Development of Automated Multistatic Active Sonar Processing in a Floating Buoy

Final Report

David Flogeras
Joey Hood
Ashley George
Jason McInnis
George Ryan
Akoostix Inc.

Prepared By:
Akoostix Inc.
10 Akerley Blvd - Suite 12
Dartmouth NS B3B 1J4

Contractor's Document Number: AI CR 2012-008
Contract Project Manager: Joey Hood, (902) 404-7464
PWGSC Contract Number: W7707-4500900254
Technical Authority: James Theriault, Defence Scientist
Contractor’s Publication Date: March 2012

Contract Report
DRDC-RDDC-2017-C136
June 2017

Principal Author

Original signed by David Flogeras

David Flogeras

Senior Software Developer

Approved by

Original signed by James Theriault

James Theriault

Project Authority

Approved for release by

Original signed by [Released By Name]

[Released By Name]

[Released By Position/Title]

Table of contents

List of figures

List of tables

This page intentionally left blank.

1 Introduction

This report documents the *Development of Automated Multistatic Active Sonar Processing in a Floating Buoy* project effort, which was executed under contract W7707-4500900254. The Project Authority is James Theriault and the work was conducted between January and March 2012.

Defence Research and Development Canada (DRDC) Atlantic's applied research project (ARP) "Enabling CF Multistatic Sonar" had a key deliverable to show a capability for in-sensor Multistatic Active Sonar (MAS) processing. The primary objective of this contract was to support this deliverable by demonstrating the required capability, building on prior knowledge, software, and system development efforts. In particular, this call-up investigated options for integration of correlation processing, main-blast detection, echo detection, and inter-sensor data fusion software into low-power computer systems that could be applied to in-sensor capability. Concepts for inter-sensor data fusion were considered, but were not implemented due to schedule and budget constraints. The completed work covered several important aspects of this problem, including:

- Designing, developing, and integrating software to upgrade Passive Acoustic Reusable Buoy (PARB) and Dynamic Active Reusable Buoy (DARB) capability.
- Investigating options for upgrading the PARB and DARB single board computers (SBC), which are no longer supported.
- Investigating options for upgrading the PARB A/D and DARB D/A converter electronics to address current deficiencies.
- Investigating options for remote interaction with the PARB and DARB using tablet computers and laptop computers that incorporate touch-screen technology.

The upgraded systems are intended to improve DRDC's ability to address their research objectives, while providing for concept demonstration of advanced capability for the Department of National Defence (DND). Specific advancements include:

- Development of in-buoy MAS processing that can be used to offer new surveillance options, including: increased sensor life, increased sensor range from primary platforms, and in-buoy processing to reduce interface bandwidth requirements. Thus, MAS receivers may be deployed in new roles such as over-the-horizon and transmitting detection data over satellite links[1] (see Section 2).
- User interface improvements, making the PARB and DARB more accessible to DRDC scientists and technicians. Specifically:
    - Simplification of the buoys' menu system, making it more user-friendly and ensuring that commonly-used functions are readily available (see Section 3.1).
    - Addition of a save and restore capability for buoy configuration, allowing less-familiar users to activate configurations generated by advanced users via a simple

[1] The PARB does not include satellite communications at this time, but this can be simulated using Freewave radios and integrated at a later date.

process. The configuration options can later be automated to run on a schedule when the buoy is not in contact with the controlling platform (see Section 3.2).

- Selection of a new SBC (Section 4) and A/D-D/A solution (Section 5) that allows for:
    - Common hardware between PARB and DARB, including D/A-A/D, which simplifies development, reduces requirements for spares, and simplifies maintenance.
    - Provides additional processing power for performing advanced processing, increased storage, and allows for longer deployments or higher data rates.
- Selection of complementary hardware (Section 6) that allows for future development including:
    - Development of the Dynamic Active Suitcase (DAS), using identical SBC and D/A-A/D hardware as the PARB and DARB, along with a touch screen interface.
    - Remote connection for control and monitoring of PARB and DARB, using tablet computers or laptops with touch screen interfaces that can be used from small watercraft.

The remainder of the report is organized to provide additional detail on how specific work elements were completed:

- Section 7 documents recommendations for future work.
- Section 8 summarizes the results of the project with conclusions.
- Annex A provides a brief description of the software used to support this project.
- Annex B provides configuration management (CM) information to help users understand which version of the software was were used for this work.
- Annex C contains the detailed benchmarking and comparison data used to evaluate the candidate single board computers in Section 4.

2 MAS processing upgrade

This section documents the design and integration of the Multistatic Active Sonar (MAS) processing capability that was added to the Acoustic Subsystem. This new capability was implemented by reusing many existing SPLIB modules, continuing development on the multistatics software libraries started under the IMPACT contract in 2011, and then integrating the components into a cohesive processing stream.

The upgrade process is described in:

- Section 2.1, which describes the overall objectives of MAS processing on-board the PARB.
- Section 2.2, which outlines the software design of the processing stream.
- Section 2.3, which describes implementation details and describes how the software was verified after the implementation was completed.

2.1 Objectives

Adding a multistatic active processing capability to the PARB system software is desirable because on-buoy processing allows trial data to be analyzed while it is being captured. Currently, this data must be post-processed after the buoys have been recovered and their data had been transferred to a computer workstation. Acoustic data cannot be feasibly transmitted in real-time, but the data resulting from processing can be, if the buoy's software performs MAS processing.

The general design objectives of the MAS processing are:

- To perform correlation processing on multiple ping types. In the future, this could be extended to multiple sensors and multiple beams, depending upon the sensor packages available.
- To perform main blast detection processing on the correlated data streams.
- To perform echo detection processing for a specified period of time following each main blast.
- To group blasts and echoes together according to their ping types.
- To apply higher-level physical constraints, such as
  - Only allowing valid main blasts once per a user-defined period of time.
  - Only detecting echoes for a period of time following a main blast (user defined).
  - Applying a vetting algorithm to choose the ping type with the best score (defined by some type of evaluation criteria) as the best candidate
- To broadcast textual messages containing blast and echoes information (e.g, time, duration, and SNR), as well as positional information of the sensor. Processed results will also be stored on-buoy, since the operator may not be within its broadcast range.
- To store the raw acoustic sensor data for further post-processing.

Although it was beyond the scope of this project, the software design considers the eventual inclusion of:

- Source information, including the following for multiple-ping sources:
    - Location.
    - Ping type(s) and schedule.
- Communication between PARB units. This could be used for higher quality blast vetting, inter-sensor data fusion, etc.

2.2 MAS Software design

A high-level design meeting was held in early February 2012 outlining several design decisions used to implement the MAS processing. They are:

- Correlation processing will be performed using existing SPLIB modules. The processing will be capable of using both real-valued ping replicas as well as complex-valued replicas.
- Main blast detection processing will be performed using existing SPLIB modules, as they are currently used in the SPPACS application *sp_main_blast*.
- The echo detection processing will be implemented with the SPLIB gap split window estimator. Threshold and detection will be performed by the SONLIB *eti_detect* module.
    - To simplify the design, this processing will run continuously rather than on the reception of a main-blast detection. This prevents complex buffering logic, and results in a much steadier system load. If this becomes a performance bottleneck in the future the design can be easily refactored.
    - Future iterations can include a two-pass split window estimator.

The *main_blast_detection* software library was initially designed and partially developed under the IMPACT contract in 2011. These libraries perform higher-level blast detection functionality such as:

- Cross-sensor and cross-processing stream decision making.
- Managing the physical constraints, such as the ping interval, echo duration, and echo grouping.
- Grouping of closely spaced echo detections into a single detection.
- Vetting invalid or poor quality detections.
- Logging the resulting information to a relational *SQLite* database.

A conceptual diagram of this overall processing design is show in Figure 1.

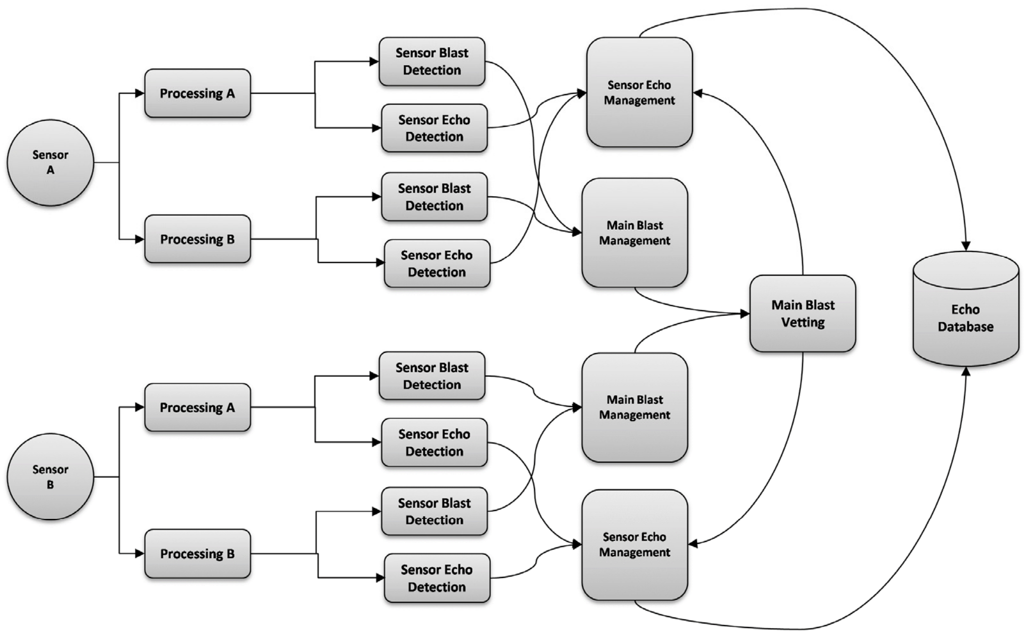


Figure 1 – Multistatic Active Sonar Processing Conceptual Diagram.

Although much of this library was already designed and implemented, new functionality still remained to be added and tested before it would be suitable for the purposes of this project. This work was performed under this contract, which included some modifications to the initial design. The database logging system did not previously exist and was designed, implemented, and tested entirely under this contract.

In order to integrate the MAS functionality into the Acoustic Subsystem (AS), the software was altered so that:

- Blast and Echo events are broadcasted as they occur, via the Freewave modem. This presented a design challenge, since the current detector runs completely autonomously with no communication with the rest of the system other than starting processing at the pre-determined time.
- Access to GPS information is available when logging the Blast and Echo events.

To meet the modem broadcasting requirement, the following design goals were established:

- When detections occur, a positional database will be used to look up the position of the buoy at that time, before creating a text-based message. This positional database is further explored in Section 3.3.
- When the user requests a textual echo log, the software will need to query both the detection database as well as the GPS database, and use information from them to create the logs.

The detector is unaware of the GPS database. It only logs blast and echo times, and signal processing values (duration, SNR, etc). Other software is responsible for combining the detections through a set of APIs. This is partly because in a glider or stealth buoy there is no access to GPS information, but more importantly it separates the concerns of the detector from the source of GPS data. These types of systems do not require broadcasting updates (the *as_host* controller is not present at all). However, if the user requests to download echo logs, the following two options can be implemented:

- Send a textual log with positional information set to null values.
- Send a binary version of the blast and echo detections, and an offline tool can be used to merge them with an existing, external GPS track.

2.3 Implementation and testing

The following subsections discuss the implementation, integration, and testing of the software design discussed in Section 2.2.

2.3.1 Signal Processing

The SPLIB software libraries were an essential part of this project. Existing library modules for correlation processing, main blast detection, and echo detection were integrated to create the core MAS processing functionality, while related modules were created or improved to allow multiple correlation, main blast detection, and echo detection streams to run in parallel. Finally, appropriate management logic was implemented to ensure that the overall system worked as designed, producing echoes only for the processing stream that was associated with the current active sonar pulse.

The following existing SPLIB and SONLIB modules were reused:

- General processing modules:
  - *splib_extract* – Used to separate data channels so they can be processed independently.
  - *splib_data_in* – Used in the SPPACS application to read DREA DAT files.
  - *splib_data_out* – Used in the SPPACS application to write DREA DAT files for examining various processing stages.
- Correlation processing:
  - *splib_apply_filter* – Used to perform correlation with real-valued replicas.
  - *splib_cpx_apply_filter* – Used to perform correlation with complex-valued replicas.
  - *splib_square* – Used to calculate the energy squared of real-valued correlations.
  - *splib_cpx_norm* – Used to calculate the energy squared of complex-valued correlations.
- Main blast detection:
  - *splib_blast_detector* – Used to detect main blasts in the correlation streams.
- Echo processing:

- *splib_estimator* framework – Used to implement various signal/background estimators. This framework was created to simplify addition and integration of new estimation algorithms into the growing list of software applications requiring this functionality (AS, *sp_eti_detect*, MAS processing).
  - *gap_split_window* – This particular estimator implementation was used to estimate echo signal and background levels.
- *sonlib_eti_detect* – Used to threshold and detect echoes in the correlation stream.

The core signal processing streams were easily integrated into the MAS processing stream using the existing C++ modules. New asynchronous modules were created to convert SPLIB detection messages into C++ signals that the *main_blast_detection* library requires as inputs. The producer-consumer relationship of this design will be discussed in Section 2.3.2.

2.3.2 Main Blast Detection Library

The Main Blast Detection library is a multistatics component that provides functionality for identifying main blast ping events and their subsequent echoes in a multi-source and multi-sensor environment. The library has been designed in such a way that it can also be used in the case where there is a single projector and single receiver. Figure 2 details the components involved in the functioning of the library.

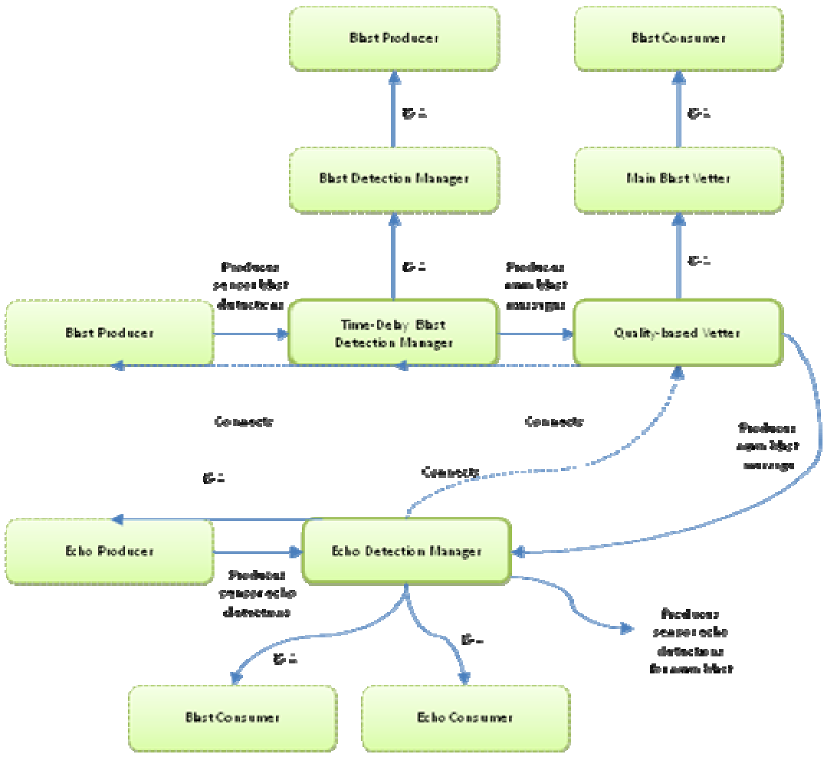

Blast Producer
Blast Consumer
Blast Detection Manager
Main Blast Vetter
Produces sensor blast detections
Produces main blast messages
Blast Producer
Time-Delay Blast Detection Manager
Quality-based Vetter
Connects
Connects
Produces main blast message
Connects
Echo Producer
Echo Detection Manager
Produces sensor echo detections
Produces sensor echo detections for main blast
Blast Consumer
Echo Consumer

Figure 2 – Main Blast Detection Components.

The components follow a stream-oriented, publisher-subscriber model for data flow. A producer for a data type (i.e., sensor blasts, main blasts, or echoes) will push data to all subscribed consumers for that data. If desired, a component can be organized to be both a subscriber and publisher of the same data type, or a publisher and subscriber of multiple data types simultaneously.

In general, sensor blast detections are produced by components using the *sensor_blast_producer* interface. Sensor blasts represent a single event from a sensor that whose data has been processed through an algorithm to find a ping event. These sensor blasts are forwarded to a *sensor_detection_manager* object that collates them into a table for the purposes of determining which sensor blasts across sensors represent a coherent main blast event. Once the detection manager has identified a main blast, it forwards that event forward to another component, the main blast vetter. It is the responsibility of the vetter to examine all of the main blast events in the system and promote one to the status of a system-wide recognized main blast event. After a main blast event is produced, the library offers the ability to obtain echo events that are associated with it.

The process of capturing echoes from the incoming data streams typically happens in parallel with the processing of the data for main blast detection, however there are no constraints upon the library that depend upon this. Like the sensor blast producers and consumers, there are components for building sensor echo producers and consumers. Components built to produce sensor echo events can be connected to an echo detection manager, which provides a limited-capacity staging area for echo events. Once supplied with a main blast event (either from a vetter or otherwise), the echo detection manager will determine which echoes that it has in its internal staging area are relevant to that main blast, and will produce a list of these associated echoes for further consumption downstream in the system.

2.3.2.1 Design

The following is an overview of the design principles behind the multistatics main blast detection library.

2.3.2.1.1 Producers and Consumers

The producers and consumers are very simple constructs used to move data from one component to another in a stream-oriented fashion. A producer is an entity that is responsible for emitting a particular data type, and a consumer is an entity that can consume it. The production of a data type is handled through a callback mechanism, whereby one or more consumers can connect to the callback function and use it to receive these data types. The registration and deregistration of a consumer to a callback is handled automatically through convenience functions provided in the consumer.

The producer classes provide implementations with convenience functions to emit their respective data types and to provide additional information regarding their current working time (more on this will be described later). The consumer classes provide their respective subclasses with virtual function signatures that must be implemented to handle the incoming data events.

Each producer and consumer within the library are either automatically assigned Universally Unique Identifiers (UUID) or can be supplied on when constructed. The use of these identifiers helps to establish uniquely identifiable data paths through the system, even where data and meta-data are similar or identical, including in a distributed environment.

The multistatic components provided by the library are designed to be used in a stream-oriented environment, where there is a continuous time-oriented flow of information. Decisions that are based on time (such as how long to search for echoes) require constant updates on data times in order to produce results. Therefore, the producers and consumers provide the means for the transmission of the current working time of producer to a consumer, in addition to any other data it can supply. A data consumer can then use this time information in any decision making that it does. The only constraint on this functionality is that a data event must obviously precede a time update.

2.3.2.1.2 Producing and Consuming Sensor Blasts

There are custom generalized classes provided for the production and consumption of sensor blast detections (see Data Types). The producer contains straightforward functions for producing a sensor blast detection and for emitting a current processing time:

```
void signal_blast(const sensor_blast_detection& blast );
void signal_processed(const util::date_time::posix_time&
      last_processed_time );
```

The consumer also has a straightforward set of functions that are called automatically when a producer emits data. The first argument in these functions is the unique identifier of the producer.

```
virtual void on_sensor_blast(const util::uuid::uuid&,const
      sensor_blast_detection& ) =0;
virtualvoid on_processed(const util::uuid::uuid& producer_id,const
      util::date_time::posix_time& ) =0;
```

2.3.2.1.3 Producing and Consuming Main Blasts

There are custom generalized classes provided for the production and consumption of main blast data (see Section 2.3.2.3). The producer contains simple functions for publishing main blast data as well as its working processing time:

```
void signal_blast(const main_blast& blast );
void signal_time(const util::date_time::posix_time& t );
```

Similarly, the consumer class provides virtual function signatures that are called when a producer emits data:

```
virtual void on_main_blast(const util::uuid::uuid&,const
      main_blast& ) =0;
virtual void on_time(const util::uuid::uuid&,const
      util::date_time::posix_time& ) =0;
```

2.3.2.1.4 Producing and Consuming Sensor Echoes

There are custom generalized classes provided for the production and consumption of sensor echo data (see Section 2.3.2.3). The producer class has the following functions for publishing sensor echoes and its current working time:

```
void signal_echo(const sensor_echo_detection& );
void signal_time(const util::date_time::posix_time& );
```

Similarly, the consumer class provides virtual function signatures that will be called when a producer that it is connect to emits data:

```
virtual void on_echo(const util::uuid::uuid& producer_id,const
      sensor_echo_detection& ) =0;
virtualvoid on_time(const util::uuid::uuid& producer_id,const
      util::date_time::posix_time& ) =0;
```

2.3.2.1.5 Sensor Detection Manager

The sensor detection manager is an abstract class that conceptually acts as a translator in the producer-consumer flow of sensor blast detection and main blast data. It is both a consumer of sensor blast detection data and a producer of main blast data, hence it provides both interfaces. While it can be used for a number of purposes, within the current implementation of the main blast detection library it is used as the base class for constructing the Time-delay Detection Manager (see XXX), which consumes sensor blast detections from one or more producers and organizes them into a main blast entity.

2.3.2.2 Main Blast Vetter

The Main Blast Vetter is an abstract class that can be used as the basis for selectively filtering main blast data. It provides interfaces for being a main blast data consumer and a main blast data producer. Within the current library implementation, it is used as the basis for the Quality-based Vetter (see Quality-based Main Blast Vetter), which selects a single representative main blast from one or more main blast data objects consumed based on the quality metric associated with the main blast data (see Data Types).

2.3.2.3 Data Types

The following data types are included in the main blast detection library.

2.3.2.3.1 Sensor Blast Detection Class

The sensor_blast_detection data type is a representation of a suspected main blast event as observed and processed from a data source (e.g., a sensor).

```
struct MAIN_BLAST_DETECT_DLL_API sensor_blast_detection {
  util::uuid::uuid identifier;
```

```
    util::uuid::uuid sensor_identifier;
    util::uuid::uuid processing_identifier;
    unsigned int receiver_index;
    util::optional<unsigned int> beam_number;
    util::date_time::posix_time time;
    util::optional<util::date_time::duration> time_standard_deviation;
    util::optional<double> quality;
  };
```

A sensor blast detection carries with it a unique identifier which allows easy cataloguing in a distributed environment. It also has unique identifiers to the sensor from which the data originated and the processing algorithm/stream that produced the detection. In addition, it can also carry a quality metric for determining the confidence that the algorithm had in the detection.

2.3.2.3.2 Sensor Echo Detection Class

The sensor_echo_detection structure contains all of the pertinent data for representing an echo, as obtained and processed from a particular receiver (i.e., sensor) by a particular echo detection processing algorithm.

```
struct MAIN_BLAST_DETECT_DLL_API sensor_echo_detection {
    util::uuid::uuid identifier;
    util::uuid::uuid sensor_identifier;
    util::uuid::uuid processing_identifier;
    unsigned int receiver_index;
    util::optional<unsigned int> beam_number;
    util::date_time::posix_time start_time;
    util::date_time::duration duration;
    util::date_time::duration peak_offset;
    util::optional<double> quality;
};
```

Conceptually, it is similar to the sensor blast detection, however the sensor echo does not have an aggregated and promoted counterpart in the same way that the sensor blast detection does.

2.3.2.3.3 Main Blast Class

A main_blast data type is conceptually similar to a sensor blast detection, but it is considered by the system to be representative of a verified main blast event across all sensors in an environment as detected by a particular algorithm. Depending upon the scenario, the main blast structure can contain a cross-reference of information to all of individual sensor blast detections that represent it.

```
struct MAIN_BLAST_DETECT_DLL_API main_blast {
    util::uuid::uuid identifier;
    util::date_time::posix_time first_detection_time;
    util::optional<util::uuid::uuid> source_identifier;
    std::list<sensor_blast_detection> detections;
};
```

Each main blast event is also tagged with a unique identifier for cataloguing. Also, the main blast's time is considered to the earliest detection time of all of the sensor blast detections from with it was formed.

The process of vetting a one particular main blast as the best representative within a system does not produce any additional information. Therefore a main blast vetter produces the same structure that it consumes (see Quality-based Vetter).

2.3.2.4 Supplied Implementations

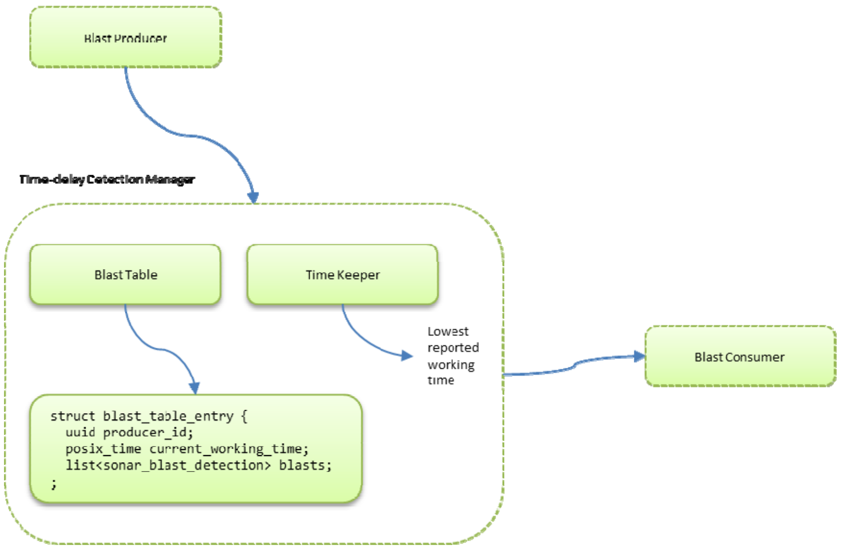
The library provides the following concrete algorithm implementations:

2.3.2.4.1 Time-delay Detection Manager

The Time-delay Detection Manager implements the detection_manager interface to provide a mechanism for producing main blast events from a series of sensor blast events. It does this by maintaining a time window from the first sonar blast event, and watching for additional blast messages during that time window. If all of the producers report blasts during that time window, then the manager can produce an immediate main blast event. Otherwise, the manager watches the current working time of each producer, and when the end of the time window is reached, it will produce a blast event using all reported main blasts.

In addition, the Time-Delay detection manager can enforce a blackout period after it emits a main blast event, during which all sensor blast reports from producers are discarded. Both the time-delay window time and the blackout time are configurable.

As sensor blast producers supply their individual blast and time events, the manager populates an internal table that catalogues them. For the purposes of synchronization, the manager maintains the lowest working time reported by any producer. This allows it to avoid producing a main blast event before all of the producers have either reported a sensor blast event, or have reported their current working times.



Blast Producer
Time-delay Detection Manager
Blast Table
Time Keeper
Lowest reported working time
Blast Consumer
struct blast_table_entry {
uuid producer_id;
posix_time current_working_time;
list<sonar_blast_detection> blasts;
;

Figure 3 – Time-delay detection manager.

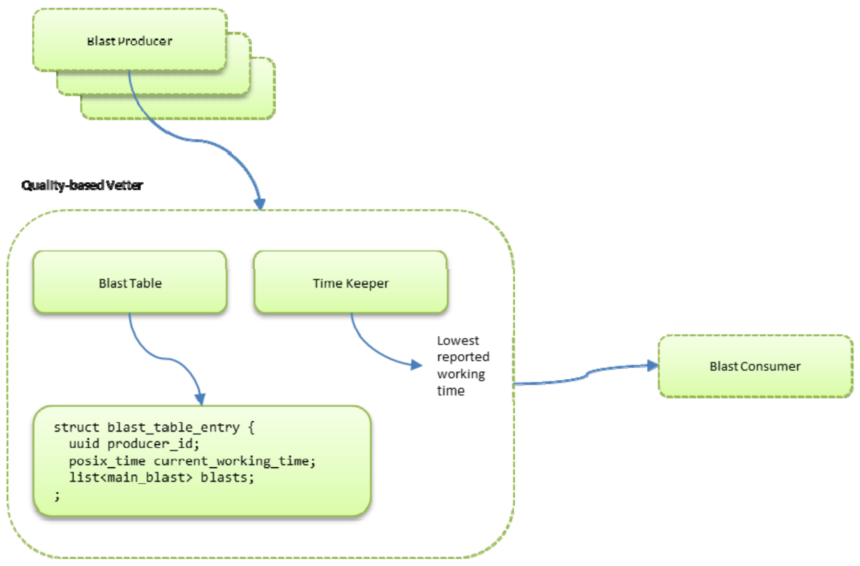
2.3.2.4.2 Quality-based Vetter

The Quality-based main blast vetter is an implementation of the main_blast_vetter interface, and it provides basic main blast vetting behavior.

Each sensor blast data structure includes an optional field for reporting the quality of the detection. This confidence value can be anything that a producer decides upon; however for the purposes of vetting all of the main blasts in the system, the final value must be comparable across all detection algorithms. The Quality-based vetter examines the confidence values of the sensor blasts and the main blast and uses a simple average to determine the overall confidence in the event. It then chooses the main blast with the highest quality value to be the system-wide accepted main blast event. In the circumstance that two or more events have the same quality metric, the vetter selects the earliest one chronologically. In the event that there is no quality information available for any supplied main blast, the vetter cannot make any decisions – it simply produces the chronologically earliest main blast event as the accepted one.

Its implementation is similar in nature to the Time-delay detection manager: it keeps an internal table of all of the producers and as main blast events and corresponding working times are reported, it catalogues them appropriately. The vetter also maintains an internal timer for how long (data-wise) it will wait before dictating a main blast event. Typically, this vetting window will be the same length of time as the time delay of the detection manager, however there is no structural reason that it must be the same.

In difference from the time-delay detection manager, the vetter does not impose any blackout period on its calculations after it emits a vetted main blast event.



```
struct blast_table_entry {
  uuid producer_id;
  posix_time current_working_time;
  list<main_blast> blasts;
;
```

Figure 4 – Quality-based Vetter conceptual diagram.

2.3.3 SPPACS Application

Once all the software library components were implemented and tested individually, the complete processing stream was implemented in an SPPACS application to enable the team to test the software in a batch-processing environment before the final integration into the real-time environment in the AS.

2.3.4 Integration into AS

After the software testing and validation was performed for the SPPACS application, the MAS processing software was integrated into a new AS processing mode (multistatic). The new mode is selectable via the menu system similar to the existing modes (passive, transient, and ambient). A set of default processing parameters are loaded the first time the AS is run. These parameters are not yet configurable via the AS menu, but can be altered manually by logging into the SBC via SSH and altering the configuration database manually.

The multistatic mode stores main blast and echo detections in the same SQLite database format used in the SPPACS software. It then broadcasts blast and echo detections via the Freewave radio. Similar to the existing processing modes, the acoustic data is stored in the AS data directory for further post-processing.

As outlined in the design goals of Section 2.2, the main blast events are also broadcasted via Freewave modem. In order to keep the AS software decoupled from the *as_host* controller, these messages are passed between the two system components using a simple Unix socket-based communication path. When the MAS processing begins, it attempts to connect to a predefined socket on the filesystem. If the socket does not exist or there is a system error, it simply does not broadcast blast events. This was a design goal for keeping backward compatibility with the glider and stealth bouy systems. It is the responsibility of the *as_host* controller software to create the communication path and to listen on the Unix socket for incoming broadcast connections, and then to send the textual messages via Freewave modem.

In the future, the *as_host* will also be responsible for performing GPS database lookups and including the positional information with the textual messages. Improvements to the GPS logging design are discussed in Section 3.3.

2.3.4.1 Verification

A single-channel version of the test case was created for testing. This DREA DAT file was converted to a wave file, and an acoustic sensor was simulated by wiring the soundcard output of a smart-phone into the General Standards A/D board via a 1/8” mini-phono jack.

The multistatic processing mode was set up to sample for 180 seconds and detections were logged to the detection database. After performing a sampling loop, the detection database and recorded wave file were verified using STAR-IDL using the process.

3 General DARB, PARB, and AS improvements

During the kick-off meeting the team identified and prioritized desirable improvements to the DARB, PARB, and AS software. The highest priority items were implemented under this contract while remaining items are listed Section 7 (Recommendations).

The improvements made under this task fall under the following categories and are documented in the following subsections as noted:

- Section 3.1 discusses improvements to user menu interfaces.
- Section 3.2 documents the new save and restore feature for configurations implemented in the PARB, DARB, and Acoustic Subsystem menus.
- Section 3.3 outlines a new design for improving the Acoustic Subsystem GPS logging.
- Section 3.4 discusses testing of the newly added functionality.

3.1 User control menu improvements

The existing user-control menus of the various system components were re-organized to present a simpler interface, with the goal of making the systems simpler to operate for new users. Items were grouped under new submenus according to their function, reducing main menu clutter and less frequently used options and advanced settings were simply moved to make a simplified interface. These changes should in no way affect the functional operation of the systems.

After the work was completed, the new menu layouts were demonstrated to the team. It was agreed that the re-organization greatly improved the menus' look and feel.

3.1.1 General

The menu re-organization task was executed with the following set of guidelines in mind:

- To add the capability to reboot the embedded system to both the PARB and the DARB control menus.
- That similar menu items shall be grouped under common sub-menus.
- That advanced and infrequently used system features shall be placed under an Advanced sub-menu.
- In the AS, the common items (sample duration, sampling bandwidth, change processing mode) shall be relocated near the bottom in all menus, making them easier to locate in all modes.

3.1.2 PARB

In addition to the items mentioned in Section 3.1.1, the PARB menu was re-organized in the following ways:

- Items relating to printing and setting the system time were placed under a new Time sub-menu.
- The following items were placed under a new Advanced sub-menu:
    - Set Filter Board Gains
    - System Shutdown
    - System Reboot
    - Toggle silent detection mode
- The following items were placed under a new GPS sub-menu:
    - Set GPS logging mode
    - Download GPS NMEA files
    - Delete GPS NMEA files

3.1.3 AS

The AS processing menus were improved greatly by grouping existing menu items under submenus by their function. All menu items perform the same functionality before, they have just been moved to reduce clutter[2]. The following re-organization was done for each of the AS processing modes' menu:

- A new "Data" submenu was created which now contains the following items (which were previously all at the top-level):
    - Download acoustic data
    - Delete acoustic data
    - Download system logs
    - Delete system logs
- New "Target" submenu was created (for passive and transient modes):
    - Delete target file
    - Change target
    - Upload a target file
- New "Time" submenu:
    - Print system time
    - Set system time
- New "Advanced" submenu:
    - Clear environment (except ambient mode)

[2] With the exception of the new Configuration menu items used to control new functionality discussed in Section 3.2.

    - Set maximum environment time (except ambient mode)
    - Toggle silent reporting mode (except ambient mode)
    - Set wavpack compression
    - Upgrade system software
    - Processing time factors (pre-multiply and post-add)
    - Set AIO16 gain code (if using the AIO16 card as an input source)
- New "Configurations" submenu:
    - Switch to new configuration
    - Switch to existing configuration
    - Delete configuration

3.1.4 DARB

The DARB menu was reorganized during work in a previous call-up, however to keep with the design goals described in Section 3.1.1, the System Shutdown and Reboot items were placed under a new "Advanced" sub-menu. A "Configurations" submenu was also created to control the adding, switching, and deleting system configurations, as was done for the AS. This should present new users with a more unified place to locate such functionality.

3.2 Save/restore configuration

DRDC requested that the PARB and DARB systems be made easier to reliably configure for multiple-deployment scenarios while in the field. Changing and verifying a myriad of settings in a sea-trial environment can be error prone, especially for new users. A simpler design allowing the system to be set up with multiple pre-set configurations indexed by suitable names was devised. These configurations can be verified during a pre-trial period, and rapidly changed through a simple menu during actual trial work.

3.2.1 General Design

After an initial design meeting, it was decided that the following be implemented:

- All modifiable system settings shall be stored under a configuration name. In order to maintain backward compatibility, the system will create an initial configuration and preload with defaults.
- The user shall be able to create and name new configurations via the control menu.
- The user shall be able to switch between named configurations.
- The user shall be able to delete configurations (with the exception of the currently selected configuration).

- Software settings shall be stored in a binary file format. Previous versions stored the user-alterable settings in text files.

This design allows for the following future improvements which were discussed during the design phase, but have not yet implemented:

- To download all configurations from a unit once it has been set up and verified.
- To upload and merge a configuration file, to quickly program multiple units with the same set of configurations. This can be handled as follows:
    - New configurations are added directly.
    - Existing configurations are replaced by incoming configurations of the same name, presenting a warning prompt to the user first.
    - All other existing configurations remain intact.
- To implement a semi-autonomous scripted mode that can be used to describe the use of multiple configurations. This is desirable for unsupervised over-the-horizon continuous operations. For example:
    - 0801-1200 use configuration “Passive 1”
    - 1201-1600 use configuration “Multistatic 3”
    - 1601-2200 use configuration “Ambient 100kHz”
    - 2201-0800 use configuration “Standby”

After identifying the functional requirements above, the following derived requirements were designed and implemented:

- Two new libraries shall be created:
    - The first is a simple tree-based key-value store using a SQLite database file for persistent storage.
    - The second shall implement the concept of multiple configurations for application integration.
- Applications shall automatically create all required default settings if they do not exist upon first loading of the configuration interface. If the user wishes to use the system in a single configuration (as it was initially designed) all changes are saved to this default configuration.
- Application-specific menus shall be created to allow the user to create, switch, and delete configurations.

3.2.2 PARB

In order to implement the save and restore configuration design properly, the PARB settings had to be separated from the AS settings. Originally, all of the settings for these systems were stored in the same text configuration file. Eventually the *as_host* controller will modify AS configurations to enable scripting in the future, but it will do so through the new configuration library API. Once these configuration items were de-coupled, the PARB control menu was upgraded to use the new configuration library for storing its persistent settings.

After implementing the changes described above, a configuration submenu was populated with items to create, switch, and delete configurations. This was added to the *as_host* controller's main menu.

The 'toggle silent mode' item in the *as_host* menu is the only AS setting modified by the *as_host* controller. It is currently implemented by directly setting the AS configuration through the configuration library's API. This may need to be modified in the future as the scripting design is implemented.

3.2.3 Acoustic Subsystem

All components of the Acoustic Subsystem were upgraded to use the new configuration library interface. This required considerable effort, as it affected almost every part of the AS. For instance, the target file setting (both in the passive and the transient modes) was initially implemented using a file-system symbolic link to indicate the selected target file. This had to be changed to use the configuration library API before implementing the save-and-restore configuration functionality in the AS.

Once all configurable settings were upgraded to use the new configuration library, the items were added to the AS menus for creating, switching, and deleting configurations.

3.2.4 DARB

The DARB system software was easier to update to the new save-and-restore configuration functionality, as its original design was much closer to the new configuration library API. Once all configuration settings were, adding menu items to create, switch, and delete configurations as was done in the PARB.

3.3 GPS Logging Design Improvements

In order to realize a fully-functional MAS capability, an interface to query GPS information was required by the *as_host* controller. A NMEA parsing library was implemented under a separate Side-scan Sonar Mine-hunting Modules (SSMM) contract. It is capable of parsing text GPS NMEA strings using the following standards:

- GPGLL
- GPRMC[3]
- GPGGA

This existing NMEA parsing library was used to parse and store telemetry data received by the GPS into a SQLite database. This database is then queried when broadcasting main blast events.

[3] This is the format used in PARB and DARB systems.

Similarly, when the user downloads a textual echo log, the system queries both the MAS and GPS databases and use information from both sources to create the log file.

During the investigation, many issues were identified that need to be addressed in the new design. This included:

- In a glider or stealth buoy, the AS does not have access to GPS. These systems will not broadcast main blast events, but will need to provide the capability to download a detection log. This can be achieved by setting latitude and longitude fields to a NULL value.
- The detector itself is not coupled to the GPS logging facility. It will only log blast times, echo times, and signal processing values (SNR etc). Other software is responsible for combining detections and GPS information.
- The implementation was performed in stages to mitigate risk. In this first iteration, only the MAS mode required the GPS database. The other legacy subsystems will continue using their text file logging strategies. Once proven, the legacy software can be upgraded to use the new GPS logging functionality.
- The database stores text strings for easy retrieval, without needing to calculate checksums or deal with floating point round-off issues when converting between text and binary formats.
- The queries needed to be able to handle any missing data.
- The NMEA library was updated to allow for fields to be blank. Many GPS units will initially report NMEA strings in this fashion when they are not locked on.
- A new GPS logging application will need to write to an SQLite database via a library API. This API will also be able to request NMEA strings back for a specific times or ranges of time.

A GPS server simulator was written in Python for the purpose testing this new design. I broadcasts NMEA strings via a Unix socket. This will enable isolated testing when the rest of teh GPS design is implemented at a later date. The simulator:

- Broadcasts NMEA sentences to a Unix socket the same way the existing gps_server application does.
- Reads from an text file containing NMEA sentences, and writes sentences to the server's socket once a second.

DRDC has generated a NMEA referemce log-file containing GPRMC sentences, including periods of invalid (V) sentences. This will used to test with during future implementation.

3.4 Software testing

Akoostix used two approaches for testing the integrated software during this contract:

- A simple, well-defined synthetic test was used to verify software logic and data flow. A desktop application was developed concurrently with the embedded software and used to perform initial tests on mutually shared software.
- A more complex computer-generated synthetic data-set was used to simulate a pseudo-real-world situation. This was used to test the subtleties of the software including algorithm configuration parameters and processing performance.

These tests were design to meet the budget and time constraints of the projects and are described in (Section X and X). Two tests that were not performed during this project but are recommended in the future as the software matures are:

- Testing with well-understood real-world data.
- Testing with a variety of real-world data.

Each of these testing approaches is designed to detect different errors.

The software developed for the SMART Land buoy was integrated from pre-existing software modules and libraries. Built-in regression tests also exist for these software components and continue to be maintained and updated by Akoostix. These tests are designed to be easily verified and help ensure that algorithms are mathematically correct. Unit tests are maintained and executed nightly for multiple target platforms to help insure that software remains functional. Descriptions for all unit tests are beyond the scope of this report, but they may be found with the STAR software release described in Annex B.

Synthetic data was created to simulate real data, but with user-defined characteristics so that the expected result (e.g., bearing of arrival) is known. These data are useful for integration testing and for evaluating the fundamental characteristics and behaviour of simple and complex algorithms.

4 SBC Hardware Selection

The current hardware implementations of both the PARB and DARB systems (as well as Slocum glider science bay and the Stealth Buoy) use the Arcom Viper single board computer. This product has reached its end-of-life and must be upgraded as replacement boards become scarce. The Viper uses PXA255 processor board (ARM architecture) in a PC104 form factor. The Viper was selected to run the first generation of the sentinel algorithm since it was a low-power embedded board, and the project benefitted from experience gained during the RDS TDP. As the AS software has matured it has outgrown the Viper's capabilities.

Modernizing the processing board will allow the PARB to perform more processing inside of the buoy. Furthermore it is desirable that the unit be able to enter a low power state to preserve battery capacity on longer missions.

This section is organized as follows:

- Section 4.1 provides technical nomenclature used throughout the following sections.
- Section 4.2 is an overview of the requirements defined for a processor board replacement.
- Section 4.3 discusses the criteria used while investigating process board replacements.
- Section 4.4 contains the evaluation results of the three candidate boards which were purchased.
- Section 4.5 summarizes the results and final board selection.

4.1 PC/104 Nomenclature

This section introduces the terms used throughout the rest of the hardware selection discussion. More information on PC/104 standards is available at www.pc104.org.

Form Factors:

- PC/104 is electrically equivalent to the PC ISA bus, but mechanically designed for industrial applications.
- PC/104+ adds the newer PCI bus for higher bandwidth expansion peripherals.
- PCI-104 also uses the PCI bus, but removes the legacy ISA bus allowing designers to reclaim real estate required by the ISA connectors.
- PCI/104-Express includes both the PCI bus as well as the newer PCI-Express bus in an industrial form factor.

4.2 Requirements

The high level objectives of this task were to investigate hardware options and select appropriate solutions:

Planned Tasks:

- Derive the hardware requirements from the application requirements for each platform including:
    - Single-board computers - eight PARB and four DARB costing no more than $9000 in total.
- Perform a hardware search, and then contact vendors to compare the hardware requirements to vendor specifications.
- Present the results of the hardware search to the PA and TA, and provide a recommendation. Then obtain a decision on hardware purchases for the investigation.
- Purchase hardware, which is likely to include:
    - Two different SBCs.
    - Solid State storage for data collection.
    - Miscellaneous hardware such as power supplies, connectors, and standoffs for mechanical integration.
- Prototype and benchmark the required functions to determine how well the hardware meets the identified system requirements.
- Present the results of the hardware investigation to the PA and TA, and provide a recommendation. Then obtain a decision on the final hardware purchases.

After a planning phase, the team identified the following requirements for the candidate SBCs:

- Power Consumption:
    - Devices should support switching to a low-power state when not required. This may be a true standby mode, or a low-power state where some activity continues.
    - Wake the system for on-demand communication and tasks (desirable to wake on timers, GPIO and hardware interrupts, Wake-on-LAN/WiFi, Freewave).
    - Switch ancillary hardware (e.g. Freewave modem, GPS, sensor electronics) into a low-power state when not required. This will most likely require more GPIO options to control power electronics.
- Health Monitoring:
    - Monitor system-health parameters such as battery voltage.
- Communication:
    - Provide 802.11g or better wireless interface.
    - Require RS-232 port for Freewave modem.
    - Require RS-232 port for GPS.
    - May require RS-232 port for system terminal (this may not be required if Ethernet port exposed).
- Storage:

    - At least 64 GB (~64GB provides 24 hours at 16-bits for 4 channels at 96 kHz)
- Derived Requirements:
    - Require for a solution for triggering system events via the GPS PPS signal (both PARB and DARB).

4.3 Hardware Investigation

After initial meetings between DRDC and Akoostix it was decided to focus on embedded PC104 processor solutions using the Intel Atom CPU. Prior investigation of the Atom-based Eurotech ISIS SBC showed reasonable processing performance for modest power consumption. It was agreed that this would consume more power than the Viper (which was approximately 2 Watts).

The SBC investigation was conducted in parallel with the data acquisition replacement investigation. For this reason there was a shift in direction of the SBC investigation as it became apparent that a PC104+ or PC104/Express form factor would be required to meet specification of the data acquisition. The following two subsections document the initial selection of PC104 based boards, followed by the second round of PC104+ based boards (the parallel investigation of data acquisition boards is documented in Section 5).

4.3.1 Initial Investigation

The first investigation focused on the PC104 based Intel Atom boards. Some of the SBC boards identified in this section were not chosen once it was clear that the data acquisition board would require a PC-104/Plus interface. They are, however, documented here for completeness.

The following subsections review the PC/104 boards that were initially selected for prototyping.

4.3.1.1 Diamond Systems

Tim Murphy had initially identified the Diamond Systems Aurora PC/104 SBC as a potential solution. Its features include:

- Intel Atom Z510/Z530 CPU.
- 1-2 GB DDR2 SDRAM.
- PC/104 form factor.
- IO features:
    - 8 GPIO
    - 4 USB
    - 4 RS232
    - SATA (via on-board PATA converter)
- Passive cooling

- Extended temperature range available.
- Although they claim to support Linux, drivers were not available on the public website to evaluate the level of support.
  - Further technical questions regarding Linux support, power management, GPIO were sent to the salesperson to get answers from their engineering staff. No response was ever received.
- Diamond noted that their current stock was depleted with more boards being manufactured. Lead time was estimated at 4-6 weeks. Tri-M (Canadian Distributor) had one bare factory refurbished board in stock but would require development kit components from Diamond in order to prototype.
- http://www.diamondsystems.com/products/aurora

This board was not investigated further since it was PC/104 only.

4.3.1.2 Parvus

Parvus is part of the Eurotech group of companies and sells several embedded solutions. Eurotech acquired the original manufacturer of the Viper (Arcom), and manufactures the Atom based ISIS board which was evaluated last year without success.

- Contact was made with sales to investigate CPU-Z530 PC/104-Plus board.
- The boards are extremely expensive. The development kits were nearly $4000 USD with bare boards close to $2000 depending on options chosen.
- Their CPU-Z530 board does not support power saving modes.
- http://www.parvus.com/product/overview.aspx?prod=CPU-Z530

No further investigation was done as it was both cost prohibitive and unable to support power saving.

4.3.1.3 Lippert GmbH (through Tri-M)

Tri-M is an embedded distributor in British Columbia carrying an extensive list of product lines. They provided excellent sales support through the RDS TDP, including their HE-104 power supplies. They suggested the Lippert Cool Literunner ECO, stating that this supplier makes excellent products with great support. This product features:

- Intel Atom Z510/Z530 CPU.
- PC/104 form factor.
- IO features:
  - 8 GPIO lines
  - 7 USB
  - 2 RS232

    - LPT parallel port
    - 2 SATA
    - Mini-PCI expansion
    - HD audio
- Passive cooling.
- Extended temperature available.
- Lead time for one board is estimated at 2-3 weeks, they test to order.
- Technical support was emailed questions regarding power savings, Linux support and BIOS boot time. An answer was received within one day with positive answers.
- Excellent manual.
- Willing to work with customers to make custom BIOS solutions. This may be useful in the future to remove unnecessary BIOS components for faster booting.
- Price approximately $760 for a development kit with heatsink and cables.
- http://www.lippertembedded.de/de/lipperts-cool-literunner-eco.html

Although this board was not chosen due to its form factor, Lippert will be revisited in Section 4.3.2.

4.3.1.4 ADL Embedded

This company offers a PC/104-Plus Atom solution, the ADLS15PC. Its features include:

- Intel Atom Z510/Z530 processor.
- PC/104-Plus (includes ISA).
- Power management.
- IO features:
    - 8 USB
    - 2 RS232
    - Parallel
    - PATA HDD interface
    - HD audio
- Price approximately $1150 USD for a development kit.
- http://www.adl-usa.com/products/cpu/datapage.php?pid=ADLS15PC

4.3.1.5 Other

A brief capability review was performed on the following Arm processor based solutions:

- Beagleboard/Beaglebone
- Pandaboard
- Gumstix

All of these solutions were too limited to meet the requirements for the PARB. For simplicity, the team preferred to use a single platform for both the PARB and DARB despite the processing requirements of the DARB being much lighter. These solutions could be an option for a future low power DARB but would require further investigation regarding their I/O capability.

4.3.2 PC-104/Plus Investigation

As mentioned, the data acquisition hardware selection occurred in parallel with the initial PC/104 based investigation. Once the data acquisition hardware began steering towards a PC/104-Plus based solution, the focus of the SBC hardware selection was changed to PC/104-Plus based solutions.

The following subsections document the results of this second-phase of the investigation.

4.3.2.1 ADL Embedded

Although initially selected as a PC/104 solution, the ADLS15PC investigated in Section 4.3.1.4 was retained as a viable PC/104-Plus solution.

4.3.2.2 Kontron

Although initially contacted for PC/104 based solutions, the change in focus to PC/104-Plus SBCs occurred before Kontron had been able to suggest a solution. Once they were made away of the change in focus, their salesperson suggested the MSM200X:

- Item Atom Z510/Z530 CPU.
- PCI-104/Express form factor (although the stock board does not include PCI connectors on both sides of the board as is standard, this can be ordered as an opion).
- 1-2 GB DDR2 SDRAM.
- I/O features:
    - 4 USB
    - 4 RS232
    - LPT Parallel port
    - Mini PCI-Express card-slot
    - HD Audio

    - Ethernet
    - Optional GPS
- The Manual on product website is still in draft. Salesperson mentioned the product CD included with the board was more up to date.
- Price approximately $1150 USD for the development kit.
- http://us.kontron.com/products/boards+and+mezzanines/pc104+sbc+and+peripherals/microspace+pc104+cpus/msm200xxpxu.html

4.3.2.3 Lippert GmbH revisited

After the change in form factor requirement the salesperson at Tri-M was again contacted to suggest a new PC/104-Plus solution. They again suggested a Lippert product:

- Cool RoadRunner 945GSE.
- Intel Atom N270 CPU.
- 1-2 GB DDR2 SDRAM.
- IO features:
    - 8 USB
    - 2 RS232
    - 2 SATA
    - HD audio
    - Ethernet
- Price $554 USD per bare board, plus $84 USD for a cable kit and heatsink. Volume discounts available.
- As was noted with the initially selected PC/104 Lippert solution, the manual seems quite comprehensive.
- http://www.lippertembedded.de/en/lipperts-cool-roadrunner-945gse.html

The lead time of this board was estimated by Tri-M to be approximately the third week of March 2012.

4.3.3 Conclusion and Ordering

A team meeting was held to evaluate the available options. It decided that three boards would be ordered to evaluate against each other:

- Lippert Cool RoadRunner 945GSE
- Kontron MSM200X
- ADL ADLS15PC

The General Standards 16AISS8AO4 data acquisition board was being purchased in parallel with this task (see Section 5). Since the 16AISS8AO4 board includes 16 GPIO lines the team was not concerned that the Lippert SBC did not include either GPIO or an LPT parallel port (which can be used for GPIO).

All three solutions include at least two RS232 ports (the minimum), but additional RS232 ports can be added with inexpensive USB devices. None of the solutions include WiFi but these can be added in the future as well.

Additionally the team agreed that one SATA and one PATA (IDE) solid state hard disk would be required for prototyping. These devices were sourced from CanadaRAM. The models purchased were:

- Kingston SSDNow V200 128 GB (SATA)
- Transcend TS128GSSD25-M 128 GB (PATA)

Some compromises were made when ordering the SBC boards. In all cases, extended temperature boards were available but would have added undesirable manufacturing lead-time. The ADL Embedded board was only available in-stock with an 8GB on-board flash option. 8GB is more than required for an operating system installation, but different flash sizes would have increased ordering lead time. The in-stock boards from Kontron do not include a PC/104-Plus connector on the bottom of the card. This made mechanical integration into the DARB/PARB more difficult since the SBC would need to be at the bottom of the PC/104 stack. They do offer a variation of the board featuring a bottom connector, but that variation would also increase lead-time, so this option was not chosen for prototyping.

During the purchasing process, Kontron support became an issue. After the salesperson was contacted for a status update, it was indicated that the board customization was causing a delay. When asked why the board was being customized he became unresponsive again. After several attempts to contact the representative, we attempted to contact the sales manager. A voice mail message was left, but he did not return the phone call. Finally, after several more voice mail messages, the original salesperson emailed an update contradicting his earlier communication. He indicated that it was not customization holding up the board, but the integration of the heat-sink.

Once the Kontron board finally arrived at Akoostix (a month after it was ordered), it had a bottom mounted PC-104/Plus connector (which was not requested) and did not have a heat-sink (which was requested). In order to work around this in a short time frame, an external PC cooling fan was used to dissipate heat from the include heat-spreader as the heat-spread alone is not sufficient as a passive cooling device.

4.4 SBC Evaluation

Each board was loaded with an operating system, configured, and then ran through several pre-defined evaluation tests which the team had agreed upon. The results of these tests can be found in Annex C. The remainder of this section provides information on how the operating system was set up for the SBCs.

4.4.1 Operating System

A minimal installation of Gentoo Linux was used on all SBCs. Gentoo was chosen for its extreme configurability and customization, making it very easy to install only those packages required to test the system, as well as customizing those packages. For instance, the package manager was used to configure the FFTW package to include support for single-precision floating point numbers, accelerated using the SSE instruction set, while disabling support for the OpenMP parallelization library. This level of control lends itself well to a tailor-made operating system with minimal dependencies. Furthermore, global compilation options can be chosen to increase system performance as will be outlined in Section 4.4.3.

The following system services were installed and included as part of the system boot:

- microcode_ctl – This is used to update the CPU firmware on Intel CPUs to the latest release, allowing Intel to correct CPU firmware bugs after product release.
- net.eth0 – This starts Ethernet networking (as a background process using the ifplug daemon).
- alsasound – This is the soundcard service.
- sshd – This service handles secure remote login over Ethernet.
- ntpd – This is the Network Time Protocol Daemon used for synchronizing the computer clock to an external source (time server, GPS, etc.)
- vixie-cron – This is a cron job server used to periodically start timed jobs.
- syslog-ng – This is the system logging facility.

4.4.2 Linux Kernel

The Linux kernel used for testing was version 3.2.11 with the Gentoo patch-set installed (this package is named gentoo-sources within the package manager). Testing began with the 3.2.1-r2 kernel, but as is outlined in Annex C the Kontron SBC had issues with the 3.2.1 version.

The Linux kernel was configured to include only those drivers necessary for the specific hardware in order to reduce size, and increase boot time.

4.4.3 GNU Compiler Optimizations

Some reports have shown that by enabling the GCC flag "–march=atom" on supported GCC versions (4.5 or later) an increase floating point performance by more than 20% is possible. For this reason, all system packages and test software were compiled with this flag set. A simple before/after benchmark performing FFTs showed an actual speed increase of over 30% (performing 8192 point FFTs over 10000 iterations). It is expected that similar speed increases will be realized throughout the system as SPLIB performs most of its processing with floating point numbers.

4.5 SBC selection

Once some free-play testing of each board was performed (configuring hardware, kernel options, setting BIOS) each board was used to run a set of tests which were agreed upon by the team. The results of these tests can be found in Annex C.

It is difficult to draw a simple conclusion from the tests regarding the best SBC choice. Each had strengths and weaknesses. It is suggested that the testing data be reviewed by the team after reviewing this report and making a team decision.

The major differences are summarized as follows:

- The ADL SBC does not support suspend-to-RAM (sleep) making it impossible to implement some desired low-power requirements.
- The Kontron SBC showed some integration issues which, if not correctable, would be extremely inconvenient:
    - Graphics subsystem corruption when GMA500 drivers were installed, resulting in system lock-ups.
    - Seemingly random boot problems when loading the Silicon Image SATA device driver.
    - The VGA output does not resume after a suspend-to-RAM cycle until the system is rebooted.
- The Lippert SBC consumes the most power, however this measurement should be re-evaluated after removing the ATX power supply that was required to power this board (and present in the other two SBC power measurements).
- All functions of the Lippert SBC worked as advertised. It included a very comprehensive user manual.
    - This board booted significantly faster than its peers. OS shutdown was roughly the same in all three cases.
    - This is also the cheapest SBC of the three.
- The sales team at Kontron was extremely difficult to deal with resulting in a long shipping delay and not receiving exactly what was specified at order time.
    - The Kontron manual is marked DRAFT. Sales alluded to an updated manual on the CD included with the SBC but not CD was delivered.
- The SATA Solid State Disk purchased (Kingston V200) draws a considerable amount of power when compared with the PATA (Transcend). It may be worth investigating other SATA SDD solutions for system integration.

5 Data Acquisition Hardware Selection

Along with the selection of a new single board computer, new data acquisition hardware was investigated, purchased and prototyped under this call-up. The motivation was two-fold:

- The existing Acces IO A/D solution for the PARB is not capable of performing simultaneous sampling on multiple channels.
- The existing DARB uses the Viper's soundcard as an output, and a playback system with better fidelity was desired.

Furthermore, the Acces IO driver and SPLIB module was written specifically for integration with the Viper hardware on a single A/D channel only. Although the board itself has other capabilities (multi-channel A/D, multi-channel D/A, GPIO), this was not implemented to date, and would require porting and updating the existing driver software. This was seen as less favourable than choosing a replacement board with existing software drivers.

The goal of this task was to identify hardware requirements from the user requirements of the PARB and DARB systems, preferably with a complete Linux driver package to reduce integration time.

The remainder of this section is organized as follows:

- Section 5.1 discusses the criteria used for selection of suitable hardware.
- Section 5.2 contains the results of a detailed hardware survey from several vendors.
- Section 5.3 outlines how the board was selected.
- Section 5.4 discusses how the board was evaluated and the results.

5.1 Requirements

The high level objectives of this task were to investigate hardware options and select appropriate solutions:

- Derive hardware requirements from user requirements for each platform including:
  - A/D boards – eight are required for PARBs.
  - D/A boards – four are required for DARBs.
  - A/D and D/A boards are to cost no more than $4000 in total.
- Perform a search and contact vendors to compare hardware requirements to vendor specifications.
- Present the results of the hardware search to the PA and TA, provide a recommendation and obtain a decision on hardware purchases for the investigation.
- Purchase hardware which is likely to include:
  - At least one D/A unit

    - At least one A/D unit

- Prototype the required functions to determine how well the hardware meets the identified system requirements.
- Present the results of the hardware investigation to the PA and TA, providing a recommendation and obtain a decision on final hardware purchases.

During execution, we discovered that there would not be sufficient budget to purchase hardware for eight PARB and four DARB units. The purchasing scope was reduced to a single A/D solution and a single D/A solution for prototyping.

The data acquisition investigation began with an internet search for products meeting the following search criteria:

- A/D (Input)
    - 16 bit resolution or greater
    - Minimum 4 channels, more are desirable
    - Minimum 96 kHz sustained sample rate per channel
    - Simultaneous sampling
- D/A (Output)
    - 16 bit resolution or greater
    - Single channel, more acceptable for future expansion
    - Minimum 96 kHz sample rate
- General requirements
    - Form factor
        - USB (preferred as it allows for the most SBC selection freedom, and would potentially be able to be used with a laptop based solution)
        - PC104
        - PC104+
        - PC104/Express
    - Linux operating system support
    - Low power consumption and/or ability to reduce power when not in use for extended periods of time.
    - GPIO desirable but not a hard requirement.
    - It is desirable to have a single solution for ease of integration, but separate A/D and D/A solutions would be considered.

5.2 Hardware Investigation

Once requirements had been determined, a hardware investigation began. This investigation was a combination of both of the teams, experience with known vendors, combined with web searches for new vendors. The following subsections list the options which were explored.

5.2.1 Pro Audio Sound Devices

The Linux kernel contains many drivers for pro-audio sound interfaces, which was used as a starting point to investigate these supported devices. However, after a short investigation it was determined that while some devices met many of the criteria, most audio grade devices have poor low frequency response (below 20 Hz). Furthermore, these devices are more geared towards desktop computer use and both power consumption and form factors became risks. It was determined quickly that effort would be better used focusing on general purpose data acquisition solutions.

5.2.2 Acces IO

Acces IO's technical salesperson was contacted by phone to discuss a solution to the requirements identified. The solution they suggested was the USB-AIO16 card. This product's highlights include:

- USB interface.
- A Linux driver is freely available based on "libusb" which is included with virtually all Linux distributions. This was downloaded and the driver and test programs compiled cleanly on a modern Linux distribution.
- 16 single ended inputs, 16 bit, up to 500 kSa/s aggregate sample rate.
- Extended temperature and ROHS available.
- Approximately $800 USD depending on options (DC-DC converter, ROHS, etc.).
- The board itself can be mounted in a PC/104 stack with a USB cable between boards, making mechanical integration with PC/104 solutions easier.
- They are willing to work with customers on software drivers if small customizations are needed.
- Cons:
    - It does not perform simultaneous sampling.
    - The D/A does not meet specification (only 4 kHz).
    - To mitigate this, the existing PC104 Acces IO card could be used D/A (it can reach 100 kHz), but this would require writing the driver code to integrate analog outputs.
- See http://accesio.com/go.cgi?p=../usb/usb-aio16-16a.html for more information.

5.2.3 Diamond Systems

While investigating Diamond's SBC solutions, their salesperson was asked about data acquisition products. They offered a few solutions but nothing that could meet the requirements without more compromises than the Acces IO solution. In particular:

- The supplied Linux drivers on their website are old and would require effort to support modern kernels.
- No products provide simultaneous sampling.
- 4 channels at 100 kHz would be pushing the limit of their fastest current product. The salesperson mentioned that two boards could be used in tandem but this might introduce more problems with synchronization.
- A new product was due in the coming weeks, but this was considered too risky.

5.2.4 Reach Technologies

Reach Technologies was contacted at DRDCs request to explore any data acquisition solutions they might have. Unfortunately, they have no off-the-shelf products that can meet requirements, and although they expressed interest in working on a solution in the future they had no resources at the time. Since timelines were extremely short and this would require designing hardware, software, and testing, it was determined that it was not feasible to explore this further.

5.2.5 National Instruments

Tim Murphy suggested that National Instruments may have a solution as they make dozens of data acquisition products. After a brief web search and contacting a salesperson, two possible solutions were identified: USB-9215 and USB-6356. These solutions could meet the specification defined (including the most difficult requirement, simultaneous sampling). After an email exchange with a technical person, it became apparent that Linux drivers would be an issue. Their Linux drivers are outdated, and only support a subset of their products (not including those being investigated), and are developed mainly for workstation-based Linux kernels (SUSE and Redhat) running LabView.

The best that National Instruments could offer was a Measurement Hardware Driver Developer Kit (or MHDDK - hardware level documentation manuals), but it is limited to a manual and support via web forums. It was determined that this would be costly to develop a driver from the ground up and only recommended if all other options were exhausted.

5.2.6 Tri-M Data Acquisition Investigation

After contacting Tri-M sales for the SBC investigation, they also mentioned that they would investigate data acquisition solutions. David Costin became the technical contact for this and was provided the data acquisition requirements. After some investigation David proposed the following two solutions to investigate further:

- Apex Embedded offers two separate PC/104 cards (one performing analog input, the other analog output) which could (together) meet specifications. As mentioned in Section 5.2 a single board solution for both A/D and D/A is desirable.
- Acquitek offers the PC104P-ADADIO, a PC104/plus card which could meet all requirements in one card.

After speaking with David on the phone about these solutions, he felt that the PC/104 bus would leave little room for future expansion so we would focus only on the Acquitek solution.

The Acquitek product website did not offer much detail such as technical manuals or drivers. David contacted Acquitek to investigate further, but he determined that this product is now manufactured and sold by General Standards who were already contacted to investigate data acquisition solutions (see Section 5.2.7). Unfortunately this exhausted both options that Tri-M had identified.

5.2.7 General Standards

Tim Murphy mentioned that General Standards products were used on another DRDC project with success. They were contacted and forwarded the requirements list in order to suggest solution(s).

James Barksdale, an engineer at General Standards, was contacted by phone to discuss their solutions. He suggested that either the 16AISS8AO4 or the ADADIO product would meet the specification. They are both single-board solutions for both A/D and D/A, and also include GPIO. The 16AISS8AO4 solution is more flexible as it offers software selectable voltage ranges for both A/D and D/A while the ADADIO ranges must be specified at time of order.

The 16AISS8AO4 specifications are:

- Native form factor is PMC, but conversion boards are available for both PC/104-Plus and PCI-Express for initial development both with an embedded computer and a desktop PC.
- 8 differential A/D converters operating at up to 2 MSa/s each.
  - Simultaneous sampling.
  - 2.5, 5 and 10 volt input ranges.
  - 256K sample hardware FIFO buffer.
- 4 D/A converters operating at up to 2 MSa/s each.
  - Simultaneous conversion.
  - 2.5, 5 and 10 volt output ranges.
  - 256 sample hardware FIFO buffer.
- 16 General Purpose IO lines.
- Mature Linux device driver available.

The Linux drivers were downloaded and evaluated before purchasing hardware. The drivers are kept reasonably up to date when compared to other vendors, and only required slight changes to their build system to compile with the latest 3.2.x Linux kernels.

While discussing the project with James, several future possibilities for better system integration are:

- Optionally ordering boards with only four input channels to reduce power.
- Laying out a native PC/104-Plus form factor to reduce size. General Standards is willing to absorb the cost of this on an order as little as five boards. James indicated that this could be done without changing the driver interface. This means that no effort would be wasted prototyping with the current form factor.
- Adding a firmware/driver option to reduce board power consumption by turning off analog circuitry when not in use.

5.2.8 SeaLevel

SeaLevel also offers several data acquisition solutions in various form factors, and includes full Linux support. They were contacted and sent a list of requirements but they did not have anything to offer. Unfortunately, their systems are designed more for industrial use, such as monitoring temperatures a few hundred times a second.

5.3 Selection

The team was presented with the results of the hardware investigation. Ultimately, the only two products which met specifications without compromises were the two General Standards products (16AISS8AO4 and ADADIO). After a discussion, it was decided that the 16AISS8AO4 would be ordered since it was more flexible than the ADAIO with respect to software selectable input and output voltage ranges.

The board was ordered in its current PMC form factor with conversion boards to PC/104-Plus and PCI-Express for rapid prototyping. A signal breakout board and interface cable was also purchased.

5.4 Evaluation and Conclusions

After receiving the hardware, the board was installed in a desktop computer using the PCI-Express convertor. General Standards' driver and test suite were compiled and used to test and evaluate how the driver software is programmed. The A/D, D/A and digital GPIO functionality of the board all work as advertised.

After initial free-play testing, an SPLIB input module was designed and implemented around the provided A/D software interface. Data is collected from the hardware driver and buffered in a separate thread transparently. This reduces the need for the SPLIB module to synchronously service the hardware buffer as it fills, allowing processing to proceed in parallel with data collection. This concurrency is hidden from SPLIB buffer consumers, complying with SPLIB's design. The resulting module will be used for integration with existing AS processing software.

General Standards reference documentation and included software examples made this task relatively straight-forward. No unexpected pitfalls were encountered while developing or testing the SPLIB module, and the software worked as expected once integrated into the AS software.

Although not completed under this work, it is expected that implementing software modules to interface the analog output and GPIO functionality of the 16AISS8AO4 would not pose any large risks.

6 Complementary computing hardware selection

Examining or modifying a sonobuoy, its software and data output in the field is prohibitive. If a sonobuoy ceases function, is damaged, misconfigured or otherwise faulty, the error may remain undetected until the device can be examined at a computer, often located on a main vessel rather than the deployment craft. The ability to examine sonobuoy output on location in the water and immediately reconfigure devices would save valuable experimental time and improve the quality of collected data. To this end, a variety of rugged and touch-based solutions were investigated for the purpose of onboard data analysis and remote interaction with sonobuoys.

6.1 General touch screen considerations and requirements

A touchscreen or tablet computer for sea use should be selected taking into consideration several properties. It should be

- Compatible with software intended to be used on the vessel
- Durable under exposure to humidity, moisture and physical shock
- Usable by touch, whether hand, glove, pen, stylus or other pointing device

The compatibility of OPD with touch-screen technology was evaluated on each device. OPD was successfully tested on both the ELO touch-monitor and the Getac V200. A sample WAV data file was loaded and displayed. The crosshair, harmonic divider, banding, annotation selector, WAV extractor, Doppler and Periodic Event tools were tested with the touch-screen on each device. Each tool performed as expected and the touch-screen interaction presented no special difficulties for these tools. To satisfy durability requirements, most rugged device manufacturers claim compatibility with MIL-STD-810F or MIL-STD-810G American military standards for device ruggedness. These standards define a baseline suite of tests including drop tests, temperature change and duration, humidity, sprayed water and vibration. The degree to which each device passes these tests varies and it is sometimes difficult to ascertain which experiments a manufacturer performed and to what extent. The standards specify minimum testing requirements and some tests may be omitted by the manufacturer.

Touch-screens are implemented using different technologies. These include resistive, surface acoustic wave, capacitive, infrared, optical, dispersive and acoustic pulse recognition. Many touch-screen technologies can be sensitive to moisture, rain, insects and dust. For extended sea use, resistive touch-screens were identified as the most robust solution, particularly for use with gloves.

Touch-screen interaction was tested under GNOME 3 using the 1547L monitor and the Getac V200 device. Normal operations are useable including dragging windows, sliding scrollbars, accessing the secondary menus (right clicking) and making selections. To enable right-clicking, the option "System Settings > Universal Access > Pointing and Clicking > Simulated Secondary Click" should be enabled. The user may then produce a secondary click (right click) by tapping-and-holding. An onscreen keyboard is available in GNOME 3 via the Accessibility menu or by installing "gok" or "onboard" for extended functionality. (It is essential to disable screen locking in the absence of a keyboard as the onscreen keyboards do not currently interact well with the screen locking application.)

6.2 Touch screen monitor selection

Devices from ELO TouchSystems, Keytec, Planar and Unytouch were considered during the sourcing process. After comparing the devices' advertised qualities including price, dimensions, weight, resolution, power usage, durability, driver availability and touch technology type, the ELO and Keytec products were selected for evaluation. The selected units are described in the following sections.

Each touchmonitor was connected to a 64-bit Debian Linux system running Linux 3.2.0 and X.org 7.6 with X server version 1.11.4.

6.2.1 ELO 1537L Touchmonitor

The ELO 1537L is a 15" touch monitor featuring a watertight resistive touch-screen seated in a metal open frame with integrated RS232 serial and USB controllers. The power and screen control buttons are positioned on the top back of the monitor frame. All external connectors are located under the bottom edge. Additional components include the USB cable for the touch-screen, VGA cable and the AC power supply. The base price is $464.89.

The 1537L was connected to the Linux system using the VGA cable. The display was recognized by the kernel and the X.org display server and was responsive to touch. Both the kernel and the X server have built-in, maintained drivers for this touch device.

The screen requires calibration before it can be used and this is done using the "xinput_calibrator" tool. After prompting the user to tap onscreen tarets, xinput_calibrator prints a standalone xorg.conf configuration section which is placed in a text file in "/etc/xorg.conf.d". Once the correct calibration is obtained for the current resolution, the device registers touches with pen or finger and was properly responsive to tapping, double tapping, dragging and tap-and-hold.

6.2.2 KeyTec KTLC-121OT

The Keytec KTLC-121OT touch-monitor is a 12.1" LCD monitor with a built-in MagicTouch touch-screen. It is an open frame device sealed with duct tape and a metal chassis and has six buttons positioned on the side for managing the onscreen display and toggling the power. It comes with a separate USB controller which converts touch-screen output to USB output as well as a VGA cable and power supply. The base price is $598.50.

The monitor and USB controller were connected to the Linux system and both were detected by the kernel and X server. No touch functionality was immediately available. The controller was recognized as an input device by the kernel but the X server indicated that a built-in driver was unavailable. Although there is no built-in driver for this device in the server, the manufacturer's CD and website contain Linux drivers. However, these drivers target the old X.org driver API. To use the KTLC-121OT with Linux, it would be necessary to use an older distribution such as Ubuntu 10.04 or Debian 5.04. The manufacturer was contacted regarding the availability of newer drivers and they indicated they were in the process of developing a new driver for this device which would soon be available.

6.3 Touch screen laptop selection

6.3.1 GETAC V200

The Getac V200 is a hybrid laptop and tablet with a swivelling convertible display. It is a ruggedized device with a magnesium alloy case and rubber corners and edges to absorb impacts. It was subjected by Getac to MIL-STD 810G tests evaluating robustness to dropping, vibration and water resistance. It includes a touch-screen and keyboard. The base price is $3400.00.

Ubuntu 11.04 and 11.10 were installed for this evaluation. Neither version of Ubuntu was immediately compatible with the V200’s touch-screen. The device was improperly recognized as a touchpad. Upgrading the kernel to Linux 3.2.0 brought in the necessary device definitions to properly identify and categorize the touch-screen. After this upgrade, the touch-screen functioned properly with the X display and responded to taps, double taps, dragging and tap-and-hold for “right clicking”.

6.4 Touch screen tablet computer selection

6.4.1 Panasonic ToughBook CF-H2

The Panasonic ToughBook CF-H2 is a rugged, portable tablet device with an Intel i5 mobile processor, integrated Wacom touch-screen, dual batteries, Ethernet adapter, wifi and optional GPS. The system is a self-contained unit with a stylus for touch interaction. It uses a resistive display so it is also possible to use fingers or gloves. The base price is $3250.00.Ubuntu 11.10 was installed on this device for testing. Installing Ubuntu Initially the touch-screen was non-responsive under Linux. Although Wacom devices are supported by the kernel and the X input layer, it was determined that the touch-screen device ID is not currently recorded in the kernel or in the X.org Wacom input driver. The proper device ID was added to the Wacom kernel module and to the X.org Wacom input driver. The kernel and input driver were recompiled and the touch-screen worked properly.

A setback for this device under Linux is that the CPU frequency scaling module (“CPUfreq”) in Linux kernels 3.0.0 and 3.2.0 doesn’t correctly support this device. The module locks the CPU frequency to the lowest possible value (800MHz) and the system performance degrades noticeably. This was resolved by blacklisting the CPUfreq driver at boot time. The cost of this solution is that, with CPU frequency scaling disabled, the power usage of the device will be less effective.

6.5 Conclusion

The recommended devices based on this investigation are the ELO 1537L touchmonitor and the Panasonic ToughBook CF-H2. The prices are better than the competitors. Each of these have tried and tested brand name touch drivers (Wacom and ELO Touchsystems) which have long been supported by the Linux kernel and X.org. Whether using Debian, Ubuntu or another Linux distribution, GNOME 3 provides a screen and pointer-oriented interface for touchscreen interaction supporting all basic and necessary operations for interacting with OPD and manipulating data.

7 Recommendations

Ideas for improvement of both the software and algorithms are frequently generated during a contract. This section documents those recommendations, capturing both a high-level description of the suggestion and a rough order-of-magnitude (ROM) estimate of the required effort. Each section contains a summary table itemizing the ideas and the related ROM. Often a reference to a Jira ticket, where more detailed information is available, is also provided.

ROM estimates are generated after a preliminary review of the requirements and therefore could have significant variance. They are broken down into the following categories:

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

Where a more accurate ROM is available it may be provided in brackets.

7.1 Main application or suggestion group

Table 1 is a summary of selected issues being tracked in Jira for future upgrades to STAR. These issues are further explained in the following subsections.

Table 1: Jira issues tracking future enhancements with regards to the call-up

Jira ID	Type	Summary	Estimated Effort
LAND-23	Defect	Clean cancel of ZMODEM transfer	Small (1 day)
LAND-36	Defect	NTPD limitation on updating the system time	Small (2 days)
LAND-37	Defect	Implement message passing communications between the child processes	Medium (5 days)

Suggested improvements for future work:

Run software as a normal user (as opposed to root). The only known problem to solve is how to shutdown/restart the system from software as a regular user.

From PARB requirements:

- Go into low-power standby
- Wake via Freewave
- Wake at time
- Provide an option to download/upload saved configurations
- Provide an option to program the date-time to switch to a save configuration
- Provide an option to extract a segment of data (start-time/duration or time from start of file/duration) and download via Freewave (or pull via wifi)
- Provide an option to connect remotely (via wireless modem) for burst uploads and downloads
- Provide the option to connect remotely (via wireless modem) and stream the current A/D output via an interface that is (or could be) compatible with OPD
- Beamform multiple sensors

From DARB requirements:

- Provide health monitoring (e.g. battery voltage)
- Go into low-power standby
- Wake via Freewave
- Wake at time
- Provide an option to download/upload saved configurations
- Provide an option to program the date-time to switch to a save configuration

General

- Implement an SPLIB module to utilitize the D/A on the General Standards board. Exact available sample rates for DARB will need to be calculated and agreed upon since the boards oscillator can general integer divisions of 40 MHz (optionally the device can be ordered with other frequency oscillators but will always be limited to integer divison).
- Implement the sleep function of SSD drives. This will require reorganizing the PARB directory structure so that the drive is not needed while not sampling.
- A new PPS driver solution will be needed for the DARB. Multiple solutions have been identified, but require testing and integration:
    - ◆ The existing kernel driver that was written as part of the DARB software could be modernized for the new hardware and kernel selected and used with an unused IRQ from the ISA bus if the selected board has this resource available.

    - The Linux ppdev driver can be used to trigger software events from parallel port interrupts. Tim Waugh's online parport book contains the details necessary to prototype this.
    - The Linux kernel now contains PPS reference drivers which can be used to interface with parallel port (if the selected board has one) and serial ports (which will require electrical conversion circuitry).
- Network the buoys so that they can collaborate.
- Review the results of this report and select the replacement SBC from the three candidates. Once selected, final hardware integration can begin including:
    - Specifying and designing (or purchasing) a DC power supply capable of generating the require voltages for the SBC and SSD hard disk. Special attention to inrush current will be necessary as all candidate boards can draw considerable currents during power-up.
    - Choose and implement a hardwired port solution for PARB and DARB units allowing for local upgrades and data transfers. It is recommended that an Ethernet connection be used for its higher bandwidth than a serial port. This was recently done with success on the legacy PARB and DARB systems.
    - Implement necessary software modules to complete hardware integration. This will include:
        - An SPLIB module to interface with the D/A for the DARB.
        - Perform an in-lab sample-by-sample waveform validation of A/D and D/A SPLIB modules. This will involve choosing suitable known signals to sample/reconstruct and modifying driver code to inject sentinel values at specific interval for validation. This task will require high precision function generation equipment and digital capture oscilloscopes.
        - Software integration and testing of GPIO functionality for the purpose of interfacing with the PARB preamplifier board, and DARB amplifier remote power switching.
        - A software module to capture the generation of hardware PPS events to trigger DARB playback in synchronous mode.
    - As the above software development stabilizes, a new development environment (including OS and system software installation procedures) must be generated and documented.
    - CPU and memory profiling and tuning of the PARB processing modes must be performed.
- Perform in-water full integration including all analog electronic components (amplifiers, transducers, projectors, etc.)
- Perform an investigation of short range, high-bandwidth wireless data transfers from the PARB. This will include purchasing of a suitable USB WiFi device and outboard antenna.

- Upgrade the wave-train definition application to allow users to specify replica waveform types and their parameters. These definition files will be transferred to DARB and PARB units (instead of generated waveforms) for playback and processing, thus reducing the need to transfer large data files.

8 Summary and conclusions

This project was very challenging both in schedule and the broadness of scope. Despite this, a large number of successful outputs were achieved. Upon project completion the following tasks from the project objectives were successfully completed:

- Development of in-buoy MAS processing that can be used to offer new surveillance options, including: increased sensor life, increased sensor range from primary platforms, and in-buoy processing to reduce interface bandwidth requirements.
    - As a by-product, an SPPACS command line application to perform multistatic post processing was created for testing/verification purposes. This application can further be developed as a viable MAS batch processing component to STAR/IDL.
- User interface improvements to the PARB and DARB, making them more accessible to DRDC scientists and technicians. Specifically:
    - Simplification of the buoys' menu system, making it more user friendly and ensuring that commonly-used functions are readily available.
    - The ability to add, delete, edit, and switch between named configurations was added to the DARB, PARB, and AS menus. This feature allows a user to create multiple processing configurations in advance, allowing novice users to easily switch between verified settings without the necessity of changing a lot of advanced parameters. The configuration options can later be automated to run on a schedule when the buoy is not in contact with the controlling platform.
- Selection and testing of new candidate SBC solutions that:
    - Allows for common hardware between PARB and DARB units, simplifying development, reducing requirements for spares, and simplifying maintenance.
    - Provides additional processing power for advanced processing and provides increased storage, allowing for longer deployments and/or higher data rates.
- A suitable replacement for data acquisition hardware in both PARB and DARB units was identified and prototyped.
- An investigation of touch-screen technology was conducted for two purposes:
    - Remote connection for control and monitoring of PARB and DARB, using tablet computers or laptops with touch screen interfaces that could be used from small watercraft.
    - Creating suitcase style DARB or PARB units using the chosen SBC with the addition of a touch-screen monitor for a graphical environment. These units are intended to be used as portable deck-units on trials for workups and prototyping.
- A list of remaining tasks in order for the systems to become trial ready was created.

As is typical for an investigative project there were multiple tasks identified for future work. The complete listing can be found in Section 7, but high priority items are noted in the following list:

- Review the results of this report and select the replacement SBC from the three candidates. Once selected, final hardware integration can begin including:
    - Specifying and designing (or purchasing) a DC power supply capable of generating the required voltages for the SBC and SSD hard disk. Special attention to inrush current will be necessary, as all candidate boards can draw considerable currents during power-up.
    - Choose and implement a hardwired port solution for PARB and DARB units, allowing for local upgrades and data transfers. It is recommended that an Ethernet connection be used for its higher bandwidth than a serial port. This was recently done with success on the legacy PARB and DARB systems.
    - Implement necessary software modules to complete hardware integration. This will include:
        - An SPLIB module to interface with the D/A for the DARB.
        - Perform an in-lab sample-by-sample waveform validation of A/D and D/A SPLIB modules. This will involve choosing suitable known signals to sample/reconstruct and modifying driver code to inject sentinel values at specific interval for validation. This task will require high precision function generation equipment and digital capture oscilloscopes.
        - Software integration and testing of GPIO functionality for the purpose of interfacing with the PARB preamplifier board, and DARB amplifier remote power switching.
        - A software module to capture the generation of hardware PPS events to trigger DARB playback in synchronous mode.
    - As the above software development stabilizes, a new development environment (including OS and system software installation procedures) must be generated and documented.
    - CPU and memory profiling and tuning of the PARB processing modes must be performed.
- Perform in-water full integration including all analog electronic components (amplifiers, transducers, projectors, etc.)
- Perform an investigation of short range, high-bandwidth wireless data transfers from the PARB. This will include purchasing of a suitable USB WiFi device and outboard antenna.
- Upgrade the wave-train definition application to allow users to specify replica waveform types and their parameters. These definition files will be transferred to DARB and PARB units (instead of generated waveforms) for playback and processing, thus reducing the need to transfer large data files.

Annex A Software tools

This section provides background information necessary to understand the role that DRDC software played in this contract. This flexible reusable software enabled the project to make better use of the available budget, advancing the work more than would be otherwise possible. Their relationship to the project is described below, while a high-level description of the tool itself is provided in the subsections below:

- The SPLIB software libraries, underlying SPPACS, were an essential part of this project. Existing library modules for main blast detection, correlation processing, and detection were integrated to create core MAS processing functionality. Related modules were either created or improved to allow multiple correlation, main blast detection, and echo detection streams to run in parallel along with appropriate management logic to ensure that the overall system could work as designed. Once these modules were created they were integrated into an SPPACS module for testing and algorithm validation, before being integrated into the AS.
- STAR-IDL was used to create test data and to visualize output data from various processing stages during testing. These tests helped to ensure that the MAS mode would function correctly, while performing test analysis with STAR-IDL helped to make the process efficient.
- OPD was used for verification of the synthetic data used to test the MAS processing stream. It was used to visualize and measure the background levels, simulated blast and echo event SNR, as well as reverb levels during a simulated ping. Finally the time cursor was used to measure and verify the spacing of simulated main blast and echo events.
- The PARB, AS, DARB, and DAS were all improved during the contracted investigation and development effort.

A.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 runtime 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 runtime 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.

A.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 (TI) 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).

A.2 STAR-IDL

The STAR-IDL[4] 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.

[4] 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.

- 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.
- 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.

A.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 indicator (ETI), amplitude-line indicator (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. The following functions summarize its capability (detailed information can be found in the OPD User Manual [2]):

- 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, decimation, sonogram compression and much more.
- OPD provides an optional beamformer and is capable of processing complex heterodyned time-series data.
- 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 crosshair 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, periodic event and Doppler cursors can be used to analyze advanced features in data and learn tactical information about potential targets.

A.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 – 40 kHz

- Acoustic preamplifier gain is variable from 0 – 35 dB in 5 dB steps. The 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 soundcards that are supported by standard API on MS Windows, OSX, and Linux. New data sources are regularly added.

A.5 DARB and DAS

The dynamic active reusable buoy (DARB) and the closely related dynamic active simulator (DAS) are valuable tools for experimentation. They are used to transmit pre-defined signals into the water. The DARB includes:

- A flotation system (buoy) which contains the electronics
- On-board computer used to control the buoy and generate signals
- Amplifier used to transmit signals
- Batteries for power
- GPS for location and time-synchronization
- Freewave modem for control and position updates

The on-board software uses a simple terminal-driven menu interface to control the buoy. All commands are logged along with other important information (e.g. NMEA track). The NMEA track is also broadcast so that the buoy's position is known, when not in the menu.

The system operates in the following acoustic modes:

- Standby – no acoustic transmissions are sent.
- Synchronous – starts transmitting pulses at a specified offset from the minute and repeats at a specified interval (based on GPS time).

- Continuous – transmits a signal continuously.
- SubGPS – the buoy transmits NMEA strings using an underwater modem with a custom modulation scheme developed by Akoostix personnel for DRDC Atlantic. The scheme allows the transmission rate to be scaled, based on required range from 1 bps to 100 bps.

Signals can be WAV formatted files, or wave trains that are a serious of WAV formatted files with user-specific gaps between the files. Wave trains are defined to avoid the requirement to create large files containing mostly zeroes to fill in gaps.

The system also comes with a user manual [3] that can be used for more detailed information and instructions.

Annex B Configuration management

The final software deliverable for this contract was provided on the STAR release CD, which generated and delivered to James Theriault, DRDC's lead scientist for STAR, on 30 March 2012. The release coincides with delivery for several STAR DISO call-ups. This section of the document describes the content of that CD.

B.1 STAR branch and release information

Each logical grouping of software modules has been independently versioned on the CD that is also versioned. The current STAR release version is 6.6.11 and contains the following:

- OPD 2.6.0
- ACDC 2.1.8
- SPPACS 1.3.0
- Analysis Tools 6.13.0 (STAR-IDL)

Installation instructions are located in the root directory on the release CD.

B.1.1 STAR software documentation

Some manuals, API documentation, and other design documents are provided with the software release CD. In a standard STAR distribution they can be found by opening the */usr/local/atools/star-6.6.11/documentation.html* file in a standard web browser. This page contains links to several sets of documentation including:

- Software revision history
- Software API documentation including, IDLDoc for the analysis tools (STAR-IDL) and DOxygen generated documents for OPD, ACDC, and SPPACs
- The STAR user manual[5]
- STAR quick reference guides
- STAR-IDL application user manuals
- STAR application user manuals
- Useful third-party Documentation

[5] This manual has not been updated for some time and is in the process of being superseded by newer documentation included on this CD.

B.2 Issue summary

The issue summary in Table 2 shows the current state of known defects for all of the software release candidates listed in B.1 as of 30 March 2012.

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 no active blocker issues but there are several critical issues. These are obscure or infrequent bugs that were discovered during current or past 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 need to be addressed before a contract can be completed successfully.

Table 2: Issue summary (severity vs. status) for all software on STAR release 6.6.11

	Opened	**Reopened**	**Resolved**	**Closed**
Blocker	0	0	0	37
Critical	12	1	0	75
Major	113	4	1	224
Minor	40	2	0	25
Trivial	7	0	0	3
Undecided	0	0	0	5

Table 3 summarizes critical issues that remain open, but only for software relevant to this contract. None of the critical 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: Known Critical issues for STAR

Module	Issue ID	Summary	Description
OPD	OPDY-275	Crash on deleting data	A crash was experienced while deleting old data. The user was processing data when the user commanded the deletion, but it appears that processing and deletion were completed at the same time, as the data being processed was also deleted, which may only occur once processing is completed. An OPD crash resulted. Effort to reproduce the crash has been unsuccessful, reinforcing that the user may have gotten the timing exactly right, deleting the data at the exact time that processing completed. See comments for system dump.
OPD	OPDY-245	OPD crashes if EADAQ changes state during processing and OPD hasn't stopped processing	OPD will hang if the user stops processing and EADAQ resets. The display will freeze and the user is unable to stop processing.
OPD	OPDY-244	OPD crashes when stopping processing after EADAQ stops	OPD will crash if the user tries to process EADAQ when EADAQ is not recording.

OPD	OPDY-77	OPD hangs when validating data sources	OPD tries to verify the data stream by reading as much data as required to determine the format and sensors. This stalls the system until it receives this information. An array server can be accepting connections but not sending data, and OPD connects to the Northern Watch array server automatically as soon as a valid IP and port are given. OPD even saves the IP and port to save users from retyping them each time. If the server stops feeding data on one port (but still allows connections) and starts another, and the user restarts OPD, OPD tries to connect to the previously saved port and hangs. Workaround: 1. Port being saved is in the registry settings. It can be changed manually before resetting OPD or 2. The user can start the array server on the previously saved port. This will free OPD from the hang. Permanent fix (tentative): 1. Prevent OPD from validating data sources automatically by using some kind of connect button. 2. Have a timeout for validating sources since we still have the issue that a socket connection can be made but the data is not flowing.
SPPACS	AKSP-72	*sp_median_nrmf* is referencing a null pointer occasionally in win32	*sp_median_nrmf* causes a crash (in OPD) by dereferencing a null pointer. This seems to only happen on win32, but may just be hidden in Unix environments (Windows has a history of being more strict, especially in debug mode). Before the crash (during construction of the module) warnings are printed to *stderr*, "Can't find remove point" An examination of the code revealed that there is a comment saying "if we got here, there is a bug".

Annex C SBC Evaluation Data

This Annex contains the results of the benchmark tests performed on the three candidate SBC boards in order to evaluate their performance and aide in choosing the best solution.

C.1 Electrical Power Requirements

Power measurements were taken for each candidate SBC using the total current drawn from the bench power supply used to power them. These measurements are simply instantaneous measurements taken with the system in various states with the intention of comparing the SBCs against each other.

C.1.1 Bare Board Test

The initial test measured the SBCs power consumption with a minimal set of peripherals connected:

- USB Keyboard
- VGA monitor
- Ethernet

After installing the operating system and various system services, power measurements were taken in the following states:

- System idle.
- System with one thread running at 100% CPU usage.
- System with two threads running at 100% CPU usage.
- System in Suspend-to-RAM sleep state (if available).

These measurements were taken with only the SBC connected to the power supply. The ADL and Lippert boards both have on-board flash disks upon which the operating system was installed. The Kontron board does not employ an on-board flash disk, so while the operating system was installed on an off-board HDD, this drive was powered using a separate power supply for these tests in order to compare results.

It should also be noted that a DC-DC ATX power supply (pico-PSU 80 purchased from minibox.com) was used to power the Lippert board from the bench power supply. This introduces power conversion inefficiency which was not present in the other two SBCs results. The Lippert SBC does not specifically require an ATX power supply (it can be powered with a single 5 volt supply). However, the required connectors were not available at the time these measurements were taken and the development kit included a power wire-harness compatible with a standard ATX supply. It is expected that for this reason the Lippert SBC measurements are at an unfair disadvantage, and should be re-measured with a level playing field at a later date.

Finally, it was noted when testing that the Kontron SBC does not seem to return from suspend-to-RAM gracefully. When woken, the VGA output remains in a suspended state. The SBC continues to operate but required a system reboot in order to restore VGA output.

The following tables contain the data collected for the tests described in Section C.1.

Table 4: Bare SBC Power Measurement Results (in Watts)

Test	ADL	Lippert	Kontron
System Idle	**6.9**	**9**	**8.8**
Single Thread Test	**7.85**	**12.84**	**10.3**
Two Thread Test	**8.1**	**13.08**	**10.6**
Suspend-To-RAM	**n/a**	**1.2**	**0.35**

From Table 4 the following points were noted:

- ADL offered the lowest power consumption.
    - The Lippert and Kontron boards draw approximately 2 Watts more at idle.
    - The Lippert board draws the most power under heavy CPU load (2.5 to 3 Watts more than the Kontron). This may be due to the different N270 Atom CPU. It may also be partly due to the ATX power supply efficiency described above.
- The Lippert board draws roughly 3.5 times more current than the Kontron board when sleeping. Again, this may be skewed by the ATX power supply's efficiency.

C.1.2 Solid State Disk Integration

In addition to the above tests, an external solid state HDD was connected to the SBCs and power measurements were taken in the following scenarios:

- System idle.
- SSD performing a large I/O test.
- System in Suspend-to-RAM (sleep) state if available.

Please note that measurements are not available for the Kontron SBC as the test setup used was not capable of measuring the current drawn for both the SBC and SDD simultaneously.

Table 5: HDD Integration Power Measurement Results (in Watts)

Test	ADL w PATA SSD	Lippert w SATA SSD	Kontron
System Idle	**6.95 – 7.45**	**11.5 – 11.9**	**n/a**
Continuous I/O Test	**8.75 – 9.25**	**16.1 – 17.76**	**n/a**

Suspend-to-RAM	**n/a**	**3.12**	**n/a**

From Table 5 the following was noted:

- The PATA SSD added very little to the total power consumption when idle.
- The SATA SSD added nearly 3 Watts of power consumption while idle.
- This gap grew even more under heavy disk I/O load.
- The SATA SSD increases the sleeping power draw considerable. Data is unavailable for the PATA SSD as the ADL board cannot support the sleep function.

C.1.3 General Standards Data Acquisition Integration

Finally, the General Standards data acquisition card was added to the system and measurements were taken with the system idle, sleeping, and running real-time MAS processing/recording. Due to testing hardware limitations, it was impossible to include the SSD in the Kontron power measurements. For this reason Table 6 compares the ADL and Lippert boards fully integrated including the SSD. Next, Table 7 compares the Lippert and Kontron solutions excluding the SSD[6].

Table 6: Power Measurement Results with Integrated General Standards 16AISS8AO4 and SDD (in Watts).

Test	ADL	Lippert	Kontron
Idle	**12.8**	**16.8**	**n/a**
Suspend-to-RAM	**n/a**	**8.4**	**n/a**
MAS Processing	**13 - 14**	**18 – 20.4**	**n/a**

From the data in Table 6 it was noted that:

- Integration of the General Standards data acquisition board added approximately 5-6 Watts, including in sleep state.

Table 7: Power Measurement Results with Integrated General Standards 16AISS8AO4 without SDD (in Watts).

Test	ADL	Lippert	Kontron
Idle	**n/a**	**15.7**	**14.9**
Suspend-to-RAM	**n/a**	**6.5**	**5.8**
MAS Processing	**n/a**	**16.9 – 18.4**	**15.5 – 16.2**

[6] It is not possible to power the PATA SSD separately with the ADL board in this comparison.

From Table 7 the following was noted:

- Integration of the General Standards data acquisition board added approximately 5-6 Watts, including in sleep state. This agreed with the results of Table 6.

C.1.4 Power Consumption of External USB Serial Port Dongle

Since the Lippert and ADL SBCs include only two RS232 Serial Ports, a simple power measurement was taken after the addition of a two port RS232 adapter via USB. This added roughly 150 milliwatts.

C.1.5 Results and Discussion

Several observations were made in the preceding power measurements:

- ADL had the best overall power consumption, however this board does not support suspend-to-RAM (sleep).
- The Lippert board draws the most power. This could be caused by:
    - Differences in the N270 and Z530 CPUs.
    - Inefficiencies in the ATX power supply which was required to power the Lippert board only.
    - Some combination of the above.
- The Kontron SBC does not recover correctly from suspend-to-RAM. This could be a driver issue and may be corrected by future Linux kernel releases.

C.2 Processing Benchmarks

Each SBC was used to run an array of processing tests to benchmark CPU intensive tasks as well as CPU with moderate I/O usage. All tests were performed using the SVN trunk of both the SPPACS and stealth projects at revision 10599.

C.2.1 FFT Tests

The first set of tests simply computed FFTs of varying sizes repeatedly. The FFT operation is core to many signal processing modules in SPLIB. The following tests were performed:

- Time to compute 128 point FFTs 10 million times.
- Time to compute 8192 point FFTs 100,000 times.
- Time to compute 128K point FFTs 1000 times.

The results of the FFT benchmark tests are shown in Table 8. It was noted that all measurements were very close to one another, with the exception of performing 128K point FFTs on the Kontron board. This could be a result of slower system RAM since the CPU is identical to that of the ADL and 128K point FFT operations most likely cannot be performed completely in CPU caches.

Table 8: FFT test results. Times are in seconds, lower is better.

Test	ADL	Lippert	Kontron
128	**17.64**	**17.71**	**17.73**
8192	**21.54**	**21.55**	**21.52**
131072	**11.73**	**11.28**	**13.07**

C.2.2 SPPACS MAS Benchmark

The second test used the new SPPACS MAS application to perform the same processing test that was used to verify the software in Section **Error! Reference source not found.**.

In particular the command used was:

```
sp_mas -T 7.0 -t 20.0 -r 3_complex_replicas_down_up_out.dat --min-seconds-between-blasts
20.0 -a 5.0 -b 0.001 --time-extent-of-blasts-in-seconds 10.0 --blast-search-width-factor 1.0 -w 0.9
-g 0.1 -s 0.004 --min-seconds-between-echos 0.004 --blast-blanking-seconds 2.0 --echo-detect-
seconds 22.0 < 4_chan_up_then_down.dat
```

The SPPACS MAS processing test results are shown in Table 9.

Table 9: SPPACS MAS Test results. Times are in seconds, lower is better.

Test	ADL	Lippert	Kontron
sp_mas	**100**	**92**	**91**

It is noted that the ADL embedded SBC performed this task roughly 10% slower than the other two SBCs. During testing the CPU usage was noted to drop considerably several times over the duration of processing. This may indicate an issue with the PATA hard disk controller on this board (although this setup outperformed the other two in disk I/O testing which will be discussed in Section C.4).

C.2.3 Integrated AS MAS Mode Test

Finally each SBC was used to perform the following integrated real-time tests through the *as_host* control interface:

- Single channel MAS mode, 2 kHz bandwidth using the General Standards 16AISS8AO4 A/D as an input device.
- Four channel ambient mode, 40 kHz bandwidth using the General Standards 16AISS8AO4 A/D as an input device.

These tests were performed with a sampling duration of 180 seconds while monitoring the average CPU usage. The integrated AS MAS processing results are shown in Table 10.

Table 10: Integrated AS MAS CPU usage test results.

Test	ADL	Lippert	Kontron
Single Channel MAS	**5 - 10% CPU**	**5 - 10% CPU**	**5 - 12% CPU**
Four Channel 100 kHz Ambient	**2 - 5% CPU**	**2 - 5% CPU**	**3 – 6% CPU**

C.2.4 Discussion

All SBCs performed the CPU benchmarks with roughly the same results. The two notable outliers were:

- The ADL embedded board performed slightly 10% worse than the others when benchmarking the SPPACS processing benchmark.
- The Kontron SBC performed roughly 15% slower than the others when computing 128K point FFTs.
- The Kontron performed slightly worse than the others when testing the integrated MAS software within the AS.

C.3 System Boot and Shutdown Times

Each SBC was timed in the following scenarios:

- Time to boot from power-on.
- Time to boot from Linux boot-loader. The difference between this and the previous measurement is the fixed BIOS boot time.
- Time to shutdown cleanly.

C.3.1 Results and Discussion

The boot and shutdown timing results for each SBC is shown in Table 11. It was noted that during this test, the Kontron board stalled randomly while loading the Silicon Image SATA controller driver. At this point, a later Linux kernel was configured and installed which seemed to alleviate the issue, but not eliminate it completely. It is unknown whether this is a Linux kernel bug or faulty hardware.

Table 11: Boot/Shutdown timing results in seconds

Test	ADL	Lippert	Kontron
Cold Boot	**23-26**	**19-20**	**27-28**
Boot OS	**12-16**	**7**	**11**
BIOS Time (calculated[7])	**10-11**	**12-13**	**16-17**
Shutdown OS	**10-14**	**11**	**10**

It was noted that the ADL board measurements had a larger variance in its measurements than the other two, but the cause was unknown.

The Lippert SBC had the best overall boot time (even though its BIOS was slightly slower than the ADL). The Kontron SBC took considerably longer to boot the OS. A significant pause was noted while loading the Silicon Image SATA controller driver.

C.4 Disk I/O

The average disk I/O bandwidth was measured by writing a large sequential file to the solid state HDD. The Linux command 'dd' was used to stream a large file containing all zeros to disk. The following command was used: 'dd if=/dev/zero of=test.bin ibs=2048 count=1048576'. This writes exactly 2 GB of data to a single file in block increments of 2048 bytes. In some tests (as noted) the count value was decreased in order to accommodate smaller flash devices.

This test was used to measure I/O performance of both the on-board flash drives included with the SBC (where applicable) as well as the external solid state HDDs used for data storage.

C.4.1 Results and Discussion

The HDD benchmarking results are shown in Table 12.

Table 12: Large HDD I/O test results in MB/s, larger is better

Test	ADL	Lippert	Kontron
Write to external SDD	**54.8 (Transcend PATA SDD)**	**48.2 (Kinsgton SATA SDD)**	**49.4 (Kinston SATA SDD)**
Write to on-board flash	**9.6**	**21.8 [8]**	**n/a**

From the results in Table 12 it was shown that all hard disk I/O is at least an order or magnitude greater than the bandwidth required to continuously stream four channels of 16-bit data at a rate of 100 kHz to disk. There should be no issue keeping up with this anticipated data rate.

[7] Calculated by subtracting the OS boot time from the cold boot time.
[8] Due to size limitations of the on-board flash drive, this test wrote 1 GB instead of 2 GB of data.

C.5 Graphical Environment

This section discusses installing a graphical environment (X server) under Linux on the candidate SBCs. The selected boards span two different graphics chipsets (integrated with the Atom processor):

- GMA 500 (Poulsbo – integrated in Z5x0 Atom processors). Used by both ADL Embedded and Kontron SBCs.
- GMA 950 (integrated in N270 Atom processor). Used by Lippert SBC.

It is noted that the GMA 500 chipset was licensed by Intel from Imagination Technologies. This chipset historically has been difficult to use with Linux since the binary drivers provided by Imagination Technologies are of poor quality and do not keep pace with modern Kernels or Xorg releases. The results have been at best sub-par if they can be made to work at all. In 2011 Intel has decided to pay to have an open-source driver written for this chipset directly in the Kernel source code tree itself. At the time of writing this support is in its early stages and is under active development. It shows promise, but there are some rough edges present:

- ADL Embedded SBC:
    - The Linux kernel console did not detect the monitor resolution without the addition of a kernel boot parameter to override the setting. Until this was done, there were noticeable graphical artefacts in the console.
    - After installing the Xorg server and running a window manager, some graphical glitches were present. No workaround was found for this, but is not unusual for a driver so early in development.
- Kontron SBC:
    - The system became unstable with the kernel graphics driver module installed. The monitor output became extremely corrupted and the system locked up. Once the graphics module was uninstalled, this issue disappeared.

In contrast, the GMA 950 chipset (used by the Lippert SBC) was designed by Intel. It has both a mature open-source driver in the Linux kernel, as well as a fully supported Xorg server driver. No issues were encountered while using the graphical drivers and installing a simple window manager.

C.6 General Purpose I/O

The digital output functionality was verified for the following:

- General Standards 16AISS8AO4 configured as digital outputs. The provided General Standards example code was used to switch on/off state. The pins' states were verified with a voltmeter.
- ADL SBC's on-board LPT parallel port. The Linux kernel ppdev driver was used with a simple test program to set pins on/off state. The pins' states were verified with a voltmeter.

- Kontron SBC's on-board LPT parallel port. This was verified in the same way as the ADL board.

All GPIO pins (configured as digital outputs) were verified to work using example test programs.

C.7 Soundcard Integration

The Linux kernel was configured with the required hd_intel hardware drivers that all three SBC candidates used. The alsa sound system service was installed as part of the OS boot and verified by playing and recording a wave-file using the alsa utilities arecord and aplay.

Unfortunately there was not enough time to test the AS legacy soundcard support. The Linux kernel required extra configuration to support the deprecated APIs (OSS) which were used in the legacy AS functionality. Alternatively, the AS could be ported in the future to use the newer alsa APIs.

References

[1] Hood, J. (2011), Introduction to the Software Tools for Analysis and Research (STAR): Tools and Processes Version 1.0 (AI MA 2011-007), Akoostix Inc., Dartmouth, Nova Scotia.

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

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

This page intentionally left blank.

List of symbols/abbreviations/acronyms/initialisms

DND	Department of National Defence
DRDC	Defence Research & Development Canada
DRDKIM	Director Research and Development Knowledge and Information Management
R&D	Research & Development