ERC32 Evaluation Report

Prepared By: _____ *Date:* 10-Feb-98

J S STEVENS

Approved By: _____ *Date:* 10-Feb-98

R J DURRANT

MATRA MARCONI SPACE UK Limited
P.O. Box 16, Filton, Bristol, BS12 7YB

Document Change Log

Issue	Date	Pages	Notes
1	Feb. 98	All	Initial issue

CONTENTS LIST

1. INTRODUCTION AND OBJECTIVES

At the ERC32 Products day in October 1996, at ESTEC, an ERC32 Evaluation Programme was announced. MMS UK, Bristol, applied to take part in this programme and submitted a proposal, in the form of a Technical Annex, which described the work to be done and identified the support required from ESTEC to facilitate the evaluation.

In summary, the objectives of the evaluation were:

- select an appropriate software system as an example; the SOHO AOCS software was proposed, with the Normal mode subset being selected, to keep the project to a manageable size.
- replace the SOHO cyclic scheduler by Ada tasking.
- re-implement assembler objects; simulate external interfaces.
- compile all Ada files.
- Derive WCET information and evaluate HRT tools.
- Link application, using ESTEC facilities.
- Evaluate Target Simulator.

In addition it was intended to attend the ERC32 training day at ESTEC; in the event this was not possible, due to the illness of the evaluator, so all training was carried out from the user manuals supplied with the software.

The remainder of this report describes the work carried out under the evaluation programme.

2. INSTALLATION OF THE TOOLS

In order to evaluate the ERC32 tools provided by Aonix and Spacebel, it was necessary for them to be installed on a suitable workstation. The unit chosen was a SPARCStation 5, currently in use as a software development workstation on the XMM and Integral projects. Thus the evaluation was run concurrently with ongoing work using the TLD Ada compiler for 1750A and the IPL AdaTEST tools. The steps taken to install the tools are described in the following sections.

2.1 THE ADA COMPILATION TOOLS

These were installed as follows:

1. The spbrx disk usage was examined and /`user` identified as an area with adequate available space for the installation.
2. A directory `/user/erc32/alsyscomp` was created to hold the compiler tools.
3. The installation script was extracted from the tape and run.
4. The environment variable ALSYCOMP was set up, using the .cshrc file. This directory was also added to the path variable.
5. Links were set up in `/usr/lib/X11` to the AdaWorld resource and key symbol files.
6. The license file entries provided by Aonix were added to the `license.dat` file, as described in section 2.3 below, and the license manager forced to reread this file. Appropriate entries were observed in the `license.log` file.

Following these steps, the AdaWorld for Motif GUI ran successfully and it was possible to create families and libraries. A simple Ada source file was created and successfully compiled, with appropriate check out and check in entries appearing in the `license.log` file.

2.2 THE SIMULATOR AND TIMING TOOLS

These were installed as follows:

1. A directory `/user/erc32/hrt_tools` was created to hold the Spacebel tools.
2. The 3 tar files `HRT_2_2.tar`, `TS_2_2.tar` and `Tools.tar` were extracted from the 8mm tape, mounted on a remote Sun workstation.
3. The tar files were unpacked, creating the various subdirectories needed to hold the executables and support tools.
4. The environment variables needed were set up, using the .cshrc file, and the path variable was modified to include the `hrt_tools/bin` directory.
5. The license file entries provided by Spacebel were added to the `license.dat` file, as described in section 2.3 below, and the license manager forced to reread this file. Appropriate entries were observed in the `license.log` file.

The installation was then tested by running the Motif GUI versions of the tools and observing that they started and terminated successfully, with appropriate check out and check in entries appearing in the `license.log` file.

2.3 THE FLEXLM LICENSE TOOLS

The machine on which the tools were evaluated was already running the FLEXlm license management software. It was already serving licenses for 2 products, the IPL AdaTEST suite and the Sun C/C++ compiler. In order to do this the license server is started automatically at reboot. This is achieved by 2 scripts, K17lmgrd and S17lmgrd, in the directory /etc/rc3.d. These files contain the following environment variable settings:

```
LIC_DIR=/opt/SUNWspro/SunTech_License
LIC_BIN=/export/home/AdaTEST/3.0/lmgr
```

The script tests for the existence of various files and then executes the following:

```
$LIC_BIN/lmgrd -c $LIC_DIR/license.dat > $LIC_DIR/license.log 2>&1 &
```

This starts the license manager program, version supplied by IPL, using license.dat and license.log files in the Sun tools directory. The license.dat file contains the following:

```
SERVER spbrx 80787694 7323
DAEMON ipld /export/home/AdaTEST/3.0/lmgr/ipld
DAEMON suntechd /opt/SUNWspro/bin/suntechd
/opt/SUNWspro/SunTech_License/options.dat
FEATURE ATI ipld 3.000 1-jan-0 1 0BAC60814218CEB865E1 VENDOR_STRING=12c
ck=23
FEATURE ATS_Compile ipld 1.000 1-jan-0 1 CBACC011216D16A1F2A2 \
VENDOR_STRING=12c ck=3
FEATURE ATS_Recompile ipld 1.000 1-jan-0 1 AB6CE041A4ACBB2B6066 \
VENDOR_STRING=12c ck=61
FEATURE ATS_TSP_Gen ipld 1.000 1-jan-0 1 5BDC00B1DF6C47285D4F \
VENDOR_STRING=12c ck=17
FEATURE ATS_TS_Gen ipld 1.000 1-jan-0 1 8BDCE0A144823FAE48F0
VENDOR_STRING=12c ck=64
FEATURE FDS ipld 1.100 1-jan-0 1 BB7C1041DD97E46C26BD VENDOR_STRING=12c
ck=250
FEATURE sunpro.cc suntechd 3.01 01-jan-0 1 2B0A6021E06EBA264557 ""
# Serial number: 12660-16301-4
FEATURE sunpro.sparcworks.tools suntechd 2.01 01-jan-0 1
2BDA3031D4792AA98569 ""
# Serial number: 87443-16201-1
```

In order to add the new ERC32 tools, it was necessary to add the daemon and feature lines for these tools to this file and restart the license manager.

3. THE SOHO AOCS SOFTWARE

This is a complex on board software application which can operate in a number of modes, corresponding to operational spacecraft modes, such as Initial Sun Acquisition, Roll Manoeuvre Wheels and so on. The software facilities which are required are provided by a number of objects which group together associated data and functions, such as Sensor_Processing, Actuator_Processing and Control_Laws. The processing which is to be carried out in each mode is programmed by an explicit scheduler, of the frequency and phase type. In order to change modes, the scheduler table is changed to that for the new mode (tasks at the beginning and end of each table look for mode change commands, check that the requested transition is permitted and carry out any transition actions, variable initialisation etc.).

Since one of the objectives of the evaluation was to replace the scheduler by the use of Ada tasks, it simplifies the activity significantly if the mode changing mechanism is omitted and just one mode is reproduced, with the scheduler and its table replaced by explicit tasks. After some study, the Normal mode was selected; this requires activity from most objects and is therefore representative of all the modes.

The study of what software was required began with the normal mode task table.

3.1 NORMAL MODE TASK TABLE

The table in Appendix A was extracted from the SOHO file ASW_CTL_ASM.MAC, which contains the task tables for all the modes. Each task is defined by 3 entries:

- procedure to be called - a label pointing to the start of a parameterless procedure.
- phase - offset of the task from the start of scheduling, in scheduler ticks.
- frequency - task repetition interval, in scheduler ticks.

The table has been simplified by combining strings of unused entries (which all point to a null procedure). Columns have been added which show:

- the file/package where the procedure is defined.
- packages which are WITH'd by that package.

In the case of the file names, the full file name is found by adding _AUX1.ADA (package specification) or _AUX1B.ADA (package body). The WITH'd packages are only listed the first time a particular file is referenced.

3.2 ADDITIONAL SOFTWARE REQUIRED.

The first table in Appendix B lists, for each of the WITH'd packages in the scheduler table, the file in which they are found and any packages which are in turn WITH'd in that file. Those WITH'd packages not already listed are shown **bold**. The remaining 2 tables show how the WITHs were followed to determine the complete set of files required to implement normal mode. The consolidated list of files appears in Appendix C, where the files are grouped by object. The standard set of files used in SOHO to implement an object is also defined in Appendix C.

Having determined the set of files needed to implement Normal Mode, these were transferred from the VAX, where SOHO software development was carried out, to the Sun workstation being used for the evaluation. The final implementation comprised 70 source files, as against 120 in the original SOHO application.

4. THE COMPILATION PHASE

After successful transfer of all the required files to the workstation, compilation was begun. This was started at the lowest level where only predefined packages were WITHd, then working back up the WITH chain, until the top level was reached.

For some objects, particularly those in the lowest layer of the software, some procedures and functions were originally implemented in assembler. In these cases, the assembler bodies were replaced by Ada bodies. In most cases these were non-functional, due to lack of time and unavailability of an ERC32 assembler, although a functional implementation was possible in a few routines. There were a number of general problems which arose, primarily due to differences between the compiler originally used on SOHO (TLD) and the Aonix compiler.

- System.Unsigned - in the TLD package System, there is an unsigned type declared, which is basically a 16-bit positive integer. This did not exist in the Aonix System package.
- Package Machine_Code - this TLD package permits direct insertion of single machine instructions in an otherwise Ada implemented unit.
- Package Bitops - this TLD package, which makes extensive use of assembler, allows various logical operations to be performed on integer variables.

These problems were all solved in a similar fashion: special ERC32 versions of the 3 packages were created, with declarations of the appropriate types and operations. In general these were non-functional, for example returning the left hand of a pair of operands input to a Bitops operation. The Unsigned type was declared in the same way as in TLD package System. In each module which used one of the problem types or operations, the relevant package WITH was changed to the ERC32 version (in some cases, ERC32_System was added, as some other facility of System was used as well as Unsigned).

4.1 SPECIFIC PROBLEM AREAS

As well as the general problem areas discussed above, which occurred in many objects, there were some specific problems in limited areas. These were:

- address calculations - these were associated with telemetry buffers and a thruster control block. In the TLD package System, Address is declared as a subtype of integer. Thus type conversions to and from integer are permitted, whereas in the ERC32 package System, Address is declared as a private type. In fact some limited address arithmetic operations are declared, but to use these would require extensive rewriting of the code.
- unchecked conversions as assignment targets - the following code section demonstrates this.

```
Formatted_Torque
     := System.Unsigned(abs(Fix(Wheel_Torque(n) * Scale_Factor)));

  declare -- local block to perform bit manipulation on
                      Formatted_Torque

     subtype Bit_Index is Integer range 0..15;
     MSB : constant Bit_Index := 0;
     type Boolean_Array_Type is Array (Bit_Index) of Boolean;
     Pragma Pack (Boolean_Array_Type); -- pack into one word
     function CUB is new Unchecked_Conversion
        (Source => System.Unsigned, Target => Boolean_Array_Type);
```

```
begin -- set bit 7 of Formatted_Torque to sign of Wheel_Torque(n)
   CUB(Formatted_Torque)(7) := (Wheel_Torque(n) < 0.0);
end; -- local block
```

In this case the variable `Formatted_Torque` is an unsigned 16 bit quantity, where the lower byte represents magnitude and the lowest bit of the upper byte represents sign. A local type declaration provides an alternative representation as a Boolean array, allowing a direct assignment to the sign bit. This is accepted by the TLD compiler but not the Aonix one, presumably because the left hand side contains a function. However it is not really a function, but merely a way of transforming a variable type. Whether this is legal Ada is debatable.

- type conversions between an enumerated type and integer - this is shown in the following code fragment:

```
type FPSS_Data_Error_Type is
 (OK, Bad_Ap, Bad_Ay, Bad_Angle_P, Bad_Angle_Y,
            No_Sun_P, No_Sun_Y);

for FPSS_Data_Error_Type use (0, 1, 2, 4, 8, 16, 32);

function CFI is new Unchecked_Conversion
 (Source => FPSS_Data_Error_Type,
  Target => Integer);

function CIF is new Unchecked_Conversion
 (Source => Integer,
  Target => FPSS_Data_Error_Type);
```

The use of these functions is rejected by the Aonix compiler as an “Implementation restriction”. This seems an unreasonable restriction. A similar problem occurred with integer to Boolean conversions; this seems a more reasonable restriction, though. The use of this conversion depended on knowledge of how a Boolean is represented by the compiler, whereas in the enumeration case the designer has specified the implementation explicitly.

- size differences in the implementation of types - the TLD Long_Integer type has no direct equivalent in ERC32; the 32 bit word length means that many types, and quantities like Storage Size, are different from the equivalents in the 16 bit 1750A system.

Various means were used to overcome these problems, some functional and some non functional. For example, Long_Integer was simply replaced by Integer and units of Storage Size were adjusted to allow a successful compilation. The use of unchecked conversion shown above was replaced by the use of ‘pos and ‘val attributes (although this led to problems later).

5. BINDING

Initial attempts at using the binder were made early in the compilation process, to determine how it worked, what reports were produced etc. The first full bind was made when the complete source file set had been compiled (although some package bodies still had problems). The bind under these conditions was unsuccessful, as expected, with the erroneous units flagged as missing. The SOHO software follows a consistent style, where separate packages are declared for different kinds of data; these packages have specifications only. Although the library manager shows uncompiled bodies for these packages, the binder reports that they are not compiled but not required, as expected. Once the errors had been removed from the package bodies, and they had been recompiled, the binder ran successfully, generating a report showing units used, offsets of units etc., and producing a '.o' file.

6. WCET EXTRACTION

6.1 COMPILER

Once a successful bind had been achieved at the functional level, the Worst Case Execution Time (WCET) extraction was attempted. To do this, the compiler WCETE option was turned on and all files recompiled, using AdaMake. This took about 5 minutes for the 70 source files involved. WCET errors were reported in 8 source files. Some of these are listed below:

- compound assignments - a typical case is shown below.

```
352       MACS_Error_Data := (Counter => 0, Time => 0,
First_Error => True);
353
354       On_Time_Increment := (others => 0);  --(array (1..8) of natural)
          1
 1    *COD WCETE subset warning: This construct is not permitted in      the
WCET Ada HRT Restrictions. The containing subprogram must not be   called within
the main body of a critical task.
```

The first compound assignment, at line 352, is accepted but the one at line 354 is not. The offending item, in this and other cases, seems to be the 'others' assignment. In this case, as shown, the data item is an array of 8 natural numbers. The reason given in the manual for exclusion of these constructs is that execution time must be bounded. The array bounds are fixed, and so the execution time is determinable; indeed, the compiler presumably generates the required code sequence quite happily, probably as a 'for' loop, so should be able to determine its execution time.

- 'pos attribute - this was introduced to overcome a functional problem (see section 4.1 above) but was rejected on timing grounds. While it is clear that some attributes may be slow to calculate, this particular one is just a type conversion between a value of a discrete type and its position in the series, which will generally be a simple relationship.
- loops with parameter as an upper limit - clearly this is a problem for the compiler, which cannot know at compile time what the loop limit will be. A pragma Loop_Count is mentioned in the user manual to allow the designer to provide this information; unfortunately this is not implemented in the prototype tool, so the problem was overcome by coding hard values for the upper limits.
- Raise statements - exception handling is clearly capable of unbounded execution time and so is not permitted. The code fragment given, however, shows that there is no explicit raise statement in the code rejected:

```
102 -- function Arctan(X : in Float) return Float is
103
104    begin
105
106       if abs(X) < X_Small then
107
108          return X;
                    1
                    2
 1    *COD WCETE warning: Block containing explicit raise statement      has
been excluded from worst case path analysis.

 2    *COD WCETE warning: Block containing explicit raise statement has been
excluded from worst case path analysis.
```

It is assumed that the raise statement is in the code implementing the 'abs' statement, which is part of the supplied run time library!

A further problem was the location and style of the task declarations, introduced to replace the operation of the original cyclic scheduler. These had originally been declared in a procedure, but were moved to a standalone package after rejection by the compiler on HRT grounds. This package was in turn WITHd by the package containing the main program. The tasks were written in a standard style, based on that defined in the user manual:

```
task body Mode7_Ctl_task is
      use Real_Time;
      Phase : constant Time_Span := Milliseconds(30);
      Period: constant Time_Span := Milliseconds(50);
      Next_Time: Time;
   begin
      Next_Time := Clock + Phase;
      Delay_Until(Next_Time);
      loop
         Attitude_Control.Do_Mode7_Pitch_Ctl;
         Attitude_Control.Do_Mode7_Yaw_Ctl;
         Next_Time := Next_Time + Period;
         Delay_Until(Next_Time);
      end loop;
   end Mode7_Ctl_task;
```

The values for phase and period were based on those used in the original frequency and phase scheduler. The statements prior to the loop ensure that the task is initially released with the correct delay. Within the loop, the procedures to be executed are called in the correct sequence, the next time at which the task should be run is calculated and used in the Delay_Until statement. The procedures called in a particular task are all those from the original scheduler table which have the same frequency and phase, and would therefore have run in the same slot.

6.2 BINDER

Before correcting the deficiencies flagged up by the WCET compilation, a bind was attempted, with WCET processing enabled in the binder. This flagged up just 3 problems - the remaining problem areas flagged by the compiler were presumably in procedures not called in Normal Mode. These were:

- Issue_RD in MACS object, undefined loop counts.
  Replaced by hard coded value.
- Read_FPSS in AOCS_Units, a compound assignment.
  Rewrote as set of individual assignments.
- Process_FPSS_Data in Sensor_Processing, a ‘pos attribute.
  Replaced by Fixed value.

With these problems corrected, and the units recompiled, the bind stage ran successfully, producing an execution skeleton file. Initially 3 tasks had been introduced; more were added, recompiling and binding as required, until a total of 10 tasks were implemented, representing most of the original Normal Mode activities, excluding telemetry. A comparison was made between the execution times generated by the tool and those in the original SOHO budgets document, shown in the following table.

Task	ERC32	SOHO
FPSS_Task	5157	955
Mode7_Ctl	1134	400
Control laws	1989	4411
Control laws HK	2124	568
Mode7 roll	473	153

SSU7	2280	2103
SSU Data	2408	3063
SSU8	119	42
RSL	202	107
Wheel	2530	2160
Total	18416	13962

This shows the ERC32 values to be roughly the same as the original SOHO figures; however it should be noted that the ERC32 clock frequency was assumed to be 1 MHz, whereas the SOHO figures were for a MAS281, running at 15 MHz, with 3 memory wait states. This demonstrates the increase in CPU power available with the ERC32 processor.

7. COMPILATION TOOL OBSERVATIONS

The compiler, binder and library manager tools were mostly invoked using the Motif X-Window interface, with the tools running on a SPARCStation and the display on a Pentium PC, using the Exceed X-Windows emulator. Under these conditions the tools were very easy to use. Advantage was taken of the ability to set defaults for the tools and then save the environment, which was reloaded on subsequent occasions. Compiler error messages were generally found to be helpful, providing an LRM reference which usually resolved the problem. The AdaMake utility was found to be very helpful, always determining which units needed recompiling and in what order.

Some minor points were noted where improvement could be made:

- units obsoleted by a source code change were deleted from the library listing. It would be better if they were left in the listing but marked as obsolete. The fact that AdaMake was able to determine the need for recompilation indicated that they were still known to the library manager.
- it appeared impossible to print from the Motif Library Manager window. A print could be obtained by using the command line version.
- a locked library, due to a system crash, caused the Motif Library Open command to hang. The library could only be unlocked from the command line interface.
- a successful compilation produced no effect in the AdaWorld window; this made it hard to know when a compile had completed, especially since the iconified compile window did not change colour until after it had been browsed.

8. THE HRT TOOLS

Once a successful bind had been carried out, with WCET processing enabled, this generated the first item of information needed by the HRT tools, namely the execution skeleton, contained in the ESF file. The second file required by the tools is the user configuration file (UCF); the format of this file was identified from the examples provided with the tools. It firstly gives the user preferences for scheduling model and blocking protocol, then provides general target information (clock speed, wait states, priority levels available); finally it lists the threads in the application, with their criticality, period, deadline and offset.

The UCF file was constructed by taking the header from the sample file and the thread list from the ESF file. DMS scheduling was chosen and the target information was set to 1 MHz, no wait states. The thread information was generated from the original SOHO scheduler table:

- criticality - all were classed as hard.
- period - the period from the original table, converted to clock ticks.
- deadline - the end of the slot in which originally scheduled, in clock ticks.
- offset - the beginning of the slot in which originally scheduled, in clock ticks.

The final information required by the tools is a run time file; this was copied from the demonstration example.

Next the analyser was run; this showed the thread set to be schedulable, with a total utilisation of around 52%.

Finally the scheduler simulator was run and a GANTT chart of the process set obtained. As is to be expected, this was very similar to the original frequency phase scheduler pattern; it is shown in Appendix D.

The HRT tools were found to be easy to use and worked well. It was not possible given the limited time available to attempt to prove whether the tools were providing a correct analysis of the real time properties of the software. Qualitatively, however, they give results which are reasonable.

9. LINKING

The evaluation was originally set up on the understanding that the Microtec linker was not available for free evaluation and that the linking would have to be carried out using the ESTEC copy of the tool. Accordingly ESTEC set up accounts and access codes to allow this to be done. Due to later changes in the MMS Internet access methods, it was likely that the remote login would no longer work; in any event the time available for the evaluation became too short for this to be attempted.

Since a successful bind had been carried out, however, the resulting COFF format object file was taken to ESTEC when attending the Evaluators Workshop and a link carried out on site. Although the link appeared to work, running the resultant executable on the simulator revealed a problem. This was diagnosed by Mr Vardanega (ESTEC) as being due to insufficient heap space. On return to Bristol, a modified configuration file was installed and a new bind carried out. The resulting object file was e-mailed to Mr Vardanega, who relinked it. The resulting executable appeared to run successfully.

Thus it can be concluded that, given a full installation including the Microtec linker, this stage would not present any problems.

10. THE SIMULATOR

Since a linked executable was not obtained until the Evaluators Workshop, very little evaluation of the simulator has been possible. Although the Spacebel evaluation license was granted to the end of February, the Aonix license expired at the end of January. It therefore proved impossible to evaluate the connection of the simulator to AdaProbe, which in any event is quite complex to set up. The executable was loaded into the simulator in standalone mode and appeared to run, although with some problems.

11. CONCLUSIONS AND RECOMMENDATIONS

11.1 CONCLUSIONS

The evaluation proved a good introduction to the commercially available ERC32 software development toolset, although the time constraints imposed by the evaluator's illness made the final stages somewhat rushed, and therefore less useful. Many of the problems encountered in the evaluation were due to it being a porting exercise; a system implemented from scratch using these tools would not encounter many of these problems.

The ERC32 tools were generally felt to be of production quality, except for a few aspects covered in the next section.

11.2 RECOMMENDATIONS

The Aonix tools, being based on a standard commercial package, were generally complete. The WCET processing elements, however, were only in prototype form and were incomplete. Efforts should be made to generate a full production version. Typical aspects which are incomplete are the implementation of special pragmas (e.g. Loop_Count) and the generation of reports including full timing information.

The Preliminary User's Guide to the WCET processing indicates some constructs which will not be permitted in software being developed using the HRT method. It is felt that the full set of forbidden constructs is too conservative and may be motivated by implementation difficulties as much as HRT considerations. A case in point is the banning of compound assignment statements, especially where they contain "others" clauses (see section 6.1 above). Since the compiler is able to generate code for these constructs, it should be possible for bounded execution time to be calculated too. The need to have to write out individual assignment statements for each element is inelegant and makes the code more difficult to understand.

Appendix A Normal Mode Task Table

Mode_Tables_NM EQU *

Table entries	File	WITHd packages
DATAT I31$Check_Mode -- 0 DATAT 1 DATAT 5	ASW_CTL	Bitops, SWS_Functions, Attitude_Control, TC_AUX3, Actuator_Processing_AUX3, Unchecked_Conversion
DATAT I31$Initialise_NM -- 1 DATAT 1 DATAT -1	- ditto -	
DATAT E33$Synchronise_ASW_To_TM -- Sync_ASW_Task_Index DATAT -1 DATAT -1	HK	System, SWS_Functions, Attitude_Control, Unchecked_Conversion
DATAT E341$Do_Mode7_Entry_Function -- 3 DATAT 1 DATAT -1	C_LAWS	System, Bitops, SWS_Functions, Math_Utilities, Sensor_Processing, Actuator_Processing, Unchecked_Conversion , Control_Laws_AUX3
DATAT E341$Execute_PCPG -- 4 DATAT 2 DATAT 50	- ditto -	
DATAT E32$Handle_ASW_TC -- 5 DATAT 6 DATAT 5	TC	Bitops, System, SWS_Functions, Attitude_Control, TC_AUX3, Unchecked_Conversion, Control_Laws_AUX3
DATAT E33$Report_ARD -- 6 DATAT 1 DATAT 50	HK	
DATAT E23$Read_FPSS -- 7 DATAT 3 DATAT 5	AOCS	Bitops, HW_Addresses, SWS_Utilities, IOS_Functions, HW_Addresses_AUX3, Unchecked_Conversion, AOCS_Units_AUX2
DATAT E342$Acquire_SSU_Data -- 8 DATAT 2 DATAT 10	SP	Bitops, SWS_Functions, Math_Utilities, Unchecked_Conversion, Math_Utilities_AUX3, System
DATAT E341$Do_Mode7_SSU_SEU_Check -- 9 DATAT 2 DATAT 50	C_LAWS	
DATAT E342$Perform_SSU3 -- 10 DATAT 2 DATAT 50	SP	
DATAT E342$Process_FPSS_Data -- 11 DATAT 3 DATAT 5	- ditto -	
DATAT E342$Perform_SSU7 -- 12 DATAT 1 DATAT 1	- ditto -	
DATAT E341$Do_Mode7_Pitch_Ctl -- 13 DATAT 4 DATAT 5	C_LAWS	
DATAT E341$Do_Mode7_Yaw_Ctl -- 14 DATAT 4 DATAT 5	- ditto -	
DATAT E341$Do_Mode7_Roll_Ctl -- 15 DATAT 4 DATAT 50	- ditto -	

DATAT E343$Compute_Control_Law_Torques -- 16 DATAT 5 DATAT 5	AP	Bitops, System, SWS_Functions, Math_Utilities, Unchecked_Conversion
DATAT E343$Send_Wheel_Commands -- 17 DATAT 5 DATAT 5	- ditto -	
DATAT E342$Perform_SSU8 -- 18 DATAT 5 DATAT 50	SP	
DATAT E341$Perform_RSL_Common_Fn -- 19 DATAT 93 DATAT 100	C_LAWS	
DATAT E212$Handle_OS_TC -- 20 DATAT 6 DATAT 5	OB_TC	Bitops, Scheduler, IOS_Functions, SWS_Utilities, TM_Buffers_AUX3, OBDH_Interface_HK, Unchecked_Conversion
DATAT E244$Null_Proc -- 21 to 33		
DATAT E23$Command_SSU -- Command_SSU_Task_Index DATAT 1 DATAT 1	AOCS	
DATAT E33$Report_IRU_Common_Fn_Vars -- 35 DATAT 154 DATAT 500	HK	
DATAT E33$Report_SAS_Common_Fn_Vars -- 36 DATAT 164 DATAT 500	- ditto -	
DATAT E33$Report_Wheel_Common_Fn_Vars -- 37 DATAT 184 DATAT 500	- ditto -	
DATAT E33$Report_PROS_Common_Fn_Vars -- 38 DATAT 204 DATAT 500	- ditto -	
DATAT E33$Report_FPSS_Common_Fn_Vars -- 39 DATAT 224 DATAT 500	- ditto -	
DATAT E33$Report_SSU_Common_Fn_Vars -- 40 DATAT 244 DATAT 500	- ditto -	
DATAT E33$Report_RSL_Common_Fn_Vars -- 41 DATAT 274 DATAT 500	- ditto -	
DATAT E33$Report_SSU_Status_Words -- 42 DATAT 294 DATAT 500	- ditto -	
DATAT E33$Report_Control_Star_Data -- 43 DATAT 314 DATAT 500	- ditto -	
DATAT E33$Report_Roll_Off_Pointing_Limit -- 44 DATAT 334 DATAT 500	- ditto -	
DATAT E33$Report_Slew_Vars -- 45 DATAT 354 DATAT 500	- ditto -	
DATAT E33$Report_Control_Law_Outputs -- 46 DATAT 374 DATAT 500	- ditto -	
DATAT E343$Perform_TC_Thruster_Command -- 47 DATAT 4 DATAT 10	AP	
DATAT E342$Download_SSU -- 48 DATAT 1 DATAT 1	SP	
DATAT E343$Acquire_Wheel_Speeds -- 49 DATAT 1 DATAT 20	AP	

DATAT E23$Read_IRU_Roll_Data -- 50 DATAT 2 DATAT 20	AOCS	
DATAT E33$Report_Roll_TM_Offset -- 51 DATAT 394 DATAT 500 DATAT E244$Null_Proc -- 52 to 55	HK	
DATAT E23$Request_SSU_BRD -- SSU_BRD_Task_Index DATAT 50 DATAT 50	AOCS	
DATAT E23$Request_FPSS_BRD -- 57 DATAT 5 DATAT 5	- ditto -	
DATAT E23$Request_IRU_BRD -- 58 DATAT 20 DATAT 20	- ditto -	
DATAT I31$Perform_Transition -- 59 DATAT 100 DATAT 100	ASW_CTL	
DATAT E244$Null_Proc -- Last_Task DATAT -1 DATAT -1		

Appendix B Analysis of required files

Table 1

Package	File	Type	Further WITH'd packages
Actuator_Processing	ap.ada		SWS_Functions, **Actuator_Processing_AUX1**, Actuator_Processing_AUX3
Actuator_Processing_AUX3	ap_aux3.ada		SWS_Functions, Math_Utilities, Math_Utilities_AUX3
AOCS_Units_AUX2	aocs_aux2.ada		**AOCS_Units_AUX2a**, **AOCS_Units_AUX3**
Attitude_Control	att_ctl.ada		**Control_Laws**, SWS_Functions, Math_Utilities, Sensor_Processing, Actuator_Processing
Bitops			
Control_Laws_AUX3	c_laws_aux3.ada		SWS_Functions, Math_Utilities, Sensor_Processing, Actuator_Processing, Math_Utilities_AUX3, **Sensor_Processing_AUX3**
HW_Addresses	hw_addr.ada		HW_Addresses_AUX3
HW_Addresses_AUX3	hw_addr_aux3.ada		IOS_Functions
IOS_Functions	ios_fns.ada		**Kernel**, **Machine_Code**, **RTU_Interface**, **MACS_Interface**
Math_Utilities	math.ada		**Math_Utilities_AUX1**, Math_Utilities_AUX3
Math_Utilities_AUX3	math_aux3.ada		-
OBDH_Interface_HK	ob_hk.ada		Scheduler, SWS_Utilities, **OBDH_Interface_HK_AUX1**, **OBDH_Interface_HK_AUX2**, **OBDH_Interface_HK_AUX3**
Scheduler	sched.ada		**Scheduler_AUX1**, **Scheduler_AUX3**
Sensor_Processing	sp.ada		SWS_Functions, Actuator_Processing, **Sensor_Processing_AUX1**, **Sensor_Processing_AUX3**
SWS_Functions	sws_fns.ada		Scheduler, **AOCS_Units**, HW_Addresses, IOS_Functions, SWS_Utilities, **OBDH_Interface**, **SWS_Functions_AUX1**, **SWS_Functions_AUX3**, **Scheduler_AUX3**, HW_Addresses_AUX3, **OBDH_Interface_TC_AUX3**
SWS_Utilities	sws_utl.ada		**TM_Buffers**, **ASM_Utilities**, **Error_Handling**, **Time_Management**
System			
TC_AUX3	tc_aux3.ada		SWS_Functions, Attitude_Control
TM_Buffers_AUX3	tm_buf_aux3.ada		-
Unchecked_Conversion			

Table 2

Package	File	Type	Further WITH'd packages
Actuator_Processing_AUX1	ap_aux1.ada		Actuator_Processing_AUX3
AOCS_Units	aocs.ada		HW_Addresses, **AOCS_Units_AUX1** & 3
AOCS_Units_AUX2a	aocs_aux2a.ada		AOCS_Units_AUX3
AOCS_Units_AUX3	aocs_aux3.ada		HW_Addresses, SWS_Utilities
ASM_Utilities	asm_utl.ada	spec	**ASM_Utilities_AUX1**
Control_Laws	c_laws.ada		**Control_Laws_AUX1** & 3, Actuator_Processing
Error_Handling	err.ada	spec	**Error_Handling_AUX1 & 3**
Kernel	kernel.ada		Machine_Code, **Kernel_AUX1 & 3**
Machine_Code			
MACS_Interface	macs.ada	spec	**MACS_Interface_AUX1 & 3**
Math_Utilities_AUX1	math_aux1.ada	exports	Math_Utilities_AUX3
OBDH_Interface	ob.ada	spec	Scheduler, OBDH_Interface_HK , **TC**, **Interrupt_Services**
OBDH_Interface_HK_AUX1	ob_hk_aux1.ada	spec + data	OBDH_Interface_HK_AUX3
OBDH_Interface_HK_AUX2	ob_hk_aux2.ada	type 2 data	OBDH_Interface_HK_AUX3
OBDH_Interface_HK_AUX3	ob_hk_aux3.ada	type defs, type 3 data	Scheduler, IOS_Functions, SWS_Utilities
OBDH_Interface_TC_AUX3	ob_tc_aux3.ada		Bitops, Scheduler, IOS_Functions
RTU_Interface	rtu.ada		**RTU_Interface_AUX1 & 3**
Scheduler_AUX1	sched_aux1.ada		Scxheduler_AUX3
Scheduler_AUX3	sched_aux3.ada		Machine_Code, Unchecked_Conversion
Sensor_Processing_AUX1	sp_aux1.ada		Actuator_Processing, Sensor_Processing_AUX3

Sensor_Processing_AUX3	sp_aux3.ada		SWS_Functions, Math_Utilities
SWS_Functions_AUX1	sws_fns_aux1.ada		-
SWS_Functions_AUX3	sws_fns_aux3.ada		Scxheduler, SWS_Utilities
Time_Management	time.ada		**Time_Management_AUX1 & 3**
TM_Buffers	tm_buf.ada		**TM_Buffers_AUX1, 2** & 3

Table 3

Package	File	Type	Further WITH'd packages
AOCS_Units_AUX1	aocs_aux1.ada		AOCS_Units_AUX3
ASM_Utilities_AUX1	asm_utl_aux1.ada		-
Control_Laws_AUX1	c_laws_aux1.ada		-
Error_Handling_AUX1	err_aux1.ada		Error_Handling_AUX3
Error_Handling_AUX3	err_aux3.ada		Time_Management
Interrupt_Services	int.ada		**Interrupt_Services_AUX1 & 3**
Kernel_AUX1	kernel_aux1.ada		Machine_Code, Kernel_AUX3
Kernel_AUX3	kernel_aux3.ada		Machine_Code
MACS_Interface_AUX1	macs_aux1.ada		Machine_Code, **MACS_IO_Addresses**, MACS_Interface_AUX3
MACS_Interface_AUX3	macs_aux3.ada		-
OBDH_Interface_TC	ob_tc.ada		Scheduler, **OBDH_Interface_TC_AUX1** & 3
RTU_Interface_AUX1	rtu_aux1.ada		RTU_Interface_AUX3
RTU_Interface_AUX3	rtu_aux3.ada		-
Time_Management_AUX1	time_aux1.ada		Unchecked_Conversion, Time_Management_AUX3
Time_Management_AUX3	time_aux3.ada		-
TM_Buffers_AUX1	tm_buf_aux1.ada		TM_Buffers_AUX3
TM_Buffers_AUX2	tm_buf_aux2.ada		-
Further from above			
Interrupt_Services_AUX1	int_aux1.ada		Interrupt_Services_AUX3
Interrupt_Services_AUX3	int_aux3.ada		-
MACS_IO_Addresses	macs_io_addr.ada		Machine_Code
OBDH_Interface_TC_AUX1	ob_tc_aux1.ada		OBDH_Interface_TC_AUX3

Appendix C Required files

File	Associated file(s)	Function	Action	Need?
aocs.ada	aocs_asm.mac	AOCS_Units package - hardware interface, configuration word, majority voter.	Stub assembler	Yes
aocs_aux1.ada	aocs_aux1b.ada			
aocs_aux2.ada				
aocs_aux2a.ada				
aocs_aux3.ada				
ap.ada		Actuator commanding		Yes
ap_aux1.ada	ap_aux1b.ada			
ap_aux3.ada				
asm_utl.ada	asm_utl_asm.mac	Block copy, indirect call, null proc, increment and modulo.	Stub	Yes
asm_utl_aux1.ada	asm_utl_aux1b.ada			
att_ctl.ada		Top level attitude control (renames)		Yes
c_laws.ada		Control laws		Yes
c_laws_aux1.ada	c_laws_aux1b.ada			
c_laws_aux3.ada				
err.ada		Error reporting	Stub?	Yes
err_aux1.ada	err_aux1b.ada			
err_aux3.ada				
hw_addr.ada		Hardware addresses		Yes
hw_addr_aux3.ada				
int.ada		Unused interrupt services		No
int_aux1.ada	int_aux1b.ada			
int_aux3.ada				
ios_fns.ada		IOS layer		Possibly
kernel.ada	kernel_asm.mac kernel_io_addr.equ	Run time kernel		No
kernel_aux1.ada				
kernel_aux3.ada				
macs.ada	macs_asm.mac	MACS bus interface	Stub	Yes
macs_aux1.ada	macs_aux1b.ada			
macs_aux3.ada				
macs_io_addr.ada	macs_io_addr.equ			
math.ada	math_asm.mac	Arctan, fix,sign		Yes
math_aux1.ada	math_aux1b.ada			
math_aux3.ada				
ob.ada		External interface to OBDH		Not on 1st pass
ob_hk.ada	ob_hk_asm.mac	Housekeeping telemetry		Not on 1st pass
ob_hk_aux1.ada	ob_hk_aux1b.ada			
ob_hk_aux2.ada	ob_hk_aux2a.ada			
ob_hk_aux3.ada				
ob_tc.ada	ob_tc_asm.mac	Telecommands		Not on 1st pass
ob_tc_aux1.ada	ob_tc_aux1b.ada			
ob_tc_aux3.ada				
rtu.ada	rtu_asm.mac rtu_io_addr.equ	RTU hardware interface	Stub	Yes
rtu_aux1.ada				
rtu_aux3.ada				
sched.ada	sched_asm.mac	Scheduler		Replace by program which starts the various tasks
sched_aux1.ada	sched_aux1b.ada sched_aux2.ada sched_aux2a.ada			
sched_aux3.ada				

sp.ada		Sensor data processing		Yes
sp_aux1.ada	sp_aux1b.ada			
sp_aux3.ada				
sws_fns.ada	sws_fns_asm.mac	Software services (layer 2)		Possibly not
sws_fns_aux1.ada	sws_fns_aux1b.ada			
sws_fns_aux3.ada				
sws_utl.ada				
tc_aux3.ada	tc.ada tc_asm.mac tc_aux1.ada tc_aux1b.ada	Various mode functions	Stub	Yes?
time.ada	time_asm.mac	Time management		Yes
time_aux1.ada	time_aux1b.ada time_aux2.ada time_aux2a.ada			
time_aux3.ada				
tm_buf_aux1.ada	tm_buf_aux1b.ada	Telemetry buffers		Not on 1st pass
tm_buf_aux2.ada	tm_buf_aux2a.ada			
tm_buf.ada				
tm_buf_aux3.ada				

Each object in the SOHO software is implemented using a number of standard files. For a specific object (shown as obj), the possible files are as follows (note that not all objects have every possible file).

obj.ada	the top level specification for the object, giving all types, subtypes, data and subprograms visible to other objects. These are all renames from lower level packages.
obj_aux1.ada	the main lower level specification, defining type 1 data and subprograms (internal and external), together with exports and imports. The type 1 data is that which is initialised to zero (or equivalent).
obj_aux1b.ada	the main lower level package body. Provides all subprogram bodies..
obj_aux2.ada	declares all type 2 data (non zero initial values).
obj_aux2a.ada	initialisation values for type 2 data.
obj_aux3.ada	declares type 3 data (data initialised by software, including elaboration).
obj_asm.mac	the bodies of any subprograms implemented in assembler.

Appendix D Task GANTT chart