sigma engineering limited

TP 13236E

MODULAR RADAR INTERFACE: ENHANCEMENTS FOR SAR

Prepared for
Transportation Development Centre
Safety and Security
Transport Canada

by
Sigma Engineering Limited

March 1998

sigma engineering limited

TP 13236E

MODULAR RADAR INTERFACE: ENHANCEMENTS FOR SAR

by

J. Ryan, S. Motty and M. Johnson
Sigma Engineering Limited

March 1998

Project Team

Joe Ryan, Project Manager, Technical Advisor
Steve Motty, Software Design Engineer
Max Johnson, Software Design Engineer
Scott Parsons, Software Developer
Gary Dinn, Consolidated Technologies, Hardware Design

Un sommaire français se trouve avant la table des matières.

Transport Canada Transports Canada

PUBLICATION DATA FORM

1. Transport Canada Publication No. TP 13236E	2. Project No. 9082	3. Recipient's Catalogue No.
4. Title and Subtitle Modular Radar Interface: Enhancements for SAR		5. Publication Date March 1998
		6. Performing Organization Document No.
7. Author(s) Joe Ryan, Steve Motty, and Max Johnson		8. Transport Canada File No. ZCD1460-362-2
9. Performing Organization Name and Address Sigma Engineering Limited 140 Water Street, Suite 602 St. John's, Newfoundland A1C 5W2		10. PWGSC File No. XSD-6-01869
		11. PWGSC or Transport Canada Contract No. T8200-66551
12. Sponsoring Agency Name and Address Transportation Development Centre (TDC) 800 René Lévesque Blvd. West 6th Floor Montreal, Quebec H3B 1X9		13. Type of Publication and Period Covered Final
		14. Project Officer C. Gautier

15. Supplementary Notes (Funding programs, titles of related publications, etc.)

Co-sponsored by New Initiatives Fund (NIF), Program of Energy Research and Development (PERD), and Defence Research Establishment Ottawa (DREO)

16. Abstract

This report describes modifications and enhancements made to the Modular Radar Interface (MRI) to increase its data throughput while adding more flexibility for multiple data formats.

The existing system architecture was reviewed to evaluate its suitability for supporting the enhanced features. A market survey was done to determine the optimum hardware configuration to support the new system. Based on the study conclusions, the system was redesigned to meet the following criteria:

- support for antenna speeds up to 120 RPM
- use of generic A/D cards for data capture
- implementation of signal processing in software
- support for high-speed data storage on tape
- support for multiple data clients
- provision of simultaneous multiple data formats

Sigma's radar interface was redesigned to provide the capability of processing gyro and azimuth data. A dedicated radar display application was also developed. The entire system was tested and an extensive data set was collected during a field trial in Newfoundland.

To support the field trial, a module was added to both the MRI and the radar display system to provide control of a Raytheon MK2 radar transceiver. A composite video decoder was designed to support this radar.

This report documents the implementation of the changes and the results of the field trial. It also contains recommendations for further enhancements of the MRI.

17. Key Words Radar, interface, scan conversion, radar display, detection, enhancement, processing, recording, tracking		18. Distribution Statement Limited number of copies available from the Transportation Development Centre		
19. Security Classification (of this publication) Unclassified	20. Security Classification (of this page) Unclassified	21. Declassification (date) —	22. No. of Pages xiv, 36, app.	23. Price —

Transports Canada | Transport Canada

FORMULE DE DONNÉES POUR PUBLICATION

1. N° de la publication de Transports Canada TP 13236E	2. N° de l'étude 9082	3. N° de catalogue du destinataire
4. Titre et sous-titre Modular Radar Interface: Enhancements for SAR		5. Date de la publication Mars 1998
		6. N° de document de l'organisme exécutant
7. Auteur(s) Joe Ryan, Steve Motty, et Max Johnson		8. N° de dossier - Transports Canada ZCD1460-362-2
9. Nom et adresse de l'organisme exécutant Sigma Engineering Limited 140 Water Street, Suite 602 St. John's, Newfoundland A1C 5W2		10. N° de dossier - TPSGC XSD-6-01869
		11. N° de contrat - TPSGC ou Transports Canada T8200-66551
12. Nom et adresse de l'organisme parrain Centre de développement des transports (CDT) 800, boul. René-Lévesque Ouest 6^e^ étage Montréal (Québec) H3B 1X9		13. Genre de publication et période visée Final
		14. Agent de projet C. Gautier

15. Remarques additionnelles (programmes de financement, titres de publications connexes, etc.)

Coparrainé par le Fonds des nouvelles initiatives (FNI), le Programme de R&D énergétiques (PRDE) et le Centre de recherches pour la défense Ottawa (CRDO)

16. Résumé

Le présent rapport décrit les modifications et les améliorations apportées à l'interface modulaire radar (IMR) pour accroître le débit des données et donner une plus grande souplesse au système dans les cas d'intégration de formats de données multiples.

L'architecture courante du système a fait l'objet d'une révision en vue d'évaluer sa capacité d'appuyer les caractéristiques améliorées. On a effectué une étude de marché pour définir la configuration de matériel optimale à l'appui du nouveau système. D'après les conclusions de l'étude, le système a été modifié pour répondre aux critères suivants :

- appui permettant de recueillir des données à des vitesses de rotation d'antenne pouvant atteindre 120 tr/min.
- utilisation de cartes génériques de conversion A/N pour la saisie des données
- mise en oeuvre du traitement des signaux dans les logiciels
- appui pour le stockage rapide des données sur les bandes magnétiques
- appui aux clients nécessitant des jeux de données multiples
- fourniture simultanée de formats multiples de données

L'interface radar de Sigma a été modifiée pour permettre le traitement des données des signaux de cap et d'azimut. On a également mis au point une application d'affichage radar spécialisée. Tout le système a été testé et un jeu détaillé de données a été recueilli au cours d'un essai en mer à Terre-Neuve.

Pour appuyer l'essai en mer, on a ajouté un module à l'IMR et au système d'affichage radar pour permettre la commande d'un émetteur-récepteur radar Raytheon MK2. On a conçu un décodeur vidéo composite pour appuyer les fonctions de ce radar.

Ce rapport fait état de la mise en oeuvre des modifications apportées au système et des résultats de l'essai en mer.
Il contient également des recommandations visant à parfaire les améliorations de l'IMR.

17. Mots clés Radar, interface, conversion de balayage, affichage radar, détection, amélioration, traitement, enregistrement, poursuite	18. Diffusion Le Centre de développement des transports dispose d'un nombre limité d'exemplaires.

19. Classification de sécurité (de cette publication) Non classifiée	20. Classification de sécurité (de cette page) Non classifiée	21. Déclassification (date) —	22. Nombre de pages xiv, 36, ann.	23. Prix —

CDT/TDC 79-005
Rev. 96

Canada

ACKNOWLEDGMENTS

The assistance of Reg Fitzgerald and Gordon Breen of Oceans Limited in setting up and conducting the field evaluation and the significant technical support provided by Charles Gautier, Transportation Development Centre, during the field evaluation are greatly appreciated. The support of the Canadian Coast Guard, Newfoundland Region, has helped make this project a success.

EXECUTIVE SUMMARY

This report describes modifications and enhancements made to the Modular Radar Interface (MRI) to enable it to be used for research in advanced processing techniques for Search and Rescue. These changes increased the data throughput of the MRI while adding more flexibility for multiple data formats.

The existing system architecture was reviewed to evaluate its suitability for supporting the enhanced features. Based on the study conclusions, the system was redesigned to meet the following criteria:

- support for antenna speeds up to 120 RPM
- use of generic A/D cards for data capture
- implementation of signal processing in software
- support for high speed data storage on tape
- support for multiple data clients
- provision of simultaneous multiple data formats

A component survey was carried out to determine the optimum hardware configuration to support the new system. Based on this study, the following hardware configuration was chosen:

- PC platform consisting of two 200 MHz dual-Pentium Pro single-board computers (SBCs) installed in a split-backplane host chassis with dual power supplies
- Exabyte Mammoth tape drive
- Matrox Pulsar-LC and Data Translation DT3152 video A/D cards

The selected video A/D cards proved unsuitable for use in the MRI. Support for previously proven technology (PDI Select 45 hardware) was then built into the video subsystem.

The existing plot extraction algorithm was reviewed to determine its accuracy. It was concluded that the plot extractor algorithm works properly.

The implementation of the modifications to the MRI is described in this report. Modifications to Sigma's radar interface were carried out to permit more flexible data input with the addition of gyro data. A dedicated radar data display user interface was also developed for use in system testing and field trials. During a field trial off the coast of Newfoundland on the Coast Guard vessel *J. E. Bernier*, the entire system was tested, and over 400 GBytes of radar data were collected in a range of sea conditions.

To support the field trial, a module was added to both the MRI and the radar display system to provide control of a Raytheon MK2 radar transceiver. A composite video decoder was also designed to support this radar.

This report documents the implementation of the changes and the state of the MRI software as tested during field trials and data collection. This version of the software contains modifications, such as a Raytheon-specific configuration and the antenna azimuth encoder error correction, that were included specifically to satisfy the requirements of the field trial. Consideration should be given to how best to remove these components from the MRI software and improve the flexibility of the software for future projects.

SOMMAIRE

Le présent rapport décrit les modifications et les améliorations apportées à l'interface modulaire radar (IMR) pour qu'elle puisse incorporer des techniques de pointe de traitement des signaux, aux fins des opérations de recherche-sauvetage. Ces modifications ont eu pour effet d'accroître le débit des données et donner une plus grande souplesse au système dans les cas d'intégration de formats de données multiples.

L'architecture courante du système a fait l'objet d'une révision en vue d'évaluer sa capacité d'appuyer les caractéristiques améliorées. D'après les conclusions de l'étude, le système a été modifié pour répondre aux critères suivants :

- appui permettant de recueillir des données à des vitesses de rotation d'antenne pouvant atteindre 120 tr/min.
- utilisation de cartes génériques de conversion A/N pour la saisie des données
- mise en oeuvre du traitement des signaux dans les logiciels
- appui pour le stockage rapide des données sur les bandes magnétiques
- appui aux clients nécessitant des jeux de données multiples
- fourniture simultanée de formats multiples de données

On a effectué une étude sur les composants pour définir la configuration de matériel optimale à l'appui du nouveau système. En se basant sur les résultats de l'étude, la configuration suivante a été adoptée :

- système d'exploitation comprenant deux ordinateurs sur carte unique, chacun doté de deux processeurs Pentium Pro double de 200 MHz installés dans un boîtier à fond de panier séparé, avec deux blocs d'alimentation
- dérouleur de bande magnétique Exabyte Mammoth
- une carte Matrox Pulsar-LC et une carte de conversion vidéo A/N DT3152

Les cartes de conversion A/N retenues n'ont pu être utilisées dans l'IMR. Du matériel d'appui dérivé d'une technologie éprouvée (matériel PDI Select 45) a ensuite été intégré dans le sous-système vidéo.

L'algorithme actuel d'extraction des plots a fait l'objet d'une révision en vue d'en vérifier la précision. On a conclu qu'il fonctionnait correctement.

Le présent rapport décrit la mise en oeuvre des modifications apportées à l'IMR. On a modifié l'interface radar de Sigma pour permettre une plus grande souplesse d'entrée des données avec l'addition des données des signaux de cap. On a également mis au point, à l'intention des utilisateurs, une interface spécialisée d'affichage de données radar qui sera utilisée au cours des tests et de l'essai en mer du système. Lors d'un essai en mer réalisé au large des côtes de Terre-Neuve, à bord du navire de la Garde côtière *J. E. Bernier*, tout le système a été testé et plus de 400 giga-octets de données radar ont été recueillies dans divers états de mer.

En vue d'appuyer l'essai en mer, on a ajouté un module à l'IMR et au système d'affichage radar pour permettre de commander l'émetteur-récepteur radar Raytheon MK2. On a aussi conçu un décodeur vidéo composite pour appuyer les fonctions de ce radar.

Le présent rapport décrit la mise en oeuvre des modifications apportées au logiciel IMR et l'état de celle-ci lors des essais effectués en mer et pendant la collecte des données. Cette version du logiciel contient des modifications comme la configuration propre au système Raytheon et la correction d'erreurs de l'encodeur en azimut de l'antenne qui avaient été installées spécifiquement pour répondre aux exigences de l'essai en mer. Il conviendra d'étudier la meilleure façon d'enlever ces composantes du logiciel IMR et d'améliorer la souplesse du logiciel pour des applications futures.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The Transportation Development Centre (TDC) is involved in a variety of projects aimed at developing techniques to enhance the small target detection capability of conventional marine radars commonly used by Search and Rescue personnel during rescue missions.

The Modular Radar Interface (MRI) has been promoted as a crucial element in the advancement of processing techniques for small target detection. The purpose of the MRI is to provide TDC with a versatile means of providing shipboard radar data to applications requiring it. The current endeavor represents the third phase of the MRI development. In phase one, the viability of the technology was established using a dedicated radar processor built by Titan Radar International Inc. Phase two of the MRI resulted in a functioning client/server application which was subsequently used in the development and testing of the AI Tracker for SAR. In the current phase, the MRI has been made capable of handling radar data generated by antenna scanning speeds of 120 rpm. In order to do this it was necessary to move to high-speed PCI bus design.

The MRI was modified to support off-the-shelf high speed data acquisition cards on the PCI bus. The tape recording media was upgraded to allow the throughputs associated with high scan rates and pulse repetition frequencies. Advanced radar processing functions such as scan to scan averaging, scan conversion, pulse to pulse processing and CFAR have been implemented in software. A special radar interface card was adapted to the requirements of the enhanced MRI. Finally, a Graphical User Interface (GUI) was developed to allow control of the MRI and live display of the captured data. The GUI and the new version of the MRI were used for data collection and review during field trials.

This report summarizes the modifications of and additions to the MRI.

2. BACKGROUND

To improve the detection capability of the AI Tracker for SAR, the antenna scan rate of a conventional marine radar was to be increased from the current 30 rpm to 120 rpm. The requirements imposed on the radar interface by such speeds could not be addressed by the existing MRI system. The existing hardware would have to be upgraded.

It was felt that any enhancements to the MRI for support of the high-speed scanner radar operation should be combined with support for other special signal processing requirements imposed by the AI Tracker for SAR project. In particular, a preference for B-scan data has been expressed. The proposed modifications would be costly to implement in hardware, so it was decided to use off-the-shelf video A/D cards to capture high rates of B-scan data across the PCI bus. Making use of these cards meant that radar-specific processing options implemented by the TITAN card-set would be sacrificed. Early studies indicated that by making use of high speed dual-Pentium Pro processors, many of the processing options required could be performed in real-time using software implementations.

The modifications were conducted by Sigma Engineering Limited. Sections 2.1 to 2.6 discuss the status of modifications for each of the following components:

- MRI Architecture Review
- Components Market Survey and Acquisition
- MRI Software Upgrade
- Plot Extraction Verification
- Radar Interface Modifications
- SAR Display Module Development

The final system was installed on the CCGS J.E.BERNIER while stationed in St. John's, Newfoundland. Field trials of the final implementation were held during the month of November 1997 to test the functions of the system and to collect data sets for processing by the AI Tracker.

2.1 MRI Architecture Review

The MRI Architecture Review concluded that the existing modular implementation served as a suitable model for the required modifications. It was decided that the final system should be packaged as a combination of executables and DLLs. The executables included the server itself, a display application and a plot extraction client/server. The primary modifications to be made were in the data flow between the A/D modules, signal processing module and the tape module. Improvements to client arbitration were also discussed. Details of the MRI Architecture Review were presented in a milestone report.

2.2 Components Market Survey and Acquisition

A number of vendors were surveyed in order to determine the best providers of processor technology, tape storage devices and A/D hardware.

The chosen PC platform consisted of two 200 MHz dual-Pentium Pro single board computers (SBCs) installed in a split-backplane host chassis with dual power supplies. One of the SBCs would host the MRI3 server, while the display applications (SAR-GUI and SetupMRI) along with the plot extractor client/server would reside on the second. Communication between the two PCs was facilitated by Windows NT networking support using RPC and TCP/IP calls.

With the exception of the CY-9000, none of the reviewed storage systems provided sufficient bandwidth for the maximum radar data rate, which is about 4.5 MB/sec. However, the CY-9000 is too expensive for use in an MRI system. Only 8 mm devices (Mammoth, SDX-300 and CY-8000) were judged adequate. The data transfer rate for the DLT4000 is not acceptable.

It was recommended that the Mammoth tape drive be used for this project, as it is directly compatible with the existing unit (it is manufactured by Exabyte). In addition, Exabyte supplies a Windows/NT device driver for the Mammoth. The Exabyte Mammoth tape drive can record at sustained throughputs in excess of 2.5 MB/sec (measured).

Two A/D cards were selected for use in the MRI - the Matrox Pulsar-LC and Data Translation DT3152. Tthe Matrox card was rejected early on due to incompatibilities between its PCI implementation and the host processors. The Data Translation card was retained and was used for early development of the video subsystem. Extended testing of the card, however, revealed undesirable tradeoffs between the PRF and the maximum programmable sampling frequency. Support for previously proven technology (PDI Select 45 hardware) was then built into the video subsystem and this card was used in the final system.

2.3 MRI Software Upgrade

A software design document was submitted to TDC by Sigma Engineering to guide the development of the MRI software upgrade. An extra module was added to support the Raytheon Pathfinder MK2 transceiver in order to convert network client requests for transceiver control and status into RS-232 I/O for serial port communication with the transceiver. The final upgrade resulted in the following software deliverables:

Server Software:	Server DLLs:	Client Applications
MRISRVR3.EXE	BASE_A2D.DLL	PlExCli.EXE
RSi3000.SYS	SIGMAIMG.DLL	SetupMRI.EXE
	SIGMAPLT.DLL	SAR-GUI.EXE
	CLIENT.DLL	

2.4 Plot Extraction Verification

Verification of the plot extraction algorithms used by Phase 2 of the MRI was conducted prior to implementation in the new system. The results of this testing are documented in Appendix A.

It was noted that large geographic masses (greater than an arbitrary value of 4096 pixels) tended to be broken down into smaller pieces before being discarded, and some remnants of these masses could be detected as false targets. It was felt that such targets could be eliminated at any time by increasing the arbitrary 4096 limit. Other than this, the algorithms proved to be reliable in reporting back the size, shape and centroid of targets. Changes to support B-scan data and extractor-specific processing were recommended to take maximum advantage of the data provided by the new system.

2.5 Radar Interface Modifications

In order to accommodate new features of the MRI an upgrade to Sigma's RSi2000 radar interface was carried out. The upgraded interface, named the RSi3000, was built for Sigma Engineering Limited by Consolidated Technologies Limited.

The RSi3000 is a full length ISA card designed to provide a flexible interface to generic radars using Azimuth Count Pulse(ACP), Synchro encoded azimuth information or a parallel azimuth word. The radar interface may be interfaced to a NMEA gyro source using a serial interface. The radar interface stores gyro, pulse period and azimuth data for each collected radar pulse. The data is stored on the RSi3000 and transferred to the PC by the RSi device driver under interrupt control. The RSi card provides synchronization signals for the PCI bus A/D card such that video digitization is absolutely synchronized with angle and gyro data.

The following are the key features of the RSi3000:

- Software control of Gain, offset, impedance and polarity of video signal.
- Measurement of pulse period in microseconds.
- Decimation and framing of radar data in order to ensure operation within the system bandwidth.
- ISA bus interface reports pulse period, antenna position and gyro to host PC.
- Software configurable signal characteristics and data formats.
- Auto-protection of inputs.
- On-board power conditioning.

The radar transceiver chosen for this project, Raytheon Pathfinder MK2, uses a composite video interface for encoding of azimuth information, making it necessary to create a Composite Video Decoder(CVD) card to convert the angle data into a parallel azimuth word. The CVD also included support for control and status of the transceiver via the RS-232 communication ports of the host PC. The CVD is a half-length ISA card that uses the host PC bus only for power. The aziumth parallel word is transferred to the RSi3000 over a ribbon cable on each radar pulse.

2.6 SAR Display Module Development

The SAR Display Module was developed to provide a real-time display of radar data during field trials. DirectDraw functions were incorporated into the GUI in order to provide a maximum display update rate of 18MB/sec. A special dialog page was added for control and status of the Raytheon transceiver via the client DLL.

The SAR Display Module supports the display of radar image data in scan converted or B-scan formats. The Display can also display plots from the Plot Extraction module and these may be overlaid on the radar image to verify plot extractor operation.

3. SYSTEM OVERVIEW

The Modular Radar Interface (MRI) is a combination of software and hardware modules that allows computers to interface to radar. Figure 1 shows the major components. The arrows represent flow of data and/or signals.

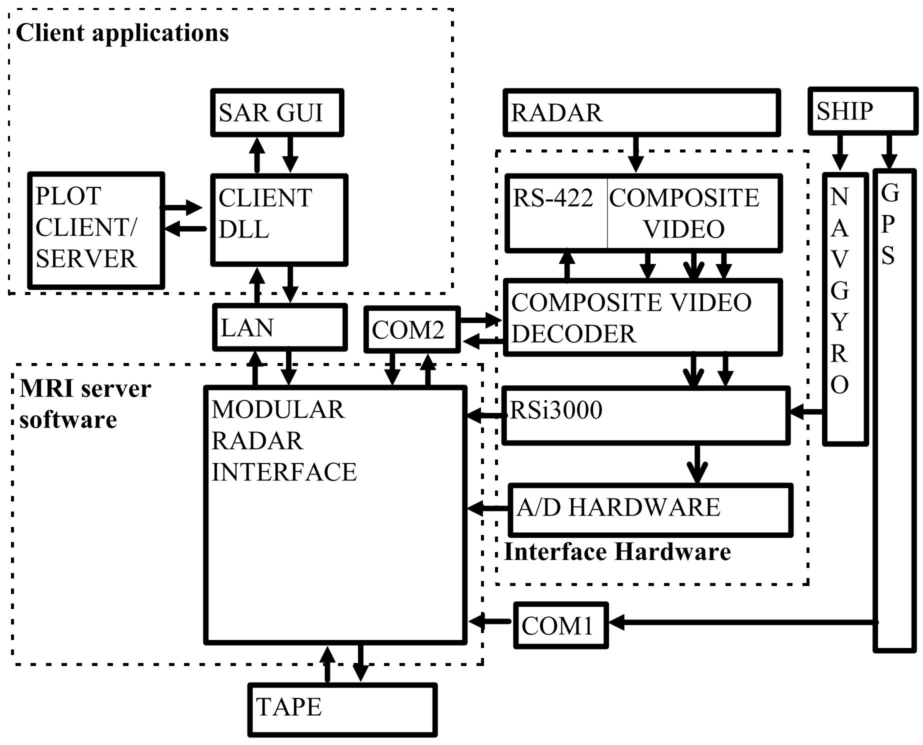

Figure 1 MRI Major Components

Components inside the dashed boxes are either software modules (Client applications and MRI server software) or MRI-specific hardware components (Interface Hardware). The other components are generic components of the host PC (LAN, TAPE, COM1 and COM2) or part of the radar and associated navigation equipment (NAVGYRO and GPS).

The server software itself is composed of software modules. A software module accepts data from a hardware device and/or other module(s). Each module applies processing to the data it receives and implements a method for making the output data safely accessible to other modules. Figure 2 shows the various modules and the communication between each. The arrows represent data and/or commands.

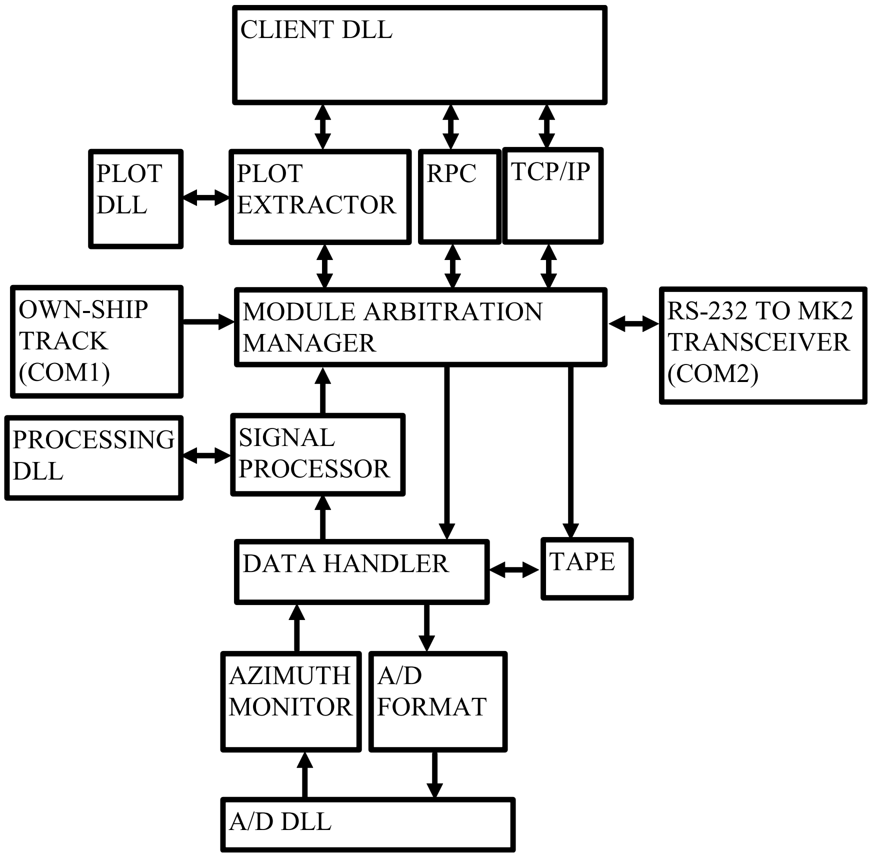

Figure 2 MRI Software Module Overview

Collected data comes from the A/D hardware to the data handler. By the time it arrives, it has been packaged with the corresponding angle and gyro data. The handler sends the raw data block directly to the tape system. It also re-formats the data into quadrants before forwarding it to the signal processor. The signal processor will perform additional processing before making it available to client processes.

The plot extractor communicates with the server either as a client application through the client DLL or directly through the manager module. The specific implementation depends on the specific installation requirements.

4. MRI SERVER (MRISRVR3.EXE)

The MRI server executable contains the necessary code for controlling the radar and interface hardware. It supports capture, recording, playback, processing, plot extraction and network distribution of radar data.

4.1 Overview

The MRI server executable consists of the following components:

- Status/configuration dialogs
- Raytheon MK1.5/MK2 transceiver interface module
- Own-ship track module (via NMEA GPS input)
- Video A/D subsystem (modules for I/O queuing, completion and routing)
- Tape subsystem (playback and recording)
- Signal processing library interface module
- LAN support (RPC/TCPIP) and module arbitration

4.2 Status/Configuration Dialogs

The status/configuration dialogs are a collection of C++ classes that perform system startup, configuration and user-interface functions. The GUI has controls for initiating selective digitization, toggling the Raytheon gear-error correction and stopping the server.

4.2.1 Startup and shutdown

Routines that perform standard startup and shutdown checks and display appropriate messages (as necessary) are stored in classes: CPhase2App, CStopExit, CExitMri, CInitErr, CCfgErr and CPhase2Dlg. These classes display the major dialogs and menus used by the system for normal operation. The menu system delivers its commands to the CPhase2App and CPhase2Dlg classes. A text area for displaying messages about the current status of the MRI server is managed by the CPhase2Dlg class.

4.2.2 Selective digitization

A set of radio buttons has been added to the Phase 3 version of the GUI. These buttons, labeled *(0 to 180)*, *(90 to 270)*, *(180 to 360)*, *(270 to 90)* and *(Off)* request selective digitization to be activated and are intended to be used for short pulse recordings on board the CCGS J.E.BERNIER.

4.2.3 Antenna error phase (TOGGLE)

It was found that the Raytheon Pathfinder MK2 transceiver reported an antenna angle that contains an error signal that repeats over a period of two scans. A correction can be applied to the angle provided that the current phase (odd or even) of the error waveform is known. On power-up and after PRF transitions, the phase of the error waveform is unknown so a phase is chosen arbitrarily. If the chosen phase is incorrect, the user must click on the TOGGLE button to switch to the correct phase. Note that the correction is not applied until azimuth gating occurs and therefore this feature has no impact on data while it is being recorded.

4.2.4 Configuration property pages

A variety of routines for initializing and then updating the property pages after confirmation by the user are contained in the C++ classes: CSetup, CPhase2Dlg and CRadarConfig.

The configuration property pages are initialized first from hard-coded values in CRadarConfig and related classes. Next, these defaults are updated based on values read from the SIGMA3.BIN configuration file (except those for the CRadarCfgSAR page). The **Configuration** menu item under the **File** menu may then be used (if enabled in the server version) to display the property pages for validation by the user. The validated values are accessible by the application directly from the GUI. For the SAR page, the updated values are also written to a SarSettings array.

Note that user validation of the values is only possible prior to starting the MRI modules. Once the modules are started, the menu item for launching the CRadarConfig dialogs is disabled.

Currently, the configuration property pages are disabled at all times in order to ensure that known values were in effect during field trials on the CCGS J.E.BERNIER. In future releases of the software, the configuration dialogs should be reenabled.

4.3 MK1.5/MK2 Transceiver Remote Interface Module

The transceiver of the Raytheon MK2 radar used during the field trial is controlled by a remote system, rather than by a Raytheon display. In this project, the MRI is the remote system but it acts merely as a pass-through system. The end user of the data sent to/from the radar is a client application such as SAR-GUI.

This control is implemented by the radar in two links - the link from the remote system to the transceiver is called the uplink and the link from the transceiver to the remote system is called the downlink. The physical uplink channel uses RS-422 protocol. The downlink channel encodes status into the video signal with each pulse output by the radar. This I/O is processed by a composite video decoder (CVD) that performs RS-232 to RS-422 conversion for the uplink and decodes the status information to complete the RS-232 downlink. Data on the downlink is formatted as a typical NMEA string. This string consists of eight bytes - a $ followed by five data bytes and terminated with a carriage return / line feed combination.

The uplink and downlink communication parameters are indicated in Table 1.

baud	data bits	stop bits	parity
1200	8	1	odd

Table 1 Transceiver Communication Parameters

While the CVD decodes the status for every pulse, the baud rate of the serial link does not permit all of the data to be sent over the downlink. New data is placed on the serial port at the rate that the baud rate permits (about 16 messages per second).

To the server, the Raytheon radar appears as a single physical unit. A module to send and receive data to/from the device was designed. The MRI uses COM2 as both the uplink and downlink. As the link to the Raytheon radar has two logical paths, two threads are used in this module, one for the uplink and one for the downlink.

4.3.1 Uplink

As long as the transceiver receives a **Transceiver Control** message more than once a second, it will continue to transmit. The module starts with the **Transceiver Control** message indicated in Table 2.

pulse length	tuning	transmit	wideband mode	STC	autotune
short	128	OFF	normal	OFF	OFF

Table 2 Default Transceiver Settings

The client application sends an array of bytes to the server to be forwarded to the transceiver. As new commands are received from the client, they are forwarded to the radar transceiver as-is, without interpretation. The server does check the message to see what type it is - normal or expansion. Expansion messages will cause the radar to generate a response message; hence the server needs to know when to expect these responses.

The MRI server uplink thread sends the **Transceiver Control** message approximately eight times per second. Expansion messages are sent immediately upon being received from the client.

4.3.2 Downlink

The downlink reads messages from the port as fast as it can. Data received is accessible to the client as an array of five 5-byte messages aligned within a forty-byte structure. The array returns each of the four possible radar responses plus the current **Transceiver Control** message being sent to the radar. The server stores the incoming messages from the transceiver into the appropriate array entry based on the message's ID.

4.4 Own-ship Track Module

The own-ship track module is responsible for providing time-stamps and location information received from a NMEA input device(GPS) in order to properly track the origin of the radar image.

4.4.1 NMEA input

NMEA input is received via the serial communication device of the PC. The software can be configured to accept a variety of NMEA-format sentences. Also, while the standard NMEA baud rate is 4800, baud rates from 1200 up to 9600 are available. Data bits, parity and stop bits must be set to 8,NONE,1. Table 3 shows the software defaults.

Serial Port	NMEA Sentence	Baud Rate	Data bits	Parity	Stop bits
COM1	RMC	4800	8	None	1

Table 3 Default GPS Parameters

Of the most commonly used GPS sentences, only the RMC string provides both time and date along with the GPS fixes. For this reason, use of RMC is strongly recommended. The sentence and baud rate can be changed via a client DLL call. The sentence may be set for auto-detection, or any of the following sentences may be specified: GGA, GLL, RMA or RMC.

4.4.2 Determination of ship's position and image origin

The own-ship track module determines the ship's position using the following C++ classes:

- CGpsSrvr
- CNMEAParser
- CRTCoordinateSystem

CGpsSrvr is the topmost derived class. It combines the other classes in a single interface, and allows the own-ship track module to be created and destroyed safely. It also declares the functions NewGpsFix and GetCurrentFix in order for the NMEA input thread to store position fixes, and for other modules to retrieve these fixes.

The CNMEAParser class contains all the logic necessary for extracting time stamps and location fixes from serial NMEA input. The class provides Start and Stop functions to begin retrieval of the NMEA input in a separate thread. The current status of the GPS input stream can be acquired via the GetStatus function. The serial port parameters and the NMEA sentence can be configured via the SetStatus function.

The CRTCoordinateSystem class contains functions for maintaining a record of the time/location fixes. Storage for the time/location fixes is defined by **`MAX_LONGLAT_HISTORY_COUNT`**. This value is currently set to 48, meaning that a history of the last 48 fixes is kept in memory.

When a module requests the ship's coordinates, the CGpsSrvr class first retrieves the key of the most current time fix from within the CRTCoordinateSystem. This allows the module to safely access the list without interfering with incoming fixes. The time and location of the ship at the specified key can be safely retrieved. In future versions, the algorithm can be easily modified to allow recursion through the time/location history if interpolation or extrapolation of the ship's course is desired.

4.5 Radar Capture, Record and Playback

4.5.1 Configuration

The configuration of the A/D modules was originally intended to retrieve its configured values from the SIGMA3.BIN file and the GUI. As implemented, values for the Heading Adjust, Gyro Adjust (North Align) Value, ACP Count, Initial Gain and Initial Offset as read from the SIGMA3.BIN file are not used by the application. The same applies for the values of the PRF upper and lower limits (for determining available pulse lengths/sampling frequencies). Future software releases will cause the application to use the values retrieved from the GUI and the SIGMA3.bin file.

4.5.2 Queuing and completion of I/O requests

Hardware specific issues are hidden from the application by making use of a DLL which handles device driver communication. The DLL expects the application to allocate and lock the buffers to which data will be transferred. It accepts a locked buffer and queues a I/O request to the underlying hardware. The application will receive notification that the DLL has new status information when the thread that queues a request enters into an alertable state using SleepEx or other similar function. The application is expected to then query the DLL associated with the request to determine the status of the I/O operation.

Queuing and completion of I/O requests are accomplished by the CLiveRadar class which itself is a collection of classes, namely the CDataObject, CStatusObject, CAzimuthMonitor and CHWFormatter classes. This collection of classes implements a generic application-level interface to the A/D DLLs. These classes work together in the following way:

- The CDataObject class is used to group together lockable buffers for azimuth, video and header information associated with a single video segment.
- The CHWFormatter uses entry points stored into the CDataObject to deliver buffers for device level access.
- Multiple CDataObjects which have been initialized for the same device/DLL are maintained by a CStatusObject class.
- The AzimuthMonitor uses entry points stored in the CStatusObject class to acquire status of the queued requests.

4.5.3 Selective digitization

The algorithm for selective digitization in azimuth works based on a count of pulses from the raw (i.e. no azimuth gating) video source. The start angle is specified as an ACP range. Video blanking stops when a video pulse is detected within the ACP range specified. Video blanking resumes when the specified number of pulses have been allowed to pass up through the higher level A/D subsystem modules. Note that even when blanking is active, the video and azimuth information is actually acquired by the hardware devices. Blanking is accomplished by making the captured video inaccessible to higher level modules.

Logic for performing selective digitization in azimuth is contained in the CStatusObject class. The CStatusObject requires that selective digitization parameters be passed to it whenever the completion status of a pending job is requested. The CAzimuthMonitor class extracts the selective digitization settings from the GUI, and converts the degree units into a start ACP range and a count of pulses. Ideally, the count of pulses would be based on the current PRF and rotation speed, and the start range would be based on configured maximum ACP count. The existing software uses hardcoded values here, assuming a maximum ACP count of 4096, and a total of 1500 pulses per scan (3000 Hz PRF @ 120 rpm).

4.5.4 Bandwidth monitoring (per quadrant)

The Handler receives selectively digitized video segments from the low-level A/D modules and packages them into quadrants (performing pulse filtering and azimuth gating as necessary) for delivery to the signal processing layer. The Handler also controls the routing of video to the tape module for recording.

Data is sent to the Handler via the NotifyHandler function of the CManager. The Handler starts up in an off-line mode and in this mode is able to accept data from the tape playback. When the radar is set online, the Handler creates a CLiveRadar class, which spawns a AzimuthMonitor thread and a HWFormatter thread. The AzimuthMonitor calls NotifyHandler to send data to the Handler in this mode. When the radar is set off-line, the CLiveRadar object is deleted, and the Handler is able to accept data from the tape playback again.

When data arrives, the Handler first determines if an adjustment needs to be made in the buffer sizes and A/D configuration in order to keep the data rate down to acceptable levels. This is usually only a concern for live capture mode. To support playback, the decided bandwidth settings are stored in the header prior to delivering the segment to the tape module for recording.

4.5.5 Recording

For live capture only, once the Handler has determined the best settings for the PRF, it locks the CDataObject and sends the data to the tape module to be recorded. If recording is not active, the SendDataToTape function returns FALSE. In this case, the Handler unlocks the previously locked CDataObject. Otherwise, the tape module is responsible for unlocking the CDataObject. Because the CDataObject may contain pointers to volatile memory, the tape module must release the CDataObject as soon as possible.

4.5.6 Playback considerations

On playback, the Handler is set up to use the recorded settings for bandwidth as is. Bandwidth, however, is primarily an issue for real-time playback. If a different format for displaying playback data is desired, the MeasureBandwidth function of the Handler may be modified to override the recorded settings.

4.5.7 Additional processing

Prior to any additional processing, the Handler checks to see if the PRF has changed or if the video pointer is NULL. If so, it requests the HWFormatter to reconfigure. If the data was successfully sent to tape and the compiler flag **`RECORDING_PRIORITY`** is defined, the Handler then releases the CDataObject and aborts processing of the current segment. This effectively prevents the data from being routed to the signal processing and client modules.

If the Handler continues with processing, it first copies the video and azimuth data into its private buffer space and releases the CDataObject. An azimuth shift can be applied to the data at this point based on the currently defined value of the **`AZIMUTH_SHIFT`** constant. Currently, the **`AZIMUTH_SHIFT`** is set to 2 to correct for a known misalignment between the video and the angles reported via the RSi3000/Composite Video Decoder.

At this point, the Handler begins updating the quadrant header to be delivered to the signal processing module. This includes reporting the mode, pulse length and PRF indexes. It then proceeds with pulse filtering and azimuth gating.

Pulse-to-pulse filtering

Pulse filtering, if requested and enabled, is performed by the Handler prior to azimuth gating. Pulse filtering uses a fixed window length as set by the compiler defined constant **`PULSE_FILTER_WINDOW_LENGTH`**. If this constant is set to 0, pulse filtering is disabled.

A utility class called CContiguousPulses stores routines for both pulse filtering and azimuth gating. When pulse filtering occurs, a 32K charge buffer is maintained in order to compensate for the half-window length delay introduced by the algorithm. By grouping the pulse filtering algorithm and the azimuth gating algorithm together in this class, the intent is for the azimuth gating algorithm to also make use of the charge buffers to correct for this delay. Currently, the delay correction is not implemented for the azimuth gating algorithm.

Azimuth gating

Following pulse filtering, the Handler performs azimuth gating in order to normalize the as captured data into a fixed image height. The image height refers to the number of pulses of radar data to be output per quadrant. The azimuth gating routine does not perform data copies itself. Instead, it generates a lookup-table mapping the offsets of the source data into a destination space with the specified height. The image height is always based on the pending bandwidth configuration. This means that if a destination quadrant buffer has already been allocated with a different height, the lookup table will not be valid for it.

4.5.8 Data delivery

The handler takes responsibility for copying the source data into the destination quadrant buffers based on the lookup tables produced by azimuth gating. If, during the current segment, a PRF change resulted in a new image height, the portion of the segment corresponding to the current quadrant will be discarded and the current quadrant will be delivered as is to the signal processing module via a call to NewDataReady. The new image height will come into effect when the Handler requests GetNextBuffer from the signal processing module.

4.6 Signal Processing Module

The signal processing module has two components, one that provides buffers for access by clients and a second that performs signal processing on the data. All processing is done in SIGMAIMG.DLL component. If this DLL is not present, the module containing the client buffers still loads.

For motion-compensated data, processing occurs in the following order:

- CFAR
- resampling
- scan conversion
- scan averaging

For uncompensated data, processing occurs in the following order:

- scan averaging
- CFAR
- resampling
- scan conversion

The processing will reset under the following conditions:

- the input header indicates a TRANSITION state
- the number of input pulses changes and motion compensation is disabled
- the number of input samples changes
- the range per sample changes
- the CFAR processing changes and motion compensation is enabled
- motion compensation has been turned on or off

If there is a discontinuity in quadrant counts, all of the quadrants in the current image are discarded. If motion compensation is enabled (resets must start at quadrant 0) then the remaining quadrants in the current scan are also discarded. Motion compensation offsets for the discarded scan are accumulated and added to the next scan.

4.6.1 Buffer management

This module provides three double buffers - two for input sources and one for output. If the processing DLL is not present, the selected input buffer is copied to the output buffer. This output buffer is the one that is normally accessed by data clients. The plot extractor normally accesses the input buffer.

When the module is started, the maximum desired input image size is passed as an argument. If this argument is zero, a maximum image size of 1 MB is assumed. The input and output buffers are allocated; the output buffer has a minimum size of 1 MB, regardless of input size, to accommodate scan conversion.

4.6.2 Resampling

If scan conversion is enabled, the DLL will scale the input data to provide the range coverage requested by the client. This scaling is accomplished by resampling the data. The number of source samples required to give the requested range coverage is mapped into 512 output samples. Source samples are replicated or decimated as required - no interpolation is performed. It is recommended that future versions of the software implement a peak detection algorithm when decimation is required.

This operation can be delayed until data is scan converted. If motion compensation is not enabled, scan conversion is the last operation so that the entire range of collected data is available to the client. If motion compensation is enabled, scan conversion is done early, so that the only range available to the client will be the one selected. If the desired range changes and motion compensation is enabled, a reset will occur.

4.6.3 Scan averaging

The current version of the signal processing module can average up to 16 scans of data. This number is limited only by the available memory on the system. The processing DLL allocates storage when it is started; by a simple modification of the DLL, more than 16 scans of data can be processed. When motion compensation is enabled, the data stored in these scan buffers is scan converted, so the minimum amount of memory required for one scan of data is 1 MB.

The basic algorithm used to implement scan averaging is a running window process, subtracting the oldest scan and adding the newest. The scan averaging routines have been optimized to give maximum performance. In order to not overload the CPU when the changing the number of scans, the maximum change has been limited to 4 when increasing and 3 when decreasing. For example, if the client changes the number of scans averaged from 2 to 10, the change will be implemented in two steps, from 2 to 6 and 6 to 10.

4.6.4 Scan conversion

Scan conversion is done using a look-up table. For each pixel in the output buffer, the look-up table has an entry containing the range (0 to 511) and bearing (0.0 to 359.99) of the source sample. The source data is searched for the proper bearing and the data is written to the output buffer.

The scan conversion algorithm has been optimized to give maximum performance. Since scan conversion only modifies a circular viewport in the square output buffer, only those pixels in the circular area are modified by the scan conversion routine. For this reason, when the data format of a buffer changes from B scan to scan converted, it must be reset by a separate operation.

4.6.5 CFAR

CFAR is performed using a rank-order filter. The data included in the data window (32 sample maximum) is sorted by increasing order, then the desired output value is selected. The selected value is subtracted from the sample at the centre of the window. Table 4 shows the parameters that are used by the CFAR process.

Window Length (1 side)	Data Offset	Rank Value
1 to 16	-150 to 150	10 to 90

Table 4 CFAR Filter Limits

The data offset is used to preserve the lower part of the noise. For a rank value of 50 and an offset of 0, half of the noise signal would be discarded.

The actual algorithm used is an indexed insertion sort. The implementation of the CFAR algorithm has been designed to maximize the speed.

4.6.6 Data delivery

Data is made available to clients when processing is completed on each quadrant. The two input channels and processed channel are delivered simultaneously. When the fourth quadrant is delivered, a complete scan is also available for access by a client, and the buffers will be toggled.

The signal processor keeps count of how many quadrants are queued for processing. There are eight quadrant buffers available. If the number of quadrants queued is six or more, the processor is falling behind. Beginning with the next quadrant 0, four consecutive quadrants are discarded. No processing is done for these quadrants (not even a copy). The output buffer will contain the last processed scan. The input buffer will contain new data. Motion offsets for the discarded scan are accumulated and added to the next one.

4.7 Plot Extractor Module

The plot extraction module analyzes radar images by consolidating areas of the image into plots based on a contour threshold. The plot extraction module has two threads. One is used for retrieval of data from the server and to scan average the data; the other is used to CFAR the data and extract plots. These threads communicate using messages. This is necessary to synchronize access to buffers in the DLL.

4.7.1 Local vs. remote implementation

The plot extractor can be run either as part of the server or as a separate client application using CLIENT.DLL. If run as a separate application, the module will fail to start if CLIENT.DLL is not successfully loaded. If run as part of the server, data transfer bypasses the TCP/IP interface and directly accesses the server buffers. However, the server would require additional memory (see sections 4.7.2, 5.3).

Due to the processing and memory requirements of the local implementation, the remote implementation was used for field trials.

4.7.2 Initialization

The plot extraction module is started with two arguments - a pointer and a base image size. The pointer is used for local implementation. If the pointer is NULL, the module will attempt to load the CLIENT.DLL to establish a connection to the server. Otherwise, the module binds to non-class functions provided in the RPCSRVR.CPP file for accessing Manager functions. The RPC module uses these same non-class functions for servicing calls that come through the RPC interface.

If the base image size is zero, a default size of 1 MB is used.

The current version of the MRI does not start the plot extractor unless **`ALLOW_PLOTS`** is defined. This permits remote implementation. For remote implementation, the plot extraction module is started in a separate EXE with the pointer parameter set to NULL.

The base module allocates two buffers for input from the server. If the memory can not be allocated, the module will not start. The module will then attempt to open SIGMAPLT.DLL. If successful, the DLL is initialized with the maximum image size. If this fails, the module will not start. Message boxes are used by both this module and the DLL to notify the user that a memory allocation error has occurred. Until the message is acknowledged, server operations will be paused.

4.7.3 Image requests

Plot extraction can run in either full scan mode or quadrant mode. Quadrant mode is preferred as it evens the load on the processor. The plot extractor will automatically switch to full scan mode from quadrant mode if it detects too many quadrant sequence errors. The current criteria for switching is 10 sequence errors. If no sequence error occurs for 15 seconds, the error count is reset to 0.

4.7.4 Masking

Masking is not enabled at this time.

4.7.5 Clutter map / CFAR

The clutter map function has not been implemented at this time. If the client enables usage of the clutter map, no processing of this type will be performed. At present, enabling the clutter map disables the CFAR process.

The DLL will accept scan converted data to be compatible with old MRI tapes. In this case, the data will not be CFARed.

The plot extraction DLL uses the same routine for performing CFAR as the signal processing DLL.

4.7.6 Scan averaging

The current version of the plot extractor can average up to 8 scans of data. This number is limited only by the available memory on the system. The processing DLL allocates storage when it is started; by a simple modification of the DLL, more than 8 scans of data can be processed.

The basic algorithm used to implement scan averaging is a running window process, subtracting the oldest scan and adding the newest. The scan averaging routines have been optimized to give maximum performance. To not overload the CPU when the changing the number of scans, the maximum change implemented has been limited to 3 when increasing and 2 when decreasing. For example, if the client changes the number of scans averaged from 2 to 8, the change will be implemented in two steps, from 2 to 5 and 5 to 8.

The DLL will accept scan converted data to be compatible with old MRI tapes. In this case, the data may be scan averaged, regardless of whether or not it has already been processed by the TITAN hardware.

If there is a discontinuity in quadrant counts, all of the quadrants in the current image are discarded.

4.7.7 Plot consolidation

A plot will only be added to the output list if its size is within the desired bounds. If a plot exceeds 4000 points, searching will stop and restart. If this happens, there will likely be remnants of the area that will now qualify as a valid plot. Searching is stopped at 4000 points because the plot extractor has only allocated enough memory to store a search list for this many points. However, this only equates to about 64 kB of storage, so this limit could be increased significantly if desired.

4.7.8 Data delivery

The plot positions are converted to range and bearing. For these conversions, the following fields in the data header must be set correctly:

- StartRange
- RangePerSample
- StartBearing
- AngleCoverage

Plots are delivered to the server on a scan basis, regardless of the data request mode.

4.8 Client/Module Arbitration

4.8.1 Manager

Requests for image data, plots and Ascope data are arbitrated and a client may only retrieve the current data once. Subsequent requests are not serviced until new data of the appropriate type arrives.

This process has not been implemented for tracks. A track client must be aware of this and handle its requests for data appropriately.

4.8.2 RPC server

The RPC module currently supports up to 10 clients. Each thread from a client application is registered with the server as a separate client. When more than 10 clients are registered, further clients are rejected.

When a client application terminates abnormally, it does not disconnect from the server. The server has no way to test for active clients. This is a problem especially when this happens frequently (i.e. during development of a client application), as the server may think it has too many clients connected. To help get around this, if a client has been inactive for more than 10 minutes, it is automatically disconnected. It may be desirable to reduce the time-out period.

4.8.3 TCP/IP server

Early benchmarking of the existing data delivery system revealed that TCP/IP data transfers were 2 to 3 times faster than RPC calls. As a result, some of the RPC calls were replaced with TCP/IP calls. To the client, this was transparent, as the changes were made within the client DLL.

This module services the following client DLL functions:

- GetData
- GetQuadrant
- GetPlotImage

A 2 MB buffer is allocated by this module when it starts. This is the maximum size block of data that can be transferred. If data was always retrieved by quadrant, a image could be up to 8 MB. This buffer is used for both normal data retrieval and plot image retrieval.

The server uses a port number of 4000. The server binds to the TCP/IP address indicated in the hosts file. It uses the **gethostname()** function to get the name of the server workstation and then calls **gethostbyname()** to get the corresponding TCP/IP address from the hosts file.

The module has two threads, one for monitoring the TCP/IP port for new connections and one for monitoring active clients for new requests.

Up to 10 clients can be active at any one time. Note: if a client terminates abnormally, the server may not be able to detect it and the socket may remain open. If this happens often enough the server may stop accepting new connection requests until some clients disconnect.

A client will send a MessageHeader to initiate data transfer. The server will respond by sending a **`DATA_RESPONSE`** structure followed by a block of data. The **`DATA_RESPONSE`** will indicate how much data is being transferred.

If an error occurs, the client socket is closed.

It should be noted that for slower CPUs where the client and server reside on the same machine, the RPC implementation may be faster than TCP/IP.

5. MRI SERVER DYNAMIC LINK LIBRARIES

In addition to the executable version of the MRI server, several dynamic link libraries (DLLs) have been produced during the course of this project. The DLLs are:

• BASEA2D.DLL	-	Used by the A/D subsystem
• SIGMAIMG.DLL	-	Used by the Signal Processing module
• SIGMAPLT.DLL	-	Used by the plot extraction module
• CLIENT.DLL	-	Used by the client application to communicate with the server

5.1 BASE_A2D.DLL Interface Documentation

The A/D Hardware DLL Interface contains the logic for control of the video and azimuth devices of which radar is comprised. By packaging this logic within a DLL, hardware-specific issues do not affect the design of application software such as the MRI server. An Application Program Interface (API) is provided for data flow control and data integrity checks at the application level.

5.1.1 Theory of operation

The DLL operates in the following way:

1. The application dedicates a HWFormatter thread to allocate and lock the buffers to which data will be transferred. The HWFormatter configures the devices when no requests are queued. It then locks the buffers and queues a request.

2. The application provides a second thread called the Azimuth Monitor. This thread must receive notice from the HWFormatter when data I/O completes. The Azimuth Monitor then queries the DLL associated with the request to determine the status of the I/O operation and to complete selective digitization of the data in azimuth.

5.1.2 Application program interface

The following entry points into the DLL are defined:

For Azimuth Monitor:

```
USHORT FAR WINAPI dllQueryVersion( PUSHORT pMonth,
                                   PUSHORT pDay,
                                   PUSHORT pYear);
```

Returns the month, day and year encoded into the DLL and/or A/D hardware.

ULONG FAR WINAPI dllGetOverallStatus(BASEA2D_STATUS * pStatus);

Returns the current mode of the DLL. The following values are valid

```
#define     BASEA2D_VALID                  0
#define     BASEA2D_FATAL                101
#define     BASEA2D_OFFLINE              102
#define     BASEA2D_SYNCHRO              103
#define     BASEA2D_SW_RESET             104
#define     BASEA2D_OVERRUN              110
#define     BASEA2D_MISS                 111
#define     BASEA2D_FAULT                122
#define     BASEA2D_ERROR                123
```

The following flags may be set to provide additional information:

Started:	The DLL is loaded
Reset:	The device is currently being reset
Configured:	The azimuth device is currently configured and a configuration is stored for the video device
Online:	The DLL has been set online
Running:	The azimuth device is responding to I/O requests
Fault:	The video device is temporarily offline
Miss:	The azimuth device detected a missed interrupt
Valid:	The video device returned a valid pointer to data
ErrorCode:	Last error code from the azimuth device
ErrorParm:	Count of errors detected since the DLL was loaded
SynchroError:	An error was returned from the azimuth device
Fatal:	Always FALSE

BOOL FAR WINAPI dllIsJobComplete(SIGMA_ASYNC_XFER * pN);

Returns TRUE if the current job is complete. Normally would update the SIGMA_ASYNC_XFER structure passed to it to reflect measured rotation speed, ACP count etc., but this has been disabled.

void FAR WINAPI dllMeasureRadarVariables(PUSHORT pRotation,
PUSHORT pHeading, PUSHORT pPulsePeriod,
PUSHORT pSequence, PUSHORT pQuadrant);

Updates the PulsePeriod value directly from the azimuth device. Returns other requested values as determined by the last call to dllIsJobComplete.

For HWFormatter Thread:

USHORT FAR WINAPI dllQueryRanges(float * pPerPelRanges,
ULONG * pPels,USHORT MaxEntries);

Returns the range and maximum collectable samples for each of the available pixel modes (up to MaxEntries) supported by the DLL. The caller must provide storage and set MaxEntries appropriately.

**BOOL FAR WINAPI dllConfigure(ULONG PixelMode, ULONG NumPixels,
ULONG Decimation,ULONG usecDesiredHSyncRate);**

Configures the hardware for the specified pixel mode, samples, decimation and timeout. Sets the azimuth A/D device online. Table 5 indicates the valid pixel modes.

Mode	Description	Samples	Block size
1	**40MHz uncompressed B-scan**	**up to 2044 x 8-bits (in 4-byte increments)**	**256 pulses (not gated)**
2	**10MHz uncompressed B-scan**	**up to 2044 x 8-bits (in 4-byte increments)**	**256 pulses (not gated)**
3	**3.75MHz uncompressed B-scan**	**up to 2044 x 8-bits (in 4-byte increments)**	**256 pulses (not gated)**

Table 5 Video A/D Modes and Parameters

BOOL FAR WINAPI dllSetVideo(USHORT Gain,USHORT Offset);

Sets the video gain and offset

void FAR WINAPI dllReset();

Prevents the DLL from queuing more I/O requests until reconfigured.

BOOL FAR WINAPI dllRestart();

Sets the video device online once the azimuth device is properly configured.

BOOL FAR WINAPI dllResyncCapture();

Stops the video device while allowing the azimuth device to continue collecting data.

BOOL FAR WINAPI dllRequestAsyncCapture(SIGMA_ASYNC_XFER * pN);

Queues a data I/O request to the hardware drivers.

BOOL FAR WINAPI dllNotifyUnlock(SIGMA_ASYNC_XFER * pN);

Reinitializes the SIGMA_ASYNC_XFER by setting the video pointer to NULL.

5.2 SIGMAIMG.DLL Interface Documentation

The signal processing DLL provides the following functions:

BOOL InitializeSystem (ULONG baseImageSize);

The InitializeSystem function is used to initialize the DLL. The argument to this function is the maximum expected image size. If the value is less than 1 MB, it is increased to 1 MB. The DLL will dynamically allocate 16 scan-average buffers plus 5 others of this size, plus a statically allocated buffer of 4 MB for the scan conversion routine. If the memory cannot be allocated, a message box will be opened informing the operator and the initialization will fail. This function should only be called once.

BOOL UpdateParameters (PROCESSING_PARAMETERS NewSettings);

The UpdateParameters function is used to load new processing parameters to the DLL. If the DLL is in a transition, this function will fail. If the CFAR processing changes from ON/OFF or vice versa and motion compensation is enabled, the DLL is reset because the scan memory buffers contain CFARed data. If motion compensation has been turned on or off, the number of scans accumulated will be reset, because the scan memory buffer format will change. These resets are queued (not implemented until processing of the next quadrant).

BOOL ResetSystem (MRI_DATA_HEADER * pNewHeader);

The ResetSystem function resets the signal processing DLL. It currently accepts a header as an argument but this is only used to check that the new image size is valid for the buffers presently allocated. This argument is probably not strictly necessary. This function clears the scan averaging accumulator buffers and resets the motion compensation offset arrays.

BOOL ProcessImage (BYTE * pInputData, MRI_DATA_HEADER InputHeader,
BYTE * pOutputData, MRI_DATA_HEADER * pOutputHeader,
MOTION_OFFSETS *pMotionOffsets);

The ProcessImage function does all the processing. Inputs to this function are input buffer and header, output buffer and header and motion offsets. Before doing anything, the input pointers and the appropriate header fields are checked. If any are invalid, this function fails. The following fields in the header must be set properly:

SamplesPerLine
Lines
RangeCoverage
RangePerSample

Before the function returns, the proper fields in the output header are modified, based on the processing performed.

void SetDesiredOutputRange (ULONG DesiredCoverageInMeters);

The SetDesiredOutputRange function copies the new desired display range into the DLL. This range coverage is only used when the data is scan converted; it does not apply to B scan format.

5.3 SIGMAPLT.DLL Interface Documentation

The plot extractor DLL provides the following functions:

BOOL InitializeSystem (ULONG baseImageSize);

The InitializeSystem function is used to initialize the DLL. The argument to this function is the maximum expected image size. If the value is 0, the function fails. The DLL will dynamically allocate 8 scan-average buffers plus 7 others of this size. If the memory can not be allocated, a message box will be opened informing the operator and the initialization will fail. This function should only be called once.

void UpdateParameters (EXTRACTION_PARAMETERS NewParameters);

The UpdateParameters function is used to load new processing parameters to the DLL. The new parameters are just written over the old ones.

BOOL DataFormatChange (BOOL FullImageMode);

The DataFormatChange function is used to inform the DLL that the update mode is changing to or from full scan mode. Changes to full scan mode are only permitted in quadrant 3.

void UpdateCoastlineMask (BYTE * pCompressedMaskBuffer,
MRI_DATA_HEADER newMaskHeader);

The UpdateCoastlineMask function loads the new mask into the DLL. The mask is expanded to 8 bits by this function.

void UpdateClutterMap (BYTE * pInputData, MRI_DATA_HEADER InputHeader);

The UpdateClutterMap function does nothing.

void GetHistogram (HISTOGRAM* pHistogram);

The GetHistogram function copies the current histogram into the supplied buffer.

BOOL ProcessImage (BYTE * pInputData, MRI_DATA_HEADER InputHeader);

The ProcessImage function does the scan averaging. Currently, up to 8 scans may be averaged. Inputs to this function are input buffer and header. Before doing anything, the input pointers and the appropriate header fields are checked. If any are invalid, this function fails. The following fields in the header must be set properly:

SamplesPerLine
Lines

void ExtractPlots (PLOTS * pPlotBuffer);

The ExtractPlots function performs CFAR and plot extraction. If data is not scan converted, either CFAR or the clutter map will be applied. The data is then masked and the histogram computed. The data is then searched for plots. If more than maximum permitted number of plots are found, further plot extraction will not be performed on the remainder of the scan. The client can configure the maximum number of plots to extract, up to an absolute maximum of 20000.

5.4 CLIENT.DLL Interface Documentation

Clients making use of the TCP/IP server and RPC server modules must use the Client DLL interface specification to control the MRI and receive data and status information. The interface specification (1) is available from Sigma Engineering Limited.

6. CLIENT APPLICATIONS

The following client applications have been tested during the development of the MRI:

Setup and Diagnostics Software	(SetupMRI.EXE)
Radar Display Software	(SAR-GUI.EXE)
Plot Extraction Client/Server	(PlExCli.EXE)

6.1 SAR-GUI

The SAR-GUI was developed in order to control the Raytheon Transceiver and display the radar data during field trials of the project. Further information is available from the SAR-GUI Manual (3).

6.2 SetupMRI

SetupMRI is a client application developed under phase 2 of the MRI. SetupMRI is intended to be a setup and diagnostic utility for the MRI server. It allows the developer/installer to exercise the majority of the functions supported by the MRI via the Client DLL interface. Status messages, radar images and A-scope data can be viewed using the utility.

SetupMRI was modified during Phase 3 in order to reflect changes made to the Client DLL Interface. Information about the operation of SetupMRI for diagnostic purposes can be found in the MRI III server manual (2).

6.3 Plot Extraction Client/Server

Plot extraction, as described in section 4.7.1, can be performed internally as part of the MRI server software, or remotely as a client/server. A plot extraction client/server was developed in order to test the remote method. This client/server makes use of reserved Client DLL function calls to retrieve images from and deliver plots to the MRI server.

6.4 Other Client/Server Support

Target tracking software has been engineered by Raytheon Canada concurrently with the upgrade of the MRI. The target tracker operates as a client/server. It uses the Client DLL to retrieve plots from and deliver tracks to the MRI server.

7. HARDWARE INTERFACES

Custom hardware interfaces were required for the server software in order to control the following components:

- Raytheon MK1.5/MK2 Transceiver Remote Interface
- MK1.5/MK2 Transceiver Composite Video Interface Hardware
- RSi3000 Interface Hardware
- Video Capture Hardware

Other hardware components of the system were controllable using software provided by the equipment manufacturer.

7.1 MK1.5/MK2 Transceiver Remote Interface Documentation

The radar transceiver chosen for the field trial was the Raytheon MK2 Transceiver. This device is controlled remotely via RS-422 input. The transceiver outputs trigger, video, antenna position and transceiver status via a single composite video signal. The speed of the transceiver antenna was modified to operate at 120 rpm in consultation with Raytheon Marine Company.

Documentation for the transceiver is available directly from Raytheon Marine Company.

7.2 MK1.5/MK2 Transceiver Composite Video Interface Documentation

A Composite Video Decoder (CVD) was produced by Consolidated Technologies under contract to Sigma Engineering. The CVD consists of a single half-length ISA card. This interface was designed to provide the physical connection for both the uplink and downlink. The uplink portion receives RS-232 signals from the MRI server and converts them to RS-422 for transmission to the transceiver. The downlink portion accepts the radar video signal and strips off the downlink data for transmission to the MRI server unit in RS-232 form.

In addition, this interface is used to extract trigger and azimuth information from the composite video output of the Raytheon transceiver for use by the RSi3000. The trigger is output on an external SMA connector; the azimuth angle is output in parallel format on an internal 14 pin connector. A ribbon cable is used to connect the CVD to RSi3000 for azimuth data.

Documentation for the composite video decoder is available from Sigma Engineering Limited.

7.3 Radar Signal Interface Documentation

RSi3000 (revision 1) and composite video decoder hardware were used for interfacing to the trigger, video, gyro and azimuth information. The outputs were conditioned by the interface in order to produce HSYNC, VSYNC and video signals suitable for digitization by OEM video capture devices. Azimuth and gyro information were transmitted to the host PC via the ISA bus.

Documentation for the radar signal interface is available from Sigma Engineering Limited.

7.4 PDI Select 45 Interface Documentation

The MRI made use of Precision Digital Images Select Monochrome 45MHz video A/D hardware for acquisition of video. This card is also used in Sigma's radar recorder product and the Windows NT driver for this card was developed by Sigma Engineering Limited for that product.

Hardware information regarding the capabilities and installation of the PDI Select 45MHz is available directly from Precision Digital Images. Device driver documentation is available from Sigma Engineering.

7.5 OEM Hardware and Documentation

Other hardware used for the recording, playback, display and distribution of the collected data was supported directly by the Windows NT operating system. This hardware includes the SCSI bus interface, Exabyte Mammoth tape drive, video display card and 100 Base-T network adapter.

8. SYSTEM INTEGRATION AND FIELD TRIALS

The system components were brought together for integration during the months of September and October 1997. Initial testing of the system was performed at the Canadian Centre For Marine Communication (CCMC). The system components were then installed on the CCGS J.E.BERNIER for final testing and field trials.

8.1 Systems Integration at CCMC

The Raytheon Radar was installed on the roof of the CCMC building for initial testing. During testing, some deficiencies in the documentation for the cabling and the serial protocol were discovered.

It was necessary to add a 12 V enable line to the uplink so that the transceiver would turn on. This was accomplished at CCMC using a DC power supply and the extra connection routed to the transceiver. For the field trial this voltage was supplied directly by the MRI.

The documentation on the Interface Protocol did not accurately specify the uplink communications parameters. A timing diagram was provided to indicate how to configure the port; however, a combination of discussions with Raytheon and trial and error process was required to determine the proper port settings. This enabled the proper operation of the basic messages required to operate the transceiver.

There are eight expansion messages (5 for the uplink message and 3 for the downlink) supported by the Raytheon MK2 radar. The uplink messages include a checksum. None of these messages is currently accepted by the radar. It appears that the checksums are not computed properly. The proper method for computing the checksum was not provided by Raytheon. All logic for handling these messages has been implemented on the server. The only modification required will be the correct computation of the checksum for the Send Transceiver Status message. The documentation indicates that this message is required to be sent to instruct the radar to resume sending normal status messages after it has received an expansion message that requires a response.

8.2 Installation of Raytheon Radar aboard the CCGS J.E. BERNIER

The installation of the Raytheon radar on board CCGS J.E. BERNIER was overseen by Oceans Limited. Radar signals for the composite video, radar trigger and radar uplink were routed via multi-core cable to the Telecom Room on the vessel.

Once the radar was installed, the MRI was installed by Sigma Engineering Limited. Initial tests with the radar revealed there was a problem that was causing target echos to be shuffled back and forth within a half-degree of their true position. This wobble had a period of two scans and was determined to be due to a periodic error in the azimuth information being reported by the Raytheon transceiver. It was found that the wobble could be removed by applying an angle correction derived from recordings of the angle data reported by the MRI. The fact that this error could be systematically removed provided confidence in the data to be collected during the trial.

During the early part of the field trial, some difficulties were experienced with the serial port. Occasionally one of the links would time out. This was especially a problem when it occurred on the uplink, as the radar would cycle on and off. To prevent this, the timeout settings on the port were lowered to 500 ms for the uplink and 900 ms for the downlink. In addition, when a timeout occurred, the port would be closed and reopened to clear any problems. It was later determined that the problem was related to a PCI bus conflict problem.

8.3 Interface to GPS

Serial GPS sentences were available from a GPS device located aboard the ship. RS-232 signals from this device were routed to the Telecom Room by the Coast Guard as requested by Oceans Limited. This input was connected to the serial port of the MRI host PC.

8.4 Interface to NavGyro

Serial gyro sentences were available from a NavGyro device located aboard the ship. RS-232 signals from this device were routed to the Telecom Room by the Coast Guard as requested by Oceans Limited. The serial input was connected to the RSi3000 interface connector. It was later determined that input errors were causing intermittent failures of the RSi3000 in parsing the Gyro sentences. A modification was made to the RSi3000 in order to correct this problem; however it is felt that using the RS-422 output signals of the NavGyro interface instead of the RS-232 outputs would have also rectified the situation.

8.5 Installation of MRI

The installation of the MRI aboard the CCGS J.E.BERNIER was conducted by Mr. Joe Ryan, Mr. Max Johnson and Mr. Stephen Motty of Sigma Engineering Limited.

Radar data collected early during the installation was used in the identification of the antenna angle encoder error. A software correction for this error was derived in order to demonstrate that the error was indeed systematic and could be removed. The correction required that a toggle button be added to the MRI Server GUI due to the fact that the error correction curve had to be applied over two consecutive scans, and toggled depending on which phase (odd or even) the antenna was in on power-up and after PRF transitions.

Problems were noted in the ability of the A/D subsystem, the tape and the MK1.5/MK2 transceiver remote interface to perform continuously in the integrated system. The problem was solved by reinstalling the SCSI, PDI, ethernet and video cards in different slots so that the SCSI and PDI cards now shared the same IRQ. The exact cause of the problem is not known, but it is assumed that a PCI bus arbitration issue exists between the SCSI card and the PDI card when they are operating at different IRQ levels.

8.6 Placement of Target Array

The placement of targets in the experiment area outside St. John’s and the deployment of drifting targets during field trials were overseen by Mr. Reg Fitzgerald of Oceans Limited.

8.7 Data Collection

Data sets were collected over a two-week period during search patterns of the target array. Each search pattern consisted of two circuit searches (one at short pulse and one at medium) and one stepped search alternating between short and medium pulse. A complete description of each search is to be presented in a summary report.

During the data collection, the radar tuning had to be monitored and adjusted using the SAR-GUI application. In general, the tuning value started at a medium value at the onset of a trip, and had to be varied as the radar warmed up. It was noted that the from time to time the radar signal strength would vary in a cyclical manner within a radar scan. It was determined that this was also related to a tuning problem with the Raytheon transceiver. This was confirmed by Raytheon after the field trial.

9. CONCLUSIONS AND RECOMMENDATIONS

In a period of less than one year, an enhanced MRI has been developed. The new MRI implements advanced processing functions in software on a Windows NT platform. It has been demonstated through extensive field trials that the system is capable of real-time processing and display for antenna speeds up to 120 rpm. At present, this technology represents the state-of-the-art in sea surveillance using microwave marine radar.

During testing and field trials, a number of unexpected problems arose. While these problems were diagnosed and solutions put into place, it was not possible to address all of the issues in as elegant a manner as would be desirable. Sections 9.1 to 9.7 present recommendations that should be considered to ensure that future work maintains an acceptable upgrade path for the system.

9.1 Extended Radar Messages

Raytheon Marine Company should be requested to provide the checksum information required for implementing the extended messages. The extended messages include control of sector blanking. This feature will be required for VTS installations.

9.2 Analysis of Antenna Angle Encoder data

The error due to the antenna angle encoder is correctable in software; however, the cause of error should be investigated and corrected prior to future data collection endeavors with this radar.

9.3 Validation of Gyro Data

While viewing the radar data in real time on the initial trial, it was noted that the north stabilization was not being maintained. The cause of the error was found to be on the input of the RSi3000 card and was corrected by changing a resistor value. It is recommended that this input be reconfigured in future generations of the interface to be RS-422.

9.4 PCI Bus Arbitration Issues

As discovered during the trial, the PCI cards used in the current version of the MRI behave differently depending on the slots in which they are installed. The behavior can result in an erratic form of operation which is difficult to diagnose without fore-knowledge of the problem. Further investigation into how PCI slots affect resources allocated to each device and the nature of the interactions between the devices may be warranted. Until the cause is known, it is recommended that new host platforms and OEM hardware be chosen with care and consideration for PCI-bus specifications and that such systems be tested soon after consignment for compatibility with the existing version of the MRI.

9.5 Signal Processing

It is recommended that a review of the signal processing algorithms be undertaken to evaluate and optimize these algorithms for the search and rescue application. The collected data set represents a significant resource in this regard and should be used to its maximum potential.

9.6 Video A/D Hardware

During the MRI upgrade, two high end video capture cards were tested for compatibility with the MRI application. The Matrox card offered great potential for use with the MRI; however, incompatibilities with the Dual Pentium Pro system proved unsolveable within the tight time schedule of the project. The Data Translation card proved to be inflexible for a generic radar application. The Precision Digital Images(PDI) card finally selected for use in the enhanced MRI proved most reliable; however, at present it is the only A/D card supported by the new MRI software. This card must be modified by Sigma to be suitable for the MRI application.

It is recommended that consideration be given to the development of a custom A/D card specifically for the MRI. The approach would be to replace the card presently being used by a card specifically suited to the radar application.

9.7 Future Work

This report documents the state of the MRI software as tested during field trials and data collection. This version of the software contains modifications, such as a Raytheon-specific configuration and the antenna azimuth encoder error correction, that were included specifically to satisfy the requirements of the field trial. Some consideration should be given as to how best remove these components from the software and improve the flexibility of the software for future projects.

REFERENCES

1. Sigma Engineering Limited, *Client DLL Interface Specification*, March 1998.

2. Sigma Engineering Limited, *MRI 3 Server Manual*, March 1998.

3. Sigma Engineering Limited, *MRI 3 SAR-GUI Manual*, March 1998.

APPENDIX A - PLOT EXTRACTOR REVIEW

Plot Extraction Algorithm

The plot extraction algorithm scans an image line by line until it encounters a pixel which exceeds the threshold. Beginning at that pixel it then searches through the blob, recursively visiting all adjacent pixels which also exceed the threshold (including diagonally adjoining pixels).

The extent of the blob is recorded by four variables which hold the points of the largest and smallest excursions for both the x and y directions. These are initialized to the first point encountered and updated if necessary as each additional point is visited. The size of the x and y extent is computed; the length of the major axis is then set to the larger. The orientation of the plot is the slope of the line between the pair of points used to determine the major axis length. The minor axis length is computed by dividing the total number of pixels in the blob by the length of the major axis. (Thus assuming a rectangular shaped blob).

The center of intensity of the plot is determined by integrating over the area of the blob with respect to intensity. The position (X,Y) is given by

$$X = \frac{\sum_i X_i * V_i}{\sum_i V_i} \qquad Y = \frac{\sum_i Y_i * V_i}{\sum_i V_i}$$

where (X,Y) is the position of the sample and V is the intensity.

Since the major and minor axes are computed separately from the center of intensity, a plot drawn on the center of intensity may have either one or both of it's axes extend beyond the geometrical extent of the blob, even for a rectangular shaped blob.

The plot extractor also reports the maximum intensity of the plot, which is simply the value of the most intense pixel found within the blob and doesn't correspond to the center of intensity of the blob.

Testing

Plot Generation

A plot generation algorithm was used to do the following:

- create blobs of various shapes (diamond, square, horizontal rectangle, vertical rectangle, plus, I, H, X)
- distribute them randomly throughout an image
- write the image to tape
- write to a log file the parameters of the blob

The MRI server was then used to read image data from tape and extract plots from it. The plot extraction parameters were set appropriately to ensure all blobs would be found and not broken up. The server was modified to output extracted plot information to a file.

Test 1

All blobs were of uniform intensity so that the center of the generated blob should correspond to the center of intensity of the extracted plot. Approximately 100 randomly placed blobs were generated for each image and 100 images were written to tape. For each image the blob parameters (position, shape, and maximum intensity) were output to a log file.

The MRI server was then run to read the images from tape, extract the plots and output to a file the plots which were extracted from each image. These results were then compared to the generated blobs using a verification algorithm, which compared the center of intensity and maximum intensity of the extracted plots to the those of the blobs.

The results were that in all cases the plot extractor correctly identified the position and maximum intensity of the blobs.

Test 2

The generation of blobs was similar to test one except that the intensity of each pixel in the blob was randomly varied. The position and intensity of each pixel was logged to a file instead of just the position of the blob. The MRI server was then used to extract the plots as before.

To verify the results, an algorithm was developed to compute the weighted sum of all points in the blob in both x and y directions, using the position returned by the plot extractor.

$$\sum_i (X - X_i) * V_i$$

$$\sum_i (Y - Y_i) * V_i$$

where:

- (X,Y) is the center of intensity reported by the plot extractor
- (X_i, Y_i) is a point in the blob
- V_i is the intensity of that point

The result should be zero for both directions if the center of intensity was correctly found by the plot extractor.

The result was zero in all cases (not exactly zero, but as close as could be expected given that (X,Y) are converted to integer values by the plot extractor), indicating that the plot extractor correctly identified the centers of intensity.

Test 3

Various oddly shaped blobs were manually generated to further test the accuracy of the plot extractor in finding the center of intensity and the orientation. These blobs were all of uniform intensity and in a variety of sizes. The plots were visually compared to the generated blobs to determine whether the resulting plots were centered correctly and with the correct orientation. In all cases the plots appeared to be centered correctly and orientated correctly.

The correct orientation of the plots is hard to verify given the problem of identifying exactly what the orientation of a blob should be. For example, given an elliptical blob, the orientation computed by the plot extractor depends on the eccentricity of the ellipse. For an ellipse which is almost circular the orientation could be in any direction. For an ellipse with a higher eccentricity (one with a larger major axis length/minor axis length ratio) the orientation tends to the orientation of the major axis of the ellipse, becoming more accurate as the eccentricity increases. For even more non-standard shapes (a half-moon shape for example) the orientation becomes even more unpredictable. However, since most targets can be expected to be somewhat elliptical in shape, this is not expected to be a practical problem.

Limitations of the Plot Extractor

Maximum Blob Size

The plot extraction algorithm searches through any blobs it encounters in the image pixel by pixel. If a single contiguous blob contains more than MaximumBlobSize pixels, the first MaximumBlobSize pixels are discarded, and the remaining pixels are used in the next search iteration. For example, a rectangular shaped blob with an area of 450 pixels and MaximumBlobSize set to 100 will result in a single plot being generated based on the last 50 pixels in the blob. Since the search of the blob is an orderly progression from one pixel to the next adjacent pixel, the previous example would give a small target centered near one end of the rectangle. Setting the MaximumBlobSize too low can result in the breakup of large targets, producing one or more smaller plots. While centered somewhere within the bounds of the original blob, these smaller plots will likely be centered near the edges of the original blob.

Jitter

For a blob with a constant geometry but with changing intensities within that area from scan to scan, the center of intensity will change from plot to plot even though the size and position of the target are constant. When continuously retrieving plots it may appear that the target is jumping around when in fact its position is not changing. This is due to the use of the center of intensity as the plot center instead of the geometrical center of the blob. No information about the geometry of the blob is returned as part of the plot other than the approximation given by the plot axes. However, any jitter will be limited by the size of the radar resolution cell and may be less, depending on the resolution of the collected data.