Copy No. _____

Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

SPPACS Maintenance and Enhancement

Joe D. Hood and Brad Glessing
MacDonald Dettwiler and Associates Ltd.

MacDonald Dettwiler and Associates Ltd.
Suite 60, 1000 Windmill Road
Dartmouth, NS
B3B 1L7

Project Manager: JD Hood 902-481-3560

Contract Number: W7707-021856/001/HAL

Contract Scientific Authority: J. Theriault 902-426-3100 ext 376

Defence R&D Canada – Atlantic

Contract Report
DRDC Atlantic CR 2005-252
April 2007

Canada

This page intentionally left blank.

SPPACS Maintenance and Enhancement

Joe D. Hood and Brad Glessing
MacDonald Dettwiler and Associates Ltd.

MacDonald Dettwiler and Associates Ltd.
Suite 60, 1000 Windmill Road
Dartmouth, NS
B3B 1L7

Project Manager: JD Hood, 902-481-3560

Contract Number: W7707-021856/001/HAL

Contract Scientific Authority: J Theriault, 902-426-3100 ext. 376

Defence R&D Canada – Atlantic

Contract Report

DRDC Atlantic CR 2005-252

April 2007

Author

Original signed by Joe Hood

Joe Hood

Approved by

Original signed by James A. Theriault

James A. Theriault

Scientific Authority

Approved for release by

Original signed by Ron Kuwahara for

James L. Kennedy

DRP Chair

Terms of release: The scientific or technical validity of this Contract Report is entirely the responsibility of the contractor and the contents do not necessarily have the approval or endorsement of Defence R&D Canada.

Abstract

The DRDC Atlantic Sonar signal Processing Package (SPPACS) has been widely used for a variety of sonar data analysis tasks. Recently, the Towed Integrated Active Passive Sonar (TIAPS) Technology Demonstration project has made significant use of SPPACS. This contract included improving the testing of SPPACS and the capability to perform the specialized beamforming of acoustic data from the TIAPS Directional Active Sensor Module (DASM). This document outlines the software enhancements used for the project.

Résumé

Le progiciel de traitement des signaux de sonar (SPPACS) de RDDC Atlantique a largement servi à toute une gamme de tâches d'analyse de données de sonar. Récemment, le programme de démonstration de technologie du sonar remorqué intégré actif et passif (TIAPS) a grandement utilisé le SPPACS. Le contrat en cause comprenait l'amélioration des essais du SPPACS et de la capacité de formation spécialisée de faisceaux à partir de données acoustiques obtenues grâce au module de détection acoustique directionnel (DASM) du TIAPS. Le présent document décrit les améliorations apportées au logiciel qui ont été utilisées dans le cadre du projet.

This page intentionally left blank.

Executive summary

Introduction

The DRDC Atlantic Sonar signal Processing Package (SPPACS) has been widely used for a variety of sonar data analysis tasks. Recently, the Towed Integrated Active Passive Sonar (TIAPS) Technology Demonstration project has made significant use of SPPACS. This contract included improving the testing of SPPACS and the capability to perform the specialized beamforming of acoustic data from the TIAPS Directional Active Sensor Module (DASM).

Results

This document primarily deals with the software processing capability, with some data used as examples. The enhancements and testing added to SPPACS are significant.

Significance

SPPACS is a scientific tool, which has become an underlying resource for much of DRDC Atlantic's signal processing studies. The results of this contract are that the available testing has been improved and the package contains new towed array beamforming software that can be used to analyze the effectiveness of not only the TIAPS Directional Active Sensor Module (DASM), but can be easily extended to other similar array architectures.

Future

The improved SPPACS will continue to be the underlying signal processing capability used by the Software Tools for Analysis and Research (STAR). It will be extended as required in order to meet future study objectives.

Joe Hood, Joe D., and Glessing, Brad. 2005. SPPACS Maintenance and Enhancement. DRDC Atlantic CR 2005-252. Defence R&D Canada - Atlantic.

Sommaire

Introduction

Le progiciel de traitement des signaux de sonar (SPPACS) de RDDC Atlantique a largement servi à toute une gamme de tâches d'analyse de données de sonar. Récemment, le programme de démonstration de technologie du sonar remorqué intégré actif et passif (TIAPS) a grandement utilisé le SPPACS. Le contrat en cause comprenait l'amélioration des essais du SPPACS et de la capacité de formation spécialisée de faisceaux à partir de données acoustiques obtenues grâce au module de détection acoustique directionnel (DASM) du TIAPS.

Résultats

Le présent document traite principalement de la capacité de traitement du logiciel, certaines données servant d'exemples. Le SPPACS a fait l'objet d'essais importants et de grandes améliorations.

Portée

Le SPPACS est un outil scientifique qui est devenu une ressource de base pour une grande partie des études de traitement de signaux de RDDC Atlantique. Le contrat en cause a eu pour résultat que les essais disponibles ont été améliorés et que le progiciel comprend un nouveau logiciel de formation de faisceaux de réseau remorqué. Ce nouveau logiciel peut être utilisé pour analyser l'efficacité du module de détection acoustique directionnel (DASM) du TIAPS, et son utilisation peut facilement être étendue à d'autres architectures de réseau semblables.

Recherches futures

Le SPPACS amélioré continuera de constituer la capacité de base de traitement de signaux utilisée par les outils logiciels d'analyse et de recherche (STAR). Son utilisation sera étendue au besoin afin d'atteindre les objectifs d'études futures.

Hood, Joe D. et Glessing, Brad. 2005. SPPACS Maintenance and Enhancement (Maintenance et amélioration du SPPACS). DRDC Atlantic CR 2005-252. R & D pour la défense Canada - Atlantique.

Table of contents

List of figures

List of tables

This page intentionally left blank.

1. Introduction

This final report outlines the work done in fulfillment of SPPACS Maintenance and Enhancement Contract No. W7707-021856/001/HAL, by MacDonald Dettwiler and Associates Ltd. (MacDonald Dettwiler). This work was performed for Defence Research and Development Canada (DRDC) - Atlantic under the direction of the Scientific Authority (SA), Jim Theriault from approximately March 2003 to January 2004.

The contract was executed in three phases. The initial contract only involved two phases of work but the contract was amended to add a third phase. The first phase involved improving software quality and creating a development approach and structure for the Signal Processing Package (SPPACS) software suite, while enhancing it for DAT32 capability. The second phase involved development of a dynamic range emulation application for use simulating the performance of an operational towed array and is to be documented elsewhere. The third phase involved enhancing or adding application modules to support Directional Acoustic Sensor Module (DASM) beamforming.

The contract provided an excellent opportunity not to only perform the specific enhancements required by this contract, but also to improve the overall design, flexibility and robustness of SPPACS. Once common errors, design issues and problems were discovered solutions were devised to avoid these pitfalls in the future. Effort was used to standardize and simplify the reading and writing of data files through better-designed library functions and templates. Library functions were developed to perform functions such as reading and writing headers and data records, completely abstracting many of the complete header management tasks, as well as the header format differences. A template program was then written to illustrate the functions' use and to standardize development. This abstraction of complex tasks from day-to-day development was important to reduce overall development time and to avoid common errors found in much of the existing SPPACS software. Standardized software design also helps to improve developer ramp-up time, as once they have seen one program they are capable of quickly understanding or developing others.

The new SPPACS software template design also allows the processing to be decoupled from the data handling. This provides three advantages. The first is to simplify the task of supporting new data formats, if the need arises. The second is to allow the simple processing tasks to be reused in different and potentially more complex multi-layer processing with little extra effort. The third is to better conform to basic software design principals that provide general flexibility and robustness in design.

The software design was iterated and improved as the contract progressed, so not all software makes full use of the latest design. It would have been too costly to attempt to upgrade all software with each design improvement. However, as software is modified to meet new requirements or funds become available, affected programs can be upgraded to the newest design.

The following section provides some background to familiarize the reader with SPPACS and the project requirements. Subsequent chapters provide the following:

- Statement of Requirements – outlining detailed information on the contract requirements and how they are met
- Project Planning and Definition – records the work done during this phase
- Process Requirements – details various processes applied to SPPACS
- Data Dictionary – describes this deliverable
- Test and Independent Validation and Verification Plan – provides general information on how the SPPACS software is tested and tracked
- Test Specification – details software testing procedures for two levels of testing
- SPPACS Clean-up – documents the work performed during the clean-up portion of Phase 1
- DAT32 Implementation – documents how DAT32 is implemented in SPPACS including legacy software support and new software design
- DASM Processing – documents the software that was developed or modified to realise this capability

Though this report documents the modifications to SPPACS for the purposes of this contract, the software continues to evolve. A number of products have been developed in order to maintain current documentation. Documentation including users guides, design documentation and algorithm information can be found in the sppacs/docs directories. The users guide is available as both html and man pages. A website has also been established and can be located at https://star.iotek.ns.ca. This limited access portal contains portal access instructions and issue tracking software that can be used to submit and view software and documentation issues, as well as enter enhancement ideas for long-term tracking. At the time of writing an effort was also being made to provide How-To type information and it is likely that the variety of products will continue to evolve as the product matures and users provide feedback. It is strongly recommended that users refer to the sppacs/docs directory or the web site for the most current information.

1.1 Background

1.1.1 SPPACS

SPPACS is a group of software programs that is based on the C programming language and is implemented on Linux based Personal Computers (PC). Each program

provides a specific processing function and a series of programs can be chained together to create a custom-processing stream using the command line or scripts. The output from SPPACS is stored in DREA formatted data files. SPPACS has slowly evolved to its present day state due to the efforts of several MacDonald Dettwiler personnel over the last 4 years.

SPPACS has been used to perform a number of mid-trial and post-trial processing functions such as the post-trial study of multistatic trial data and the mid-trial analysis of the Q265 sonobuoy test trial. SPPACS only performs data manipulation and does not provide an interface to examine the results. The processed data output is often imported into other applications that enable data display and are used to perform the detailed analysis of the results such as the Software Tools for Analysis and Research (STAR), another MacDonald Dettwiler product developed for DRDC Atlantic.

The SPPACS software suite consists of two types of software. One type is runtime executables that can be used to process DRDC Atlantic data files in a number of ways including data management and signal processing. Each program performs a specific function and the programs are designed so that they can be used in conjunction to perform more complex processing tasks. The software has proven to be very useful in simplifying data management and sonar processing tasks by providing a set of tools from which to build the necessary processing streams. These streams can be run from the command line or assembled into scripts to perform batch-processing tasks allowing for large amounts of data to be automatically processed. The second form of the software is a group of library functions that can be used by other programs to efficiently perform standard tasks. These library functions are extensively used by the runtime software, but can also used for other applications.

SPPACS is also supported by a set of signal processing libraries known at FFTW. These free open-source libraries provide optimized signal processing functions helping to ensure that the SPPACS software runs as efficiently as possible, while providing a significant reduction in coding effort.

Before the spring of 2002 many of the custom software packages at DRDC Atlantic were under little to no configuration management. At this time, a co-op student was employed to migrate some of this software to the Concurrent Versions Systems (CVS) repository where the management, development and installation of the software placed under the control of the DRDC Atlantic Software Control Project (SCP) would be standardized. SPPACS was part of this effort. The SCP version of SPPACS had a number of implementation issues and the functions used to support its configuration were in question at the time of contract award. A similar, but not identical approach to software management was chosen for SPPACS that maintains the spirit of the SCP.

Despite the work performed to continue to develop SPPACS and to place SPPACS under version control, formal test cases did not exist that allow users to confirm the correct operation of the software once compiled and installed. This lack of formal testing has occurred due to limited resource allocation and subsequent scope management on previous development contracts.

1.1.2 DAT32 Format

The legacy DREA data format was the only format supported by SPPACS. That format only allows for 64 channels of data. It also restricts the precision available for the numbers in the data header. An extended version of the format was created in an effort to remove these limitations and is know as the DAT32 format. Higher precision values are now allowed for the data header fields and extra gain blocks can be appended to allow for more channels, if required.

2. Statement of Requirements

This section provides a list of the required work and outputs from the subject contract along with a detailed description of each requirement. It is provided in fulfillment of the contractual obligation for a Statement of Requirements and is based on available contract documentation and a meeting with the SA and Technical Authorities (TA). The future tense was used to describe requirements as this section was written at the start of the contract.

2.1 Document Deliverables

The following documents are required as part of this contract. Each document is provided with a description of the required contents. It is considered acceptable to provide the required documentation as part of the final report.

Though the spreadsheet referenced in this section was created, it was decided in consultation with the SA, not to publish it as part of this report because it was in a state of flux due to other contract development. It is intended to publish this spreadsheet as part of a planned maintenance build of the software in the summer of 2004.

2.1.1 Data Dictionary

The data dictionary is provided as an excel spreadsheet.

Lists of the data formats that can be used for input to each function and that are produced by each function are required. This includes DAT and DAT32 formats, their sub-types and number types. An example of this is DAT time series (Type I) in real integer format. It will also include custom data for the library functions.

Modification to the data header that occurs inside each function will be listed. For example, spare3 is used to store the number of beams formed by sp_cardioid.

The entry arguments for each function and the acceptable limits and units for associated values will be provided.

2.1.2 Test Specification

The test specification is provided as a subsection for each program documented in the final report.

The test specification provides a description of the testing methodology that is used for the testing of that program.

2.1.3 Test and Independent Validation and Verification Plan

The test plan is provided as a section in the final report.

The test plan provides a description of how the Test Specification is applied to the SPPACS software suite and how current issues and the results of test and Independent Validation and Verification (IV&V) efforts will be tracked.

2.1.4 Test Description

The test description is provided as a subsection for each program documented in the final report.

The test description will provide a list of the elements of each function that require testing and provide a list of the tests that will be used to confirm the proper functionality of each element. A separate test description will be provided for each program or function that has a test prepared.

2.1.5 Test Report

The detailed test report is provided as a subsection for each program documented in the final report.

The test report consists of a table with each test case and a pass / fail result along with any required explanation.

2.1.6 Test and IV&V Summary

A test and IV&V summary is provided as an Excel spreadsheet for quick reference.

The test portion provides the status of the following testing stages for each program or function:

- Subject Matter Expert (SME) Confidence: This will be a subjective confidence value that provides an indication of quality with 10 being perfect quality. It will be used to prioritize test and debug efforts and stand in place of formal testing until performed.

- Test Description Completed (Y/N): Indicates that a test description has been documented.

- Test Cases Generated (Y/N): Indicates that the required test data and processing scripts or commands have been generated to execute the test description.

- Test Cases Automated (Y/N): Indicates that an automated test case exists that can be run by the user after installation to check the integrity of that software component.

- Test Cases Executed (N/Pass/Fail). 'N' means that this test case was not executed. A fail is accompanied by the specific test case that fails and any other relevant information in the notes. Details on the testing are available in the test report.

The IV&V portion provides the status of the following stages of IV&V:

- User Documentation Reviewed (Y/N): This indicates whether or not the current documentation has been reviewed to ensure that it provides all options available in the current version of the software.

- Technical Documentation Created (Y/N): This indicates whether or not a Microsoft Word technical document has been created that contains the description, requirements, design, algorithm and test documentation.

- Documentation Standardized (Y/N): This will indicate whether or not the current documentation has been edited to conform to the documentation standard provided for the User's Manual.

- Calibration / Algorithm (Y/N): This will indicate whether or not the current documentation contains the required level of detail on the software algorithms and output units or calibration.

2.1.7 User's Manual

The User's Manual will be provided as a Lyx generated Latex document using the DRDC provided style. This document will be able to be converted to HTML for online viewing and help. Individual documentation for each program or function will also be able to be converted to a man page as best able for quick reference. Formatting effort will be directed towards the DRDC formatted User's Manual and it will be considered acceptable to lose some content and / or formatting during the latex2man conversion.

The User's Manual will provide the information below. It is understood that the current documentation does not provide this level of detail and that it may not be feasible to provide this level of detail for all programs by the end of the relevant contract phase.

- An introduction that provides information about the general purpose and utility of SPPACS.

- A step-by-step software installation procedure and details on software dependencies.

- A description and examples of instructions that are common to many of the functions or general in nature such as how to index channels and how to stream together several SPPACS functions to provide a required processing stream.

- The user instructions related to each program or function with the following detail:

  - A brief description of the program, current version number and revision date.

  - The purpose or common uses for the program.

  - The entry arguments for the program, an example and an explanation of each entry argument.

  - The implementation details for the function including the algorithm, units used and normalization / calibration information. This subsection generally incorporates a reference to the related program's technical documentation in Microsoft Word format.

  - A list of related programs for reference.

2.2 Software Baseline and Deliverable

The SPPACS software on ptcruiser at DRDC will be used as the baseline version for contract start. The feasibility of continuing to support the DRDC SCP version of SPPACS will be studied. If feasible, the DRDC SCP version of SPPACS will be repaired and newer versions of the SPPACS software will be merged with the SCP version.

Checking the software into the CVS software repository and then checking it out and performing a successful build on a Linux PC running a complete installation of RedHat 8.0 must be successful to consider the delivery of the SPPACS software suite complete. This build of the software will be written to CDR and delivered to DRDC Atlantic via MacDonald Dettwiler Data Management to provide formal tracking of software delivery.

2.3 Testing Requirements

Detailed testing requirements will be provided in the Test Specification and Test Description, but as a general rule the correct functioning of the programs is what will be tested. This means that incorrectly formatted input, out of range values and thorough exception handling will not be tested.

2.4 High Priority Programs

The following programs and related library functions will be given the highest priority for the testing and IV&V process:

- sp_tape2file
- sp_file2tape
- sp_beamform

2.5 DAT32 Enhancement

A DAT32 capability will be provided for SPPACS. Preference will be given to providing a library routine that will handle DAT32 headers. This library routine would then be used to incorporate the DAT32 format capability into the required programs, while ensuring that they can also handle the required number formats. The feasibility of this implementation will be examined once investigation of the DAT32 format is completed.

The DAT32 information provided by Bruce Skinner, DRDC Atlantic, will be used as the standard for the DAT32 capability and data provided from TIAPS trials will be used for testing.

2.6 DASM Processing

On 3 September 2003, MacDonald Dettwiler received a contract requiring them to add a DASM beamforming capability to SPPACS. This requires the following additions and modifications:

- sp_beamform will be enhanced to full DAT32 compatibility.
- sp_carioid will be enhanced to full DAT32 compatibility and enhanced to allow only a single dipole and an omni-directional channel at the input.
- sp_integrate will be created as a DAT32 compatible program to convert the dipole accelerometer data to velocity data.
- sp_hack will be converted to sp_extract and sp_byteswap splitting the functionality of the program and adding full DAT32 compatibility. sp_extract will have the ability to save the extracted data to file.
- sp_merge will be created as a DAT32 compatible program to merge two data files.

2.7 Data Processing

The project team (MacDonald Dettwiler and DRDC) will endeavour to reserve 10% of the available man-days for data processing effort. This amount will not be changed without approval of the SA.

2.8 Level of Effort

It is understood that there is a fixed level of effort available to perform the contract and that only 25% of the contract value has been allocated to SPPACS Maintenance efforts, while 75% has been allocated for DRE development and data processing.

Cooperation of the entire project team is required to efficiently allocate the available resources, maintain a reasonable expectation of progress and to avoid non-essential tasks thus ensuring the best possible outcome for the project. This will include quick turn around times for information requests and reviews to avoid wasted effort.

2.9 Reporting, Direction and Tracking

Changes in project scope or direction will only be accepted if approved by the project SA, Jim Theriault. Liaison is permitted with other DRDC staff, as required and technical direction will be accepted from the two TAs, Brian Maranda and Dave Hazen.

The project progress will be tracked via short reports that will be sent via email at a minimum frequency of once per month. Each report will include:

- Details of progress to date including percentage complete. For the purposes of this contract percentage complete will entail the percentage of budget used for that task. Significant deviations between percentage complete and expected progress will be explained.
- Rationale behind design decisions
- Effort used to date
- Earned Value
- Forecast work and effort for the next 30 days
- Open programmatic risks and issues

3. Project Planning and Definition

The following tasks were performed as part of the project planning:

- A kick-off meeting was conducted to further define the contract requirements and options.
- A Statement of Requirements (SOR) was generated based on the results of the kick-off meeting. The SOR was recorded in the final report.
- Process Requirements were defined and documented in the final report.
- Draft versions of the Data Dictionary, and Test and IV&V Summary were created as Excel spreadsheets.
- A draft Test Specification was generated and documented in the final report.
- Skeleton versions of the Test and IV&V Plan, Test Description and Test Report were generated and inserted in the final report.

A draft copy of the final report and Excel Spreadsheets containing the content described above were sent to the SA for review and comment.

4. Process Requirements

4.1 Overview

The SPPACS distribution has been continually evolving over the past five years during its use in several small contracts. The requirement exists for a standard set of processes to ensure that subsequent projects can continue in a structured manner and with reduced overhead. Reduced overhead can be realised by using the project planning and process development conducted by earlier projects as a baseline. This document will outline several key processes for the SPPACS distribution that were found to be both useful and efficient. It is recommended that future work follow these same processes.

4.2 Status Tracking

For the purposes of this contract, status tracking is the process of maintaining an accurate record of the status of each component in SPPACS. The products resulting from status tracking enable managers to:

- Quickly assess the status of a component
- Determine if maintenance, scientific or development effort is required prior to conducting related work
- Prioritize and manage available effort for maximum effect

Status tracking was achieved by producing the Test and IV&V Summary Report using an Excel spreadsheet. The summary information contained in this report is supplemented by the detail contained in the issue tracking database and test reports.

4.3 Issue Tracking

Issues are problems or actions that have a limited lifespan and must be resolved in order to meet an objective. They can take a number of forms including:

- Software Problem Reports (SPR)
- Suggested Enhancements
- Programmatic Concerns or Risks

Issue tracking is the creation of, management, resolution and closing of issues. Tracking is required to ensure that issues are properly logged and managed avoiding lost data and ideas that will lead to repeated problems.

Issue tracking for SPPACS was performed using the Bugzilla issue-tracking tool. The issue database is maintained at MacDonald Dettwiler Halifax and issues can be generated and managed via a simple web based interface to the server.

It is recommended that DRDC Atlantic and MacDonald Dettwiler Halifax work together to establish a Bugzilla server that can be accessed from both facilities, thus allowing users and developers to enter and access information. In the meantime, a listing of relevant open issues can be included with the SPPACS distribution for each formal release.

4.4 Configuration Management

Configuration management is the version archiving and build maintenance of software and related artefacts. As directed by the contract, configuration control will be conducted using the CVS.

The CVS database is maintained at two locations. A development version of the CVS database is maintained at MacDonald Dettwiler Halifax. Delivery of the SPPACS software includes the check-in of the release version to the CVS repository at DRDC Atlantic.

Changes between software releases also need to be managed. This can also be done by CVS. Each time a developer checks in a new version of a source file CVS requires a log entry. CVS provides a utility called rcs2log that will be used to create a change log from the cvs log. Each new release will have a new change log generated for it.

The commands required to create a change log are shown below and will create a file called “ChangeLog” using all cvs log information for the *src_directory*.

$ cd *src_directory* (checked out directory from cvs repository)

$ rcs2log > ChangeLog

4.5 Change Process

Change management is the process of prioritizing, approving and tasking changes to an entity. In this case the entity is the SPPACS software suite.

DRDC must manage the change process for SPPACS software. It is recommended that DRDC name a configuration control authority that will coordinate contracts and approve change requests. This authority ensures that requested modifications do not adversely affect other users.

4.6 Development Process

The SPPACS development process involves the following steps:

1. The requirements of a new or modified program are established with the SA.

2. The initial design is developed and documented using Enterprise Architect.

3. Bugzilla is examined for known issues related to the subject program.

4. The requirements, design overview, any required algorithms and testing methodology are documented using the SPPACS program report template and passed to the Project Engineer (PE), SA or TA for review, as required by the SA.

5. Once the approvals referenced in Step 4 are obtained, the current SPPACS program template is used to create new programs, or old programs are modified using the current version of the software from CVS, as directed by the PE. The developer conducts initial testing on the software to ensure that the program will run and produce correctly formatted output.

6. Once initial development is complete, or as required, the PE ensures that a code review is conducted.

7. Testing and debug is performed to ensure that the program produces the correct output.

8. The Lyx based user document is created.

9. Documentation is brought up to date and reviewed, as required.

10. Tracking spreadsheets and the data dictionary are updated, as required.

11. All artefacts are confirmed checked into CVS and the work package is closed.

5. Data Dictionary

SPPACS operates on data files that use the DRDC standardized data format. It is beyond the scope of this report to fully document that format. Documentation for the standard DAT format can be found in, TM87/302, “A Standard for the DREA Data Descriptor Block”, Douglas A. Caldwell, January 1987.

At this time there is no formal documentation for the DAT32 format, though it is very similar to the DAT format with increased variable precision. A summary of the DAT32 format is provided in “DAT32 Implementation.”

Data dictionary information specific to SPPACS can be found in an evolving Microsoft Excel document. After consultation with the SA, it was decided not to publish the document with this report as it was in a state of evolution due to other contract requirements. It will be published as part of a planned software build in the summer of 2004.

6. Test and IV&V Plan

This section provides a description of how the Test Specification is applied to the SPPACS software suite and how current issues and the results of test and IV&V efforts will be tracked.

Testing takes place in three stages. First, a Test Description is generated that conforms to the Test Specification. The requirements for the Test Description are provided in Section 2.1.4. Next any required test data, instructions and scripts are created to enable the testing. Once testing is complete a Test Report is generated as described in Section 2.1.5.

IV&V involves the appropriate independent review of the testing procedure and results along with the actual software and documentation to ensure that it conforms to defined standards. At a minimum, this review is conducted by the PM, PE or an independent team member.

The testing and IV&V efforts are tracked in the Test and IV&V Summary described in Section 2.1.6.

7. Test Specification

7.1 Introduction

The test specification provides a description of the testing methodology used for the testing of the SPPACS software suite. Testing could be performed at two general levels, high-level and low-level, which is described below. The actual tests conducted are described in the test description.

The objective of this testing is to verify that the functionality expected from the utilities/routines was correct, starting with the highest-priority programs.

The testing methodology developed during this project also provides guidelines for testing any remaining or new functionality.

7.2 High-Level Testing

High-level testing involves testing of a complete program module rather than its individual components and is conducted using the following guidelines:

- The functional requirements of each utility/routine are prioritized and assessed for likelihood of failure. This assessment is primarily based on code complexity and historical software issues based on experience with SPPACS development. The significant requirements are mapped to specific test cases ensuring that the test effort is executed as efficiently as possible.

- The test cases involve running the program with known input data and examining the program output for the expected result.

- Only valid equivalence classes are utilized to verify that the utilities/routines work correctly, when provided with correct inputs. These decisions, though not optimal for quality assurance, are taken to make the best possible use of available testing effort.

- Where possible the test cases are automated to allow users to quickly verify that their build of a distribution is operating correctly on their computer. These automated tests also serve as regression tests for future development.

- Test cases and data are archived in a Configuration Management repository to ensure their continued integrity.

- For simple test cases, test automation involves examining the output or specific portions of the output for the expected result. Testing software is developed for this purpose.

- For complex test cases, test automation takes place in two stages:

  - The first stage involves manual testing and analysis that is used to produce a benchmark case.

  - The second stage involves constructing an automated test that compares the output from automated testing against the benchmark case. This comparison may also involve the comparison of floating point numbers or other variables that cannot be expected to produce an exactly equivalent result on all hardware so special software is required to assist automation.

  NOTE: Care must be taken to produce short enough benchmark data and to reuse benchmark data to avoid excessive growth of the distribution size.

- Tests are documented in a test description and the results are reported in a test report that becomes part of the work package report. The work package report is incorporated into the final report and delivered to the SA.

The baseline test cases can be supplemented with new or improved test cases as enhancements are made to the distribution or more effort is made available for testing.

7.3 Low-Level Testing

Low-level testing involves the isolation and testing of individual program components or functions. This testing method is more thorough and test cases can be more readily created without accounting for complex inter-component dependencies. The testing method can also consume considerable effort. Ideally, critical components or components such as frequently used library functions should be tested using low-level testing. Regardless, when it is decided to conduct low level testing, enhancements are unit tested using the following guidelines:

- The functional requirements of each program component are prioritized and assessed for likelihood of failure. The significant requirements are mapped to specific test cases ensuring that the test effort is executed as efficiently as possible.

- Common black-box testing methods (equivalence partitioning, boundary-value analysis and error guessing) are used to ensure a reasonable level of quality.

- Where possible the test cases are automated to allow users to quickly verify that their build of a distribution is operating correctly on their computer. These automated tests also serve as regression tests for future development.

- Tests are documented in a test description and the results are reported in a test report.

The baseline test cases can be supplemented with new or improved test cases as enhancements are made to the distribution or more effort is made available for testing.

8. SPPACS Clean-Up

The following sections outline the clean-up work performed on the various SPPACS modules. Each subsection is derived from the software documentation stored in the docs/reports subdirectory of the SPPACS distribution. These reports are based on a standard format that is modified as required for each program and serves as the technical documentation of the program. The standard outline for each report contains:

- A short description of the program and recent work performed
- A list of the program's functional requirements
- A description of the software design
- The software test description detailing the testing used
- The software test report detailing the results of testing

8.1 sp_tape2file

Initial testing revealed a number of errors with the program including problems with how data segments were extracted using start time and duration and how byte swapping was conducted. Initially repairs were started and a number of tests were generated, but it was later decided to remove a significant portion of the functionality for the reasons described above, rather than complete repairs. As a result the program complexity and final testing requirements were also significantly reduced.

The rewrite of the sp_tape2file program resulted in immediate DAT32 compatibility.

The changes to sp_tape2file include:

- Removal of start time and duration options. These options were defective and redundant in SPPACS. The old software would still sequentially read to the file start time, which would not result in program slow down with its removal and the duration calculations and algorithm was defective and overly complex. It is recommended that if these functions are re-implemented that they use information from the header and inherent tape seek functionality to extract a file segment more quickly and easily.

- Removal of channel extraction. This function would not speed the program and was redundant in SPPACS. It is recommended that sp_hack be used on the output to perform this function.

- Removal of byte-swapping. This function was defective and redundant in SPPACS. It is recommended that sp_byteswap be used on the output to perform this function.

- Addition of automatic CVS version and revision owner annotation.

- Program modularization to improve readability and ease software maintenance.

- Addition of a standard command line option parser component.

8.1.1 sp_tape2file Requirements

The sp_tape2file requirements are listed below along with their priority and the test cases that have been assigned to test each requirement.

Table 1. *sp_tape2file Function Coverage Matrix*

FUNCTIONS/INPUTS	PRIORITY	TEST CASES
Read a data set from a tape and write to stdout.	HIGH	All
Read a data set from a tape containing multiple data sets.	MEDIUM	3
Display a version message.	LOW	None
Display a usage message.	MEDIUM	None

8.1.2 Design information

The sp_tape2file software is designed to be independent of the data format and will not byte_swap data. The actual data is not interpreted by sp_tape2file, but only passed through to stdout.

The header type (DAT vs. DAT32) is required for tape to disk based file system conversion of that block of data. Headers are always stored as 1K blocks on tape.

This utility supports the following command line options/switches:

-s --skip COUNT	skip COUNT files before reading. Default 0.
-t --tape-device DEVICE	read data from DEVICE. Default /dev/tape
-d dat32	input header in DAT32 format
--help	display this help message and exit
-- version	display version information and exit

8.1.3 Test Description

The following test cases are used to test sp_tape2file. Detailed information on the tests can be found by reading the software in <SPPACS_root>/src/test/data_extraction/sp_tape2file.sh. The sp_tape2file test is run by executing this program.

The automated test cases are generated using manual analysis of the results using DISPX and sp_dh. Three tests are used:

1. Read a DAT time-series data set. Should produce a DAT data set.
2. Read a DAT32 data set. Should produce a DAT data set.
3. Read the second DAT data set from a tape containing multiple time-series data sets.

8.1.4 Test Report

DISPX and sp_dh were used to compare the original and results of a COPYDAT copy to tape and sp_tape2file copy from tape. The results of the first test run were used as a benchmark for automated tests. The block size, record size and # of records were different but valid due to manipulation by COPYDAT.

Table 2: *Test Results – tape2file*

TEST CASE	RESULT (PASS/FAIL)	FAILURE DETAILS / NOTES
1	PASS	-Used dispx to compare original and results. Benchmark from test case 1 will be used for automated tests. -Block size, record size and # of records different but valid.
2	PASS	-Used dispx to compare original and results. The data files were different. -Copydat auto-detects DAT32 headers and converts them to DAT. The file size doesn't change after copydat converts the header. -Used sp_file2tape to write aDAT32 data file to tape, sp_file2tape to read the data file back and diff to compare the two files (not automated). This worked. The files were identical.
3	FAIL	- The current implementation of sp_tape2file has problems skipping files. This is recorded as Bugzilla issue #17.

8.2 sp_file2tape

Initial testing revealed a number of errors with the program, including problems with reading past the first file on tape and how byte swapping was conducted. Initially repairs were started and a number of tests were generated, but it was later decided to remove a significant portion of the functionality for the reasons previously described,

rather than complete repairs. As a result the program complexity and final testing requirements were also significantly reduced.

The rewrite of the sp_file2tape program resulted in immediate DAT32 compatibility.

The changes to sp_ file2tape include:

- Removal of start time and duration options. These options were redundant in SPPACS. It is recommended to use sp_hack to replace this functionality. Enhancements to sp_hack would allow it to immediately go to the portion of the input file, if stdin was not used in this configuration thus saving processing time. This can be accomplished by placing sp_hack at the start of a processing stream.

- Removal of channel extraction. This function would not speed the program and was redundant in SPPACS. It is recommended that sp_hack be used on the input to perform this function.

- Removal of byte-swapping. This function was defective and redundant in SPPACS. It is recommended that sp_byteswap be used on the input to perform this function.

- Addition of automatic CVS version and revision owner annotation.

- Program modularization to improve readability and ease software maintenance.

- Addition of a standard command line option parser component.

8.2.1 Requirements

The sp_file2tape requirements are listed below along with their priority and the test cases that have been assigned to test each requirement.

Table 3. *sp_file2tape Function Coverage Matrix*

FUNCTIONS/INPUTS	PRIORITY	TEST CASES
Read a data set from stdin and write to a tape.	HIGH	All
Append a data set to a tape containing multiple data sets.	MEDIUM	3
Display a version message.	LOW	None
Display a usage message.	MEDIUM	None

8.2.2 Design Information

The sp_file2tape software is designed to be independent of the data format and will not byte_swap data. The actual data is not interpreted, but only passed through from stdin.

The header type (DAT vs. DAT32) is required for tape to disk based file system conversion of that block of data. Headers are always stored as 1K blocks on tape.

This utility supports the following command line options/switches:

Sp_file2tape OPTIONS… > OUTPUT_FILE	
-s --skip COUNT	skip COUNT files before writing. default 0.
-t --tape-device DEVICE	write data to DEVICE. default /dev/tape.
-d –dat32	input header in dat32 format
--help	display this help message and exit
--version	display version information and exit

8.2.3 Test Description

The following test cases are used to test sp_file2tape. Detailed information on the tests can be found by reading the software in <SPPACS_root>/src/test/data_extraction/sp_file2tape.sh. The sp_file2tape test can be run by executing this program.

The automated test cases are generated using manual analysis of the results using DISPX and sp_dh. Three tests are used:

1. Write a DAT time-series data set. Should produce a DAT data set.

2. Write a DAT32 data set. Should produce a DAT data set.

3. Write the second DAT data set from a tape containing multiple time-series data sets.

8.2.4 Test Report

DISPX and sp_dh was used to compare the original and results of an sp_file2tape copy to tape and a COPYDAT copy from tape. The block size, record size and # of records were different but valid due to manipulation by COPYDAT.

Table 4: *Test Results – sp_file2tape*

TEST CASE	RESULT (PASS/FAIL)	FAILURE DETAILS / NOTES
1	PASS	Used sp_file2tape to copy a file to tape, copydat the file back to disk and diff to compare the two files.
2	PASS	-Used dispx to compare original and results. The data files were different. -Copydat seems to auto-detect DAT32 headers and converts them to DAT, but the file size doesn't change after copydat converts the header. It should be reduced by 512 bytes. -Used sp_file2tape to write a DAT32 data file to tape, sp_file2tape to read the data file back and diff to compare the two files (not automated). This worked. The files were identical.
3	FAIL	- The current implementation of sp_file2tape has problems skipping files. This is recorded as Bugzilla issue #17.

8.3 sp_byteswap

sp_byteswap was created to replace the byte swapping functionality that had been built into most of the utilities. As a result, the utilities have become much simpler and thus easier to develop and maintain.

8.3.1 Requirements information

- Swap all data from little endian to big endian and from big endian to little endian.
- Handle DAT and DAT32 formats.
- Handle any precision of integer or float and real or complex data.
- Print a usage message (help).
- Print a version message.

8.3.2 Test Description

The following test cases are used to test sp_byteswap. The test are written in one script; <SPPACS_root>/src/test/data_extraction/sp_byteswap.sh >, so that they can also be used as regression tests. To run the tests execute run_test.sh (runs all tests) or cd to data_extraction and run sp_byteswap.sh (only runs the sp_byteswap tests). Three tests were generated:

1. Convert a little endian dataset to big endian (DAT).
2. Convert a little endian dataset to big endian (DAT32).
3. Convert a big endian dataset to little endian.

8.3.3 Test Report

Table 5. *Test Results – sp_byteswap*

TEST CASE	RESULT (PASS/FAIL)	FAILURE DETAILS/ NOTES
1	PASS	Convert to big endian and compared to a benchmark file. Verified benchmarks using dispx.
2	PASS	Convert to big endian and compared to a benchmark file. Verified benchmarks using dispx.
3	PASS	Convert to big endian and compared to a benchmark file. Verified benchmarks using dispx.

8.4 sp_compare

This utility was created to test data sets that contain floating-point data blocks. It is DAT and DAT32 compliant.

8.4.1 Requirements Information

- Read DAT or DAT32 data formats.
- Compare a floating-point data block based on a tolerance setting using a standard algorithm.
- Print a usage message (help).
- Print a version message.

8.4.2 Software Design information

This utility reuses several general-purpose functions from libutils to read/write headers and compare their contents.

This utility supports the following command line options/switches:

```
Usage: sp_compare –a FILE –b FILE [-c][-d][-t][-h][-v]
        -a FILE             first dataset
        -b FILE             seconds dataset
        -c                  first dataset contains a DAT header
        -d                  seconds dataset contains a DAT header
        -t TOLERANCE        set floating point comparison tolerance. Default 1e-6.
        -v                  print verbose messages
        -h                  print this help message and exit
```

8.4.3 Algorithm Information

All floating point numbers are compared using

$$|u - v| \le e * |u| \text{ and } |u - v| \le e * |v| \quad (1)$$

where u and v are the variables being compared and e it a comparison threshold. This ‘close-compare’ allows for reasonable differences in floating point numbers due to changes in processors and processing approximations.

8.4.4 Test Description

The following test cases are used to test sp_compare. Test case scripts and/or code can be found in <SPPACS_root>/src/test/data_header. The unit tests can be run by executing <SPPACS_root>/src/test/run_test.sh (will run all unit tests) or <SPPACS_root>/src/test/data_header/sp_comare.sh (only sp_compare unit tests).

The automated high-level tests are verified using sp_compare, dispx and sp_dh. Five tests are used:

1. Compare a data set containing a DAT data header and floats.
2. Compare a data set containing a DAT32 data header and floats.
3. Compare a data set containing a DAT data header and integers.
4. Compare a data set containing a DAT data header and complex floats.
5. Compare a data set containing a DAT data header and complex integers.

8.4.5 Test Report

Table 6. *Test Results – sp_convert_header*

TEST CASE	RESULT (PASS/FAIL)	FAILURE DETAILS/ NOTES
1	PASS	Compare a data file to itself and then to a different file.
2	PASS	Compare a data file to itself and then to a different file.
3	N/A	Not required for the current task.
4	N/A	Not required for the current task.
5	N/A	Not required for the current task.

8.5 sp_beamform Tests

The performance of the *sppacs* routine *sp_beamform* was tested over frequency, signal bearing and for two spatial windows. The processing noise was estimated and the processing gain was verified.

8.5.1 Requirements

The following tests were performed and documented:

- **Test 1: Performance over frequency:** Test a sufficient number of frequencies to reveal the real time delay filter response characteristics. Document the results on a graph of frequency versus signal level with text discussion.

- **Test 2: Performance over bearing:** Using a bin centred frequency around 75 Hz vary the signal bearing to test beam centred bearing closest to beamfire, endfire and +/- 30 degrees. Document the results on a graph of bearing versus signal level with text discussion.

- **Test 3: Performance with spatial windows:** Using a bin centred frequency around 75 Hz vary the shading (none, Hanning) and show that the beam width and sidelobes are as expected for a beam centred signal. Document the results showing the beam pattern polar plots and text discussion.

- **Test 4: Estimation of the processing noise level:** For at least two frequencies compare the Fast Fourier Transform (FFT) of the input and output data for the signal beam. Graph and discuss the processing noise level.

- **Test 5: Verification of the beamformer processing gain:** Show that the processing gain is 0 dB for a bin and beam centred signal, or explain the results.

8.5.2 Software Design information

The MATLAB scripts developed to generate and analyse test files were archived in the folder *sp_beamform_tests.zip.* A summary of the MATLAB package is presented in this section.

Subroutines:

- fread_drdc_time_hdr.m

- set_array_time_delays.m

- fwrite_drdc_time_hdr.m

- get_drdc_time_hdr.m

- fread_drdc_time_series_data_chunk.m

- get_beam_angles.m

- get_drdc_time_series_data_size.m

- get_signal_energy_level.m

- plot_energy_level_polar_plot.m

- write_beam_data_file.m

Test 1 routines:

- sp_beamform_analyse_test_data_task1.m
- sp_beamform_make_test_data_task1.m

Test 2 routines:

- sp_beamform_analyse_test_data_task2.m
- sp_beamform_make_test_data_task2.m

Test 3 routines:

- energy_polar_plot2.m

Test 4 routines:

- check_sp_beamform_processing_noise_level.m

Test 5 routines:

- check_sp_beamform_processing_gain.m

8.5.3 Algorithm Information

- Definition of signal energy level:

If *s* designates a time series vector of length *N*, the *energy level* of the time series *s* is calculated as follows:

$$Energy\_level = 10 * \log_{10}\left(\frac{1}{N} * \left(\sum_{n=1}^{N} s_n{}^2\right)\right) \quad (1)$$

8.5.4 Test Description

This section describes how the input and output files must be generated and analysed to meet the five test requirements. It is expected that the real time delay filter does not provide an accurate output for the first samples due to filter effects. Thus, when analysing output data files, the first 256 samples (or more) are systematically skipped.

8.5.4.1 Generation of the DRDC Time Series Input Data Files

All the input files generated for these tests represent the pressure signals that would be generated on a linear sensor array in response to a pressure plane wave ensonifying the array. All of these files have the same common configuration:

- Number of beams: 64
- Sampling frequency: 400 Hz
- Sensor spacing: 4 meters
- Number of channels (or sensors): 16
- Number of samples: 8,192

All synthetic data is bin centred in frequency and beam centred in signal bearing. This ensures that no processing effects from scalloping loss affect the results. It also ensures that any non-signal FFT bins should have zero energy given a perfect signal.

8.5.4.2 Test 1

Input files are generated using the MATLAB script *sp_beamform_make_test_data_task1.m*. The resulting input files are processed by *sp_beamform*, using the shell script *sp_beamform_tests_task1.sh.* The output files are analyzed using the MATLAB script *sp_beamform_analyse_test_data_task1.m* to generate Figure 1.

The input parameters (specific to test 1) of the input files are:

- Signal bearing: -29.4763 degrees (Beam 17)

Twenty-one input files are generated, corresponding to 21 frequency points, spanning the frequency range 0-200 Hz (200 Hz is the Nyquist frequency).

The 21 input files are beamformed to generate 21 output files.

A portion of the beam 17 data (-29.4763 degrees) from each output file is loaded (256 points: samples 2049 to 2304) and the energy level of each portion of beam is calculated and plotted.

The filter roll-off (3 dB) should be at 75% of Nyquist and the response should be flat prior to that point.

8.5.4.3 Test 2

Input files have been generated using the MATLAB script *sp_beamform_make_test_data_task2.m*. Then these input files have been processed by *sp_beamform*, using the shell script *sp_beamform_tests_task2.sh.* The output files have been analyzed using the MATLAB script *sp_beamform_analyse_test_data_task2.m* to generate the Figure 2.

The input parameters (specific to test 2) of the input files are:

- Frequency: 75 Hz
- Signal bearing angles:
    - -90 degrees (centred on beam 1)
    - -29.476 degrees (centred on beam 17)
    - -0.9095 degrees (centred on beam 32, (beamfire))
    - +29.476 degrees (centred on beam 48)
    - +90 degrees (centred on beam 64)

Five input files are generated, corresponding to the 5 signal bearings previously listed.

The five input files are beamformed to generate five output files.

For each output file, a portion of the beam centred on the signal bearing is loaded (256 points: samples 257 to 512) and the energy level of each portion of beam is calculated and plotted.

The signal energy should be identical for all beams.

8.5.4.4 Test 3

Two input files previously generated for test 2 are reused (signal bearing = -29.476 degrees and signal bearing = -0.9095). These two input files are processed by *sp_beamform* for two types of window (rectangular and Hanning), using the shell script *sp_beamform_tests_task3.sh.*. The output files are analyzed using the MATLAB script *energy_polar_plot2.m* and the corresponding beam patterns are plotted (using samples 257 to 512).

8.5.4.5 Test 4

Two files previously generated for task1 (*input_task1_f_bin_51.dat* and *output_task1_f_bin_51.dat*) are reused. These files are analysed using the MATLAB script *check_sp_beamform_processing_noise_level.m* as described below to generate plots of FFT data.

A portion of the beam centred on the signal bearing is loaded from the output file. An FFT is performed on it and it is scaled by the number of channels to account for processing gain as discussed in Test 5.

The resulting FFT data is plotted for the two following FFT cases:

a. 256 FFT point (FFT performed on samples 2049 to 2304 from input file)

b. 2048 FFT points (FFT performed on samples 2049 to 4096 from input file)

A portion of the first channel is loaded and Fourier transformed for comparison to the output data results.

Ideally the post-beamformed FFT noise level should be the same as the input data FFT noise level.

8.5.4.6 Test 5

Two files previously generated for Test 1 (*input_task1_f_bin_51.dat* and *output_task1_f_bin_51.dat*) are reused to verify the beamformer processing gain. These files are analysed using the MATLAB script *check_sp_beamform_processing_gain.m* to determine the processing gain.

The processing gain should be 0 dB or a factor related to the number of sensors.

8.5.5 Test Report

This section describes the results of each of the five *sp_beamform* tests.

8.5.5.1 Test 1

Figure 1 presents the frequency response of *sp_beamform* for a signal bearing centred on beam 17. The roll-off frequency at –6 dB is around 150 Hz, which correspond exactly to 75% of Nyquist frequency (200 Hz). The response from 0 to 140 Hz is very flat.

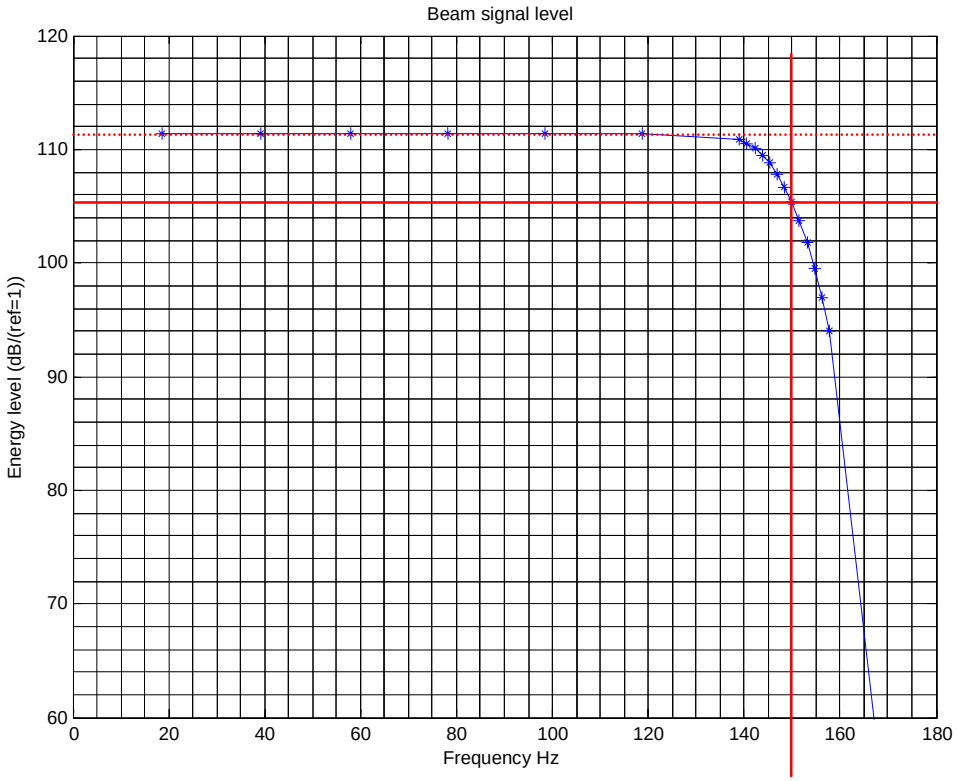

Figure 1. *Frequency Response of sp_beamform (for a signal bearing centred on beam 17)*

8.5.5.2 Test 2

Figure 2 illustrates the performance of *sp_beamform* over signal bearing. The response of the beamformer over signal bearing is flat from –90 to +90 degrees, as expected.

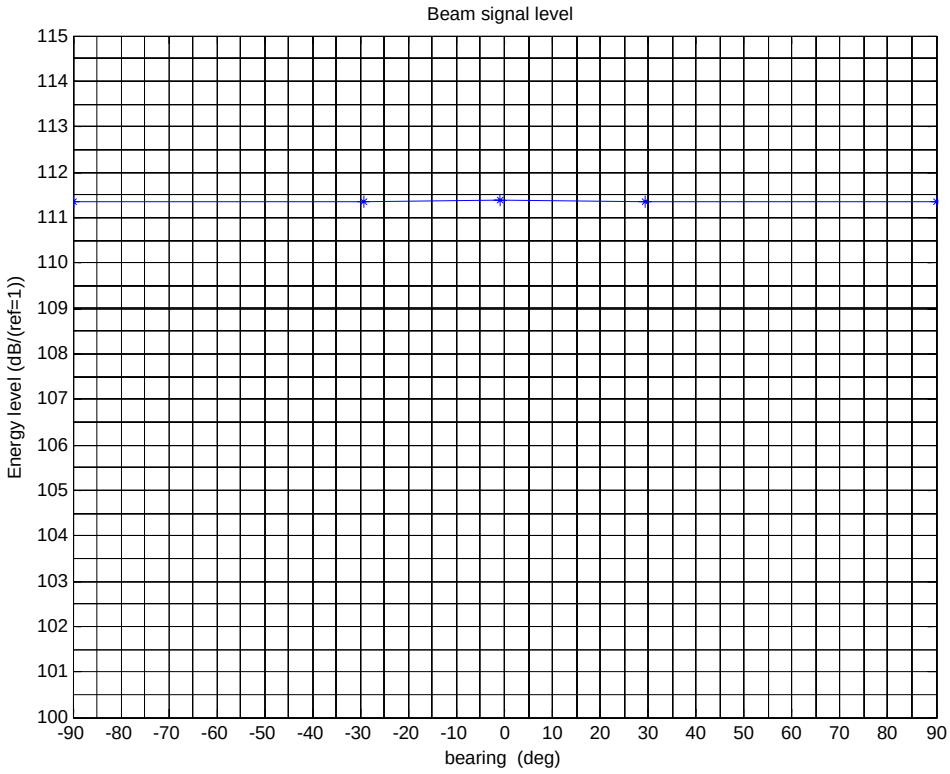

Figure 2. *Performance of sp_beamform Over Signal Bearing*

8.5.5.3 Test 3

Figure 3 to Figure 6 present the beam patterns of *sp_beamform* for two signal bearing angles:

- Around –30 degrees (-29.476)
- Around 0 degrees (-0.9095)

For the two spatial window types:

- Rectangular
- Hanning

In Figure 5, the highest sidelobe is 13 dB lower than the main lobe, exactly as expected for a rectangular window. In Figure 6, the highest sidelobe is 32 dB lower than the main lobe, exactly as expected for a Hanning window.

The beam width measured at the 3 dB down points for Figure 5 is 16 degrees. This compares well to the expected beam width of 15.3 degrees given the synthetic array geometry and a 75 Hz source. The beam width measured at the 3 dB down points is 24 degrees for Figure 6. This is also expected as an increase of 1.5 times the rectangular windowed beamformer is expected for a Hann window.

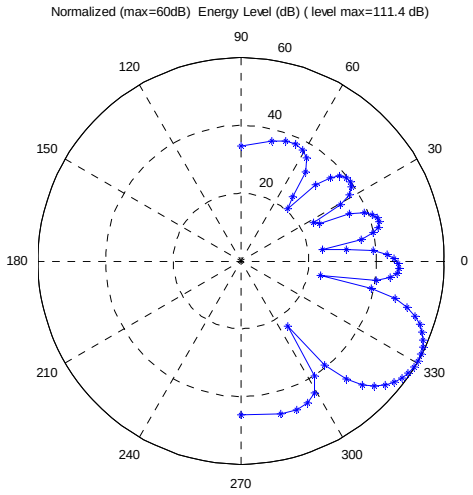

Figure 3. *Beam Pattern of sp_beamform for a Signal Bearing Around –30 degrees (rectangular window)*

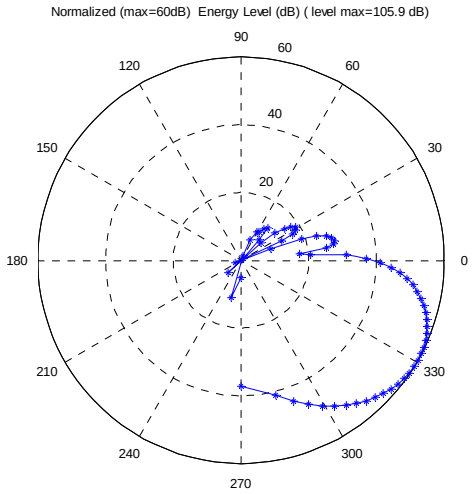

Figure 4. *Beam Pattern of sp_beamform for a Signal Bearing Around –30 degrees (Hanning window)*

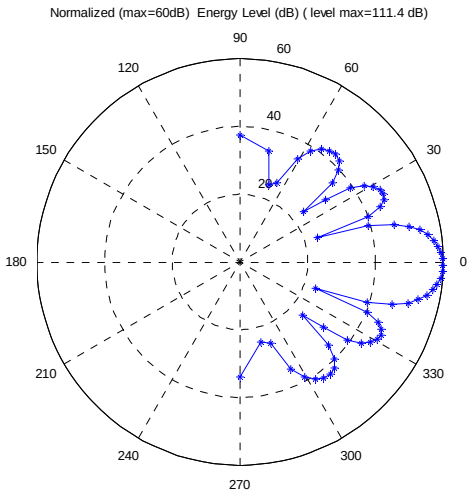

Figure 5. *Beam Pattern of sp_beamform for a Signal Bearing Around 0 degrees (rectangular window)*

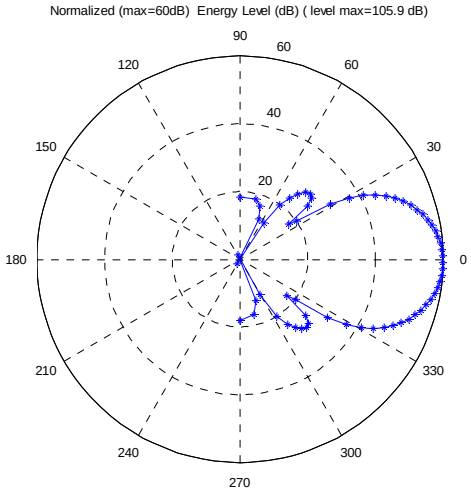

Figure 6. *Beam Pattern of sp_beamform for a Signal Bearing Around 0 degrees (Hanning window)*

8.5.5.4 Test 4

Theoretically, using a rectangular shading window, since all sensor signals add coherently, the beamformer gain on signal level is $20*\log_{10}(N)$ where N is the number of sensors. The beamformer gain on noise level is $10*\log_{10}(N)$, as noise channels add incoherently. Thus, the expected beamformer gain on Signal to Noise Ratio (SNR) is finally $10*\log_{10}(N)$. Figure 7 and Figure 8 compare signal and noise levels in the frequency domain (using 256 FFT points and 2048 FFT points respectively) before and after beamforming, using the routine *sp_beamform*. In these figures, Circles represent the beamformer output while asterisks represent the first sensor channel (channel one of the input data file). In Figure 7 and Figure 8, the beamformer output signal level was corrected by subtracting the theoretical beamformer processing gain on signal level (*i. e.* $20*\log_{10}(N)$). It can be seen that the beamformer and sensor output levels match well (after the $20*\log_{10}(N)$ correction) for the signal frequency bin (78.125 Hz for this case). Each figure presents frequency domain levels generated using a single FFT. There is not enough data averaging to allow us to observe a clear $10*\log_{10}(N)$ beamformer gain on SNR, however, looking at the noise frequency bins, it can be seen that even using single FFT, the beamformer does provide a significant SNR Gain of the order of $10*\log_{10}(N)$.

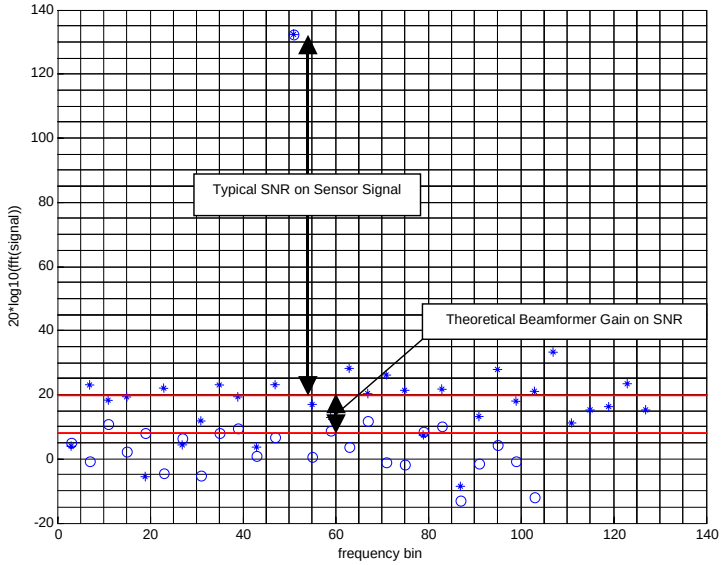

Figure 7. *SNR of sp_beamform (for 256 FFT points)*

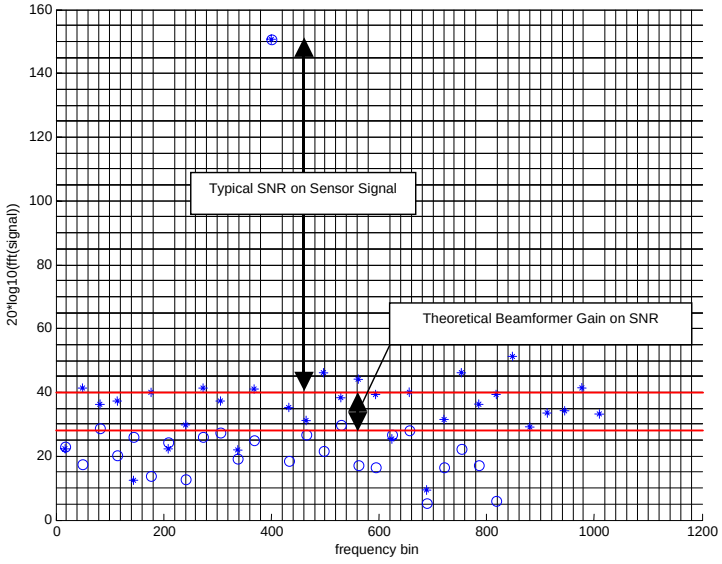

Figure 8. *SNR of sp_beamform (for 2048 FFT points)*

8.5.5.5 Test 5

Figure 9 presents the processing gain of *sp_beamform* on signal level. It can be seen that the beamformer output energy level (from *output_task1_f_bin_51.dat*) corresponds to the energy level of the first sensor channel (from *input_task1_f_bin_51.dat*) plus the beamformer processing gain on signal level $20*\log_{10}(N)$, N being the number of channels. In this example, N=16, which provides a 24.1 dB Gain on signal level.

Note: In Figure 9, the single channel energy level was calculated from the first sensor channel. Any other channel would have given the same energy level as channel signals differ only by a time shift and not amplitude.

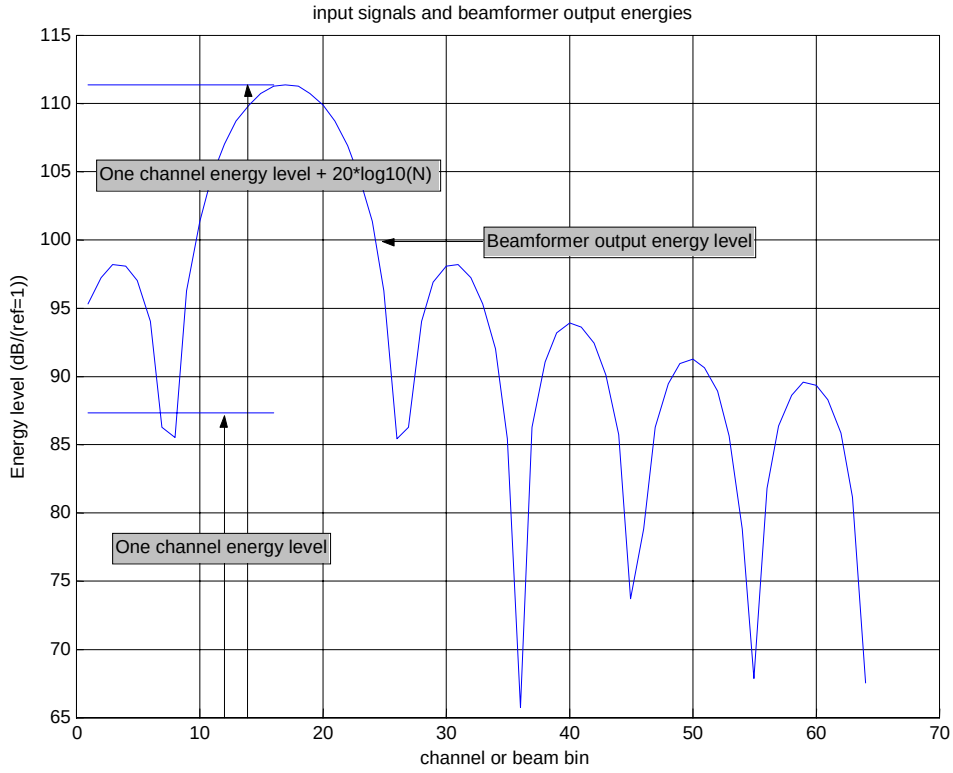

Figure 9. *Processing Gain of sp_beamform*

9. DAT32 Implementation

A requirement existed to add support for DAT32 formatted files to SPPACS. The two highest priority functions were the sp_tape2file and sp_file2tape utilities. These programs were modified to support DAT32 during the SPPACS clean-up effort.

DAT32 Input/Output (I/O) formatted files are identical to regular DRDC data files except the regular DRDC header format is replaced with a new header format as shown in Table 7. Thus the DAT format is a subset or less precise version of the DAT32 format. Programs that support DAT32 use DAT32 as the internal data format and perform any necessary header conversions at the start and end of the program using calls to a support library. The general approach of converting to an internal DAT32 header format will be used for all future development of SPPACS and will remain cost-effective, as generic library routines have been written to handle this requirement.

New programs and those modified to support DAT32 assume that DAT formatted data is input unless a command line switch is set indicating the DAT32 format. The SA indicated that there was no comprehensive automatic detection algorithm for the header type, and for this reason manual header designation is required.

Programs that do not support DAT32 can still be used by using sp_convert_header to convert the input data to DAT format, while accepting the obvious loss in precision of the header data and inability to record byte block information for more than 64 channels. The implementation details for sp_convert_header are described below.

9.1 sp_convert_header

sp_convert_header will convert between DAT and DAT32 formatted headers. It was originally created to convert DAT data files to DAT32 for testing of DAT32 functionality.

sp_convert_header can also be used to convert DAT32 files to DAT format to provide compatibility with legacy SPPACS modules that do not support DAT32. This is done at the expense of losing any extra gain information contained in the header and the extra number precision inherent with DAT32.

9.1.1 Requirements

- Convert a data file from DAT to DAT32
  - If channels > 64 then pad channel gains to number of channels
- Convert a data file from DAT32 to DAT
  - If channels > 64 then truncate channel gains to 64
- Print a usage message (help)
- Print a version message

9.1.2 Software Design information

Several libraries containing general-purpose I/O and header manipulation functions were reused. Integrating these components together took up most of the effort. This program uses:

- short2long_header and long2short_header from libutils
- libarg to construct and interface around the conversion routines
- libio for read/write data block

The utility supports the following command line options/switches:

Usage:

sp_convert_header -h --header DAT|DAT32 [--help][--version] < INFILE > OUTFILE

h --header [--help] [--version] < INFILE > OUTFILE

-h --header DAT|DAT32	INFILE header type. Will convert OUTFILE header to opposite of INFILE.
--help	display a usage message
--version	display version information

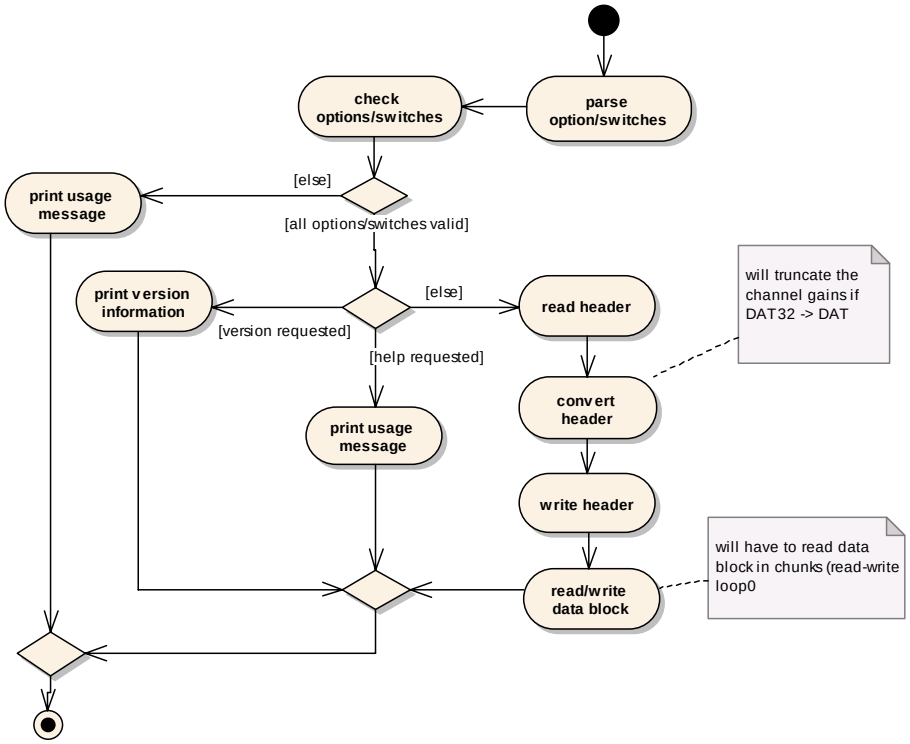

***Figure** 10: This activity diagram shows the control routine for the sp_convert_header utility.*

Figure 10 shows an activity diagram that illustrates the software design. The following table shows the structure of the DAT32 header including extra gain blocks that are appended, as required.

***Table 7:** DRDC DAT 32 Descriptor Block*

COUNT	PARAMETER	DESCRIPTION
Integer Block		
1	Block size	32-bit words
2	Record size	
3	# of records	
4	Repetition rate	

5	Number Type 1 = Real number 2 = Real floating point 9 = Complex integer 10 = Complex floating point	
6	Number of bytes / number 2 = Real integer 4 = Real floating point 4 = Complex integer 8 = Complex floating point	
7	Number of channels	
8	Multiplex length Word = 1 Record = Record size	
9	Number of averages	
10	X-axis units 1 = Time series 2 = Frequency	
11	Y-axis units 1 = Linear 2 = Square 4 = Log	
12	Sequence number	
13	Block scaling factor	
14	Spare	
15	Spare	
16	Spare	

Floating Point Block		
1	Sampling frequency	32-bit words
2	Heterodyning frequency	
3	Reference level	
4	Maximum magnitude of data	
5	Gain correction	
6	Spare	
7	Spare	
8	Spare	
ASCII Block		
1	Date	8-bit words
11	Time	
21	y-axis units	
384	User message	
Channel Descriptor Block		
1	1st channel name	16-bit words
2	Gain of above in db	
3	2nd channel name	
4	Gain of above in db	
…	…	
127	128th channel name	
128	Gain of above in db	
Extra Channel Descriptor Block		
1	129th channel name	16-bit words An extra block is handle for each additional 256 channels.
2	Gain of above in db	
3	130th channel gain	
4	Gain of above in db	

…	…	
511	256th channel name	
512	Gain of above in db	

9.1.3 Test Description

The following test cases are used to test sp_convert_header. They are contained in <SPPACS_root>/src/test/utilities/sp_convert.sh.

1. Convert a data file from DAT to DAT32.

2. Convert a data file from DAT32 to DAT. The data file should have more than 64 channels to ensure that extra gain handling is tested.

9.1.4 Test Report

Table 8. *Test Results – sp_convert_header*

TEST CASE	RESULT (PASS/FAIL)	FAILURE DETAILS/ NOTES
1	PASS	Used dispx and sp_dh to verify output file.
2	PASS	Used dispx and sp_dh to verify output file. Note: Didn't test extra gain portion of test. Extra gains were later tested manually with a supplied DRDC Data file.

10. DASM Processing

On 4 September 2003 the SPPACS contract was amended requiring MacDonald Dettwiler to provide support for DASM processing in SPPACS. The requirements related to this amendment are detailed in Section 2.6.

The following subsections describe the work performed to meet this requirement. The format of each subsection is the same as that described in the introduction to Section 8.

Sample processing scripts for DASM data are provided in sppacs/scripts.

10.1 sp_extract

This utility allows the user to extract data from a DRDC data file. It can also be used to split a file into two files. The first file is written to stdout and is based on the command line options that have been set. The second file is written to a user specified file (must use –f or –second-file option) and contains the remaining data.

The functionality offered by the original sp_hack has been moved or replaced by other utilities. This utility is the final version of sp_hack, which was renamed to better describe its purpose.

10.1.1 Requirements information

- The utility will extract a subset of data from a DRDC data file by extracting either or both a subset of data in time or channel.
    - The channel subset is designated using channel indices.
    - The time subset is designated using the subset start time, in seconds from the start of the file, and the subset duration in seconds.
- The utility will be DAT/DAT32 compliant.
- The utility will allow the user to write the undesignated channels from a subset of data to another file.

10.1.2 Software Design information

Figure 11 shows the logic for sp_extract in the form of an activity diagram. This program makes use of generic libraries for header and data handling.

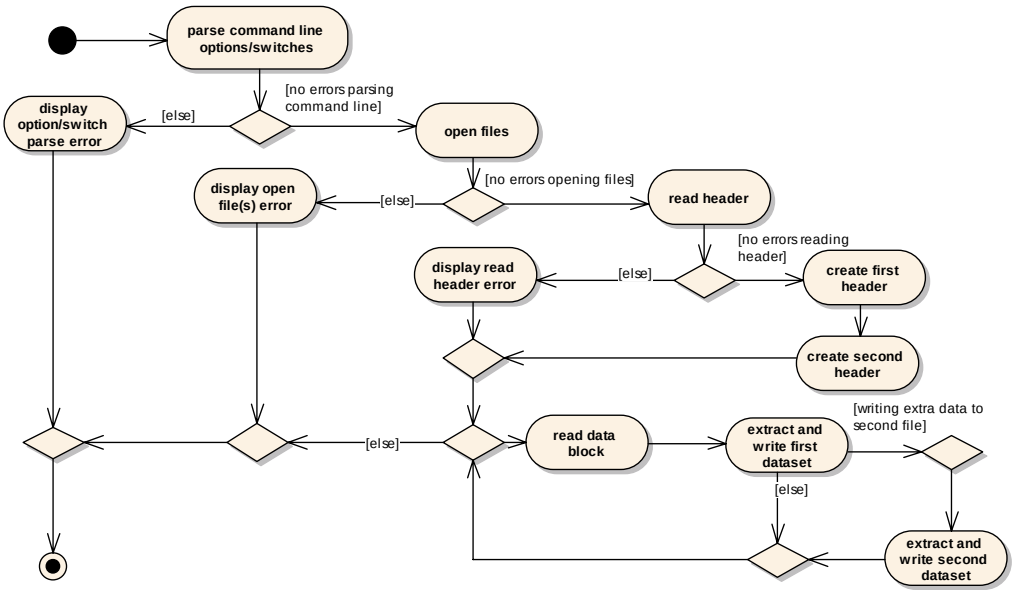

Figure 11. *Activity Diagram showing the program flow for sp_extract.*

10.1.3 Test Description

The following test cases are used to test sp_extract. Test case scripts and/or code can be found in <SPPACS_root>/src/test/utilities. The unit tests can be run by executing <SPPACS_root>/src/test/run_test.sh (to run all unit tests) or <SPPACS_root>/src/test/utilities/sp_extract.sh (to run only sp_extract unit tests).

- Split a DAT file into two files using channel, start time and duration.
- Split a DAT32 file into two files using channel, start time and duration.

10.1.4 Test Report

Table 9. *Test Results – sp_extract*

TEST CASE	RESULT (PASS/FAIL)	FAILURE DETAILS/ NOTES
1	PASS	Used a DAT file with 5 channels. Extracted channels 0,2,4 to stdout and 1 and 3 to another file. Set the offset start time 2 seconds ahead of the files date time. Set the duration to 6.0 seconds. Verified results manually with dispx.
2	N/A	Need files with extra gain blocks.

10.2 sp_merge

This utility allows the user to merge two data files together into a new data file. The newly merged data file is written to stdout. The first data file is read from stdin and the second data file is read from a user specified file.

10.2.1 Requirements information

- The utility will merge two data files.
- The utility will operate on DAT and DAT32 formatted files.
- The utility will operate on time series and power formatted files.
- The headers will be merged in the following manner:
    - The output header's block size, number of channels and record size will be recalculated.
    - For all file formats, the two input headers' number type, number of bytes, multiplex length and heterodyning frequency must match.
    - For power formatted files, the two input headers' centre frequency of first bin, frequency resolution and time interval between spectra must match.
    - For time series formatted files, the two input headers' sampling frequency must match.
    - The blabel and extra channel gains will be taken from the appropriate input header for each channel.
    - Warnings will be generated if the X-axis units and Y-axis units are not the same.
    - The file input from stdin will dominate all other header settings.
- The output data block will be padded with zeros if one data file is shorter than the other.

10.2.2 Software Design information

This utility reuses several libraries that have already been written. The argument parsing library was used to parse the command line options/switches. The log library was used to write information to a log file. The data stream library was used for DRDC file format I/O. Figure 12 provides an activity diagram showing the program flow.

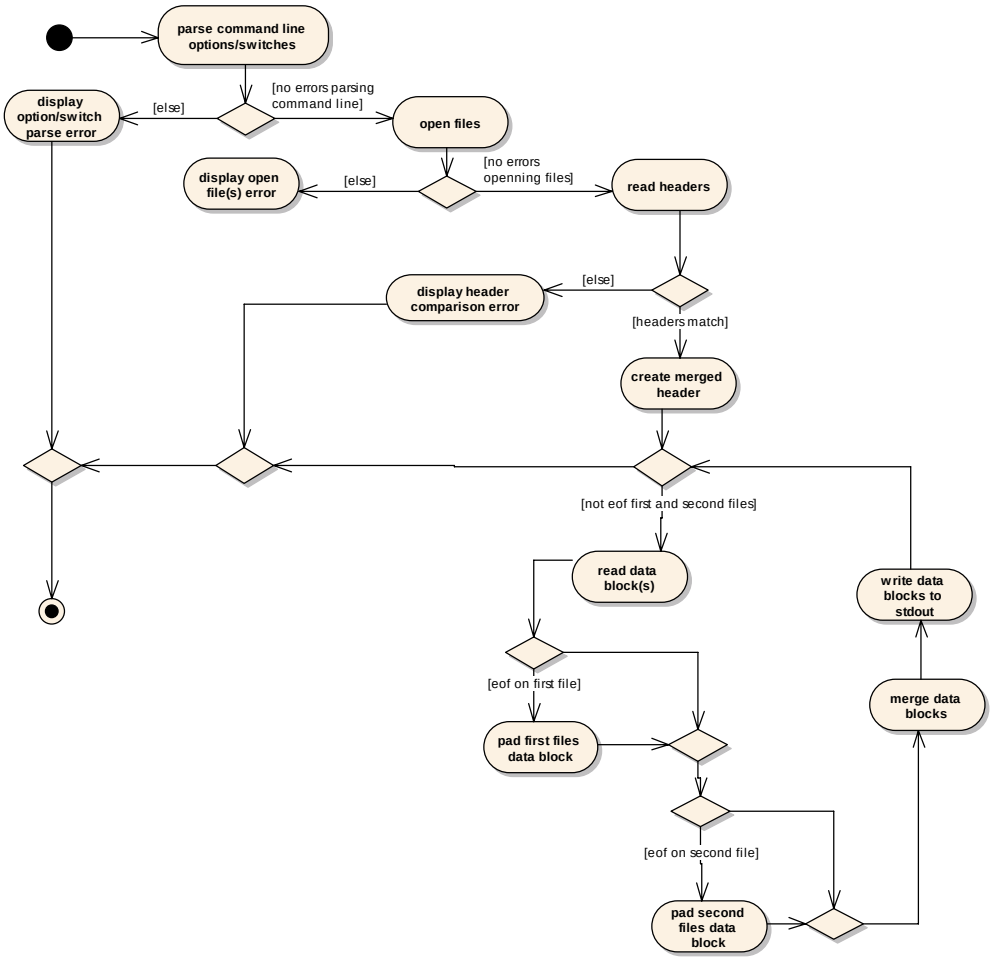

Figure 12. *Activity diagram showing the program flow for sp_merge.*

10.2.3 Test Description

The following test cases are used to test sp_merge. Test case scripts and/or code can be found in <SPPACS_root>/src/test/utilities. The unit tests can be run by executing <SPPACS_root>/src/test/run_test.sh (to run all unit tests) or <SPPACS_root>/src/test/utilities/sp_merge.sh (to run only sp_merge unit tests).

- Merge two DAT files together.
- Merge two DAT32 files together.
- Merge a DAT and DAT32 file together.

10.2.4 Test Report

Table 10. *Test Results – sp_extract*

TEST CASE	RESULT (PASS/FAIL)	FAILURE DETAILS/ NOTES
1	PASS	Used a data file with 2 channels and one with 3 channels. Mapped first file to channel 1 and 3 and the second file to channels 0,2 and 4. Verified results manually with dispx.
2	N/A	Need a file with extra gain blocks.
3	N/A	Need a DAT32 file with extra gain blocks.

10.3 sp_integrate

sp_integrate is a program to accept an input DRDC time series dataset and output it to a similar DRDC time series dataset with the selected channels integrated and otherwise manipulated as described in this section.

10.3.1 Requirements

sp_integrate accepts four main parameters, which along with Figure 13 illustrate the program's functional requirements:

- Header Type – the default is DRDC DAT format. The other option is DAT32.
- Gain setting [dB]– this is used to adjust for the difference in units between the desired input and output units. The first use of sp_integrate is to convert acceleration sensor data to velocity for beamforming with velocity sensor data and the sensitivity ratio between the two sensors is unknown. The default gain is 0 dB.
- Channels to modify – all channels are output to file, but only the selected channels are integrated. Channels that are not selected are output unmodified. This uses SPPACS standard channel definition that is zero based. Extra flags were added to allow the user to extract all even or odd channels.
- High Pass filter coefficient – a high pass filter is required to ensure that very low frequency trends or biases are removed from the data.

The output file will use the same DRDC header, number type and precision as the input file. The input file will be piped in using stdin and the output file will be sent to stdout. It is assumed that the file is in machine endian format.

The program will accept either 2 or 4 byte integer data.

10.3.2 Software Design

An overview of the software design is shown in Figure 13. Most of the program structure re-uses routines written for sp_dre and reuse is shown by "(from main dre loop)" in the figure. The program architecture and option parsing routine are similar to that of sp_dre.

10.3.3 Algorithm

All internal data manipulation uses double precision floats. Modified channel data is converted to double precision floats during the block read and quantized prior to writing back to file.

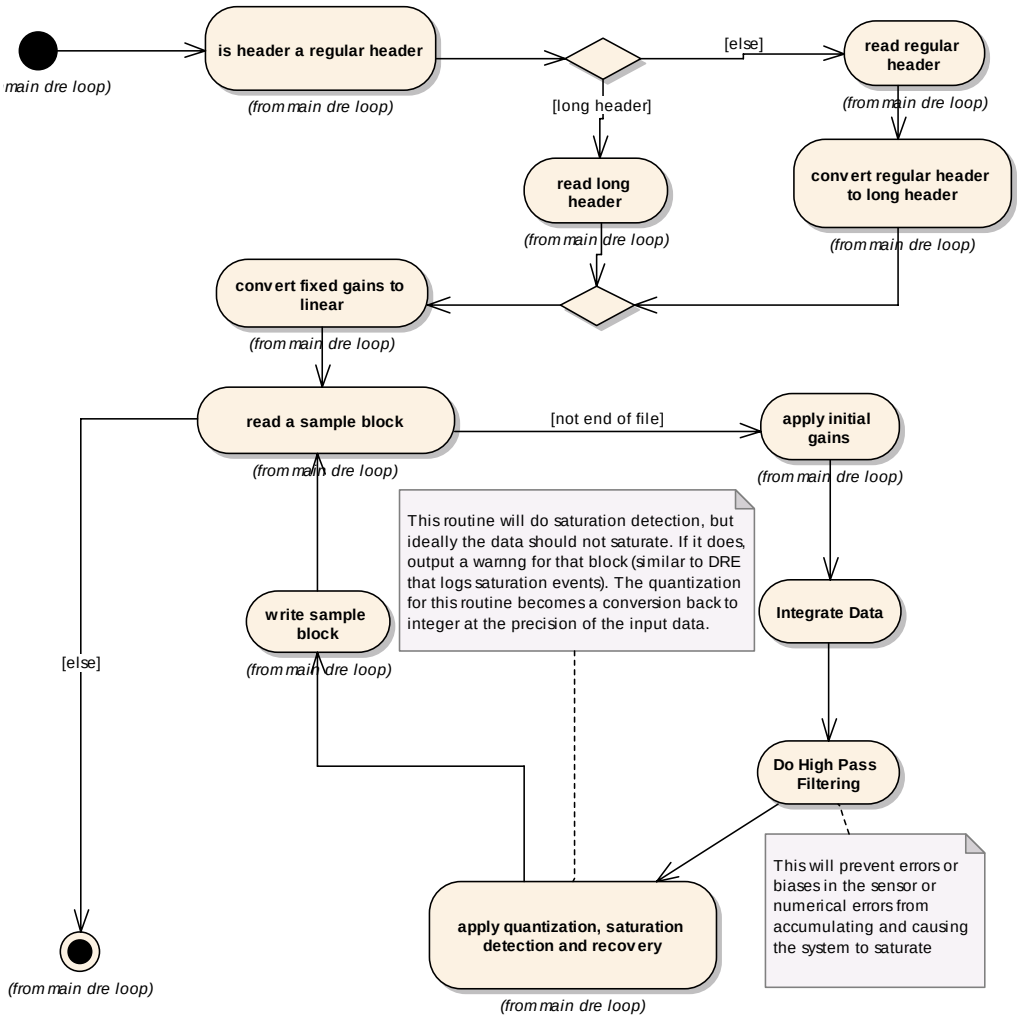

Figure 13. *sp_integrate design*

10.3.3.1 Fixed Gain application

Fixed gains are applied the same as for sp_dre. First, the input gain in units of dB is converted to a linear gain represented as a double precision float. During processing the gain is applied to each sample using multiplication.

10.3.3.2 Integration

Each input channel is integrated using

$$y_{c,n} = \frac{1}{2}\left(x_{c,n} + x_{c,n-1}\right)x_{c,n}\Delta t + y_{c,n-1} \tag{2}$$

where x an input sample for channel c and sample number n, y is the corresponding output sample and Δt is the inverse of the sample rate from the input file header.

10.3.3.3 High Pass Filtering

After integration each channel is separately high pass filtered for trend removal using

$$m_{c,n} = (1-\alpha)\,y_{c,n} + \alpha m_{c,n-1} \tag{3}$$

where m is the current average signal level and α is the filter time constant. This trend is then removed from the integrated data using

$$z_{c,n} = y_{c,n} - m_{c,n} \tag{4}$$

where z is the data sample sent for re-quantization. All other notation remains the same as for (1). For a very slow response α will be very close to 1. m will be initialized to be zero indicating that no bias needs to be removed.

10.3.3.4 Quantization

Quantization re-uses the quantization routine written for sp_dre, but the output data will use all of the dynamic range available for the output number type.

10.3.4 Test Description

The test data is created such that there are channel pairs. The element of the pair to be integrated is a copy of the non-integrated channel, the other pair element, with scaling, a trend and differentiation applied to the data. Each of these manipulations on the integrated test data are designed to confirm the correct functioning of the three algorithm components, gain, filtering and integration respectively. Data files are created and verified using MATLAB.

The tests are verified using sp_compare, dispx and sp_dh and consist of two data sets:

- Integrate a data set containing a data block (DAT) with 4 byte integers.
- Integrate a data set containing a data block (DAT32) with 2 byte integers.

10.3.5 Test Report

Test report information

Table 11. *Test Results – sp_integrate*

TEST CASE	RESULT (PASS/FAIL)	FAILURE DETAILS/ NOTES
1	PASS	
2	TBD	Need to add an appropriate DAT32 file to the dataset.

10.4 sp_cardioid

The purpose of the sp_cardioid program is to read DREA time series, or power spectra data and perform cardioid beamforming on that data. The program's operations are based on the user entered command line parameters. If the user specifies the channels to use for beamforming, only those channels will be processed, otherwise all channels will be processed. An appropriate DREA header and the beamformed data is written to an output file with similar formatting to the input file. The most recent version will work with a single or two orthogonal dipole channels and DAT32 formatted files.

Receiver in this section refers to an associated group of Omni, Sin and Cos channel data, as relevant. For example, a sonobuoy receiver would have one Omni, one Sin and one Cos channel associated to it, whereas one receiver module on a DASM array would have one Omni and one Cos channel associated to it.

10.4.1 Requirements

sp_cardioid permits user defined parameters, which along with Figure 14 illustrate the program's functional requirements. Section 10.4.3 documents how the various parameters are applied to the data.

User modifiable parameters:

- Number of Beams – specifies the number of beams to form. This is a required parameter.
- Power File – specifies the input data as a power file. The default is a time series file.

- Channels– the user can specify the indices of the Omni (O), Sin (S) and Cos (C) channels for each receiver. The number of O, C and S channels specified must be the same and equal the number of receivers to process.
    - The Sin channels need not be specified if there is only one dipole channel.
    - If the Omni channels are set to –1 (all receivers) then it is assumed that the channels are ordered sequentially as O1, S1, C1, O2, S2, C2, etc, or O1, C1, O2, C2, etc., if the Sin channel is suppressed, where the index after the channel type is the receiver number.
    - If the Omni channels are specified and the Sin or Cos channels are set to –1 (all channels) then it is assumed that the channels immediately follow the Omni channel as O1, S1, C1, O2, S2, C2, etc, or O1, C1, O2, C2, etc., if the Sin channel is suppressed, where the index after the channel type is the receiver number.
- Header Type – the default is DRDC DAT format. The other option is DAT32.
- Offset Angle [degrees]– adjusts the orientation of the first beam relative to the orientation of the cosine channel. The default is 0.
- Beamformer coefficient – used to adjust the beam pattern. 0.25 is used for limaçon and 0.5 is used for cardioid. The default is 0.25.
- Cos and Sin dipole gain – used to offset the gain for the Cos and Sin channels in linear units. For example, 1.0/1.6 (the default) is used to adjust for directional sonobuoy data. Setting Cos or Sin dipole gain to zero and specifying redundant channels can be also be used to suppress dipole data.

The output file will use the same DRDC header as the input file except for the following:

- The output data will be in single precision floats.
- Spare3 from the integer block is set to the number of beams formed.
- The byte block is not updated but can be once a standard is provided.

The output data is ordered as Receiver1Beam1, R1B2 .. R1BN, R2B0, R2B2 .. R2BN, etc. so that the number of output channels is equal to the number of receivers times the number of beams formed.

The input file will be piped in using stdin and the output file will be sent to stdout. It is assumed that the file is in machine endian format.

The program will accept either 2 or 4 byte integer, or single or double precision float data in either real or complex format.

The current CVS version number or help information can be output to the command line using the standard commands --version and --help.

10.4.2 Software Design

The sp_cardioid program is comprised of a main function, a number of logical functions, and a settings structure. The main function controls the sequences of events by calling logical sub-functions to perform operations on the input data. Figure 14 illustrates the software design in the form of an activity diagram.

10.4.2.1 Structure

The settings structure contains information relevant to the cardioid beamforming operations. The structure is responsible for providing a container to store the relevant beamform settings entered by the user from the command line. Default values are used if there are no user parameters.

10.4.2.2 Functions

The critical functions comprising the sp_cardioid program:

- main – This method is responsible for controlling the flow of the program. The main function invokes sub-functions to complete the cardioid beamforming.
- sp_cardioid_get_beamforming_coefficients - This method is responsible for calculation of the beamforming coefficients, which will be applied to the time series or power file data read in.
- sp_cardioid_set_channel_arrays – This method is responsible for setting an array that represents the channels of the receiver (i.e. omni, cos, sin) that will be used in the beamforming process. If the user specifies single dipole then the sin channels will not be used in the beamforming process.
- sp_cardioid_validate_settings_and_header – This method is responsible for checking the indices arrays to ensure that the user has not tried to read more channels then are present in the data.
- sp_cardioid_validate_channels - This method is responsible for ensuring that the user has entered the correct command line parameters. These checks ensure that the sin, cos and omni channels are valid.
- sp_cardioid_validate_settings - This method is responsible for parsing the command line parameters for the arguments array and recording the user chosen values, or the defaults, in the settings structure.
- sp_cardioid_process_float_data – This method is responsible applying the beamform coefficients to a block of the time series or power data, which has been converted to floating point. This results in beamformed data.

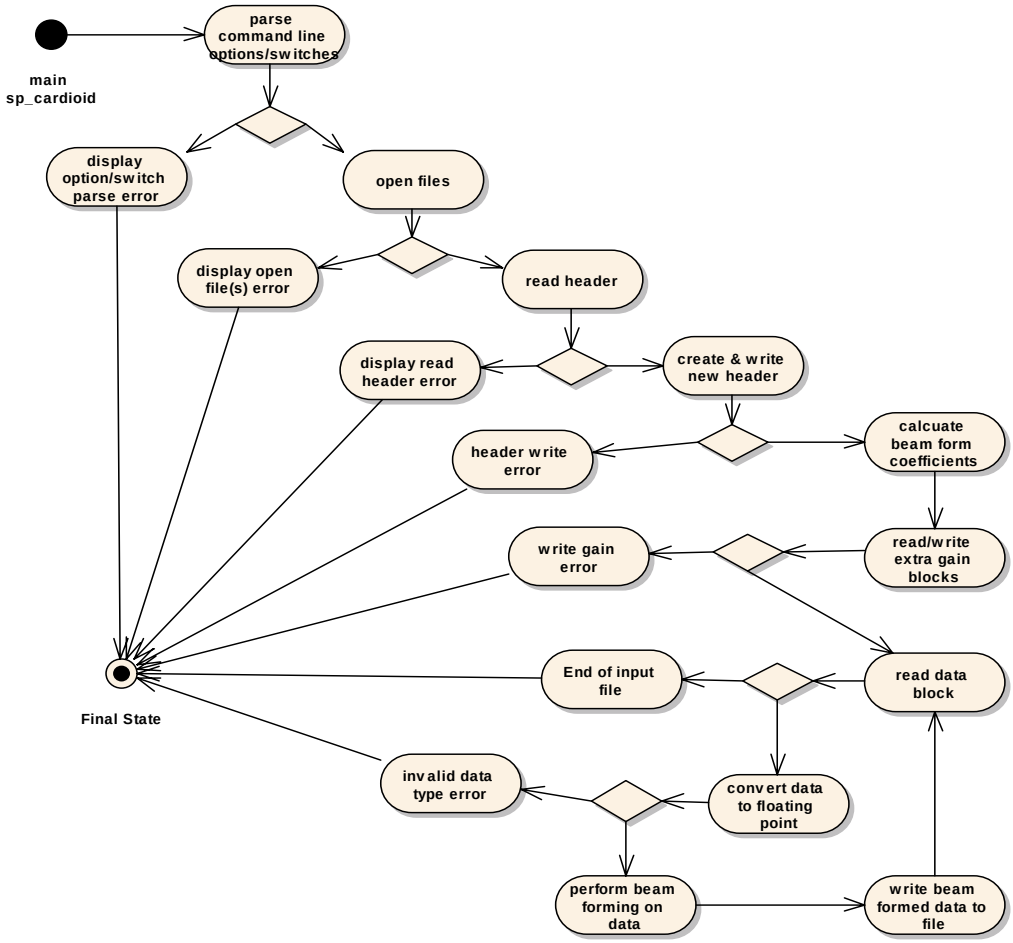

Figure 14. *sp_cardioid Activity Diagram*

10.4.3 Algorithm

All internal data manipulation uses single precision floats. Modified channel data is converted to single precision floats during the block read and the data is written as single precision floats as either real or complex data.

10.4.3.1 Cardioid Beamformer

Each input receiver is integrated using

$$y_{m,n} = \alpha \times O_m + (1-\alpha)\left[S_{gain} Sin(2\pi m / N + \frac{\Delta_\theta \pi}{180}) \times S_m + C_{gain} Cos(2\pi m / N + \frac{\Delta_\theta \pi}{180}) \times C_m \right] \quad (2)$$

where:

- $y_{m,n}$ is the output sample for sample m and beam n of N beams
- α is the beamformer coefficient
- O_m, S_m and C_m are the respective receiver's Omni, Sine or Cos channel's m^{th} sample
- S_{gain} and , C_{gain} are the respective dipole gains
- Δ_θ is the offset angle in degrees

10.4.4 Test Description

The sp_beamform test data is created such that there are channel triplets or doublets to represent the required omnidirectional and dipole channels. The data contains signals that are beam centred and consist of both narrowband and broadband data. Data files are created and verified using MATLAB.

Extensive manual unit testing are conducted to test the command line options in a free-play scenario. The unit tests are verified using sp_compare, dispx and sp_dh. Two of the manual test cases that will be used to create automated unit tests that are described by:

1. Beamform a dual orthogonal dipole data set containing a data block (DAT) with 4 byte integers to form 8 beams.
2. Beamform a single dipole spectral data set containing a data block (DAT32) with single precision floats to form 1 beam with an offset angle of 180 degrees.

10.4.5 Test Report

sp_cardioid passed extensive manual free-play testing, but these tests have yet to be converted to an automated test.

10.5 sp_beamform

Modifications were made to sp_beamform to make it DAT32 compliant.

10.5.1 Requirements information

- Must be able to read and write DREA DAT32 data files.
- Must be able to accept 2 and 4 byte integers and 4 and 8 byte floating-point data as input.

10.5.2 Software Design information

sp_beamform was modified to accept the header type definition for the input data, shading file and delay file at the command line input.

The process library was modified to meet the requirements under this task. Several functions were created to work with DAT32 format headers:

- Sp_hdr_parse_dat32
- Sp_hdr_update_dat32
- Sp_initial_seek_stream_dat32
- Sp_read_stream_dat32
- Sp_write_stream_dat32
- Sp_process_combine_dat32
- Sp_prepare_output_buffers_dat32
- Sp_read_input_buffer_dat32

10.5.3 Test Description

Some of the test cases referenced in Section 8.5.4 were made into automated regression tests in the SPPACS distribution. To execute the test run sp_beamform.sh in ssp/MDA/sppacs/src/test/detailed_beamform_tests/Task1/sp_beamform.sh or ssp/MDA/sppacs/src/test/run_test.sh. All tests can be run manually by executing sp_beamform_tests_task*.sh in each sub directory under ssp/MDA/sppacs/src/test/detailed_beamform_tests/.

10.5.4 Test Report

All tests listed in Section 8.5.4 were conducted and analyzed in MATLAB with identical results to those shown in Section 8.5.5.

List of Acronyms

A/D	Analogue to Digital
ADC	Analogue to Digital Conversion
AGC	Automatic Gain Control
CVS	Concurrent Versioning System
DASM	Directional Acoustic Sensor Module
DND	Department of National Defence
DRDC	Defence Research and Development Canada
DRE	Dynamic Range Emulation
DREA	Defence Research Establishment Atlantic
DRP	Document Review Panel
FFT	Fast Fourier Transform
HTML	Hyper Text Markup Language
I/O	Input / Output
IV&V	Independent Validation and Verification
LFA	Low Frequency Active
PC	Personal Computer
PE	Project Engineer
R&D	Research and Development
SA	Scientific Authority
SCP	Software Control Project
SME	Subject Matter Expert

SNR	Signal to Noise Ratio
SOR	Statement of Requirement
SPL	Sound Pressure Level
SPPACS	Signal Processing Package
SPR	Software Problem Reports
STAR	Software Tools for Analysis and Research
TA	Technical Authority
TIAPS	Towed Integrated Active Passive Sonar

Distribution list

Internal Distribution

2 DRDC Atlantic LIBRARY FILE COPIES
3 DRDC Atlantic LIBRARY (SPARES)

2 Authors
3 J. Theriault
1 N. Collison
1 B. Maranda
1 D. Ellis
1 J. Osler
1 P. Hines
1 S. Pecknold
1 W. Roger
1 M. LeFrancois
1 F. Desharnais

External Distribution

1 DMSS 7-3
1 NDHQ/ DRDKIM 3
1 Library and Archives Canada
Atten: Military Archivist, Government Records Branch

Total 22 copies

This page intentionally left blank.

DOCUMENT CONTROL DATA
(Security classification of the title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document, Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's document, or tasking agency, are entered in section 8.)

Publishing: DRDC Atlantic
Performing: MacDonald Detwiller and Associates

Monitoring:
Contracting:

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification is indicated by the appropriate abbreviation (S, C, R, or U) in parenthesis at the end of the title)

SPPACS Maintenance and Enhancement (U)

4. AUTHORS (First name, middle initial and last name. If military, show rank, e.g. Maj. John E. Doe.)

Joe D. Hood; Brad Glessing

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
April 2007	78	

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical document, technical note or memorandum. If appropriate, enter the type of document, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The names of the department project office or laboratory sponsoring the research and development − include address.)

Sponsoring:
Tasking:

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
11cm	W7707−021856/001/HAL
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document)	**10b. OTHER DOCUMENT NO(s). (Any other numbers under which may be assigned this document either by the originator or by the sponsor.)**
DRDC Atlantic CR 2005−252	

11. DOCUMENT AVAILABILITY (Any limitations on the dissemination of the document, other than those imposed by security classification.)

Unlimited distribution

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11), However, when further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited announcement

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

The DRDC Atlantic Sonar signal Processing Package (SPPACS) has been widely used for a variety of sonar data analysis tasks. Recently, the Towed Integrated Active Passive Sonar (TIAPS) Technology Demonstration project has made significant use of SPPACS. This contract included improving the testing of SPPACS and the capability to perform the specialized beamforming of acoustic data from the TIAPS Directional Active Sensor Module (DASM). This document outlines the software enhancements used for the project. (U)

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title).

Signal Processing
Underwater Acoustic
Sonor
TIAPS

This page intentionally left blank.

Defence R&D Canada

Canada's leader in defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca