STAR Support 2010

Final Report

Joe Hood
David Flogeras
George Ryan
Akoostix Inc.

Prepared By:
Akoostix Inc.
10 Akerley Blvd - Suite 12
Dartmouth NS B3B 1J4
Joe Hood
Contractor's Document Number: AI CR 2011-009
Contract Project Manager: Joe Hood, (902) 404-7464

PWGSC Contract Number: W7707-4500813980
Technical Authority: James A. Theriault, Defence Scientist

Contract Report
DRDC-RDDC-2017-C084
March 2012

Principal Author

Original signed by Joe Hood

Joe Hood

President

Approved by

Original signed by Robert A. Stuart

Robert A. Stuart

Head, Technology Demonstration

Approved for release by

Original signed by Calvin V. Hyatt

Calvin V. Hyatt

Chairman, Document Review Panel

Abstract

This report summarizes the work performed during the STAR Support call-up. The goal of the contract was to provide timely support for a variety of DISO-related requirements. Akoostix successfully met this goal by: (1) supporting the generation DRDC publications, (2) investigating the ISIS single board computer as a more capable and up-to-date replacement on DRDC remote sensors, (3) developing a STAR training course and facilitating that course to DRDC personnel, (4) developing a feature extraction application in support of tracking research that can extract features from multistatic data, and (5) updating STAR software and enhancing OPD's display capabilities to increase DRDC's analysis toolkit.

Executive summary

STAR Support 2010: Final Report

Joe Hood; David Flogeras; George Ryan; March 2012.

Introduction: DRDC Atlantic has been developing generic software usable for signal processing applications in order to support general research and analysis objectives. The set of software is called STAR (Software Tools for Analysis and Research), which provides a toolset for data processing, display, detailed analysis, and recording. The STAR suite currently includes the following software modules for desktop analysis, real-time remote processing, and libraries to enable the analyst or researcher to easily extend its capability.

Results: This report summarizes the work performed during the STAR Support call-up. The goal of the contract was to provide timely support for a variety of DISO-related requirements. Akoostix successfully met this goal by: (1) supporting the generation DRDC publications, (2) investigating the ISIS single board computer as a more capable and up-to-date replacement on DRDC remote sensors, (3) developing a STAR training course and facilitating that course to DRDC personnel, (4) developing a feature extraction application in support of tracking research that can extract features from multistatic data, and (5) updating STAR software and enhancing OPD's display capabilities to increase DRDC's analysis toolkit.

Significance: This effort has extended the STAR user base through training, the development of updated features related to multistatic active sonar, and general software improvements. It comprised five distinct and separable tasks, each contributing to enhancing the usefulness of the STAR components.

Table of contents

List of figures

List of tables

1 Introduction

This report documents the work performed for the STAR Support call-up for Project Authority (PA) Jim Theriault under contract W7707-4500813980. The work was performed between January and March 2011.

The objective of this work was to provide a timely support for a variety of DISO-related requirements. The overall effort was managed in consultation with the PA while individual requirements involved consultation with an appointed Technical Authority (TA).

The remainder of this document provides additional detail related to the contract and is organized as follows:

- Section 2 summarizes the work Akoostix performed to assist the generation of publications including reports related to Smart Automation, DIFAR bearing tests, and transient localization. The TA for this task was James Theriault.

- Section 3 documents the results of an investigation to replace the Viper board with the more powerful ISIS platform. The TA for this task was Tim Murphy.

- Section 4 describes the STAR training and user support provided to DRDC personnel who attended two separate training sessions hosted at Akoostix. (This work was continued with a supplementary purchase order where a third training session and some one-on-one training sessions were provided.) The TA for this task was Cristina Tollefsen.

- Section 5 describes the development of a feature extraction application to support tracking research at DRDC. The TA for this task was Garfield Mellema.

- Section 6 describes various STAR improvements and OPD enhancements performed as part of this call-up. The TA for this task was Jim Theriault.

- Section 7 provides configuration management information about the software released at the end of this call-up.

- Section 8 provides a list of recommended repairs and enhancements for consideration during future work.

- Annex A is a user instruction guide for the feature extraction application developed for support to target tracking (Section 5).

- Annex B provide background on the software that was used or updated as a result of this contract.

2 Support to generation of publications

Akoostix was required to support the generation of DRDC publications that were started as part of prior work. Akoostix effort included:

- Review of documents and DRDC author changes to those documents,
- Addressing comments produced during DRDC document reviews, which included altering text and improving figures, and
- Performing data analysis, required to improve the document and related science.

Work was planned in the following areas:

- DIFAR Bearing Bias documentation,
- Smart Automation documentation, and
- Transient Localization analysis and documentation.

The TA was provided with updated documentation and analysis, once work was completed. Only the documentation on DIFAR bearing bias [1] was supported as available resources were assigned to other tasks.

3 System investigation and integration support

3.1 Introduction

The Arcom (now Eurotech) Viper single board computer (SBC) has been used in several of DRDC's embedded systems since 2004 including:

- RDS,
- LAND Buoy (active and passive variants),
- Slocum glider science module, and
- Stealth Buoy.

The purpose of this investigation was to find a replacement for the in-service Viper board (version V1.16), since it is no longer in production, and newer processing requirements are likely to require more processing power. Viper board (V2.13) was considered, but poses porting challenges. The ISIS (also manufactured by Eurotech) was selected to as a possible replacement. It is based on a more modern architecture, the Intel Atom (versus the Intel PXA series of ARM processors).

A development kit was purchased by DRDC and the following sections outline the configuration, profiling and difficulties which were discovered during the investigation.

3.2 General

The ISIS SBC has the following general components:

- Intel Z5xx series processor (1.3 GHz for extended temperature version used in testing),
- 1 GB of RAM,
- 2 GB Intel Flash SSD drive,
- 2.5" IDE interface,
- USB,
- 8x GPIO pins (six are 5 volt tolerant),
- 2 external serial ports, and
- 1 internal serial port hard-wired to a Fastrax GPS unit.

DRDC also purchased a Silicon Devices 2.5" IDE 8 GB SSD to benchmark.

Unlike the V1 Viper, which required a special ARM-based build of Linux and development toolkit provided by Eurotech, the ISIS will run any x86 based Linux distribution. For the purposes of the investigation, a pared down version of Gentoo Linux was installed.

Gentoo was selected since it is extremely easy to customize its packages (for instance building all packages with support for threading, but without language bindings for FORTRAN or Perl). This allows the user to create a streamlined installation of only those options desired. Furthermore, a large part of reducing the Viper's boot time involved configuring a stripped-down version of the Linux kernel. This is extremely easy to do with Gentoo.

The operating system was installed on the onboard Intel flash SSD drive. When the 2.5" SSD was installed, a jumper was used to set it as the IDE bus slave.

3.3 Hardware failure

Unfortunately, a day of effort was lost when the CPU board initially installed in the development kit started malfunctioning. It would stop at the BIOS screen and never boot the operating system.

While investigating this issue, we also discovered that the CPU carrier board included with the spare setup was damaged (a surface mount part was torn from the board). Furthermore, the carrier boards were of two different hardware revisions, which would make it impossible to use the spare with the touch screen in the development kit without a new cable. The working CPU board was matched with the good carrier board, and the other two were sent back on Return Merchandise Authorization (RMA).

3.4 General purpose I/O

The processor boards integrated with the Slocum Glider and LAND Buoy use the General Purpose I/O (GPIO) pins to control other system components. This control ranges from enabling the active LAND Buoy's amplifier with a single pin, to controlling signal-conditioning gain and bandwidth on the Slocum Glider and passive LAND Buoy (requiring 7 pins).

The software shipped with the ISIS SBC includes example code for controlling the GPIO pins of the board. Unfortunately the state of this code is quite rough, and only controls six of the eight pins stating the other two as "TODO." It also states that future versions of the software should be converted to a proper Linux kernel driver, as the current implementation is simply a software library which pokes at the hardware. This implementation was tested at DRDC, and works for the six GPIO pins it is capable of controlling.

After examining the schematics it was determined that the signal conditioning board could possibly be controlled with only 5 GPIO pins by sharing the DATA and CLK pins between the two buses (gain and bandwidth each use a separate three-wire serial bus). This might require a small amount of additional digital buffer circuitry, depending on which GPIO pins are used. Initial investigation shows that the SCH 311x SuperIO chip used on the ISIS has several pins capable of sinking/sourcing the required current to drive two inputs on the signal conditioning board.

A newer version of the example code was obtained from Eurotech which they claim to supports the other two GPIO pins. Again, the code is not a proper driver and seems to be developmental quality. However, Akoostix recommends using the pin multiplexing described above to reserve additional GPIO pins for future use.

3.5 Sound card

The sound card used on systems where an analog to digital conversion board is not installed, such as the Active LAND Buoy (for output) or the Stealth Buoy (for input). The sound card on the ISIS SBC is based on an Analog Devices AD1986A codec, which is capable of sample rates up to 96 kHz.

Unlike the Viper, the audio codec on the ISIS is not included on the main board. The development kit includes a breakout board with the required codec and connectors. The required board is available separately from Eurotech so that this additional board must be considered when integrating into systems requiring a sound card.

While testing basic functionality of the sound card, it was discovered that playback was initially broken (it would play for a short duration, then repeat the time series indefinitely). After investigating various electronic mailing lists, it was found that this problem is caused by the *snd_hda_intel* driver incorrectly detecting the codec version. This problem was corrected by passing parameters to the sound driver when it is loaded. The two parameters are *enable_msi=0* and *single_cmd=1*, and were added to the file */etc/modprobe.d/alsa.conf* so they are passed when the module is automatically loaded.

With speakers connected, the audio codec produced audible pops both when the computer was powered on, as well as when the audio drivers were initialized during the boot process. This could be important in systems such as the Active LAND Buoy as it may overload the amplifier in certain situations.

3.6 USB

Initially, the EHCI (USB-2.0) driver was failing to load at boot time with an obscure error code. The failure was a result of a bug in the driver included with the 2.6.36 Linux kernel. Upgrading to version 2.6.37 fixed the issue.

3.7 Ethernet

It was discovered that using the software package *netplug* to manage the Ethernet port of the ISIS SBC could sometimes fail to detect the cable being plugged in. A bug was filed in Gentoo's bug tracking system (#354525). The software packages *ifplugd* was used to achieve the same functionality without problem.

3.8 Pulse per second (PPS) driver

The PPS signal generated by GPS devices is used for accurate playback timing on the existing Active LAND Buoys. On the Viper, this was implemented by wiring the PPS signal to an interrupt capable pin and using a kernel driver to detect the pulse and trigger audio playback.

The ISIS documentation states that the Fastrax GPS unit's PPS signal is connected internally to an interrupt capable pin. However, the manual fails to state which device/pin it is connected to.

After querying, Eurotech provided updated documentation showing it is connected to a GPIO pin of the SCH 311x SuperIO chip. No software or example code was provided to integrate with this, but it is believed that porting the existing PPS driver to work on the ISIS SBC would require minimal effort.

3.9 Serial ports

The ISIS SBC has two user accessible serial ports. One is used for the touch-screen interface in the development kit but can be re-assigned if not using the touch-screen. A third internal serial port is hard-wired to the text (NMEA) interface of the on-board GPS unit. After investigation (and requesting an updated manual from Eurotech), in was discovered that a fourth serial port is also hard-wired to the binary SiRF interface of the GPS.

The LAND buoys have the text port of the GPS wired to two serial ports, one for NTPD to use for time synchronization and the other for logging. This could be achieved in one of two ways on the ISIS SBC:

- Use the *gpsd* library (free, under BSD license) to share the NMEA messages between NTPD and logging systems.
- Use the NMEA serial port for either NTPD or logging, and use the SiRF binary port for the other. This may prove to be more difficult (it is not known if NTPD can use the binary protocol at this time), but could reduce latency in the messages used by NTPD to accurately set the system time.

It should also be noted that the manual's serial documentation is incorrect in places, sometimes stating that COM2 is connected to the GPS while it is actually COM3.

3.10 Data acquisition

A core component in the gliders as well as the Passive LAND Buoy system is a PC/104 add-on card for high speed data acquisition (AIO16E from Acces I/O). Integrating this card with the Viper had proven difficult as there were both electrical as well as software issues to resolve. Akoostix believes that porting the AIO16 driver software to the ISIS SBC would not be difficult, but unknown hardware issues could crop up as they did with the Viper.

As with the Viper, DMA is not supported on PC/104 bus. This is not required by the current configuration, but should be noted for future directions.

Another option for data acquisition might be to try an Acces I/O USB product. Several products are available (one which is very similar in features to the current PC/104 board). Furthermore, they offer a supported Linux USB driver which would reduce the risk in driver development.

3.11 Boot time

Boot and shutdown time is important in systems where the Acoustic Subsystem (see Annex B.4) is only required part of the time. The ISIS operating system boots in approximately 10 seconds

(and shuts down in about the same time). This boot time was obtained after paring down the kernel and system services, and using Gentoo features such as *openrc* (a new system services loader) as well as parallel service start up. However, the BIOS requires almost 20 seconds before the operating system even starts.

3.12 Power measurements

The power consumption of ISIS SBC was measured by stressing various system components to simulate heavy disk and processing load. The results are summarized in Table 1, while some comparative data that was gathered using the Passive LAND buoy is provided in *Table 2*. Tests were made using the standard configuration, described in Section 3.2, though in some cases other configurations were also tested (see table notes).

For these tests, the development kit monitor and touch-screen interface were disconnected, as they would not be used in an underwater system. All measurements were performed with a bench power supply set to 5.01 volts.

The Passive LAND Buoy was also measured for comparison purposes (*Table 2*). The first two measurements were taken with a supply voltage of 12.2 volts and include all system components, while the rest were measured with a 5-volt supply and include only the Viper board.

While the ISIS SBC requires substantially more power (about 7 to 9 Watts, versus 2 Watts for the Viper), it is a much more capable board. First, it must be considered that when used in computationally intensive manner, the ISIS SBC would spend much less time powered up than the Viper, as it can process more data at real time. Furthermore, the ISIS SBC has more potential to be used in situations where the Viper simply could not (e.g. sustained real-time processing of high bandwidth data).

Table 1: Power measurements of ISIS SBC.

Test	Current (A)	Power (W)	Notes
Booting	1.78 maximum 1.43 average	8.92 maximum 7.16 average	
Idle	1.37	6.86	Ubuntu was loaded for a second test and revealed a slightly lower (1.30A) idle current. It is assumed this can be achieved in Gentoo by configuring the proper CPU power saving driver.
One thread at 100% processor load	1.55	7.76	
Two threads at 100% processor load	1.6	8.02	
Writing a 200 MB file to the Intel Flash disk	1.72 maximum 1.4 average	8.62 maximum 7.01 average	
Writing 200 MB file + Two threads at 100% processor load	1.76 maximum 1.66 average	8.82 maximum 8.32 average	
Idle with Silicon devices disk installed	1.39	6.96	This measurement includes the Intel Flash as well since it is not removable.
Write 200 MB file to Silicon Devices disk installed	1.66 maximum 1.48 average	8.32 7.41	This measurement includes the Intel Flash as well since it is not removable.

Table 2 : Power measurements of passive LAND buoy components.

Test	Current (A)	Power (W)	Notes
Idle	0.40	4.88	This measurement includes the entire system (AIO16, filterboard, GPS, Freewave).
Sampling and processing	0.55	6.71	This measurement includes the entire system (AIO16, filterboard, GPS, Freewave).
Viper only	0.4 @ 5 volt	2.0	Only the Viper, negligible difference between idle and processing.

3.13 Disk throughput

The Viper hard disks have a limitation of 1 Megabyte per second throughput when using the Compact Flash interface. This is ample for the current systems but future systems requiring more data acquisition channels and/or higher sampling rates might exceed it. For example five-channel sampling at 100 kHz and two bytes per sample would require 1 MB/s alone leaving no room for writing processed data.

The raw disk I/O bandwidth of the ISIS SBC was tested using the command *dd* to measure the time required to write a large file (200 MB) to disk. The following speeds were measured:

- Intel on-board SSD flash, 20 MB/s.
- Silicon Devices 8 GB SSD, 5 MB/s.
- Secure Digital (SD) card, 60 MB/s. This test is inconclusive as it was suspected that the Secure Digital driver was buffering most disk writes to system RAM. This feature should be turned off and re-tested.

These speeds are much greater than the Viper's disk throughput and recent developments in the Linux kernel are focused on Solid State Devices (SSD Trim, batch discard), hopefully increasing this performance even more in future releases.

3.14 Processing performance versus Viper

A transient processing benchmark was performed using the SPPACS application *sp_sentinel* and an appropriate synthetic test data and target file (q302 chirp target file with synthetic q302 whale test data). The test data is sampled at 48 kHz and is 5 minutes 10 seconds in length.

The file required 3.5 minutes to process to completion using the Viper with fixed point processing. Although an unfair comparison, the floating point implementation (using FFTW version 3.1) was also tested on the Viper and required 20 minutes to complete.

The same test on the ISIS SBC using floating point math (FFTW version 3.1) processed in less than 4 seconds.

3.15 Summary and risks

The ISIS SBC is more than capable of replacing the Viper in the current (and hopefully future) implementations. It requires substantially more power than the Viper, but this may be offset by reduced on-time, especially if data can be cached at lower power for high-speed processing.

The notable drawbacks of the ISIS SBC that were discovered during this task are:

- The board's BIOS requires twice as long to boot as the operating system. This is a limiting factor on systems like the glider that require the board to boot and shutdown periodically. There is nothing that can be done in software to change this, though a low power suspend or standby state, if available, may reduce overall power-consumption (see below).

- Suspend to RAM does not work and is not supported at all. This is disappointing as it could be used to save power when not processing, while providing almost instant-on functionality.
- Eurotech's level of user support during this contract was not encouraging. Simple questions required multi-day turnaround as the support team is in USA and the technical team is in England. Furthermore their manuals, example code and drivers contain obvious errors, omissions and FIXMEs. This is common for specialized hardware, but disappointing.

The benefits of this board are:

- The Atom processor outperforms the Viper's processor in a typical signal processing task by a factor of 50.
- There are excellent options for peripheral devices (IDE, Secure Digital, ample GPIO and serial ports).
- Most hardware is supported by the Linux kernel, reducing the number of low-quality Eurotech drivers that the system must rely on.
- The Atom is a very well supported architecture, which does not require specialized variants of the Linux kernel, or cross-compiling.

Most of the benefits are not related to Eurotech's implementation, but the Intel Atom architecture itself. If the missing suspend-to-RAM functionality and/or long BIOS boot time are considered serious, Akoostix suggests that DRDC investigate another Atom-based board manufacturer. If these issues are acceptable, the ISIS SBC is a suitable candidate for replacing the Viper.

3.15.1 Future work

The remaining tasks to get a working replacement are:

- Investigate an analog input solution. Either re-use the existing AIO16 PC/104 card or investigate another solution (such as a USB device).
- Implement a solution (software and electrical) using the GPIO pins to control the filter-board components.
- Port the PPS trigger driver to use the GPIO pin on the ISIS SBC.
- Upgrade the build system and installation instructions to include the new system.
- Document the new development process since it is no longer based on the Viper development environment.
- Perform integration testing on target platforms (Slocum Glider, LAND buoys, etc.).

4 STAR training and user support

Akoostix was required to initiate formal STAR training, starting with two formal training sessions and one-on-one user support. This report section documents all of the work in order to provide the information in one coherent record. Effort, for contractual and billing purposes, was tracked separately.

After some analysis, the team decided to improve the delivery of formal training, over what was proposed. The course material was divided into two formal training sessions. The focus of the first session was an introduction to STAR, data formatting, preliminary processing, and manual analysis. The focus of the second session was detailed, automated, and custom analysis. Each session lasted approximately 4 hours, and was followed by a short hands-on session.

The course outline and lecture notes were delivered to DRDC as Microsoft PowerPoint documents. A Microsoft Word document was also created, based on the original STAR user manual [1], which contains background information on the STAR analysis process and the general purpose of STAR. Finally, a simple data set was created and copies of this data were provided to the STAR trainees, so that they would have data and examples to work through during the training.

A third group session was held, where users could ask questions and work with STAR trying to complete tasks of their choosing. A general discussion related to how to support STAR users and improve the utility of STAR was also initiated during the session. The recommendations and outcomes resulting from that discussion are provided in Section 4.1. Finally, users were given an introduction to OPD and had the opportunity to try out features in the newest release. All were very pleased with the interface and some commented that they don't understand why Prodat is still used by some at DRDC now that OPD is available. It appears that some DRDC staff are not aware of OPD and STAR.

Each trainee was provided with approximately four hours of one-on-one training. Effort for this training included preparation time, as each trainee had specific requirements, which required assembly of data and examples prior to each session. These sessions were very successful and helped to build practical knowledge and experience.

An online survey was generated and survey requests were sent to all four trainees. The results of the survey were not available at the time of delivery, but will be delivered to DRDC, once all surveys are completed.

4.1 Feedback on STAR user support

The final training session included a discussion on how to improve STAR and provide better support. The following issues were noted with recommendation and/or outcomes provided as sub-bullets:

- In some cases STAR produces warning messages that may not be important or relevant to the user. They can be confusing to those that are not familiar with STAR.

- STAR should provide logging information that is more relevant to the target user. This has been logged as an issue in JIRA AKOT-177 and has been recommended for future work in Section 8.2.2.

- Running regression tests and interpreting the output is complicated and can confuse new users. The requirement is to verify that STAR is correctly installed and configured.
  - A script that verifies the proper installation of STAR by version was created and provided with STAR 6.6.8. Users can try to use the script and provide feedback, which will be considered during future releases.
- Users need access to information, which can be difficult and expensive to capture in documentation. Also, each user learns processes and techniques that other users may benefit from. There should be an efficient way for users to support each other and for them to share information.
  - It may be possible for DRDC to create a STAR user mailing list that others can subscribe to. Ideally, this would be a proper mailing list with web-based logs and searchable archives. Users could then post tips, questions, and answers to the mailing for others to answer or benefit from. Akoostix could be a member of the mailing list, but this may be a good will gesture versus a formal requirement. The requirement for Akoostix to support, or possibly monitor and correct, issues posted to the mailing list would need to be determined by DRDC management.
  - It may be possible for DRDC to create a wiki related to STAR that could contain a FAQ, tips, related tools, and software examples (though it may also benefit DRDC to develop a software repository to hold examples and reusable STAR user code). This might be hosted at DRDC by creating a category on one of the current wikis. Someone would have to be identified who could administer the wiki, while another individual may be required to moderate and verify content, though wiki are often verified and corrected by the user community. Again, Akoostix would be willing to provide informal or formal support for the wiki, depending on DRDC access restrictions and requirements.
- Many DRDC staff are not aware of the existence of, or capabilities of STAR and advanced point-and-click tools like OPD.
  - Akoostix could provide a seminar featuring STAR tools like OPD, SPPACS, and STAR-IDL, as some trainees suggested, but it may be more appropriate for someone at DRDC to lead the seminar. DRDC management may want to consider methods of raising awareness and interest in STAR at DRDC.

5 Support to tracking research

The "Support to Tracking Research" objective was to provide a capability and to perform data extraction from multistatic raw data to a form that can be used for tracking research. An automated feature extraction application was developed during the "Support to Tracking Research" task. The *tracking_support_app* application was designed to perform batch processing over a specified range of multistatic trial data had been formatted into the STAR trial structure. User instructions for this application have been developed and are included in Annex A. Also, a set of IDLDoc entries has been developed that documents the application's software interfaces. Refer to Section 7 for information on the STAR release documentation. The IDLDoc for this particular application is located at *app/tracking_support* folder.

The tracking support application provides the following capability:

- Perform automated feature extraction for a multistatic trial. Configurable options include:
  - Start and end time of the processing;
  - If available, the real target track can be extracted;
  - Subset of sensors for processing or all available sensors;
  - Waveform to use for the processing; and
  - Detection algorithm.
- Logging the processing results:
  - The configuration used to process the data;
  - The detection features grouped by ping; and
  - Echo logs for each detected feature (optional).

6 Software defect repair and enhancement

This section describes the various defect repairs and enhancements that were implemented during this task of the project. The completion of these tasks has helped to:

- Increase SPLIB's processing performance as described in Section 6.1;
- Update the software build system's cross-platform support as described in Sections 6.2 and 6.3; and
- Increase OPD's analysis capabilities with the addition of the Harmonic, Banding and Doppler cursors as described in Section 6.4.

6.1 FFTW3

As part of this work, the SPLIB library was upgraded to use version 3.2.x of the FFTW library. Historically version 2.1.x was used, but after an investigation Akoostix worked on in the previous year, it was realized that a substantial improvement in performance could be gained by upgrading the library. Also as a result of testing, it was determined that FFTW's unpacked format Application Programming Interface (API) produced faster results than the half-complex packed format. This option was not available in version 2 of the library.

This porting involved the following tasks:

- Upgrading the SPLIB modules *apply_filter*, *c2r_spectra*, *r2c_spectra*, *ctime2cspectra*, *cspectra2ctime* to use new FFTW library API with unpacked format.
- Removing SPPACS applications *sp_beamform_old*, *sp_rds2dat*, *sp_correlate_old*, and *sp_spectra* since they were deprecated and depended on FFTW version 2. Also, some deprecated support code in the *ac_utils* library and SPPACS' *librds* library was removed, as the functionality currently exists in SPLIB.

6.2 OSX upgrades

In order to improve support for the OSX platform, and support the latest 10.6 release, the following tasks were performed:

- Fixed the Qwt and KISS libraries to properly use the *install_name* field in the installed binaries to avoid the need of defining a DYLD_LIBRARY_PATH environment path. This will ease installation, and allows for multiple installed versions of a library/application on the same system.
- The *install_name* field of Boost's *program_options* library could not be easily adapted due to the nature of the Boost build system. To preserve budget, Boost was modified to install and link statically, eliminating the need of the *install_name* which only applies to dynamic libraries. This method was chosen since the *program_options* library will be phased out at a future time, and is not worth spending additional budget on.

- Upgraded to Qt 4.7.x as this fixes some build and run-time issues with the OSX 10.6 platform as well as 64-bit support.
- A python script was created to create OSX bundles for the applications OPD and ACDC. This script gathers required binaries and resets the *install_name* field in all libraries to create a self-contained bundle. It packages this bundle in a disk image (.DMG) without setting environment variables (see first bullet).

6.3 Windows upgrades

The following items were addressed in order to provide better support on the Windows platform, as well as preliminary porting for Windows 7 and 64-bit support:

- Upgraded the TRE regular expression library to build and work with Windows 7.
- Upgraded SQlite to version 3.6.23 and build from source on Windows versus distributing a binary DLL. This was required for preliminary 64-bit support.
- Upgraded to CMake 2.8.x as it is required for Windows 7 64-bit support.
- Builds can be made to work on 64 bit but not fully automated yet. Some ideas were added as inline comments in the system-build scripts.

6.4 OPD upgrades

Additional measurement cursors were added to OPD during this contract. The OPD user manual [3] was updated with user instructions for each cursor type. They include:

- The harmonic cursor was developed and integrated with the *spdisplay* library using the existing cursor framework. Features that were developed for this cursor include:
  - Multiple cursor creation. More than one harmonic cursor can be created and used in OPD.
  - Cursor interaction. The user is able to select and interact with more than one harmonic cursor in order to move the cursor to a new time or adjust the harmonic spacing.
  - Comprehensive cursor readouts including time, frequency, harmonic ratio, spacing, and comb numbers.
  - Look-and-feel options including changing line styles, thickness and colour. A list of various symbols can be selected by the operator for the comb positions.
  - A manual entry option for adjusting the harmonic spacing.
- A banding cursor was developed and integrated in OPD. This was a side-effort that was not billed to this call-up. The banding cursor is similar to the harmonic cursor and provides the same options as the harmonic cursor described above. In addition, the banding cursor allows the root comb to be reposition to a value other than 0 Hz.

- A Doppler cursor was developed and integrated in OPD. This was also a side-effort that was not billed to this call-up. The Doppler cursor allows the operator to measure the radial velocity of a target. It provides the following features:
  - Modes for both Active and Passive sonar, and
  - Manual adjustment of the centre frequency to match the source's centre frequency.

7 Configuration management

The final software deliverable for this contract was provided on the STAR release CD - version 6.6.8. The CD was generated and delivered on March 31st, 2011. This software release contains the original project deliverables. This section of the document describes the content of that CD.

7.1 STAR branch and release information

Each logical grouping of software modules has been independently versioned on the CD. The current STAR release version is 6.6.8 and contains the following:

- OPD 2.4,
- ACDC 2.1.6,
- SPPACS 1.1.9, and
- Analysis Tools 6.11 (STAR-IDL).

The 6.6.8 release CD was generated in conjunction with other DISO call-ups. Installation instructions are located in the root directory on the release CD.

7.1.1 STAR software documentation

Some manuals, API documentation, and other design documents are provided with the 6.6.8 software release CD. In a standard STAR distribution they can be found by opening *the /usr/local/atools/star-6.6.8/documentation.html* file in a standard web browser. This page contains links to several sets of documentation including revision history, the IDLDoc for the analysis tools (STAR-IDL), the manual for the analysis tools, and DOxygen generated documents for OPD, ACDC and SPPACs.

PDF versions of user manuals for more mature software can be found in the *manuals* directory on the root of the CD.

7.2 Issue summary

The issue summary in Table 4 summarizes the blocker and critical issues that remain open, but only for software relevant to this contract. None of these issues had any effect on the success of this contract. Resolution of these issues may increase efficiency during the execution of future call-ups or contracts.

Table 3 shows the current state of known defects for all of the software release candidates listed in Section 7.1 as of March 31st, 2011.

The distribution of issues is indicative of the maturity of the software. Though maturing, much of this software is composed of various evolutions of an iterative design, especially command line SPPACS applications and STAR-IDL components. This software would benefit from general design improvements and refactoring. There are two active blocker issues and several critical issues. These are obscure or infrequent bugs that were discovered during current work, but budget or schedule has been insufficient to address them yet. Critical issues are issues that still allow the operator to perform their function but could cause erroneous results or loss of data in those instances. These bugs should be fixed in the near future. Only Blocker issues do not have a work-around and may need to be addressed before a contract can be completed successfully.

Table 4 summarizes the blocker and critical issues that remain open, but only for software relevant to this contract. None of these issues had any effect on the success of this contract. Resolution of these issues may increase efficiency during the execution of future call-ups or contracts.

Table 3: Issue Summary (Severity versus Status) for all software on STAR release 6.6.8

	Open	**Reopened**	**Resolved**	**Closed**
Blocker	2	0	7	24
Critical	14	1	5	58
Major	109	5	39	160
Minor	32	1	3	17
Trivial	7	0	0	3
Undecided	2	0	0	2

Table 4: Known Blocker and Critical issues for STAR release 6.6.8

Module	JIRA Issue ID	Summary	Description
STAR-IDL (Tracking Support)	AKOT-164	Crash on invalid Time - Tracking Feature Extraction	A crash occurs when trying to run on the CFMETR data. This is probably just a case of not checking for an invalid response and assuming that you got a time.
STAR-IDL (Tracking Support)	AKOT-163	Failure with Ping to Source Mapping -- Tracking Feature Extraction	Mapping a specific ping to a source when trying to run the tracking feature extraction application fails. I suspect that the error is similar to the issue affecting the main clutter application. A temporary workaround involves removing pings from the ping.txt in the NAD that are not being processed.
STAR++	STPP-3	Potential error in STAR IDL hyperbola fixing	Starting from line 1102 of *star_cross_data_support* in the *create_hyperbola_hyperbola_contact* function there may be a problem. The variables *xbh1*, *xbh2*, etc are not used but instead a set of vectors *xbh*, *ybh*, *BB*, and *SS* are created. These vectors contain all three buoys in one shot instead of just the two from one hyperbola. These variables are what is used in *getbc* and then for *bayesprobs*. The original IDL code from Joe Maksym doesn't make this simplification (idemo.pro about lines 23813 and 25263). We need to check and make sure that this is valid or the posterior probabilities will be incorrect.
OPD	OPDY-245	OPD crash if EADAQ changes state during processing but the user hasn't stopped OPD processing	OPD will hang if the user tries to stop processing (reconfigure in this case), if EADAQ was reset while you were processing. (When EADAQ resets the display freezes as one might expect, but the user is unable to stop OPD.)
OPD	OPDY-244	OPD Crash trying to stop processing after EADAQ stops	OPD will crash if you try to process EADAQ when EADAQ is not recording. It is interesting to note that once the users starts a data stream the user can stop EADAQ recording and OPD will continue to process the data.

Module	JIRA Issue ID	Summary	Description
			In this case you get a filled in header, but it appears that no data runs.

Module	JIRA Issue ID	Summary	Description
OPD	OPDY-174	Bad combination of zero padding/overlap in the processing parameters dialog can cause a hang	If the user chooses something like 8192 FFT size, 8191 zero pads, and anything but a 0% overlap, you end up with (8192-8191)*(1-overlap) which for any overlap but 0% will cause an integer round to 0. A typical user would not configure it this way.
OPD	OPDY-160	Crash with extreme processing parameters	A crash occurs using the following processing parameters: File: meg//scratch/OPD_test_data/bof_99.dat Channels: 1-16 Default settings except: 1M pt FFT 99% overlap
OPD	OPDY-77	OPD hangs when validating data sources	OPD tries to verify the data stream by reading as much as required to determine the format and sensors. It stalls the system until it receives this information. An array server can be accepting connections but not sending data. This is particularly annoying since OPD connects to Northern Watch Array Server automatically as soon as a valid IP and port are given. OPD saves the IP and port to save from retyping them all the time. So if you stop feeding data on one port (but still allow connections) and start it on another, then restart OPD, it tries to connect to the previously saved port and hangs. Current Workaround: 1. Port being saved is in the registry settings. It can be changed manually before resetting OPD or 2. You can start the array server on the previously saved port. This will unclog it.
SPPACS	AKSP-91	*sp_correlate* output time not correct	The output file time for *sp_correlate* is being set to 31-DEC-69 23:59:59 when the input time is actually 01-JAN-80 03:00:00 they should be the same.
SPPACS	AKSP-72	*sp_median_nrmf* is referencing a	*sp_median_nrmf* causes a crash (in OPD) by dereferencing a null pointer down in its lower

Module	JIRA Issue ID	Summary	Description
		null pointer occasionally in win32	processing layers. This seems to only happen on Win32, but may just be being hidden in Unix environments (windows has a history of being more strict, especially in debug mode).
STAR-IDL (ITAC)	AKOT-153	Animation with selected tracks - bad behaviour	Selecting tracks in ITAC then started animation produces odd track display errors. Once animation is turned off it behaves normally again.

Module	JIRA Issue ID	Summary	Description
STAR++	AKOT-150	Crash / Hang in STAR++	I got a hang when using STAR++ to investigate AKOT-149 on OSX (tracking3) and a crash on TMAST02 data (Linux and likely slightly different version). I was running the Monte-Carlo simulation for the runs on OSX and had them off once and on another time for the TMAST02 runs (crashed both ways). Generally settings were: 2 detections, 10% threshold 2 Clusters, 3km radius El-El, El-Brg, El-Hy crossings Monte-Carlo simulation with all error types and cluster results Search time adjustment of 0.5 seconds On TMAST 02 I had designated on channels 2,3,4,5 For Tracking3 I set a threshold of 550.
STAR-IDL (Trial Reconstruction)	AKOT-111	Spits out an array when you exceed the number of colors in the color array	Bug occurred in the trial reconstruction tool (stack trace is provided in ticket comments).
STAR-IDL (Tactical Plot)	AKOT-107	Ownership of overlays and contained data	There is a general problem with ownership of overlays and contained data by the tactical plot. The tactical plot can be closed and reopened several time during an application's lifetime, so if it destroys all data that it contains it will be lost to the application and cannot be used on subsequent instantiation of the tactical plot, or usage by other modules (i.e. tracker). It may be reasonable just to own the overlay itself and not the contained data, but then we need to assign ownership of the contained data to someone. Another option may be to tell the tactical plot when it owns an overlay. Right now image overlays are owned by the tactical database, because it would have been more work to create a data container in the database and then force the creation of an overlay after the fact. This might

Module	JIRA Issue ID	Summary	Description
			need to change depending on how this issue is resolved. If two objects are created (image container and image overlay) then we need to be careful how data is passed between them to avoid expensive data copies for big images.
STAR-IDL (Trial Reconstruction)	AKOT-97	Fail on trial recon - Signal Excess versus Total Distance	On first selection of the Signal Excess versus Total Distance option in trial reconstruction (q320/tracking3), a null string error is produced (error captured in comments of ticket).

Module	JIRA Issue ID	Summary	Description
ACDC	ACDC-154	ACDC (2.1 branch) crash on OSX upon startup	Using *spdisplay* 2.7.1, ACDC crashes on OSX. It seems to be running in circles given the stack trace upon crash (provided in ticket). Reverting back to 2.6.4 does not crash. This effect is not seen on Linux (with same Qt and 3rd party libraries).

8 Recommendations for future work

This section is used to capture ideas and suggestions for maintaining or improving the software used during not only this call-up but for other call-ups under the current DISO. It is in addition to addressing the known blocker and critical issues identified in Section 7.2. Please refer to the CFMETR 2011-1 call-up final report for recommendations on suggested improvements to the trial analysis process and related data analysis software. The recommendations contained in this section are divided by software application.

Each section contains a summary table containing the estimated effort to implement each of the recommendations. These estimates are rough order-of-magnitude (ROM) estimates generated after a preliminary review of the requirements and therefore could have significant variance. Estimates are broken down into the following categories:

- Small is between one (1) and five (5) days of effort,
- Medium is less than two (2) weeks of effort, and
- Large is more than two (2) weeks of effort.

8.1 SPPACS

The SPPACS tools continue to be an integral part of trial data analysis for DRDC. These tools continue to grow in functionality and importance. It is important that the current state of the software is maintained and that designs are upgraded as necessary to meet the growing size and complexity of the various applications. As well, there is currently software being developed and integrated with IMPACT that would prove to be valuable as SPPACS tools.

Table 5 is a summary of selected issues being tracked in JIRA for future upgrades to the SPPACS software. These issues are further explained in the subsections.

Table 5: SPPACS Improvements

JIRA ID	Depends On JIRA ID	Summary	Estimated Effort
AKSP-57	None	Update SPPACs framework	Medium
AKSP-85	None	Add DEMUX application	Medium
AKSP-86	None	Add Arctan for ETI data	Medium
AKSP-93	None	*sp_main_blast* should log its settings in the file.	Small
AKSP-94	None	Integration of Main Blast Detector library (IMPACT) with *sp_main_blast*	Medium
AKSP-95	None	Create a *sp_make_clutter_data* using Clutter Data Generator library (IMPACT)	Medium

8.1.1 Update SPPACS framework

SPPACs Framework could use some maintenance. It is important to continue to normalize and simplify designs and code in order to keep it maintainable and flexible. Below is a list of suggested enhancements for continued maintenance and improvement of the SPPACS framework:

- Ensure that the framework provides the option to use the header files if specified for a specific input file.
- Provide a better class hierarchy for implementations:
  - One layer would handle parsing in command-line parameters, defining input and output data sources, and outputting help and version information. This layer may also add simple options for WAV, byte-swapped, channel subsets, time subsets, etc.
  - An internal layer would handle running the processing and be reusable beyond SPPACS, similar to what is done for *sp_transient_processing*.

8.1.2 Add DIFAR demux application

The current DIFAR demultiplexor (demux) is run as a stand-alone application that cannot be connected to an SPPACS stream, as it will not accept and produce data from pipes (*stdin*, *stdout*). It is also not modularized so that it could be integrated into applications such as OPD to enable DIFAR processing. Finally, the demux application requires filters be available from the current directory (i.e. directory from which the application is run). This limitation reduces the potential for reuse and batch processing efficiency.

Akoostix recommends that the demultiplexor be ported into an SPLIB module, but contained in its own library to simplify control over the source code. This integration effort would allow for either static definition of filters, or an environmental variable that defines the location of the

filters on the file system. This change would also allow the demultiplexor to be used (e.g. licensed) as an object code library with defined interfaces, which would increase its value to other users. Once this work is completed it would be straightforward to integrate the module into an application compatible with SPPACS, and to use it in other applications (e.g. GUI-based demultiplexor, OPD, etc.).

8.1.3 Add Arctan for ETI data

An arctan bearing processing capability should be created for ETI data (in SPLIB) and then integrated into an SPPACS application. This would allow the direct performance comparison of these two methods of bearing estimation. (MVASP currently uses arctan bearing estimation for ETI data.)

The processing should be prototyped in STAR-IDL before developing the SPLIB module to ensure that the algorithm is correct. Methods for storing the data should also be considered. The SPPACS application *sp_arctan* currently packs the intensity and bearing data into one fixed-point 32-bit value. This approach avoids the requirement to synchronize two data files (one for bearing and another for intensity), and is currently used in other STAR-IDL applications. Other options could include an independent bearing file with one bearing for each time sample, and multi-beam intensity data. This would allow users to benefit from the beamforming gain on the displays, while having access to the pre-computed bearing estimate. A new design should also consider how other bearing estimation methods might be integrated for future analysis and comparison.

8.1.4 *sp_main_blast* should log its settings in the file

It is important to know what settings were used for *sp_main_blast*. These should be logged as a comment at the top of the output along with the system time and version used to do the processing. This approach is becoming a standard entry for logs in STAR-IDL and should become common to STAR.

8.1.5 Integration of main-blast detector service with *sp_main_blast*

The library modules developed for IMPACT's main-blast detection could be integrated with *sp_main_blast* to improve its processing. This integration would include the simultaneous processing of multiple data streams that picks the "best" stream for each main blast. Additional policies that use positional data could be added as IMPACT is improved.

Merging the main-blast detector capability with SPPACS would increase the DRDC user base for this service. The library and its components would be used and tested in more varied scenarios. A properly configuration managed, common code base would ensure that feature enhancements and bug fixes would benefit both IMPACT and SPPACS.

8.1.6 Create a *sp_make_clutter_data* application using the clutter-data generator service

The library modules developed for IMPACT's clutter mitigation processing could be used to create a *sp_make_clutter_data* SPPACS application. This application would use the main-blast logs to create a *.pwr* formatted clutter file and associated log file. This application would be designed to be compatible with STAR-IDL clutter analysis, but could also include prior normalization and decimation processing. STAR-IDL (or an OPD-like application) could then be enhanced use multiple resolutions and allow for rapid zoom in/out during processing.

Merging the clutter data generator capability with SPPACS would increase the DRDC user base for this service. The library and its components would be used and tested in more varied scenarios. A properly configuration managed, common code base would ensure that feature enhancements and bug fixes would benefit both IMPACT and SPPACS.

8.2 STAR-IDL

The STAR-IDL tools continue to be an integral part of trial planning and trial data analysis for DRDC. These tools continue to grow in functionality and importance. It is important that the current state of the software is maintained and that designs are upgraded as necessary to meet the growing size and complexity of the various applications. This is particularly important for how tactical data is managed, stored and queried since this has a significant impact on the performance of various STAR-IDL applications.

Table 6 is a summary of selected issues being tracked in JIRA for future upgrades to the STAR-IDL software. These issues are further explained in the subsections.

Table 6: STAR-IDL Improvements

JIRA ID	Depends On JIRA ID	Summary	Estimated Effort
AKOT-165	None	Change the method for grouping main blasts	Medium
AKOT-177	None	Rationalize the log system for different use cases including scientific, developer, and operator	Medium
AKOT-178	AKOT-181	Introduce an SQLite database	Large
AKOT-179	None	Change representation of time	Small - Medium
AKOT-180	None	Upgrade remaining track sources to track container design.	Small
AKOT-181	AKOT-180	Finish upgrading the tactical database to the latest container-based design.	Large

8.2.1 Change the method for grouping main blasts

Several places throughout STAR-IDL need to know which main blasts are part of the same ping and therefore end up using the *cluster_events* method in order to group them. Some of these areas include:

- Smart Automation: localization is performed on a main blast group at a time.
- Animation in Tactical Plots: clusters main blasts to determine an animation timeline.
- Tracking Support: groups main blasts in order to log analysis on a per-ping basis

The problem is that each of these applications may cluster main blasts differently rather than this should be determined consistently across applications. This is a potential source of error and increases the amount of code to maintain. There are two possible solutions:

1. When main blasts are read into the tactical database they are clustered. This would occur once. Queries to the tactical database could be updated to retrieve a main blast group

2. When main blasts are written to a log they are assigned a main blasts group UUID (Universally Unique Identifier). This would require a change in the message format.

8.2.2 Rationalize log system

The different STAR-IDL applications are logging various errors, warnings and information without consideration of the user of the system. This approach has the potential to increase confusion between the various users as to the importance of the information being logged. A

logging system is most effective (and has less a chance of being ignored) when the information is concise and relevant.

The logging information would be more useful if it was targeted to the individual needs of the user depending on their specific use case:

- Scientific: These users need to know about information that could invalidate their analysis or that a particular algorithm is not processing as expected.
- Developer: These users need to know about logic errors and be given enough information to debug the errors efficiently.
- Operator: These users need information on potential errors when using tools or configuring displays/processing.

The logging system could be updated such that the output for the different users can be configured using modes in the system that can be individually toggled on and off. (i.e. Scientific + Operator could be viewed but Developer is suppressed). How log information for the different use cases is presented should be an extensible/adaptable part of the system. For example, warnings could show up in red for a Developer but yellow for an Operator.

8.2.3 Introduce SQLite database

Integration with an SQLite database would reduce data-management code, simplify data-integrity checks and provide mechanism for cross-tool support. The non-acoustic information used by the STAR-IDL applications is a good candidate for a low-overhead database such as SQLite.

8.2.4 Change representation of time

The current time structure in STAR does not allow for simple comparison when calculating tracks between start and end times. The performance of the STAR software may benefit from using a built-in or simple type to represent time which would allow sort and comparison operations to be optimized. This is especially important when dealing with arrays of time structures.

8.2.5 Upgrade remaining track sources to track container

The recorder, target, and wreck main track files are still being parsed into the trial structure. The latest tactical database design uses various independent containers including the track container to manage data. Previous effort was spent integrating the main tracks and source tracks into the track container. Once the recorder, target and wreck tracks are integrated, it will complete the track container integration.

8.2.6 Finish upgrading the tactical database to the latest container based design

Further work is required to completely deprecate the trial structure as well as the non-OO based tactical database. Only a portion of the container-based design has been implemented. This

design has increased the tactical database efficiency for areas that have been integrated such as the track container and the echo container. The current trial structure is set to a static maximum size which can take a significant amount of memory. The containers use much less memory since they grow dynamically as data is imported.

8.3 OPD

The OPD program is a successful processing and display application, which provides a large number of tools for data processing and analysis. The current version of OPD provides the operator with the means to configure processing parameters for a single data source and view the results in a number of graphical plots. However the processing parameters cannot be altered after processing has commenced, and the processing itself is limited to a single stream on a single data source at a time. It is also limited to viewing a single channel of processed data at a time.

To reach another level of utility, OPD will need to be enhanced in a number of ways to provide a more productive and dynamic analysis environment. For example, providing the operator with the means to control some processing parameters in real-time during processing, exporting processed and raw data, and showing multiple channels of processed data simultaneously. This level of enhancements will require a combination of back-end and front-end changes. (Here back-end changes are upgrades to the design that may not provide visible improvement, but that add value by enabling more complex features to be implemented in the future, or by increasing reliability.) Table 7 provides a selected list of issues from JIRA that relate to recommended OPD upgrades. For each issue a summary is provided, including a list of dependencies (items that must also be addressed to realize the required benefit). The estimated effort for these tasks is broken down into ROM estimates (small, medium, and large), as described in the introduction to this major section

Table 7: OPD Recommendations

JIRA ID	Depends On JIRA ID	Summary	Estimated Effort
OPDY-17	SPLB-1, WIDG-1	View current processing without stopping	Small
SPLB-1	None	Enhanced processing parameter validation	Medium
WIDG-1	SPLB-1	Modularize processing configuration widget	Medium
SPDY-13	None	Create a plot item database	Large
SPDY-12	DMGR-1	Upgrade displays to support on-the-fly processing parameter changes	Small
OPDY-46	None	Display multiple processed channels	Large
SPLB-2	None	Upgrades to support multiple, concurrent clients of a single data source.	Small
OPDY-49	SPLB2	Raw data output	Small
OPDY-50	None	Zoom capability – add heterodyning	Medium
DMGR-1	None	Upgrade memory and data management capabilities.	Medium
DMGR-2	None	Enhance data manager client notification mechanism.	Small
OPDY-81	DMGR-1	Improved memory resource configuration	Small
OPDY-83	DMGR1, SPLB1 DMGR2	Output processed data to file	Small
OPDY-120	None	Time-compress processed data	Medium - Large
OPDY-122	ANN2	Build on annotation tags and quick comment text on the fly	Small
ANNN-2	None	Upgrade the annotation database.	Small
OPDY-123	OPDY50	Perform correlation processing	Small
OPDY-217	None	Add use of separate header files to the WAV data source to get time stamps	Small
OPDY-220	None	ETI formation at the output of the normalizer	Small
OPDY-230	None	Despeckle sonograms (most useful before compression)	Medium
OPDY-238	None	Select optional filters for the time-series display	Medium
OPDY-253	DMGR-1	General Time series display improvements	Large
OPDY-254	None	Autosave annotations to file periodically	Small
OPDY-255	None	Create icons for toolbar buttons	Small - Medium
OPDY-257	None	ICI Cursor	Medium
OPDY-258	None	CPA Cursor	Medium
OPDY-259	None	Target Specific Cursors	Large
OPDY-260	None	Windows 7 Support	Small to Medium
OPDY-261	None	Multiple connections to a single data source	Large

8.3.1 View current processing without stopping

It would be useful to be able to view the current processing configuration without stopping the current processing run. The parameters that can be changed without requiring a processing restart (e.g., averaging time, ETI frequency range, ALI time, quantize values, etc.) should also be enabled and available for the operator to manipulate.

8.3.2 Enhanced processing parameter validation

The current processing parameter validation functionality resides within the OPD processing configuration dialog, and is designed to handle complex cross-references and validation of a large number of parameters. To further increase its usefulness, this functionality should be modularized and moved into the SPLIB library to make the extension and maintenance easier and to enable it to be functional without being embedded within a GUI component.

8.3.3 Modularize the processing configuration widget

The existing processing configuration dialog for OPD contains a large number of controls for various groups of processing parameters. These controls would be useful for other applications, and their extension and maintenance costs improved. These widgets would then be used to construct the OPD processing configuration dialog, and other user interface components for other applications using the same SPLIB processing components.

8.3.4 Plot item database

The current *spdisplay* library that is used to create the OPD plots, graphs, cursors, and annotations works well; however, each plot and associated plot item are separate entities with limited knowledge of each other. This flexibility is desirable for creating a framework for building displays with various layouts, but makes it more difficult to provide markers, cursors and other plot items that need interaction with and awareness of each other. The enhancement to the existing *spdisplay* library of a plot-item database would allow for the plots, markers, annotations, and cursors to be able to query and control a central repository of entities across all plots in an application, providing the means for more sophisticated interaction.

8.3.5 Upgrades to displays to support on-the-fly processing parameter changes

Currently, the *spdisplay* library gathers its plot information from the data manager, but it assumes that the processing parameters and associated metadata is the same for the entire processing duration. In order to support mid-stream changes to processing, the *spdisplay* library will need to be modified to retrieve a series of processing parameters and to be aware of which parameters apply to which times on a plot or graph. It is also desirable for the *spdisplay* library to visually demark changes in processing parameters for increased operator comprehension of the data.

8.3.6 Display multiple processed channels

OPD allows multiple channels of data to be processed, but only one channel can be displayed at a time. Users must page through channels and therefore cannot compare the data directly. Work has already been done to create a framework for multi-channel displays, but it hasn't been completed and integrated into the application. The multichannel displays should be created with user-selectable options to quickly switch between different layouts, potentially including:

- 1, 2, 4, or 8 panels;
- Vertical or horizontal alignment (i.e. all in one row or all in one column);
- Shift axis to only the left-most or bottom-most display, or turn them off;
- Selectively lock panels (time, frequency, or channel) to maintain synchronization as scroll bars are shifted or, or unlock to allow independent scrolling. For example, it might be useful to view different time-frequency ranges for the same channel then scroll through channels;
- Shift ALI and/or ETI data onto a single panel and color code;
- Add a button bar and/or short-cuts to allow users to intuitively change display options; and
- Record / recall complex display layouts for quick set-up.

8.3.7 Upgrades to support multiple, concurrent clients of a single data source.

The current SPLIB library allows for multiple clients associated with a single data source, but it does not allow for multiple clients that may be in different simultaneous threads of execution. In order to efficiently support higher-function features, it will be necessary to enhance the existing implementation of the data-source objects to allow for concurrent access.

8.3.8 Enhanced data manager client notifications

The OPD data manager currently allows for event-based notification for clients when new data is available, and for updates to existing data; however, it does not emit notifications when a data run has completed (it is currently implicit). This functionality will need to be amended for clients that control resources, such as file writers, so that they can be informed when to release their hold on those resources. Adding this functionality will require a few changes to the manner in which data manager resources are created and released.

8.3.9 Raw data output

The ability to save data from a live source would allow the user to capture any selected data on the user's computer. This would alleviate any accessibility problems or data-transfer issues (for instance during trials) and would provide the OPD operator instant access to the data should they wish to reprocess it from file. User interface controls could include a start/stop recording or a recording schedule that allows for recording data between specified intervals of time. Things to consider would be:

- Accessibility of the captured data from OPD and the ability to quickly replay the captured data with different processing parameters.
- The bandwidth capability of a networked source such as EADAQ.
- Outputting the captured data to WAV format.

8.3.10 Zoom capability – add heterodyning

The data zooming capability in OPD consists of setting frequency bands and using a high frequency resolution. This method doesn't reduce any processing, rather it just saves the specified band which is restricted at the output of the processing chain. There may be some benefit to use the heterodyning capabilities that exist in the processing to limit the computation. This may in certain circumstances reduce the amount of computation required, especially when performing such operations as beamforming.

8.3.11 Upgrade memory and data management capabilities

The OPD data manager has an internal circular-buffer implementation that helps to control some of the issues of memory consumption when running a processing stream for a long period of time. However, this solution is limited in its usefulness due to the budget and priority constraints thus far. There are two main issues that still need to be addressed with the data-management layer. They are:

- The inability to manage the memory usage across buffers as a whole. While a single processing run may only consume a certain amount of memory, there are no good constraints about how much overall memory across many processing runs can be used. It is still possible to run out of memory if too many processing runs are done in a single session.
- There is currently no ability to purge expired or unneeded data from the application to a persistent storage area in order to free up space. Therefore, capturing or using history data from long runs is difficult or impossible.

The recommended solution for these issues would be to extend the circular buffer concept to a overall memory-management and data management layer that can more deftly control the amount of memory being consumed by any aspect of the application, and that can store overflow data to a persistent place and then read it back in again when required. Of course, the necessity to have persistent storage might not be served in all uses of the application or the underlying layers, so this part would be optionally configurable.

8.3.12 Improved memory resource configuration

Building on the data-manager functionality from Section 8.3.11, OPD should provide the means for an operator to view the current memory usage and the current memory usage limits. The operator should also be allowed to change memory usage limits and free up data as necessary to perform processing.

Some operator and/or application configurable parameters would include:

- The overall amount of in-memory buffer storage allowed,
- The overall amount of capacity for specific circular buffers (processing runs),
- The overall amount of persistent storage to be used (if desired), and
- The ability to dump in-memory buffers to persistent store on exit.

While there are many trade-offs to be balanced, this solution would essentially provide the application with storage capacities that are only constrained by hard drive sizes and processor performance, rather than an arbitrary configuration within the operating-systems maximum allowed executable size.

8.3.13 Output processed data to file

Exporting processed data files would give other tools such as IDL or MATLAB access to the processed data from OPD. It also would allow tools such as ACDC to read the processed data and display it for further analysis. Eventually, OPD may provide a capability for reading in processed data as well saving the operator time to reprocess. This requirement would depend on the data management solution implemented in OPD.

8.3.14 Time-compress processed data

It would be useful to provide data-compression options to reduce the update rate and allow more history on a screen, while not compromising detection quality. These options would involve a variety of ORing operations described in [5].

Compression schemes would be created and tailored to specific transients such whale clicks and squeals [narrowband]. Ideally these options would provide a degree of zoom capability allowing the operator to select a region of time-compressed data and zoom through the different resolutions.

8.3.15 Build on tags and quick comments on the fly

It would be useful to be able to save comments made on-the-fly to the quick comments box for future use. It would also be useful to be able to update the tag-list live. The user could then save an updated version of the tags / annotations to the current files.

8.3.16 Upgrades to annotation database

The current implementation of the annotation database does not allow for the on-the-fly creation of new annotation quick messages and annotation tags. Being able to add these items while an application is running will improve the annotation feature's functionality by avoiding having to edit text files and reload or restart the application.

8.3.17 Correlation processing

Proper correlation processing and display (monitoring) of frequency modulated (FM) data is required to evaluate the effectiveness of various experiments. MVASP monitoring of this data proved very useful during Q320. The underlying processing is available in SONLIB, and the *spdisplay* ETI widget could display the data.

This effort should include completion of the multi-channel display options (Section 8.3.6), and would likely benefit from the enhancement to support multiple simultaneous processing streams. Multiple processing streams would also reduce the load on the EADAQ data server.

8.3.18 Allow header file for WAV data source to get time stamp

Akoostix has *.hdr* files that allow the user to specify the start / end time and floating-point sample rate for a WAV file. It would be useful to improve the WAV source to look for an *.hdr* file and use it, if it is present. Units like ADAC would then be able to make better use of annotation that requires an accurate time stamp.

8.3.19 ETI formation at the output of the normalizer

Allowing the user to specify ETI formation at the output of the normalizer versus the output of the averaged PSD (power-spectral density) module would allow it to operate more like a detection function, showing peaks for signals that are above the estimated background level.

8.3.20 Despeckle Sonograms

OPD could be integrated with the despeckle algorithm currently implemented in ACDC. This would often be performed before compression and allow images to be compressed more without losing important signals. It would also help to highlight the most significant signals on screen. The existing despeckle functions would be integrated into an *splib* module and modified to work on streaming data. The user interface would require an additional configuration widget to allow the user to set/modify despeckle parameters.

8.3.21 Apply filters to time-series display

The new OPD time-series display should allow the user to define a filter and apply it to the data. This would allow the operator to isolate frequency bands of interest. A filter module is currently available in SPLIB that could be used to define high-pass, low-pass, and band-pass operations with a fairly simple GUI.

8.3.22 General time-series display upgrades

The time-series display is an important, initial indicator of the quality of the data being processed. Its continued improvement will allow users to quickly evaluate the quality of raw data. The following are recommended improvements to the time-series display:

- Set up the time-series display to 'trigger' like a scope and capture pulses or signals that exceed some level.
- Provide on-the-fly zooming, giving fine-grained zooming capabilities. It is expected that zooming may take the form of +/- 10% options, finer scroll wheel adjustments or a manual entry of the zoom factor and/or range desired.
- Connect the time-series window and the GRAM/ALI/ETI windows to allow for cross-plot interactions and communications. This would provide synchronization of the GRAM and time-series displays when scrolling. Markers in the time-series window could be related to visible GRAM data, cursor interactions, etc.
- Provide multiple channels of time-series data viewable on a single window, in a similar manner to the ETI data (including legend and colours). Some complex logic would be required to track which displays are showing which channels of data, so that paging makes sense.
- Integrate the ability for the data manager to use one or more data files to overflow time-series data for longer history viewing without consuming all of the application's memory. It is not expected that this history data would be persistent, and would be deleted when OPD exits.

8.3.23 Autosave annotations periodically

The annotations file should be saved periodically as a precautionary measure in case OPD crashes. This will reduce the chances of loosing important analysis information.

8.3.24 Create icons for toolbar buttons

Icons should be used instead of text for the toolbar buttons. This would save on display real-estate. The cost of this change is highly depending on how specialized and professional the icons would be although integration of icons is trivial. Additional consideration should be given to how icons are used at an application level, including their design, colour, and format.

8.3.25 Inter-click interval (ICI) cursor

An inter-click interval measurement tool would help classify species of marine mammals for DRDC experiments and trials. The tool would look very similar to a harmonic divider but measure across the time axis instead of frequency axis in a GRAM display. The cursor would also have error bars to allow for variable ICI.

8.3.26 Closest point of approach (CPA) cursor

A cursor to measure CPA (closest point of approach) would be useful for determine tactical information about a target. This would be useful for Acoustic Data Analysis Centre (ADAC) during passive monitoring as well as to DRDC for trials involving the Expendable Mobile Anti-Submarine Warfare Training Target (EMATT).

The basic version of this tool would mark the up and down Doppler and the CPA time. Advanced versions of the cursor would model the vessel speed either based on Doppler shift or a set speed. Another option would be to use the current center-frequency to help fix some variables. The user should be able to adjust the CPA parameters including:

- Frequency minimum,
- Frequency maximum,
- Time of CPA,
- Range of CPA,
- True Speed, and
- Doppler Speed

The best solution would be to have a "best-fit" option. This has already been prototyped in STAR-IDL and has shown that it can outperform an operator.

8.3.27 Target specific cursors

A target specific cursor would help ADAC identify specific targets based on known set of signature information. This cursor would behave like a harmonic cursor, but with specific combs at known ratios, and others at fixed frequencies. Development of this cursor would be complicated by the fact that it would have to be configured independent of OPD due to the classified nature of the target information.

8.3.28 Windows 7 support

It would be good to keep OPD up-to-date with the latest Window 7 operating system. This is important as DRDC upgrades their infrastructure (many computers on the CFMETR 2011-1 trial were Windows 7 computers). Compiling for the latest operator systems and using the latest build tools also helps to make the software more robust.

8.3.29 Multiple connections to a single data source

Data capture systems such as EADAQ only allow one connection per computer (IP address). It would be useful (especially during DRDC trials) to use ACDC and OPD concurrently on the same computer (or allow multiple OPD connections for users remotely logged on the same computer). In order to do this, a data source multiplexing service could be created that connects to the single data source and relays the data out to multiple connections.

8.4 ACDC

ACDC continues to be a useful application for detecting and analyzing transient signals. Its utility can be further enhanced by adding features to make current tools easier to use as well as integrating advanced processing that could automate some analysis tasks or provide even more information to an analyst.

Table 8 is a summary of selected issues being tracked in JIRA for future upgrades to the ACDC software. These issues are further explained in the subsections.

Table 8: ACDC Recommendations

JIRA ID	Depends On JIRA ID	Summary	Estimated Effort
ACDC-34	None	Click on annotations to select	Small
ACDC-61	ACDC-67	Logging for ACDC	Medium
ACDC-62	None	Data recording	Large
ACDC-65	ACDC-159	Displaying multiple ETI bands	Medium
ACDC-67	None	SQLite backend database	Very Large
ACDC-68	None	ETI level meter during live processing	Small - Medium
ACDC-80	None	Show detection on display	Small
ACDC-85	None	Shortcut keys	Small
ACDC-103	None	Beamform data	Medium - Large
ACDC-104	ACDC-103	Sense beams and filter detections	Medium
ACDC-105	None	Detect pulse trains	Large
ACDC-106	None	Two-axis plot for detection summary	Small
ACDC-107	None	Remember more settings	Small
ACDC-108	None	Filter detections by detecting band / channel	Small
ACDC-111	None	Live tweaking of processing	Small - Medium
ACDC-112	None	Send data to external tools such as OPD, Audacity, etc.	Small - Medium
ACDC-118	None	Voice annotations	Large
ACDC-119	OPDY	Compress processed data in time or post-process at new resolution	Medium - Large
ACDC-120	None	Alarm on contact	Medium
ACDC-148	None	Audio playback feedback	Small depending on API support.
ACDC-159	None	Upgrade to latest version of *spdisplay*	Medium
ACDC-160	None	Filter detections in post processing	

8.4.1 Click on annotations to select

There are several modifications to the annotation features of ACDC that would make them easier to use and increase the users/analysts efficiency. For instance, it would be very useful to be able to select annotations from the GRAM view and manipulate them. There are cases when an annotation isn't mapped completely into detection and thus doesn't show in the detection list. In these cases you may want to modify it, or delete it. It would also be useful to be able to change the extent of the box after selecting it by dragging it.

8.4.2 Logging for ACDC

It is important to know how a system was configured for a number of reasons including proof of due diligence, fault finding, quality assurance, and system optimization. ACDC should log important settings and changes to settings to facilitate meeting these objectives.

For processing runs, the time and date of processing along with the active software version and primary settings should be logged at the start of each run. A short version of this information should be logged in the detection log file. Items that should be logged are those that would affect detection performance.

Initial thoughts:

- Target file or even better a dump of target parameters such as threshold etc. as the target file may not be version controlled,
- Output directory and output / run settings (file names, delays, suppression, etc.),
- Data source and settings (sample rate, gain [if applicable], etc.), and
- Include freeform notes to add in detail not available to the system.

Selected channels and changes in channel selections should be logged, ideally with the option to add free form text to explain why the change was made.

Detection time delays and changes should be logged, ideally with free form text to explain the change.

8.4.3 Live data recording

Now that ACDC has live data feeds it would be useful to record the raw-data stream for post-analysis and debugging. This functionality could become a base option for any data source (reusable) as its needed elsewhere also (See OPDY-49).

8.4.4 Displaying multiple ETI bands

The current ETI display in ACDC only shows the detecting band. It may be helpful to have the option to display all bands at once or be able to scroll through bands denoting whether a band is a signal or noise band.

8.4.5 Upgrade to an SQLite database

Currently, the database used to hold both the transient and the annotation data is a combination of an application-specific, in-memory data structure and the operating system's file system. While this is a workable solution for a small-scale application, the shortcomings become readily apparent as more functionality and complexity is added to the system. For example, it becomes problematic to address issues around concurrent application access and configuration because disk files have no inherent protection from corruption from simultaneous reads and writes; fault tolerance becomes difficult and costly to achieve because the application and the file system are disjointed applications; and managing larger amounts of data with more complex queries becomes increasingly costly to develop and maintain.

Our recommended solution to these issues is to migrate the back-end data storage and manipulation layer to use an SQLite database. The SQLite database is small, fast, and reliable embedded database API that uses ordinary disk files to maintain a real SQL database. The main benefits are as follows:

- Allows for focusing on scientific coding rather than data storage and manipulation.
- Embeds within the application as a library; no third-party tools to install or maintain - no setup or administration is needed.
- Small code footprint: approx. 300K.
- Transactions are atomic, consistent, isolated and durable, even after system crashes and power failures.
- Issues around concurrency and portability are solved already.
- SQL statement support provides flexibility in data queries and manipulation that would have to be custom-written otherwise.
- The database is a single disk file that is binary cross-platform (Linux/Unix, OS/2, OSX, Win32) and 64-bit compatible.
- File could be easily served via network or serial communications and will stay intact.
- Dozens of language binds would allow for easy scripts to be provided to insert/extract data into any custom format used with other systems or for analysis.

The current in-memory/file-system transient-database API already provides a layer of abstraction and encapsulation behind which an SQLite database could be seamlessly inserted. It would require little extra modification to the actual applications themselves. If desired, a full client/server database such as MySQL could also be used for compatibility or other requirements-driven reasons.

8.4.6 ETI level meter during live processing

The transient-detection capability has been integrated with ACDC. Now that it is part of one run-time application, several real-time tools would be useful to determine the behaviour of the processing, providing instant feedback to the user.

One useful tool would be to view the current ETI level for the different processing bands. Additions, such as threshold levels indicators would allow the user to view that signal as it approaches detection. ACDC currently doesn't manage data in real-time similar to OPD so if historical data is required then a real-time memory management capability would be required.

8.4.7 Show detection on display

It would be very useful to overlay a box on the ACDC snapshot display that encompasses the detection (as seen by transient detector). This would allow the operator to know exactly what the underlying algorithm is doing and help diagnose configuration or software issues. The temporal bounds are already available, but the frequency bounds would be harder. A simple annotation on the time axis (such as a red bracket) could be added to highlight the detection time and duration. Ideally, frequency bounds would also be added, but ACDC does not have that information (yet) except for live runs.

8.4.8 Shortcut keys

When using ACDC operationally, users will need to do things like ignore a contact, mark it as a target quickly, and do other operations quickly. Some level of task flow analysis should be performed and where appropriate logical, short-cut keys or context sensitive menus should be created to speed up analysis.

8.4.9 Beamform data

It would be useful to be able to beamform data and bring it into ACDC without having to pre-process it, such as via a feed from EADAQ. This would reduce the number of tasks required before analyzing the data.

8.4.10 Sense beams and filter detections

When bringing in beamformed data (see 8.4.9) it is likely that detections will be produced on neighbouring beams. It would be useful to be able to filter the detections and present only the "best" detection to reduce the amount of information that gets logged and analyzed.

8.4.11 Detect pulse trains

Marine Mammals often vocalize as part of a pulse train. It would be useful to be able to detect the pulse trains and present them as a single detection or allow a higher level of detection that groups these associations and presents statistics on elements such as inter-click interval.

8.4.12 Two-axis plot for detection summary

It would be useful to be able to plot the time and band on different axis as their range can often be quite different.

8.4.13 Remember more settings

It is useful for ACDC to remember which target files, data files, etc were used between running the application. The last settings used are often appropriate. This would reduce setup time and decrease the chances of making a processing error due to incorrect configuration.

It would also be useful for ACDC to remember which data channels, and minimum time between detections was used between runs. Some error checking would be required to see which channels are still available as the input may have changed. This is done in OPD and a similar approach could be used to decide on which channels to present.

8.4.14 Filter detections by detecting band / channel

It would be useful to be able to configure ACDC to filter the detection list by the band that detected. It would also be useful to be able to filter by channel. This would provide additional options for the user to adjust their view and focus on specific set of detections.

8.4.15 Live tweaking of processing

It would be useful to be able to adjust the detection threshold, bands, etc., on-the-fly. This would include being able to simply turn them on/off (at least from detection). This would eliminate the need to reconfigure and restart the detection processing which could result in a loss of important information during trials.

8.4.16 Send data to external tools such as OPD or Audacity

It would be useful to be able to quickly send a command to another application such as OPD or Audacity to bring up a selected ACDC detection for processing. OPD currently does this when extracting WAV files. This provides easy access to external tools that may have additional processing capabilities and analysis tools.

8.4.17 Voice annotations

It would be nice to be able to voice annotate the data (voice clip with timestamp). The user could then review the data and voice clips in some integrated manner. This may be easier and quicker than text annotations especially in environments where it might be difficult to use a keyboard.

8.4.18 Compress processed data in time or post-process at new resolution

It would also be useful to be able to display compressed data. This would allow the user to see more of the data at once without losing important information. This would include compression in time and would likely require an OR operation vice averaging the data. This option would also be useful on OPD with some compression factor.

8.4.19 Alarm on contact

It would be very helpful if ACDC could produce an audible and/or visual alarm when new contact is generated. The alarms may change to indicate the confidence in the detection / detection rate. This would allow the operator to setup ACDC and monitor it periodically.

8.4.20 Audio playback feedback

Right now there is no progress indicator or any other type of feedback for the user when they play a file. So it's hard to tell if it's actually working or not. Some kind of feedback mechanism like those used in media players should be added. This feedback would be a static text display showing the current play time and total file time.

8.4.21 Upgrade to latest version of *spdisplay*

The ACDC application is currently using *spdisplay* version 2.7.3. The latest version of *spdisplay* is 2.9.x which is currently used by OPD. ACDC should be upgraded to the latest *spdisplay* version to take advantage of bug fixes and new features provided by the latest library including harmonic, banding and Doppler cursors.

8.4.22 Filter detection in post processing

ACDC could perform some post-detection processing to do logical filtering for click trains. This would automate some marine mammal analysis tasks for the user and increase efficiency.

It would also be useful if ACDC continued processing until the end of a click train in order to get bigger samples to look at as a group. This would reduce the number of detections for the analyst that are related to the same click train.

8.5 Build system and configuration management

It is crucial that the build system provides the necessary level of support for the growing number of tools in STAR. The build system and proper configuration management of the DRDC repository work together to ensure that the software remains easy to compile, test, install and versioned.

Some ideas for future build system maintenance work are provided in Table 9. Note that unlike previous sections, all issue related information is directly in the table.

Table 9: Build System Improvements

JIRA ID	Depends On JIRA ID	Description	Estimated Effort
INST-25	None	Improve Windows 7 and 64-bit support. This will involve better detection of the compiler and platform since 32-bit cross compiles are possible on a 64-bit Windows host.	Small
INST-24	None	Improve OSX 64-bit support. This will involve better detection of the compiler and platform since 32-bit cross compiles are possible on a 64-bit OSX host.	Small
INST-26	INST-25	Once platform detection is improved on Windows, automatically choose the correct version of the FFTW3 DLL to install (32 bit or 64 bit).	Small
INST-27	None	Incorporate embedding of license files into the Python OSX Bundle script, which will be displayed when a user installs the software.	Small

9 Summary and conclusions

Akoostix addressed a variety of STAR-related technical support tasks. As a result the following has been accomplished:

- Documents reviewed by Akoostix contain updated or new information for publication.
- The software development performed on OPD increases its value for future trials and data analysis efforts by providing advanced measurement and analysis tools.
- The STAR training facilitated by Akoostix will help DRDC scientists and developers to directly access the extensive suite of analysis software for their own individual areas of research.
- The ISIS SBC is more than capable of replacing the Viper in the current (and hopefully future) implementations. It requires substantially more power than the Viper, but is much more capable.
- The feature extraction software will aid in future tracking research being performed at DRDC. The application will increase the accessibility to features in multistatic data.

References

[1] Theriault, J.A., Collison, N.E., Mosher, Hood, J.D., and Bougher, B.B. (2011) DIFAR Sonobuoy Bearing Accuracy, DRDC TM 2010-244, December 2011.

[2] Hood, J., and Glessing, B. (2004), The Software Tools for Analysis and Research: Data Analysis and Technical Manual - Revision 1, MacDonald Dettwiler and Associates Ltd., Dartmouth, Nova Scotia.

[3] McInnis, J. and Ryan, G. (2011), OPD User Manual Version 1.5 (AI MA 2008-001), Akoostix Inc., Dartmouth, Nova Scotia.

[4] Flogeras, D. (2010), Active LAND Buoy Operator Manual Version 1.0 (AI MA 2010-009), Akoostix Inc., Dartmouth, Nova Scotia.

[5] Hood, J. and Bougher, B. (2010), MMOS 07 and Sirena 08 Analysis Report, DRDC Atlantic CR 2010-060), in review.

This page intentionally left blank.

Annex A Tracking support user instructions

The Tracking Support Application provides automated multistatic detection and feature extraction from DRDC trial data. The following section provides user instructions for this application. The Tracking Support application is one of many tools in the STAR-IDL analysis tools collection that works with the STAR trial structure [1] including the Non-Acoustic Data (NAD) for tactical information. This application also uses the processed correlation data files and corresponding main blast logs that are often processed during a multistatic trial. These user instructions do not cover setting up the trial structure.

This application automates two common analysis tasks including:

- Designating data that is of importance or of interest for analysis. The Tracking Support application designates data using the main blast information that matches the user specified properties. Designations are created by defining a search window (in time) in relation to each main blast.
- Detection and feature extraction are performed on the designated data. The features are logged to an analysis log in human readable format. This log can then be used to do further analysis in tracking research.

A.1 Initial setup

A template start-up script for the Tracking Support application is located in the STAR-IDL source distribution at *src/template/tracking_support_template.pro.* Like most STAR-IDL tools, this script is typically copied and renamed to the *idlprog* directory for the target trial structure. This template is self documenting and provides a description of all the parameters that can be configured as defaults when running the main application. This includes parameters such as:

- Default start and end time. See Section A.2.
- Speed of sound used during processing.
- Default active sensors. See Section A.4.
- Target name. See Section A.2.
- Echo log path. If this is defined, the detector also produces an echo log that is compatible with other STAR-IDL analysis tools.
- Default analysis log path. See Section A.5.

Note that some of these are *only* configurable through the script and cannot currently be accessed by the operator interface including speed of sound, echo log path and analysis log path.

A.2 Main application

The application can be run once the application script has been copied and configured for the trial (see Section A.1 above). During initialization the software will read in the Non-Acoustic Data

(NAD). Once initialization is complete the application's main user interface appears as shown in Figure A.1.

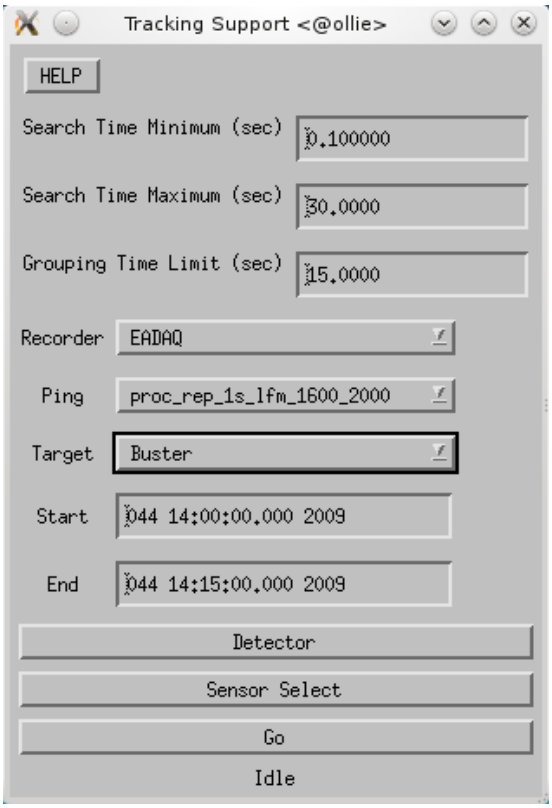

Figure A.1: Tracking support application main configuration dialog

The following is an explanation of each configuration item:

- The *Help* button provides an abbreviated version of the user instructions as well as any additional warnings or tricks to aid the user when using the Tracking Support application.
- The *Search Time Minimum* is the amount of time in seconds after the main blast time to start detection and feature extraction. The search time minimum and maximum define a search window.
- The *Search Time Maximum* is the amount of time in seconds after the main blast time to stop detection and feature extraction.
- The *Grouping Time Limit* is used to cluster main blasts in order to determine which main blasts belong to a ping event. The difference for any main blast times within a clustered group can be no more than the specified time limit in seconds. This value should be set based on the maximum geographical spread of active sensors for a particular trial.
- The *Recorder* drop down box allows the operator to select a recorder to use during processing. Only main blasts that are associated with the specified recorder are processed.

- The *Ping* drop down box allows the operator to select the ping waveform to use during processing. Only main blasts that are associated with the specified ping waveform are processed.
- The *Target* drop down is optional. It allows the operator to select a known target from the trial database in order to compare the processed results with the expected results. Expected results are logged to the analysis file (described in Section A.5). Note that the default target name can be configured in the application script (see Section A.1).
- The *Start Time* edit box allows the operator to specify the beginning processing time other than the default start time. This is useful if the operator only wants to extract features from the trial data within a specific time range. The default start time can be configured in the application script (see Section A.1). The start time will not be set until a valid time is specified in DDD HH:MM::SS.ss YYYY format. An 'X' will appear on the right-hand side of the edit box while editing to let you know that the current text is an invalid time.
- The *End Time* edit box allows the operator to specify the end processing time other than the default end time. Only main blasts with times between the start time and end time will be processed. See *Start Time* text in bullet above.
- The *Detector* button displays the configuration dialog which is used to configure the contact detection and feature extraction processing as described in A.3.
- The *Sensor Select* button displays the sensor selection dialog which is used to select the active sensors for processing as described in Section A.4.

A.3 Detector configuration

The contact detection and feature extraction processing can be configured using the configuration dialog shown in Figure A.2.

Fields whose labels begin with *Echo* belong to the feature extraction processing while those starting with *Detect* belong to the detection processing:

- The *Echo Noise Window* sets the noise window size in seconds.
- The *Echo Select Max Beam* is a Boolean value (0 or 1) that turns off/on the select max beam function. When turned on, the echo analysis algorithm selects peaks from the beam with the most energy.
- The *Echo Search All Beams* is a Boolean value (0 or 1) that turns off/on the search all beams function. When turned on, the echo analysis algorithm searches through all beams in the time search window to find the peak.
- The *Echo Peak Search Window* sets the size of the search window in seconds, which represents the designation error.
- The *Echo Notch Window* adjusts the notch window in seconds, which represents the duration of the echo.
- The *Detect Threshold* is the signal to noise ratio, in dB, used to determine detection.

- The *Detect Gap Size* is the detector gap window size in seconds and is used during SNR estimation, which uses a split window estimator.
- The *Detect Signal Window* is defined in seconds. An average over the window is used to estimate the signal level for the SNR calculation.
- The *Detect Noise Window* is defined in seconds. An average over the noise window, minus the gap is used to estimate the noise level for the SNR calculation.

Tip: *The Echo Peak Search Window* is the time limit used to cluster detections into a single feature. The operator can lower this value to reduce the number of *simultaneous* features that are being produced.

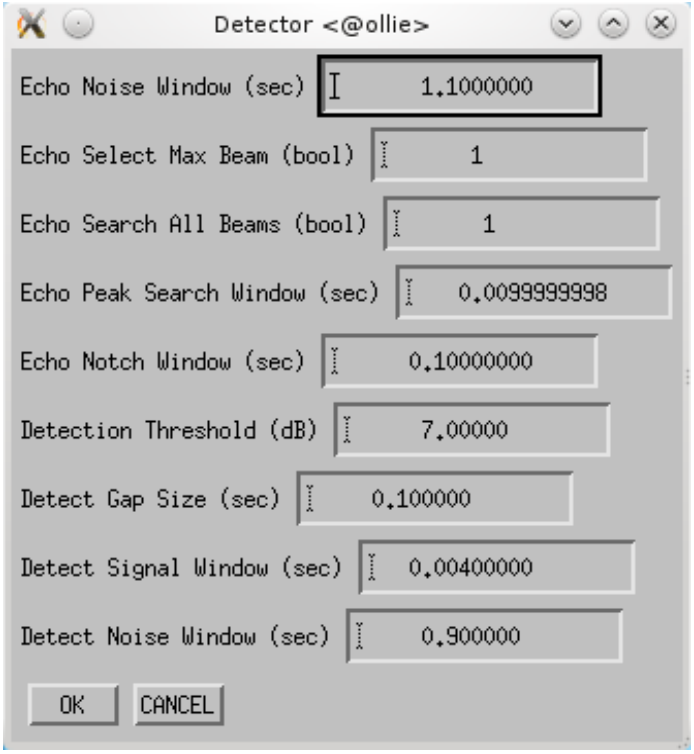

Figure A.2: Detector configuration dialog.

A.4 Active sensor selection

The Tracking Support application can be configured process for specific sensors using the configuration dialog similar to that shown in Section A.4. Note that the appearance of this dialog will vary based on the number of available sensor as well as their ID (or RF) assignment.

This dialog allows the operator to filter any unwanted or defective buoys. You must press the *OK* button for the settings to take effect.

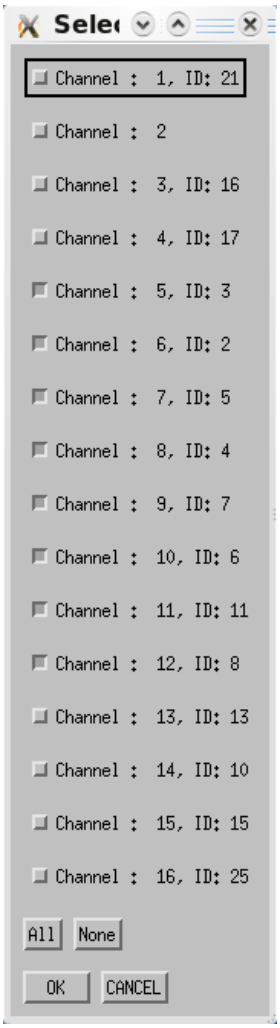

Figure A.3: Sensor selection dialog

You can set the default active sensors in the *tracking_support_template.pro* script for example:

receivers = [4, 5, 6, 7, 8, 9, 10, 11]

Note that receivers are zero-offset where channels in Figure A.3: Sensor selection dialog are 1-offset channels. i.e receiver 4 equals channel 5.

A.5 Analysis logging

The results of the feature extraction processing are logged to disk in the file specified by the *log_filename* variable in the *tracking_support_template.pro* script. The log consists of ASCII messages with one message per valid input main blast. Tokens are separated by spaces. Each message contains the data from all fields described in Table 10. Here there are 3 types of information; metadata that describes the message (*italics*), user-defined data (**bold**), and data computed by the algorithm (normal font). The fields are written into the log message in order. Along with the AUTO messages, the detection and feature extraction configuration is logged as comments in the analysis log. Comments are denoted by '#' as the first non-white-space character on the line.

Table 10: AUTO Log format

Field	Description	Value
Serial Number	A sequential number denoting the position of the message in the log.	Integer
Message Type	The string 'AUTO' to indicate that this is an auto association statistic message.	String Token
Major Version	The major version of the message format.	Integer. The current major version is 1
Minor Version	The minor version of the message format.	Integer. The current minor version is 0
Main Blast Time	The time of the main blast on the searched receiver.	DDD HH:MM:SS.sss YYYY
Hooked Sensors	A list of sensor identifiers (RF for sonobuoys) for the receivers used to hook the data. This is set to NONE for Tracking Support.	String tokens contained in []¹ separated by a space.
Recorder	Identifier for the recorder used to record the data.	String Token
Channel	The channel that was processed to produce the detections.	1-offset Integer
Sensor ID (RF)	Sensor identifier (RF for sonobuoys) of the searched receiver.	String Token
Search Time Adjustment	Number of seconds that were subtracted from the minimum and added to the maximum time window. This is not used for Tracking Support (default 0.0).	Floating point number
$\mathbf{T_{min}}$	Seconds past the main blast that was used to start searching for detections on the searched receiver. For Tracking Support this is the *Search Time Minimum*.	Floating point number
$\mathbf{T_{max}}$	Seconds past the main blast that was used to end searching for detections on the searched receiver. For Tracking Support this is the *Search Time Maximum*.	Floating point number
T_{expect}	A list of expected detection times in seconds past the main blast time, one for each reported target location.	Floating point number contain in [] separated by spaces
$\boldsymbol{\theta_{left}}$	The left-hand bearing in degrees True used to search for detections. This is fixed at 0.0 degrees for Tracking Support.	Floating point number.

[1] [] are delimiters used to mark the start and end of a list in the log.

Field	Description	Value
$\theta_{\mathbf{right}}$	The right-hand bearing in degrees True used to search for detections. This is fixed at 360.0 for Tracking Support.	Floating point number.
θ_{expect}	A list of expected detection bearings in degrees True, one for each reported target location.	Floating point number contained in [] separated by spaces.
$T_{detects}$	A list of detection times in seconds past the main blast in the searched receiver.	One or more floating point values in a space delimited list within []
$\theta_{detects}$	A list of detection bearings.	One or more floating point bearing values in a space delimited list within []
$SNR_{detects}$	A list of SNRs, one for each detection.	One or more floating point values in a space delimited list within []
Time Errors	A list of time errors in seconds, one for each detection. Currently unused by Tracking Support. Default value is 0.0	One or more floating point values in a space delimited list within []
Bearing Errors	A list of bearing errors in degrees True, one for each detection. Currently unused by Tracking Support. Default value is 0.0.	One or more floating point degrees True values in a space delimited list within []
Ping Wave Form	The ping waveform associated with the main blast.	String Token
Source Latitude	The latitude of the source at the time of detection.	Floating point bearing in degrees
Source Longitude	The longitude of the source at the time of detection.	Floating point bearing in degrees
Sensor Latitude	The latitude of the sensor at the time of detection.	Floating point bearing in degrees
Sensor Longitude	The longitude of the sensor at the time of detection.	Floating point bearing in degrees
Expected Values Good	Either a 0 or a 1. This is set to 0 if no target information was provided during processing and therefore the expected values cannot be trusted.	Boolean

Annex B Software tools

This section provides background information necessary to understand the role that DRDC software played in this contract. Their relationship to the project is described below while a high-level description of the tool itself is provided in the subsections below:

- SPPACS was used perform transient data-analysis in support of generating publications for DRDC. SPPACS was also used during the STAR training course since it play a key role processing DRDC data for data analysis.
- STAR-IDL was the central focus of the STAR training course. As part of the training course facilitated by Akoostix, DRDC was exposed to many of the data and tactical analysis applications that comprise the STAR-IDL framework.
- OPD was upgraded during this contract and used for basic quality-assurance tasks. The tools that were added to this application will continue to provide an important data-analysis capability for DRDC and other centres such as ADAC.
- A replacement board for the Viper is being investigated which is currently the host platform for the Acoustic Subsystem. This board has been used extensively by DRDC as a low-power embedded signal processing / capture device integrated with such platforms as the Stealth Buoy, LAND Buoy, and the Slocum Gliders.

B.1 Signal processing packages (SPPACS)

SPPACS is a group of software programs that are written in the C/C++ programming languages, with each application providing a specific processing or utility function. They are designed to run on Linux and OSX based PCs and typically work with Defence Research Establishment Atlantic (DREA) formatted data files (DAT), though format converters are also contained in the suite. SPPACS has slowly evolved to its present-day state.

The SPPACS software suite consists of two types of software. One type is run-time executables. These applications have 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, often very customized, processing streams. These streams can be run from the command line or assembled into scripts to perform batch-processing tasks allowing large amounts of data to be automatically and incrementally 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 run-time software, but can also used for other applications, such as OPD. There are several types of libraries of which three are most commonly used in SPPACS:

- Utility (e.g. math, geo, filesystem, …) libraries that consist of utility routines for performing tasks, such as header manipulation, geospatial data representation, and command line parsing.

- Signal processing (e.g. *splib*) libraries that contain modules for low-level signal-processing. A new SPPACS module typically consists of one or more SPLIB modules linked together with an SPPACS user interface.
- Sonar processing (e.g. *sonlib*) libraries that contain modules consisting of several SPLIB modules linked internally to create a complex sonar module, such as passive processing.

B.1.1 Background and design information

More generic and reusable software was created by separating the library code above from SPPACS. These modules are independent of the data header format, time-stamping method, etc., and are suitable for integration in real-time processing systems. The libraries can be built to run on a number of Unix, OSX or Microsoft Windows platforms and on less common processors such as the ARM core and Texas Instruments DSP. Once successfully ported, the CMAKE build environment supports subsequent builds with a command line option.

The C and C++ elements of the libraries are intentionally separated to ensure that the core capability, found primarily in the C modules, can be readily ported to systems that don't support the more complex language features employed in the C++ version of the libraries. For the most part, the C++ layer consists of a wrapper on the C layer that provides a more generic method of instantiating, connecting and running modules. This is provided by inheritance that is, in part, the adoption of a common interface from a base class allowing parts of the system to interact with a module without knowing the details of the module. Connection of SPPACS applications using Unix pipes provides similar functionality at the application layer.

SPPACS is also supported by a number of libraries, such as the Fastest Fourier Transform in the West (FFTW), helping to ensure that the SPPACS software runs as efficiently as possible, while providing a significant reduction in coding effort. These dependencies, and the associated licenses, are tracked for those projects that require knowledge of intellectual property (IP).

B.2 STAR-IDL

The STAR-IDL[2] tools were developed to support general research and analysis objectives at DRDC Atlantic. The actual software goes hand-in-hand with an analysis process that is intended to help formalize a reliable and consistent research and analysis methodology [1]. The primary objectives of the STAR-IDL tools are:

- Provide scientific-grade analysis tools that allow for efficient, detailed quantitative and qualitative analysis of a data set.
- Provide scientific-grade algorithm prototyping and refinement tools that can be used to quickly realize a variety of algorithm options, validate the basis of the algorithm, and determine the best approach to use for system prototypes.

[2] The STAR-IDL tools were formerly referred to as the Software Tools for Analysis and Research (STAR). The STAR Software Suite has now come to mean the greater tool set, including OPD, ACDC, SPPACS, etc.

- Support synergy between DRDC groups and the Department of National Defence (DND) by providing a common software base for analysis. This synergy encourages inter-group communication and simplifies user training, analysis process development, documentation and data portability.
- Support cost and analysis efficiency by providing software reuse and common tools and data formats. Examples of efficiency would be using the output of analysis from one group to feed the inputs of another, or using common software components to lower development cost of several custom analysis tools.

Most STAR-IDL components are currently implemented using Interactive Data Language (IDL), though the design is not restricted to IDL. For example, localization algorithms contained in C++ libraries are accessed from IDL.

Applications in the STAR-IDL tools are built using a combination of reusable and custom components that meet the requirements of each application. The layered design and common components allow for rapid and logical development of new capabilities. Though currently focused on two main areas - sonar data processing and analysis, and target localization, tracking, and multi-sensor data fusion - the tools are capable of expanding to meet other analysis and research requirements.

B.3 Omni passive display (OPD)

OPD is a standalone signal-processing application designed to run on UNIX, OSX, and Microsoft Windows platforms. It can be used to quickly produce sonogram, energy-time integration (ETI), amplitude-line integration (ALI), and time-series output from DREA digital acoustic tape (.DAT/.DAT32) files, wave files, sound card, EADAQ, Rapidly Deployable System (RDS), and Northern Watch arrays. The following functions summarize its capability (detailed information can be found in the OPD User Manual [3]):

- A user can quickly set up the desired signal processing by loading in a preset configuration from storage, or by simply defining the desired frequency and time resolution. A more-sophisticated user can define a wide range of parameters, including Fast Fourier Transform (FFT) size, zero padding, overlap, quantization range and much more.
- OPD provides an optional beamformer
- Annotations can be added to the data.
  - The user can assign a category (or classification) to the annotation from a list of presets as well as provide free-form text to associate with the annotation.
  - Previously-generated annotations are displayed on screen when processing data associated with the annotation.
  - The annotation format is compatible with STAR-IDL and ACDC.
- Each processing result is stored in memory and can be selected for viewing and analysis. Analysis tools include a cross-hair cursor for time-frequency measurements.
- The entire sonogram can be saved to an image file to capture the output for reports, etc.

- A WAV extraction tool allows the operator to define a region within a sonogram and clip the raw data associated with the selected bounds into a wave file..
- Operational measurement tools such as harmonic, banding, and Doppler cursors can be used to analyze advanced features in data and learn tactical information about potential targets.

B.4 Acoustic subsystem (AS)

The AS is a general-purpose embedded acoustic recording and detection system composed of an integrated set of reusable software modules. The AS operates in one of three modes:

- Transient detection and recording mode – In this mode onboard detection processing is performed, the entire sample period is recorded, and individual captures (WAV files) are created for each detected transient along with an ASCII detection log. When operating at 40 kHz bandwidth the maximum sampling duration, on the current hardware, is 5 minutes with a duty cycle of ~50%.
- Target (vessel) detection and recording mode – In this mode onboard detection processing is performed, the entire sample period is recorded, and an ASCII detection log is created. In this mode, the bandwidth is often limited allowing the AS to process at real-time.
- Ambient recording mode – In this mode the AS records acoustic data and operates real time, so recording duration is not limited on the current hardware, provided enough flash memory is available.

The system is intended for soft real-time operation and is normally installed on a low-power, fixed-point, general-purpose processor and paired with other technology that acts as the vehicle (e.g. Slocum Glider, LAND Buoy, Stealth Buoy).

When used for marine mammal detection, it is most commonly paired with the Slocum Glider, where the current acoustic sensor bandwidth (40 kHz) supports detection of a broad range of species. The AS is designed to work with ACDC, where ACDC provides post-processing, if required, and data visualization.

The AS is composed of a number of technology layers. Its capability will be described at each layer, as capability varies significantly. At the topmost layer, the AS is:

- Single channel.
- Data is sampled at 16-bit resolution at rates up to 100 kHz.
- Acoustic bandwidth is variable from 2 to 40 kHz.
- Acoustic preamplifier gain is variable from 0 to 35 dB in 5 dB steps. The analogue to digital (A/D) also has adjustable input voltage ranges that are ± 1, 2, 5 and 10 Volts.
- Data is recorded on a standard compact flash card and recordings can be taken up to the limit of the card capacity
- The AS provides a serial interface for a host controller interface with a basic command set to allow an external system to control it. The extensible command set includes control of the sample period, time setting, query of detection status, and power off. Where an external

interface is not available the same interface is controlled by an AS host controller via a socket and onboard software. The AS host controller provides additional control and functionality over a serial user interface (terminal interface).

- The AS provides a serial user interface (terminal interface with text menus) for direct user access to manage modes, logs, data, etc.

Underlying the AS is modular technology with greater potential. It allows for processing of any number of data channels at any sample rate, using floating-point numbers. This includes both detection processing and recording. This software is written so that any data format can be supported via an appropriate module at the front of the processing stream. WAV, WAV64, and other formats supported by *libsndfile* are currently supported along with a number of DRDC proprietary formats, one PC-104 A/D, and various sound cards that are supported by standard API on MS Windows, OSX, and Linux. New data sources are regularly added.

List of symbols/abbreviations/acronyms/initialisms

[]	Delimiters used to mark the start and end of a list of entries.
A/D	Analogue to Digital
ACDC	Acoustic Cetacean Detection Capability
ADAC	Acoustic Data Analysis Centre
ALI	Amplitude Line Integration
API	Application Programming Interface
ARM	An incorporated technology company
AS	Acoustic Subsystem
ASCII	American Standard Code for Information Interchange
BIOS	Basic Input/Output System
BSD	Berkley Software Distribution
CFMETR	Canadian Forces Maritime and Experimental Test Ranges
CMAKE	A cross platform build tool
CPA	Closest Point of Approach
CR	Contract Report
CSA	Contract Scientific Authority
DAT	Digital Audio Tape
DAT32	32-bit version of DAT
dB	decibel
DDD	Debugger Software Tool
DEMUX	De-Multiplexor
DIFAR	Directional Frequency and Ranging
DISO	Departmental Individual Standing Offer
DLL	Dynamic Link Library
DMA	Direct Memory Access
DMG	An OSX Disk Image File
DOxygen	An in-line documentation generator
DSP	Digital Signal Processing/Processor
EADAQ	Environmental Acoustic Data Acquisition
El-Brg	Ellipse/Bearing Crossing

El-El	Ellipse/Ellipse Crossing
El-Hy	Ellipse/Hyperbola Crossing
EMATT	Expendable Mobile Anti-submarine Warfare Training Target
ETI	Energy Time Indicator
FFT	Fast Fourier Transform
FFTW	Fastest Fourier Transform in the West
FFTW3	Version 3 of FFTW
FM	Frequency Modulation
FORTRAN	A procedural programming language
GB	gigabyte
GHz	Giga-Hertz
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GRAM	Sonogram
GUI	Graphic User Interface
Hz	Hertz
ICI	Inter-Call Interval
IDE	Integrated Drive Electronics
IDL	Interactive Data Language
IDLDoc	IDL Documentation (similar to DOxygen)
IMPACT	Integrated Multi-static Passive Active Concept Testbed
IP	Internet Protocol
ISIS	A single board computer manufactured by Eurotech
ITAC	Integrated Tracking and Aural Classifier
JIRA	A tick tracking system by Atlassian
kHz	Kilohertz
KISS	A portable FFT library
LAND	Lagrangian Ambient Noise Drifter
MATLAB	A mathematical modelling language by MathWorks
MB	Megabyte
MS	Microsoft
MVASP	Multi-static VME-based Acoustic Signal Processor

MySQL	A full featured client/server database
NAD	Non-Acoustic Data
NMEA	National Marine Electronics Association
NTPD	Network Time Protocol Daemon
OO	Object Oriented
OPD	Omni-Passive Display
OS	Operating System
OSX	Operating System X (Ten) - Apple OS
PA	Project Authority
PC-104	An embedded PC standard
PDF	Probability Density Functions
PPS	Pulse Per Second
PSD	Power Spectral Density
PWGSC	Public Works and Government Services Canada
RAM	Random Access Memory
RDS	Rapidly Deployable Systems
RF	Radio Frequency
RMA	Return Merchandise Authorization
ROM	Rough Order-of-Magnitude
SBC	Single Board Computer
SD	Secure Digital
SNR	Signal to Noise Ratio
SONLIB	Sonar Library
SPDISPLAY	Signal Processing Display
SPLIB	Signal Processing Libraries
SPPACS	Signal Processing Packages
SQL	Structured Query Language
SQLite	A lightweight SQL compliant database
SSD	Solid State Devices
STAR	Software Tools for Analysis and Research
STAR-IDL	IDL specific applications of STAR
TI	Texas Instruments

TMAST	TTCP Multi-scale Active Sonar Technology
TRE	A portable regular expression library
TS	Target Strength
UNIX	A computer operating system
UUID	Universally Unique Identifier
WAV	Wave file format (i.e. .wav)
WAV64	64-bit Wave format