MHz and zero/zero waitstates on the DEM32 board. The test run were the ESTEC-B Image Compression program, the Hartstone program, the ESTEC JIAWG selection, a PIWG selection and the ACVC version 1.11.

The result of this work package was presented in the Ada Compilation System qualification report [GD33] and the IU and FPU User´s manuals [GD87] and [GD87]

3.2. Design, Verification and Implementation

This chapter describes the design and verification methodology used for the development of the ERC 32 hardware devices as well as the technology used to manufacture these devices.

3.2.1. Design

3.2.1.1. Design Methodology

There was no standard design methodology used for the devices. However, this was primarily due to the different heritage.

The IU and the FPU devices were further developed from existing commercial devices whereas the MEC development started from the created VHDL model.

3.2.1.2. IU Design and Layout

The IU circuit was based on the Cypress CY7C601 processor.

The design of the IU with fault tolerant extensions has been done entirely in the GDT/Mentor environment:

- Debug of functional models of CY7C601 (M Model)
- Schematic entryto create MHS CY7C601 database
- Validation of database with Cypress test patterns
- Introduction of fault tolerant mechanism and test logic in CY7C601 database to create an IU database
- Generation of RT test patterns and validation of the RT database
- Replacement of control PLAs by logic gates
- Fault simulations on the IU database to analyze test pattern coverage
- Validation of netlist concerning the application of space rules
- Transistor size optimization
- Back annotated simulations of critical paths on final layout

At this step, all the dynamic parameters of the IU were compliant with the IU specification [GD61].

The main steps for Layout generation are described below:

- Generation of hardened SPARC-RT library for:
  - Standard cells
  - Data path cells
  - I/O blocks (pads)
- Mega cells (Register file)
- Adaptation and generation of the new floor plan
- Generation of all blocks (alu, program counter, special registers...)
- Assembly and routing of the chip
- Layout verification (ERC, DRC, LVS)
- Sizing and MEBES tape generation

3.2.1.3. FPU Design and Layout

Since the original database is based on the MEIKO design written in Verilog, the first design phases have been done in Verilog and there after transfer the database to a GDT/Mentor database.

- Adding of interface with the IU
- Validation of this database with MEIKO test patterns and new test benches.
- Introduction of fault tolerant mechanism and test logic in the database to create a FPU database
- Generation of RT test patterns and validation of the RT database
- Generation of high level VHDL model of the FPU and validation
- Replacement of ROM by logic gates
- transfer from Verilog to GDT/Mentor environment by using synthesis tool
- Validation of this the FPU database by running all test patterns
- Fault simulations on the FPU database to analyze test pattern coverage
- Validation of netlist concerning the application of space rules
- Transistor size optimization
- Back annotated simulations of critical paths on final layout

At this step, all the dynamic parameters of the FPU were compliant with the FPU specification [GD62].

The main steps for Layout generation are described below:

- Generation of hardened SPARC-RT library for:
  - Standard cells (same as for the IU)
  - I/O blocks (pads)
- Mega cells (Register file)
- Adaptation and generation of the new floor plan
- Generation of all blocks
- Assembly and routing of the chip
- Layout verification (ERC, DRC, LVS)
- Sizing and MEBES tape generation

3.2.1.4. MEC Design and Layout

The MEC circuit was based on the MEC behaviour model written in VHDL during phase 1. The first design phases dealt with the task to convert the behaviour model to a synthesizable model.

- corrections of bugs in the behaviour model,
- redefinition of the sequencing of the RAM access control signals to achieve a 14MHz ERC32,
- modification of the sequencing of the MEC in order to get only one clock and one edge active,
- balancing of the internal data bus in order to optimize the critical paths,
- optimization of the address decoder in order to reduce the number of gates of the decoding structure,
- redefinition of the access protection scheme,
- removing of MEC master/slave capability in order to save gates and pads
- implementation of the modifications of the MEC specification done during the detailed design,
- implementation of the real MCRT input and output buffers,
- removing timing checkers and output delays.

From the behaviour point of view, the VHDL simulation model and the VHDL synthesizable model are totally identical. So it was possible to check the behaviour of the synthesizable model against the simulation model using the regression test suites developed during phase 1.

The synthesizable VHDL model of the MEC was used as an input for the SYNOPSYS synthesis tool. The synthesis has been done using MHS MC RT library. Each time a block was considered as critical on the timing or on the area point of view, a structural synthesis approach was preferred.

- Transfer of MEC VHDL model to Verilog netlist format by using synthesis tool.
- Elaboration in order to constrain the synthesis tool to be compliant with the MEC specification.
- Validation of this MEC netlist by running all test patterns at 15 MHz, 0 waitstate
- Fault simulations on Mentor Quick Fault simulator to analyze test pattern coverage. The chosen fault models were Stuck at 1 Stuck at 0 on the output.
- A floor planning of the MEC.
- Clock tree tuning.
- Back annotated simulations of critical paths on final layout at 14 MHz.

At this step, all the dynamic parameters of the MEC were compliant with the MEC specification [GD71].

3.2.1.5. Design Tools

Following design tools are used for each step of the design flow:

- Schematic & manual layout entry..... Led (MENTOR)
- Logic simulations..... Lsim (MENTOR) + XP200 (ZYCAD)
- VHDL simulations..... QVSIM (MENTOR)
- Functional simulations..... Lsim (MENTOR)
- Functional simulations..... Verilog (CADENCE)
- Synthesis in gates..... Design Compiler (SYNOPSYS)
- Synthesis in gates..... AutoLogic (MENTOR)
- Scan insertion in netlist..... DFT ADVISOR (MENTOR)
- Generation of standard cells blocks..... AutoCells (MENTOR)
- Generation of data path blocks..... DP compiler (MENTOR)
- JTAG patterns generation..... FASTSCAN (MENTOR)
- Floor plan of the chip..... MicroPlan (MENTOR)
- Routing of the chip..... MicroRoute (MENTOR)
- Sizing & MEBES tape generation..... Dracula (CADENCE)

3.2.2. Verification

3.2.2.1. Verification Methodology

The ERC32 chipset has been verified by both VHDL model verification and design database verification before manufacturing and by characterisation after manufacturing.

The validation plan of the ERC32 chipset has been described in the document, Processor Validation Plan [GD55].

3.2.2.2. VHDL Model Verification

The high level VHDL models are generated in order to perform simulations at system level with the three components: IU, FPU and MEC.

IU VHDL model:

To validate the IU VHDL model, the existing production test patterns of the Cypress CY7C601 have been used. Those patterns have been developed by Cypress and are convenient for the architecture of the chip. Those test patterns represent 100K vectors split into 135 independent test files. Each of those files test a special functionality of the circuit. After translation of the test patterns into a Lsim format, these patterns are simulated using QVSIM VHDL simulator. Comparison is performed vector by vector with respect to original production test patterns.

FPU VHDL model:

No existing test vectors were available to validate the FPU since the FPU (from MEIKO) has not the same Instruction sequencing as the 90C602 (from WEITEK). So Test vectors were generated using SPARC source files (assembly code) from different Databases:

- Source Files from MEIKO to test MEIKO core (87 files)
- Source Files from TSUNAMI database (chip with MEIKO FPU) (14 files)
- Source Files from MMS and MHS to test global functionality of the FPU (6 files)
- Source Files from IEEE to test IEEE conformity (9 files)

To validate the FPU Verilog Model in a real system, the simulations were achieved with the CY7C601 product connected to the Verilog simulator via a hardware simulator (LM700). The test patterns represent 450K vectors split into 116 independent test files. These patterns are simulated using QVSIM VHDL simulator. Comparison is performed vector by vector with respect to original LSIM patterns.

MEC VHDL model:

The test benches developed during phase 1 have been modified and used for VHDL model validation. The modification was done mainly for 3 reasons:

- to take into account MHS test oriented simulation rules. For example, it is not possible to perform a data transition directly from a value D1 to a value D2 because of potential contention problems.
- if a signal generated by the test bench is generated on the same clock edge than it is sampled by the MEC, there could be one clock delay in the sampling of the signal. So the test benches have been modified to generate a signal on the rising (resp. falling) edge of the clock when it is sampled on the falling (resp. rising) edge of the clock by the MEC.
- to upgrade the fault coverage.

The tabular files generated by these modified test benches have also been used as an input for the gate level simulations.

3.2.2.3. Design Database Verification

3.2.2.3.1. IU and FPU Circuits

Design database verification before wafer manufacturing has been described in the IU-RT and FPU-RT Final Design Documents [GD69], [GD69] and summarized for:

- design rules check DRC
- layout versus schematics LVS
- electrical rules check ERC

DRC verifications:

The 2 chips have been layouted in order to be processed in two different technologies:

- z86H called SCMOS - RT (previously called w86rt)
- z86R called SCMOS - RH (previously called w86rtp)

So, design rules must be checked for these two technologies.

LVS Verifications:

Comparison between Layout and Schematics is performed in both technologies.

ERC Verifications:

Electrical rules are verified with the same software. This check detects:

- All transistors powered or grounded by other layers than MET1 or MET2
- Short or Opens
- Transistor gates connected to VCC or VSS

3.2.2.3.2. MEC Circuit

The MEC is verified in accordance with MHS verification methodology for circuits implemented on a MC gate array matrix.

Design database verification before wafer manufacturing has been described in the MEC Design Report [GD70].

3.2.2.4. Characterisation

Characterisation of the ERC32 propotype chipset consists of:

* functional tests
* electrical tests
* radiation tests

A complete set of manufacturing test vectors were used for functional tests with a fault coverage of 95% for the IU circuit, 98.3% for the FPU circuit and 94.6% for the MEC circuit.

The results of the characterisation of the ERC32 chipset can be found in chapter for "achieved results for the ERC32 hardware".

3.2.2.5. Verification Tools

Following verification tools are used for each step of the verification flow:

- Electrical simulations..... Eldo (ANACAD)
- Fault simulations..... XP200 (ZYCAD)
- Fault simulations..... Quick Fault (MENTOR)
- Static Analysis..... PathMill (EPIC)
- Static Analysis..... QuickPath (MENTOR)
- Layout verifications (ERC, DRC, LVS).. Dracula (CADENCE)

3.2.3. Implementation

The IU, the FPU and the MEC have been manufactured in different technologies.

The IU has been manufactured using the Matra MHS W86H process, a space hardened 0.8 µm SCMOS-RT technology. The mask number is 2821.

The FPU has been manufactured using the Matra MHS W86H process, a space hardened 0.8 µm SCMOS-RT technology. The mask number is 2859.

The MEC has been manufactured using the Matra MHS gate array MC 50K, processed on the space hardened 0.8µm SCMOS-RT technology.

3.3. Problems Encountered

Since the device development approach was different for each circuit different problems where encountered.

The design of the IU and the FPU circuits was based on existing commercial designs with fault tolerant mechanism extensions. The MEC circuit was a totally new design developed with a top down design methodology.

The problems are primarily identified in form of detected bugs in the circuits.

3.3.1. IU Circuit

- Revision 0: 3 bugs identified
- Revision A: Not functional
  * clock skew problem
- Revision B: All bugs from Revision 0 corrected. 2 new bugs discovered
  * power on reset
  * jmpl/restore
- Revision C: All bugs from Revision B have been removed

No bug discovered on Revision C during functional evaluation.

3.3.2. FPU Circuit

- Revision 0: 9 bugs identified
- Revision A: All bugs from Revision 0 corrected. 5 new bugs identified
  * failing stdf and stdfq
  * failing lddf
  * failing mds/mhold/fhold sequence
  * failing sequences with waitstates
  * failing parity during fhold

- Revision B: All bugs from Revision A corrected. 3 new bugs discovered
  * failing store double floating-point instruction
  * failing store floating-point status register with waitstates
  * failing load floating-point instruction with waitstates

Three bugs discovered on Revision B during functional evaluation.
Problem description and work around is available from Matra MHS (TEMIC) [GD89].

3.3.3. MEC Circuit

The work with the transfer of the MEC behaviour model to a synthesizable model took a lot more time than expected. The timing of the MEC behaviour model was implemented in a timing shell and the true timing of the synthesized MEC was not acceptable, that lead to some redesign of MEC functionality.

- Redefinition of the sequencing of the RAM access control signals to achieve a 14MHz ERC32.
- Heavy modification of the architecture of the EDAC block in order to speed up the error detection.
- Modification of the sequencing of the MEC in order to get only one clock and one edge active.

- Revision 0: 12 bugs identified

- Revision A: All bugs from Revision 0 corrected. 3 new bugs identified
  * CPU Halt indication in MEC Error and Reset Status register not cleared at soft reset,
  * Erroneous UART status after UART clear,
  *System Fault Status register not updated during EDAC Non-Correctable Error)

Three bugs discovered on Revision A during functional evaluation.
Problem description and work around is available from Matra MHS (TEMIC) [GD89].

3.3.4. DEM32 Board

The first DEM32 board version 1, was based on the ERC32 chipset revision 0A0 and had problems with the bugs detected in the ERC32 chipset:

- Power on reset might fail in the IU circuit.
- Internal parity error.
- Wrong polarity on bus parity.
- FPU / MEC operation bug.

The second version of the DEM32 board version 2, was based on the ERC32 chipset revision CBA and all problems from the version 1 board was solved but some new was detected in the ERC32 chipset:

- CPU halt indication not cleared at soft reset.
  => Error injection during CPU halt not possible.
- Erroneous UART status after UART clear.
  => modification of the DEM32 monitor software.
- Data parity timing not consistent with EDAC check bits timing.
  => bug fix on the board to used the data write strobe for check bits store instead of the dedicated check bits write strobe.

This version of the board could, however, be used during the ERC32 compilation system qualification and validation testing.

A third version of the DEM32 board was manufactured and delivered to be used in the ERC32 evaluation program.

The DEM32 V3, is identical to the previous version, DEM32 V2 board, but has an expanded EEPROM memory.

4. ACHIEVED RESULTS FOR THE ERC32 HARDWARE

This chapter describes the functionality of the ERC32 chipset, performance where applicable, electrical characteristics and radiation hardness capabilities as well as the currently known limitations and bugs.

The ERC32 chipset has successfully completed verification according to plan .

The ERC32 chipset has given satisfactory results concerning the different evaluations:

- The functional evaluation on breadboards with various programs has demonstrated the full compatibility with the SPARC V7 standard. 3 bugs have been discovered on the FPU which can be work around by software and 3 bugs have been discovered on the MEC which have a minor effect on the system.

- The electrical evaluation has demonstrated the full compatibility to the 14Mhz specifications.

- The radiation results gathered during various test campaigns show that the hardening of very complex components can be achieved by taking into account all possible degradation and weakness of the elementary devices. Up to 97.5% of all single bit errors generated during worst case test are detected by hardware at the system level. The use of MASTER/CHECKER option would reduce this remaining error rate.

- Manufactured using the 0.8µm SCMOS-RT technology, the three devices show typical degradation (power supply increase) after irradiation but with no functional limitation. Since the overall consumption is far below the specification and because of previously demonstrated dose rate effects, total dose is not a concern for designers who have to deal with 100Krad.

4.1. Description of the ERC32 Chipset

4.1.1. Functionalities

The ERC32 chipset is a SPARC V7 compatible processing core implemented in three devices; the Integer Unit (IU), the Floating-Point Unit (FPU) and the Memory Controller (MEC).

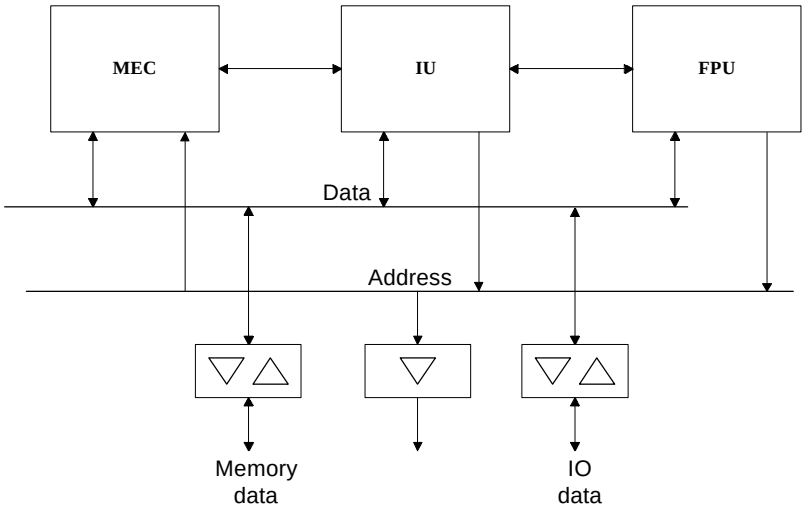

Figure 3 - ERC32 Computer core with buffers

The ERC32 interfaces directly to external memory and IO devices. The MEC includes all system functions required to form an embedded computer and to host a real-time operating system. The most important features are:

- Address decoding
- Memory interface
- Interrupt controller
- Block protection unit
- 32-bit SEC/DED Error Detection And Correction unit (EDAC)
- Two 32-bit timers
- Two UARTs
- Boot prom interface
- Direct Memory Access interface (DMA)
- Error manager
- Watchdog

The ERC32 does not use a cache memory, it runs directly from a fast SRAM-based main memory. There is no memory management unit (MMU), address translation and paging is typically not used in space-based embedded systems.

4.1.1.1. Integer Unit (IU)

The Integer Unit (IU) is the main computing engine of the ERC32 chipset. The IU is based on the SPARC 32-bit RISC architecture, which defines a processor capable of execution at a rate approaching one instruction per clock cycle.

The IU is based on the CY7C601 a SPARC V7 integer unit from Cypress Semiconductors. To the basic functions of the Cypress Integer Unit a number of concurrent error detection and handling capabilities has been added.

4.1.1.1.1. Basic Functions

Registers
A total of 140 32-bit registers are accessible to the programmer, divided into 136 general and four special purpose register. The SPARC architecture uses register windowing, the general purpose registers are divided into windows of 24 registers, with an overlap of eight. Only one window at a time is accessible, selected through the Current Window Pointer (CWP) in the Processor Status Register (PSR).

Exceptions
Two types of exceptions (traps) are supported, synchronous and asynchronous. The asynchronous traps are generated by external interrupts and can be masked, while the synchronous traps originate from internal events and cannot be disabled.

4.1.1.1.2. Concurrent Error Detection Capabilities

The concurrent error detection capabilities consists of parity protection and program flow control.

Parity Protection
The register file, containing the general purpose registers, is provided with one parity bit per register. Other registers in the IU, such as general purpose registers, special purpose registers and temporary registers are also provided with one parity bit per register. More than 98% of all latches in the IU are covered with parity protection.

To check the integrity of the external address bus an address parity bit is generated. A parity bit on the external data bus is generated during stores and checked during loads and instruction fetches.

Three parity bits are used to protect the control buses, on bit on the bus going from the IU to the FPU (checked by the FPU), on bit on the bus going from the FPU to the IU (checked by the IU), and one bit for the remaining output control signals (checked by the MEC).

Program Flow Control
A program flow control functions is included in the IU using the embedded signature-monitor technique (ESM), in order to complement the register-targeted error-detection methods. The concept is to calculate a signature from each executed instruction and to compare this signature at appropriate points with a predefined checksum, to insure that the correct instructions have been executed.

4.1.1.1.3. Concurrent Error Handling Capabilities

Adhering to the general exception mechanism, the internal error-detection logic is evaluated during the execute stage of each instruction. If an error is found, an error trap is taken. The error traps are divided into six types, grouped after error location and possible recovery action.

4.1.1.2. Floating-Point Unit (FPU)

The Floating Point Unit (FPU) is a high-performance, single-chip implementation of the SPARC reference floating-point unit. The FPU is designed to provide execution of single and double-precision floating-point instructions concurrent with execution of integer instructions by the IU. The FPU is compiant with the ANSI/IEEE-754 floating-point standard.

The FPU is based on the MEIKO floating-point core. To the basic functions of the MEIKO floating-point core a number of concurrent error detection capabilities have been added.

4.1.1.2.1. Basic Functions

Execution
The floating-point instructions are started by the IU using the INS1/INS2 signals and then execute independently inside the FPU. Most FP instructions execute in parallel with IU operation. During the parallel execution, the address and data of the current instruction is held in the floating-point queue, FPQ.

Exception
If an exception (e.g. overflow) occurs during the execution of an floating-point instruction, the FPU will assert FEXC and enter pending_exception state. The IU will recognize the floating-point exception at the start of the following floating-point instruction and take a floating-point trap.

4.1.1.2.2. Concurrent Error Detection Capabilities

The FPU error-detection scheme is similar to the IU scheme. All registers are provided with parity bits, and FPU generates and checks parity bits for all buses (address, data and control). 100% of all latches in the FPU are covered with parity protection.

4.1.1.2.3. Concurrent Error Handling Capabilities

The error-handling is slightly different from the IU; the FPU cannot handle the detected errors on its own, but flags them to the IU which have to take corrective measures. When and error is detected, the FPU enters pending_exception state and indicates the error type in the ftt in the floating-point status register. Three types of errors are defined; restartable error, non-restartable error and data bus error.

4.1.1.3. Memory Controller (MEC)

The MEC is designed to interface the IU and the FPU to external memory and I/O units thus forming a system, with which computers for on-board embedded real-time applications can be built.

The Memory Controller implements important system support functions such as chip select decoding, wait state generation, EDAC, timers and UARTs as well as concurrent error detection and handling capabilities.

4.1.1.3.1. Basic Functions

The MEC constitutes all necessary support and on-chip resources:

- System start up control and reset
- Power down mode control
- System clock
- Watchdog function
- Memory interface to RAM ranging from 256 Kbyte to 32 Mbyte
- Memory interface to PROM ranging from 128 Kbyte to 4 Mbyte
- I/O interface to exchange memory (e.g. DPRAM) ranging from 4 to 512 Kbyte.
- I/O interface to four peripherals
- DMA interface
- Bus arbiter
- Programmable wait-state generator
- Programmable memory access protection
- Memory redundancy control
- EDAC, with byte and halfword write support
- Trap handler including 15-level interrupt controller
- One 32-bit general purpose timer with 16-bit scalar
- One 32-bit timer with 8-bit scalar (Real-Time-Clock)
- UART function with two serial channels
- Built-in concurrent error detection including support for master/slave checking of IU and FPU
- System error handler
- Parity control on system bus
- Test support including a minimal TAP interface

The MEC interfaces directly to the address, data, and control buses of the IU and FPU, requiring no additional components. It also interfaces directly to external memory and I/O units only requiring additional buffers for the address and data bus.

4.1.1.3.2. Concurrent Error Detection Capabilities

Error-detection is implemented by providing each MEC register with a parity bit which is continuously checked. The parity of the external address, data and part of the control bus is checked by the MEC.

4.1.1.3.3. Concurrent Error Handling Capabilities

The MEC also contains an error handling manager, where the error signals from the IU, FPU and the MEC itself are sensed. For each error type, the error manager can be programmed to either ignore the error, issue an interrupt, reset the ERC32 or halt.

4.1.2. Performance

The ERC32 chipset is compatible to its specification of 14MHz system clock frequency. Benchmarking for the ERC32 chip set has been detailed in the document: "Benchmarking of 32-bit processors for space applications" [RD5].

During the benchmarking exercise, it has become apparent that computational requirements can never be expressed in MIPS figure using a particular assembly level instruction mix. For the ERC32, benchmarks were compiled and run both with and without optimization. The assembly level instruction mix was equal in both cases, even though the un-optimized bench marks executed more that twice as slow. Performance requirement for systems where high level languages are to be used should therefore be expressed as a maximum execution time of a test program written in the same language. It is also important that the selected test program uses similar language elements and calculations as the target application.

4.1.3. Electrical Characteristics

The ERC32 chip set is specified and has been tested for:

Maximum ratings:

Storage Temperature :	-65°C to 150°C
Ambient Temperature with power applied:	-55°C to 125°C
Supply Voltage:	-0.5V to +7.0V
Input Voltage:	-0.5V to +7.0V

Operating range (Military):

Ambient Temperature with power applied:	-55°C to 125°C
Supply Voltage:	+5.0V ± 10%

DC characteristics over the Operating range:		min	max	Units
V_{OH}	Output high voltage	2.4[1]		[V]
V_{OL}	Output low voltage		0.5	[V]
V_{IH}	Input high voltage	2.1	V_{CC}	[V]
V_{IH}	Input low voltage	-0.5	0.8	[V]
I_{IZ}	Input leakage current	-10	10	[µA]

IU Circuit:		min	max	Units
I_{OZH}	Output leakage current (Vout=V_{CC})	-10	10	[µA]
I_{OZL}	Output leakage current (Vout=V_{SS})	50[2]	240[2]	[µA]
I_{SC}	Output short circuit current	-30	-350	[mA]
I_{CCop}	Supply current (@ 14MHz)		200	[mA]
I_{CCsb}	Supply current (@ standby)		1	[mA]

FPU Circuit:		min	max	Units
I_{OZH}	Output leakage current (Vout=V_{CC})	-15	15	[µA]
I_{SC}	Output short circuit current	-30	-350	[mA]
I_{CCop}	Supply current (@ 14MHz)		180	[mA]
I_{CCsb}	Supply current (@ standby)		3	[mA]

MEC Circuit:		min	max	Units
V_{OH}	Output high voltage	3.7		[V]
I_{OZH}	Output leakage current (Vout=V_{CC})	-15	15	[µA]
I_{SC}	Output short circuit current	-30	-350	[mA]
I_{CCop}	Supply current (@ 14MHz)		100	[mA]
I_{CCsb}	Supply current (@ standby)		1	[mA]

Capacitance ratings [3]		max	Units
C_{IN}	Input capacitance	10	[pF]
C_{OUT}	Output capacitance	12	[pF]
C_{IO}	Input/Output bus capacitance	15	[pF]

AC characteristics over the Operating range:		max	Units
f	System clock frequency	14	[MHz]

Notes:
[1] MEC Output high voltage >3.7V
[2] On chip pull-up resistor = 20kΩ
[3] Test conditions are: Vcc=5.0V, Ta=25°C, f=1MHz

4.1.4. Radiation Harness

The radiation harness has been evaluated for total dose, ESD, latch up and SEU characteristics.

4.1.4.1. Total Dose

The Total Dose evaluation for the ERC32 chip set has been detailed in the Total Dose test reports [GD78], [GD79], [GD80].

Test conditions

The three chips (IU-RT, FPU-RT & MEC) have been irradiated according to the standard ESA/SCC No 22900:

- Gamma rays from Cobalt 60 at 300rad(Si)/hour
- Biasing at 5.0V during irradiation
- Clock generated at few Hertz during irradiation
- Sensitive parameters monitored during and after irradiation
- High temperature annealing to assess low dose rate behaviour

Irradiation has been stopped at 70 & 80 Krad because of time constraint.

Main results

Since the three components are using the SCMOS1/2 RT technology, the expected most sensitive parameter is leaking current. Large dose rate effect is also illustrated by the recovery of the pre-rad value for sensitive parameter which confirms minor degradation in the outer space environment.

The absolute maximum power consumption for the three chips remains below specification (2.25W @ 5.5V) after 70Krad. No functional failure is observed all over the test.

4.1.4.2. Electrostatic Discharge (ESD)

IU ESD test:

- 3 parts stressed at 3500V --> 3 parts good at retest.
- 3 parts stressed at 4000V --> 1 part fail and 2 parts good at retest.

FPU ESD test:

- 3 parts stressed at 3500V --> 3 parts good at retest.
- 3 parts stressed at 4000V --> 3 parts fail at retest.

MEC ESD test:

The ESD sensitivity of the MC Gate Array series applies to the MEC.
No specific testing has been performed.

Conclusion:

IU and FPU; the level of ESD sensitivity is greater than 3500V;
both chips are classified in class 2 of the MIL STD (> 2000V).

MEC; the device is classified in class 2 of the MIL STD (> 2000V).

4.1.4.3. Latch Up

IU , FPU and MEC:

- Single event latchup (SEL) > 72 MeV/(mg/cm^2)
- Electrical latchup: no latchup observed

Conclusion: The devices are latchup free.

4.1.4.4. Single Event Upset (SEU)

The SEU evaluation for the ERC32 chipset has been detailed in the document "ERC32 single event upset test results" [RD4].

The ERC32 radiation Tolerant RISC processor is designed to be used in critical space applications where single event upset (SEU) phenomena can not be avoided. It therefore incorporates several mechanisms to detect and isolate SEU induced errors. Following are provided some test system description and results gathered during radiation tests campaigns.

SEU test concept

The following parameters were to be determined:

- LET threshold and cross-section
- Coverage of on-chip error-detection mechanisms in IU-RT, FPU-RT and MEC
- Coverage of system level error-detection mechanisms

The test system consists of a host computer and a target system connected together via a serial RS-232 link. The host computer has a monitoring software that logs events in the target system. The target system consists of an ERC32 chipset, 2 Mbyte RAM and 512 byte flash-PROM. The IU-RT and FPU-RT are running in master/checker mode, i.e. two devices in parallel. The target system is configured to use all error-detection functions available; parity on address, data and control buses is checked and the memory is provided with EDAC check bits. A dedicated test software has been developed to insure correct error detection and reporting. Three application programs are also developed to allow single device test:

- IUreg (IU test)
- FPUreg (FPUtest)
- the well-known Paranoia (application).

Test Results

Error calculation of the three devices have been done using CREME software from Naval Research Laboratory (NRL). The results are worst case assuming solar minimum conditions and a shielding of 1g/cm^2. The Upset rate has been calculated for two orbits taking into account a "normal" use of the devices:

- Low Earth Orbit (LEO), altitude 400 km, inclination 53°.
- Geostationary (GEO), altitude 36000 km, also valid for eccentric orbits.

Device	LEO (upset/day)	FITS (all errors)	Error detection coverage	FITS for undetected errors	MTBF for all errors (years)	MTBF for undetected errors (years)
IU	4.4E-5	2062	99.3%	11	55.4	7906
FPU	2.5E-6	106	94.2%	6	1074	18520
MEC	9.3E-6	388	99.1%	4	295	32373
ERC32	1.6E-4	2556		21	44.7	5436

Figure 4: The main results for Low Earth Orbit (compensated):

Device	GEO (upset/day)	FITS (all errors)	Error detection coverage	FITS for undetected errors	MTBF for all errors (years)	MTBF for undetected errors (years)
IU	4.1E-4	17292	99.3%	121	6.6	942
FPU	2.2E-5	896	94.2%	156	127	732
MEC	8.1E-5	3375	99.1%	30	33.8	3758
ERC32	1.3E-3	21563		307	5.2	372

Figure 5: The main results for Geostationary Orbit (compensated):

4.1.5. Known Limitations and Bugs

The document "Current Errors in the ERC32 (CBA)" [GD89] lists all known limitations and bugs. This should be consulted in the first place to ensure the status of the ERC32 chipset.

The following sections list the limitations and bugs in the ERC32 chipset delivered as part of the programme.

4.1.5.1. IU Circuit

No bug has been discovered on IU, Revision C during functional evaluation.

4.1.5.2. FPU Circuit

Three bugs have been discovered on FPU, Revision B during functional evaluation as follows:

* failing store double floating-point instruction
* failing store floating-point status register with waitstate
* failing load floating-point instruction with waitstates

4.1.5.3. MEC Circuit

Three bugs have been discovered on MEC, Revision A during functional evaluation as follows:

* CPU Halt indication in MEC Error and Reset Status register not cleared at soft reset
* Erroneous UART status after UART clear
* System Fault Status register not updated during EDAC Non-Correctable Error

4.1.5.4. ERC32 System

One bug has been discovered on the ERC32 system level, Revision CBA during functional evaluation as follows:

* Data parity timing not consistent with EDAC check bits timing

Workaround to this bug is to use the *data write strobe* for check bits instead of the dedicated *check bits write strobe*. For further details see [GD74].

4.2. Use of the ERC32 Chipset

The ERC32 is to be used as a building block only requiring memory and application specific peripherals to be added to form a complete on-board computer. All other system support functions are provided by the core.

The ERC32 incorporates the followings functions:

- Processor, which consists of one Integer Unit (IU) and one Floating-Point Unit (FPU). The processor includes concurrent error detection and handling capabilities.

- Memory Controller (MEC), which is a unit consisting of all necessary support functions such as memory control and protection, EDAC, wait state generator, timers, interrupt handler, watch dog, UARTs, and test and debug support. The unit also includes concurrent error detection and handling capabilities.

- One or two oscillator(s).

- Buffers necessary to interface with memory and peripherals.

The figure below schematically shows a basic ERC32 computer with external functions added to form a complete system.

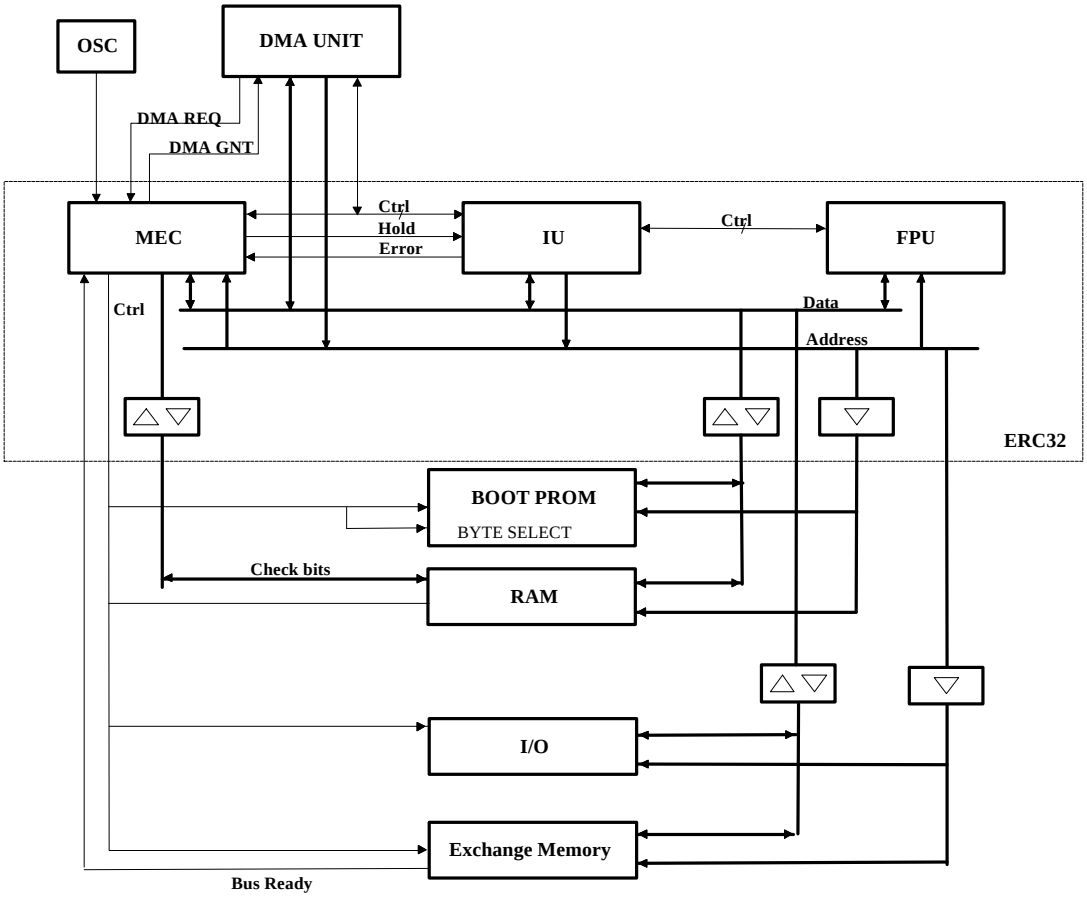

Figure 6 - ERC32 Computer with typical peripherals

4.3. Availability of the ERC32 Chipset

The ERC32 chipset is now available from Matra MHS (TEMIC):

IU revision C, as the TSC691E.
FPU revision B, as the TSC692E.
MEC revision A, as the TSC693E.

Information and chipsets can be obtained from the following:

TEMIC / MATRA MHS S.A.
La Chantrerie - Route de Gachet
CP 3008
44087 Nantes Cedex 03
France

Tel: (33) 40 18 18 18
Fax: (33) 40 18 19 20

The ERC32 VHDL models are now available from ESTEC:

ERC32 VHDL Models Version 1.0

Information and VHDL models can be obtained from the ERC32 Home page:

http://www.estec.esa.nl/wsmwww/erc32/erc32.html

The ERC32 Home page is managed by:

Mr Jiri Gaisler
European Space Agency (ESA/ESTEC)
Box 299
Noordwijk 2200 AG
The Netherlands.

Tel: (31) 71 5654880
Fax: (31) 71 5654295

4.4. Future improvements for the ERC32 Chipset

Future improvements for the ERC32 chipset can be divided in performance, system and functional aspects.

The performance could be improved in terms of speed (>20 MHz).

The system improvements could be achieved by implementing the three components on a single chip.

Concerning the functionality, a redesign of the FPU could be developed in order to correct all identified limitations to the system.

5. DEVELOPMENT OF THE ERC32 SOFTWARE

This chapter describes the work performed to develop the ERC32 software. The design, verification and implementation aspects as well as problems encountered are described separately.

5.1. Description of the work performed

This section summarises the activities performed for the software tools development. By way of introduction the work package breakdown is presented for phase 1, 2 and Development Finalisation Phase. This is followed by subsections addressing specific aspects of the development.

5.1.1. Work Breakdown Structure

5.1.1.1. Phase 1

Phase 1 consisted of a single work package for the specification of the tool requirements and planning for the implementation.

This work package was sub-divided into four sub work packages:

- overall co-ordination of tool specification
- specification of software tools
- design and development plan for tools
- test plan for tools

5.1.1.1.1. Overall Co-ordination of Tool Specification

This work package dealt with the overall management and control of the phase. This involved the co-ordination of the various inputs for the tools' specifications, planning for phase 2 and the review and approval of the individual specifications at completion of the phase.

In addition, there was a need to ensure that the tool requirements were consistent with the hardware development, such as overall system requirements and specific constraints on the tools due to the hardware implementation, e.g. Ada Compilation System Board Support Package and ERC32 Target Simulator implementation.

5.1.1.1.2. Specification of Software Tools

This work package dealt with the analysis and production of the software requirements documents. The requirements were a synthesis of general requirements such as the Platform and specific requirements identified in the SOW.

In addition, this work addressed the interfaces between the tools. Provision of an industrial quality set of tools requires that, where appropriate, the tools interact and/or transfer data consistently.

For this toolset, the ACS shall be able to use the ERC32 Target Simulator for debugging and in order to provide accurate HRT analysis the Schedulability Analyser and Scheduler Simulator need information regarding the Ada Run-Time System.

The result of this work package was the individual specifications and interface control documents.

5.1.1.1.3. Design and Development Plan for Tools

The purpose of this work package was to produce a development plan for the implementation. This plan was produced in accordance with PSS-05-0 and defined the managerial, technical and commercial issues of the implementations.

The phase 2 software project management plan, software configuration plan and software quality assurance plan were output from this work package.

5.1.1.1.4. Test Plan for Tools

This work package addressed the overall verification and validation approach to be taken for the implementation.

As there were a number of tools, it was necessary to both co-ordinate the verification process and also to plan that the interfaces between the tools were verified.

A software verification and validation plan was the outcome of this activity.

5.1.1.2. Phase 2 and Development Finalisation Phase

For phase 2 three overall work packages were identified for the software part of the programme:

- Ada Cross Compilation System Development
- Advanced Supporting Tools (Debugger)

- Hard Real-Time Supporting Tools (Other Tools)

Finally, the following structure of work packages was agreed as seen in figure below:

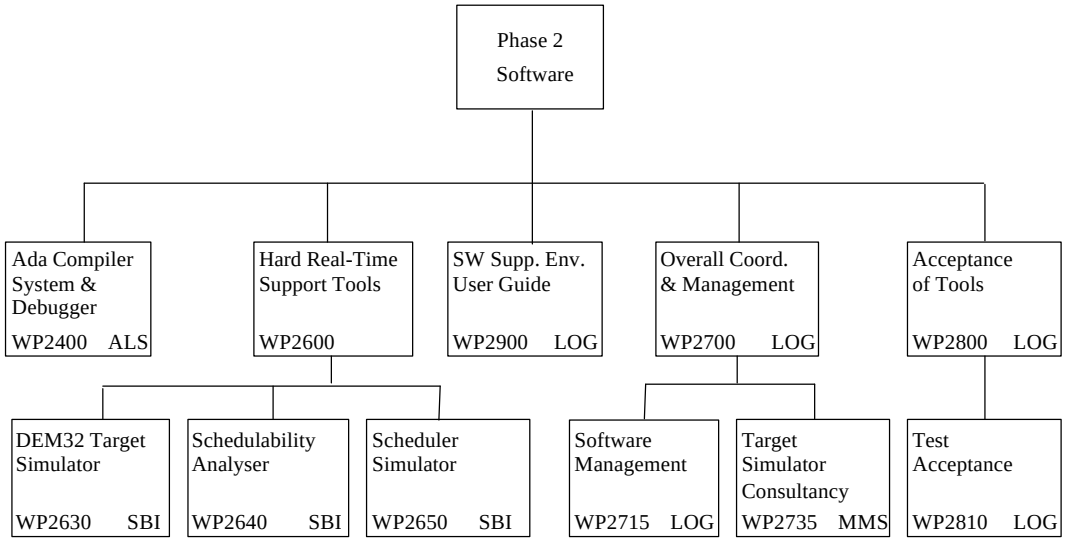

Figure 7: Phase 2 work breakdown structure

Initially, it had been planned to go through to a Phase 3 of the programme, where the software tools were to undergo formal integration and validation and the ACS would be subject to the Ada Compiler Validation Certificate (ACVC) process.

However, following the initial release of the ERC32 chipset, there were bugs identified in the chipset. This meant it was not possible to provisionally accept the ACS using the DEM32 breadboard.

These events coincided with the advanced release of the Schedulability Analyser and Scheduler Simulator. These were examined and a number of errors and refinements were identified.

As a consequence, it was agreed to modify the software programme through removal of the ACVC process as well as of the formal integration and validation process of the tools.

The efforts were instead directed to refinement of the HRT tools and modification of the ERC32 Target Simulator to the new ERC32 chipset. The ACS was also upgrade with additional functions and modified to the new chipset.

This work was finalized as a direct follow-up for the phase 2 during redirected Development Finalisation Phase.

5.1.1.2.1. Ada Compiler System and Debugger

The work consisted of taking the standard Alsys Baseline Compiler Technology and performing the necessary modifications to incorporate the ERC32 programme requirements. This consisted of the following activities:

- to develop a complete Ada program development and execution environment hosted on Sun SPARC under the Solaris operating system and targeting the 32-bit Embedded Real-Time Computing Core (ERC32), including the ATAC.
- to provide support for Hard Real Time scheduling analysis by implementing efficient Ada primitives in such a way as to model the basic entities and operations assumed by Rate Monotonic Analysis and Deadline Monotonic Analysis scheduling theories.
- to provide the necessary parameterization to permit use of an ERC32 Target Simulator (also developed within the project) as the target for the Ada Compilation System.

The development was completed with provisional acceptance tests using a commercial SPARC board from Force Computers Ltd (CPU-2CE) and an ERC32 simulator supplied by ESTEC, SIS version 1.4. This demonstrated that the main functionality was implemented correctly.

Following this acceptance, it was agreed that the ACS was to be modified for the revised ERC32 chipset and to allow acceptance to be carried out on an actual ERC32 board.

In addition, extra functionality was added for indirect procedure calls, interface to C and to control the ROM data size on Preelaboration.

As a result, the ACS development was completed with acceptance using a DEM32 board with the revised ERC32 chipset.

5.1.1.2.2. HRT Support Tools

This work package was split into two sub-phases, known as phase 2a and phase 2b, with each having three sub-packages corresponding to one for each tool development.

Phase 2a consisted of requirements consolidation and preliminary architectural design, while phase 2b took the preliminary design and worked through to complete the implementation and acceptance of the tools.

During the development advanced deliveries of the tools were provided. These allowed an initial assessment of the functionality and were used to identify refinements before the actual acceptance. In addition, the ERC32 Target Simulator was directed towards the revised ERC32 chipset.

The development of each of the tools followed the same lifecycle.

Phase 2a consisted of a mid-term Baseline Consolidation Review (BCR) and was completed by a Preliminary Architectural Design Review (PADR).

Phase 2b consisted of an Architectural Design Review (ADR), which followed finalisation of the design and then consisted of a Code and Unit Test Key Point (CUT/KP), which coincided with completion of development and provision of advanced release; an Integration Review (IR), which completed testing and reviewed the acceptance tests; and the Acceptance Review (AR), at which the tools underwent formal acceptance.

5.1.1.2.2.1. ERC32 Target Simulator

The ERC32 Target Simulator development differs from the other tools in that it was primarily an integration of a number of existing products, with the addition of some specific functionality.

The core of the tool is based upon SPARCSim, the SPARC architecture simulation from Sun Microsystems. Added to this to provide the ATAC functionality was ATACsim, produced by R-Tech the developers of the ATAC. Around this it was decided to use TCL/TK to integrate these products and to provide the user interface.

The additional functionality focused on providing additional interfaces to the ERC32 Target Simulator to allow a user to set up a complete external environment (e.g. define DMA transfers and external I/O) and to allow connection of the ACS debugger to the simulator.

The connection to the debugger used existing technology for communication between the debugger and a simulator. This relies on an implementation of a gateway to mimic the actions of the monitor typically used for embedded debugging. An object library of the debugger to gateway communications protocol was provided, which required an implementation of a procedural interface to translate the monitor commands to ERC32 Target Simulator commands.

An advanced delivery was provided and allowed minor modifications to be made to the functionality and user interface. The major change to the user interface during the development however, was the introduction of the scoreboard window. This provides a user-defined snapshot of the processor upon occurrence of a break in simulation.

The acceptance of the ERC32 Target Simulator was performed using a set of scripts, which verified the complete operation, including detailed instruction timing, of the ERC32 Target Simulator.

5.1.1.2.2.2. Schedulability Analyser

The Schedulability Analyser development was focused on correct definition and implementation of the analysis required.

During phase 2a, the requirements specification was overhauled and subject to an in-depth review.

While during phase 2b, precise implementation was continually discussed.

The advanced delivery provided the opportunity to assess the implementation. A number of comments and error reports were also provided. In particular, the process identified a number of refinements that could be made to the tool to provide more precise results.

In particular, the analysis model was changed to calculate blocking time based on protected objects within the worse case paths of threads. In addition, an alternate analysis was introduced to provide a report to the user in case blocking time was increased when all protected objects were considered.

As a result, the tool was modified in order to incorporate these refinements and errors identified.

The acceptance of the tool was based on a set of test cases ensured that each of the requirements were correctly implemented and performed correctly for a realistic test application.

5.1.1.2.2.3. Scheduler Simulator

The Scheduler Simulator development was similarly focused as the Schedulability Analyser and a similar approach was taken to the development.

Phase 2a consisted of an in-depth examination of the requirements at the BCR.

The advanced delivery of this tools also provided the opportunity to assess the implementation and a number of refinements and errors were identified. This resulted in an in-depth analysis of the modelling of the Ada Run-Time System operations and their accounting. These were summarised in a technical note, which formed the basis of the modifications performed.

The simulation model underwent a significant change in the accounting mechanism and also the user interface was modified so that the user was able to clearly see the RTS operations performed within a thread.

Again, the acceptance of the tool was based on a set of test cases ensured that each of the requirements were correctly implemented and performed correctly for a realistic test application.

5.1.1.2.3. SW Support Environment User Guide

A user manual was produced for each of the individual tools by the respective development. In addition, an overall user guide was produced to illustrate the use of the toolset in developing a HRT application.

The manual was developed from the experience of earlier ESTEC programmes, such as HRTOSK [RD1 & RD2] and HESSE [RD3] as well as the experience gained on this programme.

5.1.1.2.4. Overall Co-ordination and Management

This activity controlled the software part of the ERC32 programme. It is split into two sub-packages.

5.1.1.2.4.1. Software Management

This activity consisted of overall day-to-day management of the subcontractors and also the technical direction. As such, it included the planning and reporting of the developments.

One critical activity was the co-ordination of the debugger and target simulator interface between the subcontractors. Due to the nature of the interface and the unusual mechanism, it was necessary to ensure that dialogue was maintained during the integration of the target simulator.

In addition, this activity included the consolidation activities, which took place following the advanced deliveries of the tools.

Primary amongst this was the production of the HRT Requirements Consolidation note and ERC32 Simulator Requirements Consolidation note. The former of these provided the basis for the latest modification to the Schedulability Analyser and Scheduler Simulator . The second summarised the changes to the ERC32 Target Simulator requirements due to the change in ERC32 chipset functionality.

5.1.1.2.4.2. Target Simulator Consultancy

This activity provided for the involvement of MMS and in particular their knowledge of the requirements and operations of a target simulator. Input was provided during the requirements consolidation stage to facilitate the understanding of the requirements and especially the important role a target simulator can provide in overall system development.

5.1.1.2.5. Acceptance of Tools

Originally, this task was to provide for the integration and validation of the toolset. However, the re-direction restricted it to the process of accepting the individual tools.

5.1.1.2.5.1. Tool Acceptance

The acceptance of each of the tools was ultimately based on a formal acceptance test and review.

However, prior to this milestone reviews were carried out and the acceptance tests themselves were subject to review.

5.2. Design, Implementation and Verification

5.2.1. Design and Implementation

5.2.1.1. Design methodology

There was no standard design methodology used for the tools. However, this was primarily due to the different heritage.

The ACS is baselined on existing Alsys/TSP technology and therefore the implementation consisted of modifying that technology. Therefore the nominal company design approach was used.

In the first instant each requirement was designated as being either within the baseline technology or as a functional enhancement see ACS design document [GD21]. For the functional enhancements, the implementation within the baseline technology was defined and these were documented in a six design specifications, each of which covered a functional entity of the ACS, see [GD22-27].

For the ERC32 Target Simulator, Schedulability Analyser and Scheduler Simulator the HOOD method was used. The use of the method was, however, limited to the architectural design phase.

Primarily, this was due to the use of the HOOD method for detailed design, i.e. to produce PDL was considered onerous for a development of this type. Therefore, the level of detail in the architectural design was such that it was possible to move straight to the coding phase.

In addition, there were a number of design decisions taken which meant that the value of the detailed design would be limited. For the ERC32 Target Simulator, the implementation was based around the existing SPARCsim product produced by Sun and the TCL/TK tool suite. The impact of this was that the activity for this tool became primarily one of integration.

TCL/TK was also used for the Schedulability Analyser and Scheduler Simulator.

5.2.1.2. Design Decisions

There were a number of design decisions taken during the programme, some of which were fundamental to the operation of the tools, the following sections provide a summary of these.

5.2.1.2.1. ACS

The following were the main decisions for the ACS:

Interrupt Handling Model

This issue concerned the handling of interrupts in the context of developing HRT software, in particular to implement the interrupt sporadic trigger in Ada. Originally, it was intended that an interrupt handler should call an entry in a passive task. This entry call would then open (i.e. set to true) a guarded entry upon which a sporadic task would wait. This mirrors the Ada95 model of interrupt handlers calling into protected objects.

However the ATAC uses the classic Ada83 model of *interrupt entries* whereby the interrupt handler performs an entry call directly to the waiting sporadic task, which was called the *Rendezvous model*. This meant that in order to support both an ATAC and non-ATAC environment, two interrupt handling mechanisms would be required.

Consequently, it was decided that it was preferable to have source compatibility between the two hardware environments and so the Rendezvous model was adopted in the non-ATAC runtime.

Priority Range

The original intention of the ERC32 software programme was to support a large range of priorities. This would ensure that the user could design a HRT application without being constrained by the a limitation in the number of tasks. Arbitrarily this number was fixed at 128. It had also been determined that for simplicity purposes the selection of and ATAC or non-ATAC Run Time System should be a bind time decision rather than have the user specify this in the code or at compilation (through selection of an ATAC or non-ATAC package system).

However, the ATAC is limited in its support for priorities to 64 priority levels. During the detailed design of the compiler, it was realized that this would mean that if a priority of greater than 64 was assigned for a task when the ATAC was in use a *PROGRAM_ERROR* would be raised. Although this may have been acceptable, it also meant that some of the ACVC tests would not be able to pass.

Consequently, it was decided to restrict the priority range for both the ATAC and non-ATAC RTS to 1 .. 63.

Ada Debugger Support for ERC32 Simulator Commands

It was important that the ACS and the ERC32 Target Simulator would run together. In addition, it was considered beneficial if ERC32 Target Simulator native commands could be executed through the debugger, when they did not conflict with the debugger operation.

Therefore, it was decided to incorporate a facility in the debugger that allowed the user to input ERC32 Target Simulator commands and for the resultant output to be displayed. Consequently, the *TRANSPARENT* command was added to allow such commanding. Also in order to ensure that there was no conflict, functionality was added to the ERC32 Target Simulator to vet the commands before executing them.

5.2.1.2.2. ERC32 Target Simulator

The design decisions for the ERC32 Target Simulator are detailed in the architectural design document [GD49], the following summarise these:

Debugger/Target Simulator Interface

There were a number of options for providing this interface. The simplest form was to treat the ERC32 Target Simulator as another instance of a board and to run the ACS *monitor* with the application running on top of this. However, this would have

immediately introduced a time overhead in execution and debugging programs. Given that both the tools were being developed within the scope of the ERC32 programme it was decided to investigate alternatives.

It was decided that a more efficient and effective mechanism for this interface would be for the ERC32 Target Simulator to perform the function of the *monitor* and implement the internal details.

The Alsys/TSP debugger already used a protocol for communication with the target monitor and therefore it was considered appropriate to utilise the same mechanism. This resulted in the identification of a *monitor gateway* program which took the debugger protocol units and translated them to simulator operations. The approach also meant that the actual communications handling between the debugger and the *monitor* could be reused .

Integration

The primary decision for the implementation of the ERC32 Target Simulator was the mechanism by which the various component parts were to be integrated. The ERC32 Target Simulator core was to be provided by SPARCsim, while there was an interface to the ACS debugger, an interface for environment simulation and a user interface. These placed constraints on the design as there was a need for integration, while at the same time some of these need to be separate processes.

The solution was to place SPARCsim into a simulator core, which performed the simulation and served the user interface, which also provided the input windows for environment simulation, and the *monitor gateway*. The UNIX pipe mechanism was selected to provide the communications, with the simulator core interpreting strings input from either the user interface or the *monitor gateway*.

User Interface

In addition to the interface from the debugger there was a requirement for the ERC32 Target Simulator to provide an interface for use in a stand-alone mode. A number of alternatives were available.

SPARCsim itself contains a debugger, however this would have required a significant amount of modification. An alternative was the GNU debugger, but this was considered too rich and too large, especially since it also required modification. Consequently, it was decided to use the Tool Command Language (TCL/TK) package.

This provided the necessary user interface command parsing, but also had the advantage of being easy to integrate to an existing product and could be used to encapsulate existing or new functions. So for example, it was possible to incorporate existing GNU functions to allow loading of different formats.

5.2.1.3. Schedulability Analyser and Scheduler Simulator

Analysis and Modelling of application

The algorithms for performing the analysis and simulation are critical to the accuracy of the results. This programme has paid close attention to ensure that the algorithms do not present a incorrect result (i.e. schedulable declared for an unschedulable application), but that the results are not unduly pessimistic.

This requires attention not only to be paid to the actual definition of the algorithms, but also the definition of application and the understanding of the RTS. During phase 1 and the early part of phase 2, the development focused on ensuring that the definition and information required for the analysis and simulation was complete and well-understood.

The algorithms were based on those identified in the HRTOSK [RD1 & RD2] project, but were refined based on the exact RTS being implemented for the ERC32 programme. For the schedulability analysis, the refinement concentrated on accurate incorporation of blocking. While for the scheduling simulation, it was necessary to ensure that the time calculation was correct and that the allocation of processing time to tasks was appropriate. During the development, particularly following the advanced delivery of the tools, both of these were refined.

For the information requirements of the schedulability analysis and simulation, it was decided to partition the information into two parts, the application definition and the RTS definition. The application definition was then further sub-divided to the user definable part and the automatically generated part. The result was the three files: the RTS file, for the Run-Time characteristics; the UCF, for the user defined task characteristic such as period and criticality; and the ESF, for the execution profile of the tasks.

Syntax for Formulae in RTS Characteristics file

In order to perform accurate analysis and simulation of the scheduling activities of an application it is important that the RTS timing information is as precise as possible. Initially, the RTS timing information was based on a model of the RTS operations and assigned a fixed time for each of the operations.

However, it became clear during the development that some of the operations were dependent on the number of tasks and/or the number of protected objects. For example the time for performing a *DELAY_UNTIL* operation is dependent on the number of tasks in the delay queue.

Therefore the syntax for the use of formulae in expressing runtime overheads which are proportional to some count was added to the RTS characteristics file. This allowed for the maximum value to be used in the Schedulability Analyser, while the Scheduler Simulator was able to perform RTS operations precisely according to the number of tasks in a queue.

5.2.2. Verification

For each tool a verification and validation plan was produced. The actual technique differed reflecting the heritage and nature of the tools.

5.2.2.1. Ada Compilation System

The ACS being baselined on an existing product already had a defined qualification approach. This is internal to Alsys/TSP and consists of running the ACVC, a number of specific tests, e.g. to verify the user interface, and the running of some standard benchmarks.

Therefore, the validation approach adopted was to use this basis and to build specific tests for the additional functionality being developed for the ERC32 programme. An SVVP [GD29] was produced detailed the test cases for each of the functional enhancements. These were then converted into test cases and added to the company qualification approach. In addition to these tests being added a set of benchmarks were provided. Testing was performed using a Force Computers board first complemented with testing on DEM32 V2 board.

The complete qualification including the functional enhancements was performed and the benchmark results were provided as part of this report produced.

5.2.2.2. ERC32 Target Simulator

The ERC32 Target Simulator VTP contained a definition of the test cases to be run and also provided a trace to the requirements. This was complemented with a Validation Test Procedures document which detailed the exact tests were to be performed.

Although the test cases concentrated primarily on showing the requirements were met (e.g. each instruction was tested) they also included the execution of benchmark programs, such as JIAWG and ESTEC-B, which had been generated by using the ACS.

Primarily, the verification and validation was performed using the simulator command interface, through which the procedures were entered. Typically each test corresponded to a command procedure, which included the verification of the test also.

5.2.2.3. Schedulability Analyser and Scheduler Simulator

For the Schedulability Analyser and Scheduler Simulator the VTPs identified each test case, with a trace to the requirement to be verified. Again, validation test procedures were produced for each of the test cases.

In addition, both tools included a realistic example of an application for testing. This was based on the Olympus Attitude and Orbit Control System, which had been created as an example for the HRTOSK programme [RD1 & RD2].

The test procedures were again primarily automated using Perl and Awk scripts to verify the output in response to each test case.

5.3. Problems Encountered

There were three problem areas within the ERC32 software development. These related to the target simulator/debugger interface, the first version of ERC32 chipset, ATAC and the HRT analysis and simulation. A summary of each of these follows.

TS/ACS interface:

The TS/ACS interface design approach was intended to provide a more integrated solution than available with the normal target/debugger situation. This took advantage of the fact that both tools were being developed/enhanced under the same programme and therefore benefits in terms of performance could be gained by integrating. The use of the Alsys/TSP provided library to handle the communications between the debugger and the target simulator required a strong definition of the interface. However, although this had been performed within Alsys/TSP on a number of occasions, this was the first time the interface had been provided to an external company for integration.

The interface was to be provided at a procedural level and although a document defining the interface was provided, it proved incomplete for a detailed implementation.

Therefore, in order to address the discrepancies, it was necessary to set up a discussion process between Alsys/TSP and Spacebel. The process was very slow to operate, primarily because there was no official “deliverable” between the two companies. Despite these problems, the subsequent completion of development and acceptance showed that the interface operated correctly in all aspects, except for a very minor limitation in performing a step operation.

ERC32 chipset, ATAC:

The DEM32 V1 board primarily presented a problem to the ACS development, but also had a minor impact on the ERC32 Target Simulator development.

The ACS development required the DEM32 board in order to perform testing and provide the basis for acceptance. However, the ERC32 chipset had some failings.

As a consequence the initial acceptance of the ACS had to take place on a replacement board , Force SPARC, complemented tests with the ESTEC simulator. The lack of acceptance on an actual board presented problems.

However, the chipset underwent a re-work and the acceptance testing was completed on DEM32 V2 board.

During acceptance testing ATAC HW failings where detected. Work arounds for these problems where presented and ATAC testing where completed on DEM32 V2 board.

HRT analysis and simulation:

The problems encountered in this area were due to the evolution of the requirements. Primarily, this was related to the fact that a clear understanding of the detailed requirements could only become achieved as the implementation progressed. During requirements definition in phase 1 and phase 2a, it was well understood what the purpose of the HRT tools was, however the requirements addressed the high level functionality only. As the development progressed, those issues that had not been addressed became evident (e.g. allocation of Run-Time execution times to tasks).

Although, some of these were resolved during the development, the advanced deliveries of the HRT tools showed that there were disparate views in the detailed implementation. However, the advanced deliveries allowed a more thorough assessment of these detailed requirements and their refinement was performed during Development Finalisation Phase.

6. ACHIEVED RESULTS FOR THE ERC32 SOFTWARE

This chapter describes the functionality of the ERC32 tools, performance where appropriate and the currently known limitations and bugs.

All tools have completed acceptance testing successfully.

6.1. Description of the ERC32 Toolset

The software toolset has been developed to provide an environment to produce embedded software for the ERC32 processor. The toolset provides all the necessary support for the coding and testing of an real-time software which is typical of the on-board space applications. This allows the development of the more complex and flexible software, which is to be expected from future space missions, while maintaining a high degree of confidence in meeting the timing constraints of the application.

The toolset provides both the tools typically associated with an embedded software development environment, a cross-compiler and a target simulator, and also functionality to allow the exploitation of the benefits offered by Priority Based Preemptive Scheduling (PBPS) and the Ada Tasking Coprocessor.

This additional functionality uses the advances made in recent years in the area of Hard Real Time (HRT) systems. This research has identified techniques which allow the use of Ada tasking, through PBPS based on Rate Monotonic Scheduling (RMS) and Deadline Monotonic Scheduling (DMS) [RD1 & RD2]. Within this research algorithms have been developed to calculate the timing requirements of software developed using these techniques. For this programme, these algorithms have been extended to implement Arbitary Deadline Scheduling (ADS) in addition to DMS.

The toolset builds on a number of previous ESA-ESTEC programmes to investigate these HRT techniques and to promote their use within the space industry.

Two of these programmes, the Hard Real-Time Operating System Kernel (HRTOSK) programme [RD1 & RD2] and the HRT Embedded Software Support Environment (HESSE) [RD3], established a framework for the practical application of these HRT techniques by identifying how the Ada language could be used to support PBPS schemes. The toolset has implemented the tools to perform HRT analysis and simulation.

The toolset consists of the following:

- an Ada compilation environment for the generation and debugging of programs, with specific functionality for developing HRT applications;
- a Schedulability Analyser that provides an analytical assessment of the schedulability of HRT programs;
- a Scheduler Simulator that provides graphical and textual representation of the scheduling behaviour of an HRT program;

- a ERC32 Target Simulator and/or target breadboard to provide a realistic environment on which to run an application.

The following diagram provides a logical representation of the structure of the toolset:

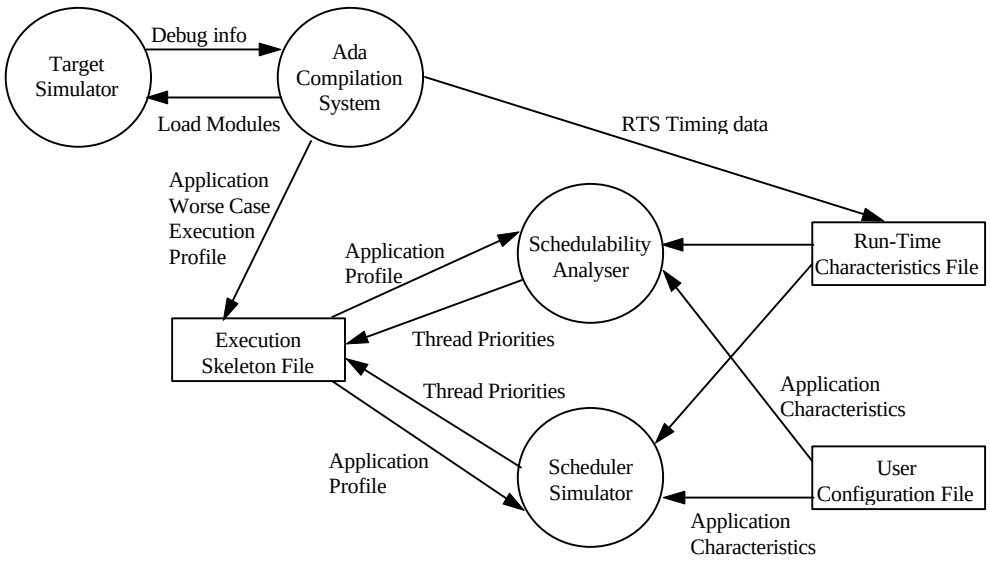

Figure 8: Logical Model of ERC32 Software Toolset

6.1.1. Functionalities

6.1.1.1. Ada Compilation System

The ACS allows for compilation, binding and linking of the application source code. The tool is based on the standard Alsys/TSP baseline compiler system technology, which is well-proven in a range of application areas. This baseline technology provides a complete Ada development environment, with a command line and GUI interface, and provides a Board Support Package capability to tailor the code generation and execution to a specific instance of an embedded processor (e.g. interrupts masking).

The compiler supports Ada 83, with specific modification and extensions introduced to support HRT analysis. Specifically, these are:

- Pragma PASSIVE for the implementation of protected objects (passive tasks) which correspond to Ada 95 protected types, including provision of Priority Ceiling Emulation (PCE) for the implementation of IPCI. Passive tasks provide mutual exclusion and allow the user to perform synchronisation between tasks and protected access to data, for example:

```
TASK Buffer IS
   PRAGMA PRIORITY (30);
   PRAGMA PASSIVE;
   ENTRY Read (Item : out Data_Item);
   ENTRY Write (Item : in Data_Item);
END Buffer;

TASK BODY Buffer IS
   Item_Store : Data_Item;
BEGIN
   Forever:
   LOOP
      SELECT
         ACCEPT Read (Item : out  Data_Item) DO
            Item:= Item_Store;
      OR
         ACCEPT Write (Item : in Data_Item) DO
            Item_Store := Item;
      OR
         TERMINATE;
      END SELECT;
   END LOOP Forever;
END Buffer;
```

- Accurate delay timing through absolute delays (i.e. a DELAY_UNTIL construct);

```
   LOOP
      DELAY_UNTIL (Writer_Time);
      Store.Item := Write_Item;
      Controller.Signal;
      Write_item := Write_Item + 1;
      Writer_Time := Writer_Time + Period;
   END LOOP Period_Action;
```

- Up to 64 distinct priority levels to provide large range of unique task priority assignment and also a Dynamic Priorities package to allow the user full control over priority assignment;
- A user option to Pre-elaborate data for inclusion in ROM images. The user is also able to control the size of this pre-elaboration;
- Provision of package to provide for dynamic code replacement;
- Integration of the RTS with the Ada Tasking Coprocessor (ATAC) chip, though a user bind time option;
- Tailoring of the RTS to ensure time bounding of RTS operations used in constructing a HRT application. The timing characteristics of the RTS operations have been measured and recorded;
- Tracing of RTS scheduling activity to allow comparison with scheduling analysis provided by the Schedulability Analyser and scheduler simulator;
- A user option to generate worse case timing data for each task in an application;
- Provision of an interface to allow the import of C and assembler code and the export of Ada code and structures to C programs;
- Watchpoint debugging capability through the ERC32 hardware functionality.

The tool also provides full debugging capability via an interactive debugger, which may run on either the ERC32 Target Simulator, developed as part of the tool set, or on the target ERC32 processor.

The ACS is fully configurable for any ERC32 configuration through a BSP which may be tailored by the user.

6.1.1.2. Hard Real-Time Tools

The Schedulability Analyser Tool and the Scheduler Simulator Tool provide the means by which the user may conduct a comprehensive analysis of the HRT system at a very early stage in the lifecycle. Priority assignment of all the tasks constituting the application takes place automatically within each tool, and is based upon the scheduling algorithm (Arbitrary Deadline Scheduling, ADS, or Deadline Monotonic Scheduling, DMS) and the blocking protocol selected by the user (Immediate Priority Ceiling Inheritance, IPCI, or Inhibit Interrupts, INHIB), as well as the interaction between tasks.

The user provides both tools with an outline description of the application, the Execution Skeleton File, and a characterisation of the RTS (the Run Time System File), as well as a User Configuration File detailing the salient HRT characteristics.

Both tools are suitably equipped to handle the inclusion of the ATAC within the HRT system.

Execution Skeleton

A file built either by the Ada Compilation system WCETE functionality or by the user. This identifies the calculated worst case timings and synchronisation points for the threads in an application. It is supplemented with the thread priorities when these are calculated by the Schedulability Analyser or scheduler simulator.

For example:

```
PROGRAM HRT
   THREAD Real_Time_Clock
      TYPE CYCLIC
      PRIORITY 122
      WCET 430,0,0
   END

   THREAD Read_Bus_IP
      TYPE CYCLIC
      PRIORITY 120
      CALL_PO Bus_Ip_Fifo Pop
      WCET 2134,0,0
      CALL_PO TC_PO Send_TC
   END
END
```

User Configuration File

This file is the primary source of user input, currently produced by the user through a UNIX editor. The file contains the scheduling approach (DMS or ADS) and the blocking protocol (IPCI, immediate priority ceiling inheritance, or, INHIB, interrupt inhibit) against which the analysis and simulation takes place; the scheduling characteristics of the threads of an application; override or default worse case execution times for operations or threads; and the processor clock speed and memory characteristics for a particular application.

For example:

```
PREFERENCES
   DMS
   BLOCKING PROTOCOL IPCI
END

TARGET CHARACTERISTICS
   NO ATAC
   PRIORITY LOW 1
   PRIORITY HIGH 124
   CPU CLOCK FREQUENCY 1 MHz
   WAIT STATES READ 0 WRITE 0
END


THREAD DEFINITION

   THREAD Real_T ime_Clock
   CRITICALITY HARD
      PERIOD   50000
      DEADLINE  9000
      OFFSET    0
   END

   THREAD Read_Bus_IP
      CRITICALITY HARD
      PERIOD   10000
      DEADLINE 10000
      OFFSET    0
   END
```

RTS Characteristics File

The RTS characteristics file is provided by the Ada compilation system vendor and contains the worst case timing characteristics associated with the RTS provided with the compilation system. The file contains the timings, in processor and memory clock cycles, for each non pre-empting RTS operation and the entry and exit times for each pre-empting operation. The timings may contain a part which is proportional to the number of threads in an application or undergoing a particular process at the time of the operation (e.g. those in the delay queue for a delay operation).

For example:

```
-- RTS Characteristics for DEM32 at 10 MHz (non-ATAC runtime)
-- Characteristics are in Processor Cycles
ENTER PASSIVE TASK 80,0,0
EXIT PASSIVE TASK 110,0,0
ENTER SOFTWARE SPORADIC WAIT (70,0,0)
EXIT SOFTWARE SPORADIC WAIT 30,0,0
ENTRY QUEUE SERVICING (60,0,0)
CONTEXT SWITCH TIME 340,0,0
SCHEDULER SELECT TIME (50,0,0)
ENTER DELAY UNTIL OVERHEAD AT HEAD (390,0,0)
ENTER DELAY UNTIL OVERHEAD NOT AT HEAD (220,0,0) + (30,0,0) * C
EXIT DELAY UNTIL OVERHEAD 80,0,0
TIMER INTERRUPT OVERHEAD 210,0,0
READY AFTER DELAY 120,0,0
INTERRUPT HANDLING OVERHEAD 670,0,0
ENTER INTERRUPT SPORADIC WAIT (30,0,0)
EXIT INTERRUPT SPORADIC WAIT 30,0,0
MAXIMUM NON PREEMPTION 1300,0,0
```

6.1.1.2.1. Schedulability Analyser

The Schedulability Analyser Tool is used to carry out a complete timing analysis of the application without having to execute the application: an analysis report, summarising the results of the timing analysis, is produced and basically informs the user whether or not an application is schedulable (i.e. is it able to meet all of its deadlines). This report also provides information pertaining to the processor usage and quantifies how much headroom, in terms of computation time, is left in the HRT system before any deadline is violated.

Specifically, the tool is able to report:

- overall thread utilisation and application schedulability;
- individual task worst case computation time (processing time for task) and worst case response time (time from release of task to task completion);
- blocking time suffered by task;
- detailed individual task schedulability based on deadline and worst case response time;

- margin of sensitivity for a task, to identify the increase in computation time the task may have while still remaining schedulable;

6.1.1.2.2. Scheduler Simulator

The Schedulability Simulator Tool performs detailed simulations of the scheduling behaviour of the application and provides an indication of when particular threads are executed and the interaction between one thread and another and between the thread and the Run Time System.

It highlights main scheduling events including missed deadlines, task initialisation and all RTS processor usage.

In effect it allows the user to visualise the scheduling and execution of tasks. The user may also explore the effect of various release patterns of interrupts upon the HRT system. The user may obtain outputs in both graphical and textual forms.

Specifically the tool:

- automatically generates priorities for tasks based on the scheduling algorithm;
- configuration of external interrupts according to a number of different distribution functions;
- reports all scheduling events, both on-line and to file to allow later analysis, with the on-line reporting being user-definable;
- generates either textual or graphical (Gantt charts) reports from file;
- allows searching for specific scheduling events in an application trace;
- generates utilisation data for individual tasks, the RTS and the overall application;

6.1.1.3. ERC32 Target Simulator

The ERC32 Target Simulator Tool provides an accurate and fast simulation of the execution of the application code on an ERC32 microprocessor. The behaviour of the ERC32 core and the ATAC (should the user so wish) are all comprehensively modelled.

The operation of the simulator can be controlled via the debugger that comes as part of the ACS tool, or if the user prefers, the simulator can be operated in stand-alone mode using its own dedicated interface.

The tool has a powerful interface based on the TCL/TK tool allowing the user to specifically tailor the interface and also add functionality which simulates the various ERC32 interfaces e.g. I/O and DMA.

Specifically, the ERC32 Target Simulator:

- simulates the IU, FPU and MEC of the ERC32 and the ATAC to clock level accuracy;
- is able to simulate asynchronous events such as timers, interrupts, DMA, UART and I/O activity;
- is configurable to allow a user to exactly simulate a Board implementation of the ERC32;
- provides a fast simulation of the hardware, approaching 1% of chipset operation;
- a user definable scoreboard, which is updated on occurrence of simulator events, e.g. breaks, watchpoints;
- provides a flexible interface through UNIX pipes, that allows integration of the simulator with a spacecraft level simulator and/or test environment;

In addition, the simulator can be integrated with the Ada Debugger to provide an Ada source level debugging facility for HRT applications.

6.1.2. Performance

Performance figures are available only for the Ada Compilation System and the ERC32 Target Simulator. These have been generated using standard benchmarks such as, PIWG and JIAWG and also ESTEC-B.

The following sections provide the results of running the various benchmark tests at the tool acceptance.

6.1.2.1. ACS benchmarks

The following benchmark programs have been run with DEM32 board at 10 MHz:

6.1.2.1.1. PIWG

The following PIWG results were obtained without ATAC:

	CHECKS=ALL	CHECKS=STACK	CHECKS=NONE
A000091 (µs)	102.00 (+/- 5.100)	86.60 (+/- 4.330)	82.70 (+/- 4.135)
A000092 (ms)	497.77 (+/- 24.888)	492.97 (+/- 24.648)	492.96 (+/- 24.648)
A000092 (kwips)	2009	2029	2029
A000093 (ms)	276.34 (+/- 13.817)	272.17 (+/- 13.608)	270.90 (+/- 13.545)
A000093 (kwips)	3619	3674	3691
A000094A (s)	0.21 (+/- 0.011)	0.20 (+/- 0.010)	0.18 (+/- 0.009)
A000094B (s)	0.28 (+/- 0.014)	0.25 (+/- 0.013)	0.24 (+/- 0.012)
A000094C (s)	0.26 (+/- 0.013)	0.18 (+/- 0.009)	0.18 (+/- 0.009)
A000094D (s)	1.01 (+/- 0.050)	0.42 (+/- 0.021)	0.42 (+/- 0.021)
A000094E (s)	0.83 (+/- 0.041)	0.19 (+/- 0.009)	0.19 (+/- 0.009)
A000094F (s)	1.53 (+/- 0.077)	1.16 (+/- 0.058)	1.16 (+/- 0.058)
A000094G (s)	0.18 (+/- 0.009)	0.14 (+/- 0.007)	0.14 (+/- 0.007)
A000094H (s)	0.31 (+/- 0.015)	0.31 (+/- 0.015)	0.31 (+/- 0.015)
A000094I (s)	0.29 (+/- 0.014)	0.26 (+/- 0.013)	0.25 (+/- 0.013)
A000094J (s)	0.39 (+/- 0.020)	0.31 (+/- 0.016)	0.31 (+/- 0.016)
A000094K (s)	12.33 (+/- 0.616)	12.40 (+/- 0.620)	11.70 (+/- 0.585)
B000001A (ms)	1.84 (+/- 0.092)	1.60 (+/- 0.080)	1.59 (+/- 0.080)
B000001B (ms)	1.61 (+/- 0.080)	1.60 (+/- 0.080)	1.59 (+/- 0.080)
B000002A (µs)	962.29 (+/- 48.114)	518.59 (+/- 25.930)	517.69 (+/- 25.885)
B000002B (µs)	528.69 (+/- 26.434)	518.59 (+/- 25.930)	517.69 (+/- 25.885)
B000003A (µs)	1.39 (+/- 0.069)	791.69 (+/- 39.584)	790.49 (+/- 39.525)
B000003B (µs)	803.39 (+/- 40.169)	791.69 (+/- 39.585)	790.49 (+/- 39.525)
B000004A (µs)	999.54 (+/- 49.977)	527.69 (+/- 26.385)	526.79 (+/- 26.340)
B000004B (µs)	539.50 (+/- 26.975)	527.69 (+/- 26.385)	526.79 (+/- 26.340)
B000010 (ms)	1037.82 (+/- 51.891)	1011.82 (+/- 50.591)	1001.37 (+/- 50.068)
B000011 (ms)	198.30 (+/- 9.915)	166.36 (+/- 8.318)	166.37 (+/- 8.318)
B000013 (ms)	20.38 (+/- 1.019)	18.50 (+/- 0.925)	18.50 (+/- 0.925)
C000001 (µs)	531.44 (+/- 26.572)	531.41 (+/- 26.570)	530.41 (+/- 26.521)
C000002 (µs)	539.84 (+/- 26.992)	539.13 (+/- 26.957)	538.90 (+/- 26.945)
C000003 (µs)	525.89 (+/- 26.295)	526.35 (+/- 26.318)	525.76 (+/- 26.288)
D000001 (µs)	3.40 (+/- 0.170)	2.90 (+/- 0.145)	2.50 (+/- 0.125)
D000002 (µs)	776.66 (+/- 88.833)	703.89 (+/- 35.194)	703.51 (+/- 35.175)
D000003 (µs)	9.50 (+/- 0.475)	7.10 (+/- 0.355)	5.80 (+/- 0.290)
D000004 (µs)	565.00 (+/- 128.250)	807.23 (+/- 40.361)	806.20 (+/- 40.310)
E000001 (µs)	87.69 (+/- 4.385)	86.89 (+/- 4.345)	87.10 (+/- 4.355)
E000002 (µs)	147.70 (+/- 7.385)	139.99 (+/- 7.000)	140.50 (+/- 7.025)
E000003 (µs)	211.80 (+/- 10.590)	211.99 (+/- 10.600)	212.79 (+/- 10.640)
E000004 (µs)	251.50 (+/- 12.575)	259.00 (+/- 12.950)	258.80 (+/- 12.940)
E000005 (µs)	507.51 (+/- 25.375)	499.31 (+/- 24.965)	498.98 (+/- 24.949)
F000001 (µs)	0.35 (+/- 0.017)	0.35 (+/- 0.017)	0.35 (+/- 0.017)
F000002 (µs)	0.45 (+/- 0.022)	0.45 (+/- 0.022)	0.45 (+/- 0.022)
G000005 (µs)	167.12 (+/- 8.356)	167.12 (+/- 8.356)	167.11 (+/- 8.355)
G000006 (µs)	609.35 (+/- 30.468)	609.35 (+/- 30.468)	609.35 (+/- 30.468)
H000001 (µs)	9.70 (+/- 0.485)	9.70 (+/- 0.485)	9.70 (+/- 0.485)
H000002 (µs)	20.51 (+/- 1.025)	20.51 (+/- 1.025)	20.51 (+/- 1.025)
H000003 (µs)	140.01 (+/- 7.001)	144.82 (+/- 7.241)	144.82 (+/- 7.241)
H000004 (µs)	87.21 (+/- 4.360)	87.21 (+/- 4.360)	87.21 (+/- 4.361)

H000005 (µs)	87.21 (+/- 4.360)	87.21 (+/- 4.360)	87.21 (+/- 4.361)
H000006 (µs)	6.10 (+/- 0.305)	6.30 (+/- 0.315)	6.30 (+/- 0.315)
H000007 (µs)	18.63 (+/- 0.932)	18.64 (+/- 0.932)	18.63 (+/- 0.932)
H000008 (µs)	4.30 (+/- 0.215)	3.90 (+/- 0.195)	3.90 (+/- 0.195)
H000009 (µs)	26.80 (+/- 1.340)	26.80 (+/- 1.340)	26.20 (+/- 1.310)
L000001 (µs)	0.35 (+/- 0.018)	0.35 (+/- 0.018)	0.35 (+/- 0.018)
L000002 (µs)	0.45 (+/- 0.023)	0.45 (+/- 0.023)	0.45 (+/- 0.023)
L000003 (µs)	0.55 (+/- 0.028)	0.35 (+/- 0.018)	0.35 (+/- 0.018)
L000004 (µs)	0.38 (+/- 0.019)	0.38 (+/- 0.019)	0.38 (+/- 0.019)
L000005 (µs)	0.38 (+/- 0.019)	0.38 (+/- 0.019)	0.38 (+/- 0.019)
P000001 (µs)	1.89 (+/- 0.094)	1.89 (+/- 0.094)	1.89 (+/- 0.094)
P000002 (µs)	1.10 (+/- 0.055)	1.10 (+/- 0.055)	0.80 (+/- 0.040)
P000003 (µs)	1.10 (+/- 0.055)	1.10 (+/- 0.055)	0.80 (+/- 0.040)
P000004 (µs)	0.60 (+/- 0.030)	1.10 (+/- 0.055)	0.80 (+/- 0.040)
P000005 (µs)	1.60 (+/- 0.080)	1.60 (+/- 0.080)	1.30 (+/- 0.065)
P000006 (µs)	1.50 (+/- 0.075)	1.50 (+/- 0.075)	1.20 (+/- 0.060)
P000007 (µs)	1.60 (+/- 0.080)	1.60 (+/- 0.080)	1.30 (+/- 0.065)
P000010 (µs)	3.90 (+/- 0.195)	3.90 (+/- 0.195)	3.60 (+/- 0.180)
P000011 (µs)	8.00 (+/- 0.400)	8.00 (+/- 0.400)	7.70 (+/- 0.385)
P000012 (µs)	4.60 (+/- 0.230)	4.60 (+/- 0.230)	4.30 (+/- 0.215)
P000013 (µs)	7.60 (+/- 0.380)	7.60 (+/- 0.380)	7.30 (+/- 0.365)
T000001 (µs)	107.20 (+/- 5.360)	106.60 (+/- 5.330)	107.10 (+/- 5.355)
T000002 (µs)	107.10 (+/- 5.355)	106.90 (+/- 5.345)	108.00 (+/- 5.400)
T000003 (µs)	108.00 (+/- 5.400)	108.05 (+/- 5.403)	108.00 (+/- 5.400)
T000004 (µs)	133.75 (+/- 6.687)	132.05 (+/- 6.603)	133.00 (+/- 6.650)
T000005 (µs)	105.98 (+/- 5.299)	106.24 (+/- 5.312)	106.80 (+/- 5.340)
T000006 (µs)	216.48 (+/- 10.824)	217.18 (+/- 10.859)	216.88 (+/- 10.844)
T000007 (µs)	63.95 (+/- 3.198)	63.85 (+/- 3.193)	64.45 (+/- 3.223)
T000008 (µs)	291.19 (+/- 14.559)	289.58 (+/- 14.479)	289.09 (+/- 14.454)

The following PIWG results were obtained without ATAC:

T000001 (µs)	92.90 (+/- 4.645)	92.90 (+/- 4.645)	92.60 (+/- 4.630)
T000002 (µs)	92.80 (+/- 4.640)	92.81 (+/- 4.640)	92.50 (+/- 4.625)
T000003 (µs)	93.60 (+/- 4.680)	93.55 (+/- 4.678)	93.10 (+/- 4.655)
T000004 (µs)	109.61 (+/- 5.480)	109.50 (+/- 5.475)	109.05 (+/- 5.452)
T000005 (µs)	92.00 (+/- 4.600)	92.00 (+/- 4.600)	91.70 (+/- 4.585)
T000006 (µs)	154.01 (+/- 7.701)	154.01 (+/- 7.700)	153.71 (+/- 7.685)
T000007 (µs)	51.90 (+/- 2.595)	51.90 (+/- 2.595)	51.90 (+/- 2.595)
T000008 (µs)	202.51 (+/- 10.126)	201.81 (+/- 10.090)	201.50 (+/- 10.075)

6.1.2.1.2. JIAWG

The following JIAWG results were obtained:

	CHECKS=ALL	CHECKS=NONE
CAP (ms)	4.10	2.90
DES1 (ms)	30.73	26.05
FLTMULT (ms)	364.88	221.39
INTMULT (ms)	523.57	486.57
KALMAN (ms)	211.31	181.15

6.1.2.1.3. ESTEC-B

The following ESTEC-B results were obtained from the ACS:

CHECKS=ALL

Q_step = 4 <=> max RMS error = 1.15 (Times are in milliseconds)

block	dct	r&q	bit	invbit	invr&q	invdct	Nbytes	RMSerr
1	1.75	0.44	1.20	1.48	0.39	2.02	26	0.18
2	2.03	0.56	1.67	2.08	0.68	2.11	44	0.13
3	1.96	0.64	1.90	2.16	0.77	2.07	57	0.13
4	1.98	0.63	1.84	2.16	0.81	2.18	53	0.00
Avrg:	1.93	0.57	1.65	1.97	0.66	2.10	45	0.11

Q_step = 8 <=> max RMS error = 2.31 (Times are in milliseconds)

block	dct	r&q	bit	invbit	invr&q	invdct	Nbytes	RMSerr
1	1.76	0.42	0.79	1.13	0.32	1.99	17	0.31
2	2.03	0.49	1.51	1.89	0.56	2.10	34	0.50
3	1.96	0.59	1.76	2.02	0.73	2.06	48	0.38
4	1.98	0.58	1.73	2.03	0.75	2.17	45	0.38
Avrg:	1.93	0.52	1.45	1.77	0.59	2.08	36	0.39

CHECKS=NONE

Q_step = 4 <=> max RMS error = 1.15 (Times are in milliseconds)

block	dct	r&q	bit	invbit	invr&q	invdct	Nbytes	RMSerr
1	1.52	0.32	0.98	1.19	0.27	1.66	26	0.18
2	1.76	0.44	1.37	1.71	0.51	1.74	44	0.13
3	1.68	0.52	1.55	1.75	0.58	1.73	57	0.13
4	1.71	0.51	1.50	1.77	0.62	1.81	53	0.00
Avrg:	1.67	0.45	1.35	1.60	0.49	1.74	45	0.11

Q_step = 8 <=> max RMS error = 2.31 (Times are in milliseconds)

block	dct	r&q	bit	invbit	invr&q	invdct	Nbytes	RMSerr
1	1.53	0.30	0.64	0.90	0.22	1.63	17	0.31
2	1.76	0.38	1.24	1.55	0.41	1.73	34	0.50
3	1.68	0.47	1.44	1.64	0.54	1.72	48	0.38
4	1.71	0.46	1.42	1.66	0.56	1.80	45	0.38
Avrg:	1.67	0.40	1.18	1.44	0.43	1.72	36	0.39

6.1.2.1.4. ERC32 Target Simulator

The performance of the ERC32 Target Simulator is dependent on the Host platform and also the level of environment simulation. The dependency on the level of environment simulation is because the simulator will access the environment interface for command information. Maximum performance is gained when no environment simulation is performed , see ERC32 Target Simulator User Manual [GD52] for information.

The product itself contains information on the performance enhancement and details of the expected operation. However, as a very generalised summary, on a standard current Sun SPARC machine performance of the order of 0.5%-1.0% of real-time can be expected. This of course will improve as the host platform processing improves.

6.1.3. Known Limitations and Bugs

The delivery and release notes for each of the tools list all known limitations and bugs. These should be consulted in the first place to ensure the status of the current version of the tools.

The following sections list the limitations and bugs in the tools delivered as part of the programme.

6.1.3.1. ACS

The ACS limitations, include limitations and bugs in the underlying technology and the reader is referred to the Release Note [GD34] for complete information. Below are the limitations relating specifically to the ERC32 programme.

6.1.3.1.1. AlsyMonitor

When the monitor regains control it does not flush its I/O buffers. Therefore if a breakpoint is planted following calls to TEXT_IO.PUT the text will not necessarily be displayed before the breakpoint is hit.

6.1.3.1.2. User Interface

The following errors exist in AdaWorld for Motif:

- Opening a library results in a lock being placed on that library. Adam does not always remove this lock. This causes libraries not to be compressed and if NFS is being used for hidden files (with filenames beginning with .nfs) to be left inside libraries. To force removal of the lock, either close and re-open the family, or else exit from Adam.
- When importing multiple libraries from an export set, the libraries are created in reverse correspondence to the export sets.
- Protected units can be erased from a library.

6.1.3.1.3. IEEE 754 Exceptions

There is currently no support for the IEEE 754 exception unfinished_FPop. On the FORCE CPU2CE this exception is generated by performing the operation 1/LONG_FLOAT'LAST. Performing such operations may produce erroneous results. This exception will not be generated by the ERC32 FPU.

6.1.3.1.4. AdaProbe

Any control characters contained within output text received by AdaProbe from the target as part of the Lowlevel.Transparent command (typically from a ERC32 Target Simulator) will not be interpreted correctly. For example, a new line character will not cause a new line to occur in the output window.

6.1.3.2. ERC32 Target Simulator

The following limitations and problems are known within the Scheduler Simulator version 2.1, see installation manual [GD53]:

- The ERC32 Target Simulator behaves as the ERC32 chipset and therefore has the same limitations, such as maximum 32Mbyte RAM and no extended memory area.
- In I/O area handling, the only area currently handled is the ATAC, which has an integrated device simulator. Consequently, it is not possible to add user definable memory in the I/O area. This can be worked around by using the Exchange Memory area to add an I/O simulation.
- The ATAC simulator functionality has only been subject to a limited test set.
- Polling for environment activity can decrease performance significantly. Therefore to overcome this if no environment activity is required, the polling can be avoided by scheduling a wait event, such as *WAIT_TIME 20000000*, which will effectively disable the polling.
- When using the Alsys/TSP debugger and executing a *STEP OVER* command on a recursive routine, which invokes to a depth of more than 8 levels, an error message is generated by the debugger warning "*AFTER STEP: . . .* " which means the debugger has lost synchronisation with the execution. This can be avoided by using a breakpoint to step around the call or by performing an *EXECUTE.CONTINUE* following the warning.

6.1.3.3. Schedulability Analyser

The following limitations and problems are known within the Schedulability Analyser version 2.1, see installation manual [GD45]:

- Using both the Schedulability Analyser and Scheduler Simulator together sometimes causes a spurious syntax error to be generated. This can be solved by quitting both tools and deleting the file "DEFAULT_HRT_FILES" from the home directory.
- The access to the context sensitive help takes some time to display.

6.1.3.4. Scheduler Simulator

The following limitations and problems are known within the Scheduler Simulator version 2.1, see installation manual [GD46]:

- Using both the Schedulability Analyser and Scheduler Simulator together sometimes causes a spurious syntax error to be generated. This can be solved by quitting both tools and deleting the file "DEFAULT_HRT_FILES" from the home directory.
- The access to the context sensitive help takes some time to display.
- Using the Scheduler Simulator context restore facility from the "Simulation" window sometimes fails.

6.2. Use of the ERC32 Toolset

The ERC32 Toolset has been developed to run on any Sun-4 configuration under Solaris version 2 or later, with all user interfaces employing the X/11 MOTIF standard.

The tool set is intended to be used throughout the software development lifecycle, allowing the user to analyse and verify that all timing constraints are met each stage of the development, and that the software will execute on the hardware prior to its availability.

It also allows early prototyping of applications, particularly useful during the early system design phases.

The following steps summarise the intended use of the tool set within the context of the HRT software lifecycle:

During the architectural design phase the user performs the following activities:

- produces a high level design for the application which is analysable via the use of the tool set;
- estimates the processing times of tasks (estimates may be made either empirically or by prototyping) and then uses these in the HRT analysis that he/she undertakes via the Schedulability Analyser and Scheduler Simulator tools;
- examines the results of the HRT analysis to confirm the feasibility of the design, which is then refined by adding detail (e.g. subprogram calls, or making modifications to the tasks);
- estimates (by prototyping if necessary), if required, the timing characteristics of the subprograms and repeats the HRT analysis;
- continues the process of refining the design until the he/she is sufficiently confident to proceed to the detailed design;

During the design, coding and test phase the user performs the following activities:

- produces the detailed design and updates the estimates/prototype values of the processing times, if required;
- repeats the HRT analysis and refines the detailed design and continues with this iterative procedure until he/she is confident to proceed to coding;
- generates the application code and calculates a worst case path execution time, based on the object code produced by the compilation process; for each task and subprogram, and then repeats the HRT analysis
- refines the code and repeats the HRT analysis until the developer is confident to proceed to formal testing;
- builds a machine code version of the application and commences with testing using the target processor simulator i.e. the ERC32 Target Simulator Tool;

- corrects and re-tests the code (and the design also as required) as errors are detected and this continues until the developer is confident to move to integration and testing on the actual target;
- continues with testing on the target processor and extracts timing data which can then be compared with the results of the analysis. If need be, the extracted timing data can be used in any further HRT analysis. He/she also carries out the Acceptance Test at which the software is provisionally tested.

At any stage, the HRT analysis may reveal that significant changes to the application are needed. In this situation the development process can return to the architectural design, detailed design or coding and test phase as appropriate.

A more detailed description of the use of the toolset and the individual tools can be found in the respective user manuals [GD18], [GD34], [GD44], and [GD52]

6.3. Availability of the ERC32 Toolset

The toolset elements and information can be obtained from the following:

ERC32 Ada Compilation System v5.5.4:

Alsys/Thomson Software Products
Partridge House
Newtown Road
Henley-on-Thames
Oxon, RG9 1EN
England

ERC32 Target Simulator, Schedulability Analyser v2.1 and Scheduler Simulator v2.1:

Spacebel Informatique
111 Rue Colonel Bourg
B-1140 Bruxelles
Belgium

Hard Real-Time Software Development with ERC32 toolset:

Logica UK Ltd - Space Division
Wyndham Court
94 Portsmouth Road
Cobham
Surrey KT11 3LG
England

6.4. Future Improvements for the ERC32 Toolset

This section identifies a number of areas and issues that may be addressed at a future date or following an evaluation of tools.

ACS

One of the primary issues to address with regard to the ACS is the relationship of the product to the Ada95 standard. Although all the extensions to the compiler system have adopted an Ada95 style, the precise syntax is not the same and they do not address all the Ada95 additions. While some of the Ada95 additions may not be appropriate (e.g. some of the Annexes) in the context of on-board processing, it may be that other features (e.g. object oriented) become a necessity for future developers. In this case, a close examination of the update strategy would be required. Determination of this strategy, including its need, should be a major consideration.

A second issue to be addressed concerns the ACVC. The present product does not have certification, although the acceptance process used within Alsys/TSP includes execution of the test suite. The lack of formal certification may present a problem for certain types of application (e.g. safety critical, safety related, mission-critical). The approach adopted on the ERC32 programme was that formal ACVC was appropriate, when a definitive need was identified. This need should be assessed.

The third issue which should be addressed here is the worse case execution time generation facility, known as the WCETE. Currently, the ACS provides a prototype with limited functionality. Taking, this function beyond this prototype stage will enhance the overall toolset functionality as it will increase the accuracy of HRT analysis and simulation.

Finally, the ACS requirements specification identified a number of “future” requirements. While some of these dealt with extremely detailed issues (e.g. levels of optimisation), a number would represent significant benefits if required (e.g. generation of additional formats).

ERC32 Target Simulator

The ERC32 Target Simulator provides a degree of flexibility in adding functionality through its use of the TCL/TK tool. Particularly, this allows the user to add to the environment and so incorporate either complex models or incorporate the simulator into a larger spacecraft level simulation.

One area which should be given some consideration however, is the incorporation of symbolic information in the stand-alone user interface. This interface currently provides a disassembler function, however with some of the load formats additional symbolic information is provided and this could be used as the basis for providing a more

sophisticated debug capability. This would be particularly used in a mixed language environment, where multiple sources are used.

Scheduler Simulator

For the Scheduler Simulator, a similar addition in post-processing capability would also be beneficial. Currently a Gantt chart provides a good visualisation of the execution of an application and provides the best way for investigating scheduling events. However, a summary display for the textual report, on a task and application basis would provide additional benefit and would also allow the user a swift comparison of the schedule simulation with the analysis.

Schedulability Analyser

Currently, the user interface for the Schedulability Analyser is limited. The presentation of the results is currently provided in textual form only. While, this provides the necessary information for performing HRT analysis, post-processing of this data using nominal graphical representations may considerable enhance the tool.

For example, pie chart would provide a quick and concise summary of the analysis results, which could be used in the overall development documentation. In the case, where a scheduling problem had been identified, it would also act as a more visible indicator to the user that further investigation was required.

7. ERC32 SYSTEM

7.1. ERC32 System Overview

The ERC32 system consists of both hardware components and tools for software development for on-board embedded real-time computers.

The ERC32 core is characterized by low circuit complexity and power consumption. Extensive concurrent error detection and support for fault-tolerance and reconfiguration is emphasized.

A basic computer implementation using the ERC32 chipset would consist of the following components:

ERC32 core, including:

- Processor, which consists of one Integer Unit (IU) and one Floating Point Unit (FPU). The processor includes concurrent error detection facilities.
- Memory Controller (MEC), which is a unit consisting of all necessary support functions such as memory control and protection, EDAC, wait state generator, timers, interrupt handler, watch dog, UARTs, and test and debug support. The unit also includes concurrent error detection facilities.
- One or, if a separate watchdog clock source is required, two oscillators.
- Buffers and latches necessary to interface with memory.
- Memory
- Serial communications (UART) interface circuitry

The above configuration is the simplest possible and is probably only usable for performance measurements etc. A computer in a space application is most likely to have some additional Input/Output interfaces apart from the serial communication implemented by the UART function of the MEC.

In addition to the basic bus interface, an exchange memory area, providing simultaneous access from the ERC32 and the external bus, is supported by ERC32 and could thus be easily implemented.

For efficient Ada code execution, an Ada Tasking Coprocessor (ATAC) can also be added to the system. The ATAC is supported by all the software tools, and the user may easily switch between ATAC and non-ATAC utilization by a single binder option.

All of the above functions typically fits onto one printed circuit board in space applications, depending on memory size.

For high-reliability applications, the concurrent error checking capability of ERC32 can be used. This would then double the number of IU/FPU components in the system.

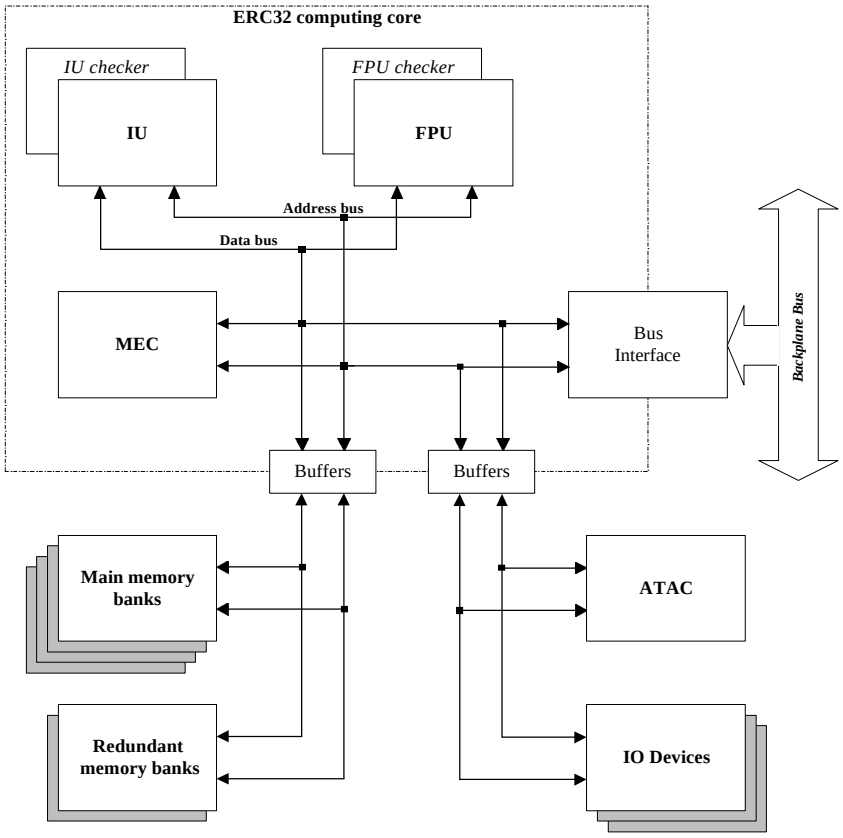

Figure 9 - ERC32 Computer for high-reliability applications

The ERC32 computer core is accompanied by a dedicated real-time software development environment, including basic components as an Ada cross-compiler, cross-debugger, and a very accurate target simulator, as well as specific tools for analysing the real-time behaviour and assertion of the schedulability of the software system. The tools cooperate in various ways, e.g. to enable automatic analysis of worst-case execution times (including the runtime system) in relation to hard real-time requirements defined by the user.

The software tools run on any Sun-4 configuration under Solaris version 2 or later.

7.2. Foreseen Applications

The developed ERC32 hardware and software products are foreseen to be used specifically in Fault Tolerance Systems and in demanding Hard Real-Time applications. However, the ERC32 products can also naturally be used in less complex environment where performance requirements are emphasized.

7.2.1. ERC32 Chipset in Fault Tolerant Systems

The ERC32 system can be configured to very high fault tolerant applications. This is primarily reached by means of implemented concurrent error detection capabilities in the ERC32 chipset.

The objective of concurrent error detection is to detect every error in the ERC32 when the ERC32 is running an application.

The concurrent error detection methods have mainly been designed to detect transient faults because the transient fault rate, caused by SEUs, is at least a magnitude higher than permanent fault rate.

Furthermore, the concurrent error detection in the ERC32 system is mainly aimed at detecting the manifested faults (errors) in the system, not the latent faults.

Therefore, the latent faults in the memory should be processed by concurrent on-line test software, in order to avoid undetectable multiple faults in the memory. The on-line test software should include "Memory Scrubbing", read with writeback in a repetitive manner.

The error detection methods are defined on different levels in the ERC32 system:

- Unit concurrent error detection:

  Each unit in the ERC32 (IU, FPU, MEC) is responsible for detecting and reporting faults within itself, e.g. parity error, memory error, permanent faults etc.

- ERC32 system concurrent error detection:

  The ERC32 system is responsible for detecting and reporting faulty units and erroneous behaviour to the next level, e.g. master/slave compare faults, memory circuit fault, double errors in memory, software execution errors such as watchdog timeout and program flow errors.

Unit concurrent error detection.

Since internal errors in units are not visible outside the unit in most cases, it is necessary to detect these errors internally within each unit. Nevertheless, only the IU of the ERC32 core is able to process errors by using trap routines. This leads to a concept where each unit detects an error, signals it to the IU, either directly or via the MEC. The error status of each unit will be monitored and sampled in the MEC to allow the trap routine to trace and process the error.

ERC32 system concurrent error detection.

For system application with very high demands on reliability the basic ERC32 core itself can not be used without adding redundant units to increase the reliability.

By duplication of units and comparators one can detect almost 100% of the internal errors, especially those errors that are not detected by internal unit concurrent error detection mechanism.

Duplications of units and introduction of comparators normally require external buffers and extra logic, and thus the area, power and execution time is increased. In ERC32 the comparation logic is implemented in the pads of each unit, the units can monitor the pin values without driving the outputs, and the overhead is reduced.

The ERC32 chipset supports the handling of faults both in normal mode and in master/slave mode.

In master/slave mode the IU and FPU are doubled and the extra devices compare instead of drives all output pins.

The ERC32 scheme for a master/slave concept only adds two pins to each unit, the ERC32 area is doubled due to the need for duplication of units, the timing impact is negligible and power impact minimal.

The master/slave concept not only detects internal errors in circuitry, it also covers pad, bonding, pin and interconnection faults.

For master/slave solutions there could be a possibility that the system can continue with only one correct unit, or with two after restoration of state of the faulty unit. This is only possible if the IU and/or the FPU fails, not if the MEC fails.

The MEC requires error signals from both the masters and from the slaves of both the IUs and the FPUs. In case of total corruption the system behaviour is defined by the MEC error-handler. If hardware support is required it has to be implemented outside the ERC32 core.

7.2.2. ERC32 Toolset in Hard Real-Time Applications

The ERC32 software toolset is primarily aimed at providing support for the development of Hard Real-Time software of the type typical of on-board applications within the Space industry.
For such applications the software performs one of a number of roles, for example:

- Forming part of the overall satellite Attitude and Orbit Control System (AOCS), where the software performs specific functions such as orbit manoeuvring and fine station keeping.

- Providing an overall spacecraft management function, the On-Board Data Handling (OBDH), where the software monitors the on-board environment (e.g. power

consumption) and responds to ground commanding (e.g. telecommand action and distribution).

- Providing a subsystem or instrument control role, such as an Instrument Control Unit (ICU) which controls the instrument as well as performing telecommands, telemetry collection and instrument data collection/direction.

It is to be expected that the role will increase as satellite applications become more complex. As the complexity of the overall satellite functionality increases, software will become an ever larger component and itself more complex.

This envisaged increase in complexity and functionality has been a major driver within the ERC32 toolset development, with the fundamental aim being to provide a framework to allow the development of efficient, yet comprehensible and analysable HRT software.

Therefore, it is to be expected that the tools can be used for any future space missions where the timing constraints of the software are paramount.

The toolset, however is not restricted to on-board HRT applications. In addition to providing the specific support for this type of software, the Ada Compilation System and ERC32 Target Simulator are of a sufficiently general nature for them to be used within other types of applications, though of course the restriction exists that the tools are for development of applications for the ERC32 processor.

Indeed, the ACS with its heritage can be foreseen to be used on any application, where the benefits provided by the Ada programming language would be advantageous. These may not be HRT in nature, but would be applications which either required the power of the ERC32 or where the advantage or Ada for large scale program development would offer considerable development savings.

For the ERC32 Target Simulator, the applications are limited only to the application scope of the ERC32 processor itself. It provides an accurate simulation of the processor and therefore can provide benefits to a developer right from the start of a programme, including the feasibility stages, where it could be used for the assessment of the progressing capabilities. In addition, the ERC32 Target Simulator is not tied to the Ada language or the ACS debugger, as it also provides its own stand-alone interface and disassembler function.

8. CONCLUSIONS

This 32-bit Microprocessor and Computer Development programme has developed an infrastructure of sophisticated hardware components and software tools to support the development of 32-bit on-board computers for future space applications.

The hardware development resulted in a computer core (ERC32) which was built of three chips, i.e an Integer Unit, a Floating Point Unit and a Memory Controller. The core is based on SPARC V7 architecture, including all functionality required to form an embedded 32-bit on-board computer. Only adding external memory and required I/O devices a complete system is configured. I.e it is easy to design systems with the ERC32 core.

The ERC32 chipset development has resulted in a Rad hard latchup free chipset with performance up to 10 MIPS at 14 MHz clock frequenzy and a power consumption below 2,25 W at 5,5 V.

The hardware was first simulated extensively in VHDL and then procured in prototypes and characterized accordingly on-site. Finally, the functionality was verified within frame of 32-bit Demonstration Breadboard (DEM32) used by both hardware and software co-ordinators and by software toolset developers.

The ERC32 chipset is now available for procurement for flight application purposes.

The software development has resulted in a suite of tools, the “32-bit toolset”, which support the user in the development and verification of hard real-time software written in the Ada programming language.

The toolset supports the complete Hard Real-Time software development lifecycle and consequently, can be used from the start of a programme, for example in the prototyping and conceptual design, through to the final verification and validation of the application.

The software development has focused on ensuring that the toolset provides realistic and pragmatic implementations. For the ERC32 Target Simulator, hardware experts have been involved in reviewing and evaluating the product to ensure accurate simulation. While for the HRT tools, there has been significant technical exchange to ensure that they perform a precise logical analysis and simulation of the Ada Run-Time System, thus ensuring that the tools could be ported to another system with minimal effort. For the Ada Compilation System, the Run-Time System implementation has focused on providing an efficient implementation, as well as ensuring that the execution time for Run-Time System functionality is measurable, in order to ensure that it can be analysed.

The tools have undergone a verification and validation process, which includes both a limited evaluation performed as part of the programme and also a formal validation of each tool.

The ERC32 toolset is now ready to be used for flight application purposes.

APPENDIX 1 - Generated documents

This section provides a definitive list of the documents generated by software part and the hardware part of the programme. The software and hardware parts are split into the three phases.

SOFTWARE DOCUMENTS

Phase 1

[GD1] Analysis of a Worst Case Execution Timing Tool for the 32bit Microprocessor Project
MCD-LOG-P1-TN-004, Issue Draft, Rev. 1A, 28 Jan 1994

[GD2] Software Quality Assurance Plan
MCD-LOG-P1-PLN-001, Issue 1, 16 May 1994

[GD3] Software Project Management Plan
MCD-LOG-P1-PLN-003, Issue 1, 16 May 1994

[GD4] Software Configuration Management Plan
MCD-LOG-P1-PLN-004, Issue 1, 16 May 1994

[GD5] Software Verification and Validation Plan
MCD-LOG-P1-PLN-002, Issue 1, 16 May 1994

[GD6] Software Tools Overall Definition Document
MCD-P1-LOG-TN-007, Issue 2 11 July 1994

[GD7] SRD for the Ada Development Environment
DTI-32B-ST-001, Issue 1, Rev 4, 24 Jun 1994

[GD8] SRD for the DEM32 Simulator
DTI-32B-ST-100, Issue 4, 25 May 1994

[GD9] SRD for the Scheduler Simulator
DTI-32B-ST-003, Issue 1 Rev 2, 1 Jul 1994

[GD10] SRD for the Schedulability Analyser
DTI-32B-ST-002, Issue 1 Rev 2, 1 Jul 1994

[GD11] SRD for the WCET Estimator
DTI-32B-ST-004, Issue 1, Rev 3, 1 Jul 1994

[GD12] ICD for the Ada Development Environment
DTI-32B-ID-001, Issue 1, Rev 2, 1 Jul 1994

[GD13] ICD for the HRT Tools
DTI-32B-ID-002, Issue 1 Rev 2, 1 Jul 1994

[GD14] ICD for the DEM32 Simulator
DTI-32B-ID-100, Issue 4, 25 May 1994

Phase 2 and Development Finalisation Phase

[GD15] Software Project Management Plan
MCD-LOG-P1-PLN-003, Issue 2, 6 Jun 1995

[GD16] HRT Tools Requirements Consolidation Document
MCD-P2-LOG-TN-008, Issue 1, 7 Jun 1996

[GD17] ERC32 Target Simulator Requirements Consolidation Document
MCD-P2-LOG-TN-009, Issue 1 revision 1, 20 Dec 1996

[GD18] ERC32 Software Development Toolset User Manual
MCD-P2-LOG-TUM-001, Issue draft, 15 Jan 1997

Ada Compilation System

[GD19] SRD for the Ada Development Environment
MCD-LOG-SRD-001, Issue 1, Rev 1, 22 Nov 1996

[GD20] ICD for the Ada Development Environment
MCD-LOG-ICD-001, Issue 1, Rev 1, 22 Nov 1996

[GD21] Ada Compilation System Design Document
MCD-ALS-P2-DD-001, issue 2.1, 29 Nov 1996

[GD22] ACS Compiler Design Specification
MCD-ALS-P2-DS-001, issue 2.1, 29 Nov 1996

[GD23] ACS Binder Design Specification
MCD-ALS-P2-DS-002, issue 1.1, 14 Jun 1995

[GD24] ACS Run-Time System Design Specification
MCD-ALS-P2-DS-003, issue 2.1, 29 Nov 1996

[GD25] ACS DEM32 Board Support Package Design Specification
MCD-ALS-P2-DS-004, issue 2.1, 29 Nov 1996

[GD26] ACS Debugger Modifications Design Specification
MCD-ALS-P2-DS-005, issue 2.1, 29 Nov 1996

[GD27] ACS ERC32 Monitor Design Specification
MCD-ALS-P2-DS-006, issue 2.1, 29 Nov 1996

[GD28] ACS Software Project Management Plan
MCD-ALS-P2-PLN-001, issue 2.0, 29 Feb 1996

[GD29] ACS Software Verification and Validation Plan

MCD-ALS-P2-PLN-002, issue 3.2, 16 Dec 1996

[GD30] ACS Software Quality Assurance Plan
MCD-ALS-P2-PLN-003, issue 2.0, 29 Feb 96

[GD31] ACS Software Configuration Management Plan
MCD-ALS-P2-PLN-004, issue 2.0, 29 Feb 96

[GD32] Work Package Report
MCD-ALS-P2-DOC-018 Issue 2 revision 1, 16 Dec 96

[GD33] Qualification Report
MCD-ALS-P2-QA-020 Version 6, 11 Dec 96

[GD34] ACS User Manuals
AdaMake User's Guide, MCD-ALS-P2-DOC-012, Issue 1 revision 0
AdaMath User's Guide, MCD-ALS-P2-DOC-016, Issue 1 revision 0
AdaProbe User's Guide, MCD-ALS-P2-DOC-011, Issue 2 revision 0
AdaReformat User's Guide, MCD-ALS-P2-DOC-014, Issue 1 revision 0
AdaWorld for Motif User's Guide, MCD-ALS-P2-DOC-004, Issue 1 revision 0
AdaXref User's Guide, MCD-ALS-P2-DOC-013, Issue 1 revision 0
Appendix F, MCD-ALS-P2-DOC-008, Issue 2 revision 0
Application Developer's Guide, MCD-ALS-P2-DOC-007, Issue 2 revision 0
Cross-Development Guide, MCD-ALS-P2-DOC-015, Issue 2 revision 1
Flexible License Manager User's Guide, MCD-ALS-P2-DOC-005, Iss. 1 rev. 0
Installation Guide, MCD-ALS-P2-DOC-002, Issue 2 revision 0
Project Development Guide, MCD-ALS-P2-DOC-006, Issue 1 revision 1
Release Notes, MCD-ALS-P2-DOC-001, Issue 2 revision 1
User Interface Guide, MCD-ALS-P2-DOC-009, Issue 1 revision 0
User's Guide, MCD-ALS-P2-DOC-003, Issue 1 revision 0
Viewing Guide, MCD-ALS-P2-DOC-010, Issue 1 revision 0
Welcome to AdaWorld, MCD-ALS-P2-DOC-019, Issue 1 revision 0

[GD35] Consolidated Schedulability Analyser SRD
32B-SBI-SRD-0189-001, Issue 1 rev 4, 31 Jan 96

[GD36] Consolidated Scheduler Simulator SRD
32B-SBI-SRD-0189-001, Issue 1 rev 4, 31 Jan 96

[GD37] ICD for the HRT Tools
32B-SBI-ICD-0189-0001, Issue 1 rev 4, 31 Jan 96

[GD38] Schedulability Analyser ADD
32B-SBI-ADD-0189-0001, Issue 2 rev 1, 31 Jan 96

[GD39] Scheduler Simulator ADD
32B-SBI-ADD-0189-0001, Issue 2 rev 1, 31 Jan 96

[GD40] Schedulability Analyser Validation Test Plan
32B-SBI-VTP-0189-0001, Issue 1 rev 3, 20 Nov 96

[GD41] Scheduler Simulator Validation Test Plan
32B-SBI-VTP-0189-0001, Issue 1 rev 2, 1 Dec 96

[GD42] Schedulability Analyser Test Procedure Document
32B-SBI-TP-0189-0001, Issue 1 rev 1, 20 Nov 96

[GD43] Scheduler Simulator Procedure Test Document
32B-SBI-TP-0189-0001, Issue 1 rev 2, 16 Dec 96

[GD44] Schedulability Analyser and Scheduler Simulator User Manual
32B-SBI-SUM-0189-001/2, Issue 1 rev 3, 20 Dec 96

[GD45] Schedulability Analyser V2.1 Delivery
32B-SBI-iM-0189-0001, Issue 2 rev 1, 20 Dec 96

[GD46] Scheduler Simulator V2.1 Delivery
32B-SBI-IM-0189-0002, Issue 2 rev 1, 19 Dec 96

[GD47] ERC32 Target Simulator SRD
32B-SBI-SRD-0189-003, Issue 1 rev 3, 13 Sep 95

[GD48] ERC32 Target Simulator ICD
32B-SBI-ICD-0189-003, Issue 1 rev 3, 16 Feb 96

[GD49] ERC32 Target Simulator ADD
32B-SBI-ADD-0189-0003, Issue 2 rev 1, 16 Feb 96

[GD50] ERC32 Target Simulator Validation Test Plan
32B-SBI-VTP-0189-0003, Issue 1 rev 2, 16 Feb 96

[GD51] ERC32 Target Simulator Validation Test Procedures
32B-SBI-VTP-0189-0003, Issue 2 rev 1, 14 Jan 97

[GD52] ERC32 Target Simulator User Manual
32B-SBI-SUM-0189-003, Issue 2 rev 1, 19 Dec 96

[GD53] Scheduler Simulator V2.1 Delivery
32B-SBI-IM-0189-0003, Issue 2 rev 1, 19 Dec 96

[GD54] MEC Device Specification,
MCD/SPC/0005/SE, Issue 8, 1 Dec 1995

HARDWARE DOCUMENTS

Phase 1

[GD55] Processor Validation Plan,
MCD/PLN/0001/MHS, Issue 1, 21 Dec 1992

[GD56] Preferred Technologies List,
MCD/PLS/0002/SE, Issue 2, 5 May 1995

[GD57] Error Detection and Testability Methods,
MCD/TNT/0003/SE, Issue 1, 26 Oct 1992

[GD58] ERC32 Technical Specification,
MCD/SPC/0001/SE, Issue 7, 20 Apr 1994

[GD59] ERC32 Trade-Off Document,
MCD/TNT/0004/SE, Issue 2, 28 Apr 1993

[GD60] ERC32 System Design Document,
MCD/TNT/0009/SE, Issue 3, 20 Apr 1994

[GD61] IU-RT Preliminary Specification,
MCD/TNT/0016/MHS, Issue 6, 28 Aug 1995

[GD62] FPU-RT Preliminary Specification,
MCD/TNT/0017/MHS, Issue 6, 28 Aug 1995

[GD63] MEC Device Specification,
MCD/SPC/0005/SE, Issue 8, 1 Dec 1995

[GD64] MEC VHDL Model Test Report,
MCD/TRP/0003/SE, Issue 1, 2 May 1994

[GD65] VHDL Release Notes,
MCD/TNT/0028/MMS, Issue 1, 3 May 1994

[GD66] ERC32 VHDL Models User´s Guide,
Version 1.0, Oct 1996

[GD67] Device and Software Development Plan W.P. 1.5 Final Report,
MCD/PLN/0003/MHS, Issue 1, 2 Feb 1994

Phase 2

[GD68] IU Final Design Document,
MCD/TNT/0021/MHS, Issue 1, 31 Mar 1995

[GD69] FPU Final Design Document,
MCD/TNT/0022/MHS, Issue 1, 25 Apr 1995

[GD70] WP 2.1 Report, MEC Design Report,
MCD/TNT/0033/MMS, Issue 1, 14 Nov 1995

[GD71] MEC Rev. A Device Specification,
MCD/SPC/0009/SE, Issue 4, 10 Apr 1997

[GD72] IU Product Validation Report,
MCD/TNT/0025/MHS, Issue 1, 13 Oct 1995

[GD73] FPU Product Validation Report,
MCD/TNT/0024/MHS, Issue 1, 13 Oct 1995

[GD74] ERC32 System Overview Rev. CBA,
MCD/TNT/0020/SE, Issue 3, 10 Apr 1997

[GD75] DEM32 User´s Manual,
MCD/TNT/0017/SE, Issue 1, 13 Dec 1995

[GD76] Test Report for DEM32,
MCD/TNT/0020/SE, Issue 2, 10 Apr 1996

Development Finalisation Phase

[GD77] ERC32 Single-Event Upset Test Results,
WSD/JG/320/NL, Issue 1, 22 Oct 1996

[GD78] IU Total Dose Test Report,
MCD/TNT/0027/MHS, Issue 1, 19 Sept 1996

[GD79] FPU Total Dose Test Report,
MCD/TNT/0029/MHS, Issue 1, 19 Sept 1996

[GD80] MEC Total Dose Test Report,
MCD/TNT/0028/MHS, Issue 1, 12 Sept 1996

[GD81] IU Electrical Characterization Test Report,
MCD/TNT/0031/MHS, Issue 1, 1 Sept 1996

[GD82] FPU Electrical Characterization Test Report,
MCD/TNT/0032/MHS, Issue 1, 1 Sept 1996

[GD83] MEC Electrical Characterization Test Report,
MCD/TNT/0030/MHS, Issue 1, 1 Sept 1996

[GD84] IU Endurance Test Report,
MCD/TNT/0037/MHS, Issue 1, 19 Sept 1996

[GD85] FPU Endurance Test Report,
MCD/TNT/0038/MHS, Issue 1, 19 Sept 1996

[GD86] MEC Endurance Test Report,
MCD/TNT/0039/MHS, Issue 1, 12 Sept 1996

[GD87] IU User´s Manual,
Issue 0, 2 Dec 1996

[GD88] FPU User´s Manual,
Issue 0, 10 Sept 1996

[GD89] Current Errors in the ERC32 (CBA),
Issue 1, 2 Apr 1997