1997

DEVELOPMENT OF A REAL TIME GEOGRAPHIC INFORMATION SYSTEM TOOLBOX FOR OCEANOGRAPHIC SURVEY DATA ACQUISITION, MONITORING, AND PROCESSING

Stephen Michael Dzurenko Jr.
University of Rhode Island

DEVELOPMENT OF A REAL TIME GEOGRAPHIC INFORMATION SYSTEM TOOLBOX FOR OCEANOGRAPHIC SURVEY DATA ACQUISITION, MONITORING, AND PROCESSING

BY

STEPHEN MICHAEL DZURENKO, JR.

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

OCEAN ENGINEERING

UNIVERSITY OF RHODE ISLAND

1997

Abstract

A real time geographic information system (RTGIS) toolbox has been designed and implemented to assist in all phases of an oceanographic survey; including pre-cruise planning, data acquisition, survey monitoring, post-cruise report generation, data processing, and data archiving. The RTGIS Toolbox is a software system, written mainly in C and shell scripts, with interfaces to data acquisition hardware; such as DGPS, LORAN, gyro compass, echo sounder, sidescan sonar, and multibeam sonar. At the core of the RTGIS Toolbox is the public domain geographical information system (GIS) GRASS4.1. The functional capabilities of GRASS4.1 are augmented with a suite of peripheral public domain software which specifically addresses oceanographic surveying needs; including GMT for map making, Xsonar for sidescan sonar display and processing, and MB-System for multibeam sonar display and processing. The RTGIS Toolbox runs on the public domain Linux operating system which results in a workstation $10,000-$15,000 less than comparable commercial products (based on 1997 cost estimates). The Tcl/Tk graphical user interface (GUI) development system is used to provide a consistent, Motif-like interface for the RTGIS Toolbox and all its tools; such as the line planning and the data viewing tools. Real time operations are permitted by adding to the base GRASS4.1 software programs which include semaphore resource sharing, serial data acquisition, and interprocess communication via network sockets. The RTGIS Toolbox satisfies many of the surveying needs of an ocean engineering department and investigates survey solutions for commercial and government hydrographers and oceanographers.

Acknowledgments

Support for this project was provided by the "Real Time GIS" project of the NSF sponsored Ocean Technology Center (OTC) at URI and by a NOAA sponsored project for "Developing a Coastal Multibeam Hydrographic Software Toolkit."

Considerable contributions were made to this project by SeaBeam Instruments, Inc., a member of the OTC. SeaBeam granted the Ocean Mapping Development Center (OMDC), under the direction of Dr. Robert Tyce, permission to use their proprietary SeaView code. The SeaView software was ported to Linux and enhances the multibeam sonar portion of this thesis by providing real time multibeam sonar data acquisition and additional multibeam processing tools.

I would like to acknowledge my major advisor, Dr. Robert Tyce, for providing a highly technically challenging environment which promotes both personal and professional growth. I would also like to thank Dr. Tyce for his support and guidance during my study with him; his motivation has helped me perform the best that I can.

The other members of my thesis committee have each contributed to my educational growth and I would like to thank each one; Dr. Peter August who first taught me about GIS concepts and applications, Dr. John King who provided me with valuable ship time and feedback, and Dr. John Hall who was always there to help me keep things in perspective.

There are a number of people who deserve thanks for providing technical assistance along the way. Thank you to John Gann at USGS in Redwood City CA. who provided much support in the very beginning and who unwittingly taught me C programming; Bill Danforth at USGS in Woods Hole MA. who provided assistance with Xsonar and who unwittingly taught me X Window programming; Paul Cohen at Seabeam Instruments and Dave Caress at SeaBeam Instruments and Lamont-Doherty Earth Observatory both of whom provided assistance with MB-System and with whom I will soon have the privilege of working; and Scott Venckus, formerly of the University of Tennessee and "knower of all that has to do with computers."

I would like to acknowledge a very special group of people. Colleagues and friends who have helped me immeasurably both technically and personally. To tell how each has helped would fill a chapter or two. Each of you should know how important you have been. Thank you to Ron Allen, Dan Clapp, Shef Corey, Li Erikson, Bob Gampert, Michael Ingram, Erin Kammonn, Tor Pedersen, and Jim Squadrito. In case you didn't notice, that was in alphabetical order, so as not to make any of you feel less special than any other. A special thank you to Shef Corey, Li Erikson, and Bob Gampert for their time in reviewing my thesis in the final weeks.

And the best for last, the deepest thanks and gratitude goes to my family. It goes without saying that all that I do would not be possible if not for them. Their support and guidance and my desire to do my best for them guides all that I do. I dedicate this and all that I do to them. My mother, Diane Dzurenko, my father, Steve Dzurenko, my sister Holly

Dzurenko, and my everything, Jill Douglass. Thank you Mom, Dad, Holly, and Jill, I love you all very much and I thank the Lord for each of you.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

In 1990, the masters thesis of John Shannon Byrne presented the design and implementation of a software and hardware system for digital sidescan sonar data acquisition, display, and processing (Byrne, 1990). Byrne's system acquired the digital data from the EG&G 960 Seafloor Mapping System's Tape Write port, wrote the data directly to the hard disk of a MicroVax computer, and simultaneously displayed the seafloor coverage in real time (Byrne, 1990). Byrne then applied a series of digital image processing techniques to the collected sidescan imagery using the U.S. Geological Survey's mini image processing system (MIPS) and routines developed at the Ocean Mapping Development Center (OMDC). The end result was a digital mosaic of the acquired sidescan sonar data (Byrne, 1990).

In 1993, the masters thesis of Gerald Hatcher investigated the application of GIS technology to the collection of spatial data from the ocean environment on properties of the seafloor (Hatcher, 1993). Hatcher believed that the functionality of a GIS could potentially improve the management of data collection from the ocean environment (survey management) resulting in less data loss, faster data analysis, and more complete data interpretation. In addition, Hatcher realized that the survey management potential of the GIS could be significantly augmented by customizing the GIS as a real time tool. Indeed, the GIS was shown to be a very powerful tool when integrated with the existing

data acquisition systems at the OMDC, including the system created by Byrne (Tyce and Hatcher, 1993).

1.2 Problem Statement

This thesis investigates the use of a Real Time Geographic Information System (RTGIS) as the next generation oceanographic survey system. The GIS workstation becomes the primary component of the oceanographic survey system and coordinates all data acquisition, logging, and display during survey operations; permitting advanced real time survey monitoring techniques. The GIS workstation also becomes the primary tool for survey planning, data processing, and data archiving. Creating the RTGIS survey tool requires the following steps: 1) porting all software capabilities previously developed by Byrne and Hatcher to PC hardware running the Linux operating system, 2) upgrading GRASS4.0, used by Hatcher, to the next generation GRASS GIS, GRASS4.1, and 3) augmenting the capabilities of GRASS4.1 by adding a suite of peripheral software specific to oceanographic data through a consistent and easy to use graphical user interface (GUI).

1.3 Methodology

In the past, survey operations commonly required a single computer to be dedicated to acquiring and displaying the data from a single instrument. The result is an expensive waste of resources. With today's faster computers this is no longer a restriction. The work presented by this thesis moves the data acquisition from all instruments to a single computer which also logs, displays, and processes all data as desired. Moving all

software to a Linux PC workstation immediately provides the basis for a comprehensive, central, and inexpensive survey workstation.

In order to give the GIS real time capabilities, a number of utilities were added to provide high speed serial data acquisition via a direct connection to devices. A wide range of instruments have been accommodated, including differential GPS receivers, water quality sensors, single beam echosounders, and sidescan sonar systems. The MIPS system utilized by Byrne for sidescan sonar processing was replaced with Xsonar (Danforth, 1996), a fully GUI-based sidescan sonar processing system providing complete geometric and radiometric corrections, image enhancements, geographic gridding, and a number of automated processing steps, including navigation filtering and smoothing and mouse-driven bottom tracking. Then a suite of additional software was added; MB-System for multibeam sonar processing (Caress and Chayes, 1996), Generic Mapping Tools (GMT) for map making (Wessel and Smith, 1995), Data Transfer Mechanism for reliable interprocess communication (Terstriep and Weber, 1993), Ghostscript and Ghostview for hardcopy output (Merz, 1997), a suite of custom-created utilities addressing needs such as line planning, data viewing, and survey monitoring, and a suite of other software which acts in support of the aforementioned software (see Appendix A for a complete list). Several filter programs were created to import raster output from peripheral software (e.g., the raw raster swath data created by Xsonar) to GRASS4.1 data layers. The GRASS4.1 GIS provides the means for data management and advanced geographic data processing, display, and analysis techniques. GRASS4.1 and all peripheral software is

coordinated by a single top level menu, called the RTGIS Toolbox, created by the Tcl/Tk GUI development system.

1.4 The GIS

There are many definitions for GIS resulting from the wide range of applications and heterogeneous groups of users (Maguire, 1991). Most simply, a GIS is a computer based system for entry, update, edit, analysis, and display of spatially referenced data (Guptill, 1989; Maher, 1987). A GIS is distinguished from a computer mapping or computer drafting system by its ability to integrate apparently separate pieces of information through their geography, and to present combinations of original data in new ways and perspectives thereby creating entirely new information (Maguire, 1991). GISs were developed as a result of the need to evaluate multiple geographic data sets simultaneously and were conceived as far back as the mid-1960s in order to address the problems of managing and interpreting increasingly complex and voluminous earth sciences data sets (Aronoff, 1989). Today, GISs have proven to be powerful tools for the manipulation of spatial data and have provided an efficient means for data organization, analysis, and presentation (Hatcher, 1993).

GIS methods are well suited to seafloor data for several reasons. Since the majority of ocean survey data is in digital form and easily entered into the GIS database, the problem of data entry is merely in transport and transformation, as opposed to the laborious and expensive data digitizing more common with terrestrial data sets (Hatcher, 1993). Seafloor data are also characterized as having uneven distribution and quality, resolutions

which vary over several orders of magnitude (e.g., 10cm sidescan data to 50km global gravity data), and relative incompleteness (Hatcher, 1993). Therefore, the GIS's ability to manipulate and combine data with differing scales and resolutions can be applied to matters of survey planning and data storage, update, and presentation (Hatcher, 1993). Oceanographic survey vessels have high operational costs. The ability to present historic survey data to assist in optimal survey planning and prevent redundant data collection increases the cost effectiveness of the survey and reduces the dollar/data ratio. In addition, the ability of the GIS to create data formats which can be transported to other computer systems implies a greater number of users can access the data, thereby reducing the cost/user ratio of the seafloor data (Hatcher, 1993). Lastly, seafloor data collection surveys (and oceanographic surveys in general) are commonly conducted far offshore, away from any fixed reference points. The geographic framework of the GIS, in combination with ship navigation instruments (such as global positioning systems), provides an excellent reference system for locating survey areas and positioning data to a high degree of accuracy.

1.5 The Oceanographic Survey

The term "oceanographic survey" encompasses many topics. In order to assist in setting the scope of this research, this section discusses exactly what this thesis considers to be an "oceanographic survey."

Our oceanographic survey consists of the collection of bathymetry data, side scan sonar data, and water quality data. Bathymetry data can be acquired with a single beam echo

sounder because it is readily available. However, bathymetry acquisition via a multibeam sonar is provided for, if access to a multibeam sonar is available. The collected data will be geo-located by the concurrent collection of differential GPS and LORAN navigation data, magnetic heading, and ship's gyro heading. Several tools have been created and assembled for the acquisition and processing of all the aforementioned data types. These tools are discussed in chapters 2 and 3.

In order to maximize a survey's efficiency and productivity, I have organized the survey into several logical phases: pre-cruise planning, data acquisition, survey monitoring, post-cruise data reporting, post-processing data, and data archiving. The pre-cruise planning phase is typically performed in the office, days in advance of the cruise. The goal of this phase is to construct a model of the intended survey area and plan the most efficient cruise, including what data will be collected and where the survey vessel will go. The data acquisition phase entails carrying out the planned cruise and collecting the planned data. The data are acquired by computer and then available for processing. The survey monitoring phase runs concurrent with the data acquisition phase. The monitoring phase consists of tracking the progress of the data acquisition relative to the planned procedure and adjusting the remaining survey plan when necessary. This phase is made possible only due to the ability of having immediate access to the survey data. The post-cruise reporting phase is usually accomplished aboard ship after the survey is completed. In this phase, a summary of the collected data is prepared via text reports and printed maps. The last two phases operate on the collected data and do not have a well defined ending period. The post-processing phase consists of any additional data processing and

data reporting necessary. This typically consists of full image processing of side scan sonar data for creating mosaics and full cleaning of bathymetry data for hydrographic data analysis. This phase also entails preparation of high quality presentation maps. The final phase is the data archiving phase. Data archiving is actually begun in the data acquisition phase as data is logged in the data base structure at its pre-planned location. The difference between the data acquisition phase and the data archiving phase is that the latter marks the completion of a survey as the raw data and any intermediate processed data is moved off line (e.g., written to CDs) to free hard disk space for new survey data. What remains is a set of final geographic data layers (site, vector, and raster) which represent the collected survey data and become part of the permanent GIS database. Tools have been created to assist in each phase of the oceanographic survey. These tools and their role in each phase will be discussed in chapters 2 and 3.

1.6 Real Time

The term "real time" is used to describe many applications and can be misused, if not put into proper perspective. For the purpose of setting the scope of this thesis, this section discusses the term "real time" and defines it as it is used in this thesis.

In the realm of software engineering, "real time" can be generally categorized as either hard real time or soft real time. Hard real time most often refers to the design of an operating system. A real time operating system can guarantee that each program instruction will be processed within a specified minimum time. The task of the real time operating system engineer is to decrease that minimum time. Hard real time also refers to

software which runs on real time operating systems. Such software inherits the same response specifications as the real time operating system and is typically used for control systems. In soft real time, no mechanism exists to guarantee that an instruction will be processed within any specified time. Soft real time usually refers to the design of user software and implies that processes are executed such that there is no perceptible delay "to the human eye" in the response of software to its real world inputs. However, the response specifications for soft real time systems are implementation-dependent and most often imply that things happen "sufficiently" fast enough. At a minimum, soft real time software must respond fast enough to handle the input data rates.

For the scope of this thesis, "real time" means soft real time. Data are acquired, logged, and displayed to the user with no perceptible delay. Where appropriate, the response specifications of real time modules will be given.

2. System Design

Chapter 2 details the design and implementation of the RTGIS Toolbox and is divided into sub-sections which detail a specific RTGIS Toolbox sub-system. Since a large amount of original software was engineered for this thesis and numerous existing software systems were modified, at the end of most sections there is a table summarizing the software written and modified for the sub-system discussed in that section.

2.1 Concept Design

The initial Project Description for the OTC-sponsored "Real Time GIS" project states that the "project will undertake to develop a Real-Time Geographic Information System (RTGIS) as the basis of the next generation of shipboard survey data collection, monitoring, and processing systems." To this end, the following preliminary block diagram was created to begin defining the RTGIS.

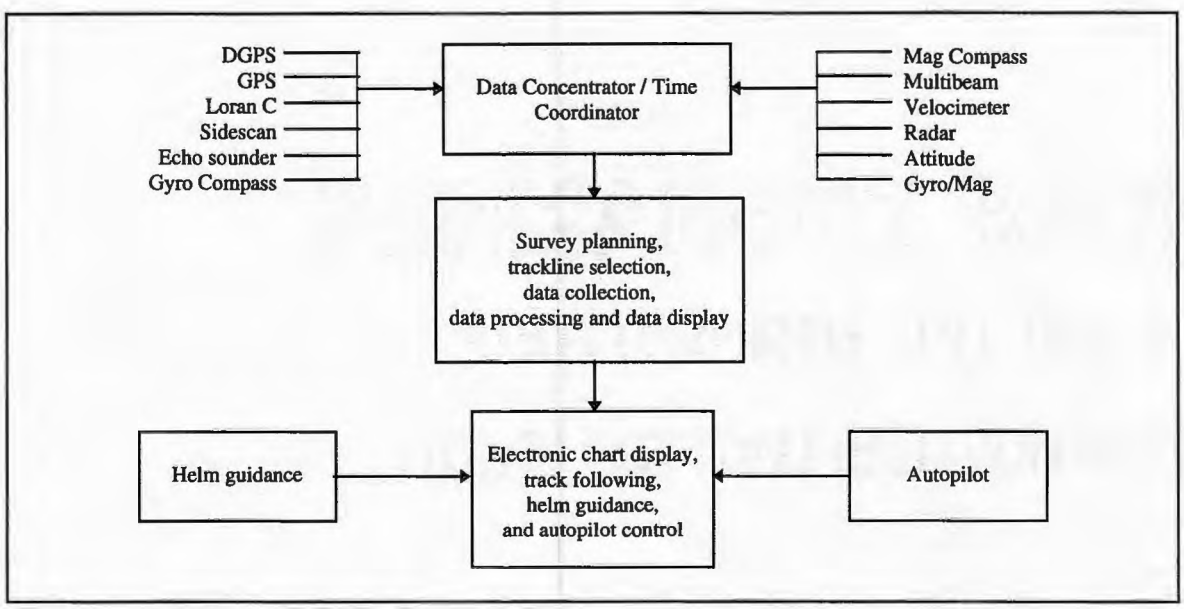


Figure 2.1 Preliminary RTGIS Toolbox block diagram.

A GIS will provide the basis of the system described by Figure 2.1. The GIS will perform the data concentration and time coordination, the survey planning, the data collection, and the data processing and display. The GIS will interface to a helm guidance system and additional data processing and display systems. The GIS chosen for this project was the Geographic Resources Analysis Support System (GRASS). GRASS will be discussed in more detail in Section 2.4. From this beginning, the RTGIS Toolbox development proceeded by identifying a set of representative inputs, outputs, and features. Table 2.1 below lists the RTGIS Toolbox selected test inputs. Table 2.2 lists the RTGIS Toolbox selected test outputs. Table 2.3 lists the RTGIS Toolbox features.

Table 2.1. RTGIS Toolbox selected test inputs.

Input	Source	Specifications
Real time navigation data	OMDC Magnavox DGPS CT-1 Furuno 920 GPS OMDC Furuno LORAN C	1 samp/sec, GGA/VTG NMEA messages 1 samp/sec, GLL/VTG NMEA messages 1 samp/sec, GLL/VTG NMEA messages
Real time gyro compass data	CT-1 ship's gyro	1 samp/sec, single floating point value
Real time mag compass data	OMDC Datamarine magnetic compass	1 samp/sec, single floating point value
Real time ship speed thru water data	OMDC Datamarine paddle wheel speed transducer	1 samp/sec, single floating point value
Real time bathymetry data	OMDC Datamarine single beam echosounder SeaBeam 2100 multibeam sonar	5 samp/sec, single floating point value or Ethernet broadcast of raw ping data
Real time water quality data	OE Hydrolab H-20 water quality sensor	variable sample rate up to 1 samp/sec or sample on command
Real time side scan sonar data	OMDC EG&G 272-TD towfish and 960 Deck Unit	8-bit parallel data at 12kHz parallel data output from deck unit paper records also acquired by deck unit
side scan sonar data files for post processing	QMIPS data files previously logged by the RTGIS	complete geometric and radiometric corrections to swath data mosaic creation and output
multibeam sonar data files for post processing	various data files previously logged by various multibeam systems	complete multibeam sonar processing and gridding

Table 2.2. RTGIS Toolbox selected test outputs.

Output	Device	Specifications
Real time navigation to helm display software	Serial port	< 1 second latency Rebroadcast NMEA messages GGA, VTG, DBT
Real time ship track display	RTGIS display monitor	Update at user-defined time interval Display track in colors indicating on or off survey line
Real time waterfall sidescan sonar display	Modified Xsonar drawing window	Display ping data as 0-255 grey scale Update every ping Provide logarithmic and linear mapping of backscatter pressure levels over the 0-255 grey levels
Real time sonar swath coverage	RTGIS display monitor	Update at user-defined ping interval Line perpendicular to ship course shows swath coverage Swath width adjusted based on water depth and user-input fish depth
Real time data logging status	RTGIS display monitor	Text displays of most recently logged data record at user-defined time interval Separate display for each device being logged Warn user of apparent log failures with audible beeps and text warnings
Hardcopy text descriptions of collected data	HP DeskJet 855Cse printer on parallel port	List of newly collected survey data Data file names and types Data file sizes and recording times
Hardcopy maps	HP DeskJet 855Cse printer and HP DesignJet 650C plotter	User-defined maps including site, vector, and raster data layers
Post processed side scan swaths	Raytheon TDU 850 thermal printer	User defined scale in Xsonar
Raw Survey Swath Data	Maxoptix Optical Drive	Optical disk mounted on survey database
Data backups	Exabyte Tape Drive	User-selected files set written to tape
Archived data	CD-ROM Recorder	User-defined file set written to CD

Table 2.3. RTGIS Toolbox features.

Feature	Specifications
To assist in survey planning, a tool should allow the user to browse the data base and view data layers previously collected in the intended survey area.	Site data layer viewing: Selection of site icon and site color Vector data layer viewing: Selection of vector color Raster data layer viewing: Selection of overlay mode Selection of GRASS display monitor Display monitor erase View data layers in uncompressed and UNIX-compressed formats
To assist in survey planning, a tool should allow the user to view the coverage of raw data files	Raster coverage of raw sidescan files Vector coverage of raw sidescan files Vector coverage of raw navigation files View raw data in uncompressed and UNIX-compressed formats
To assist in survey planning, a tool should allow the user to easily and quickly layout a series of survey lines to cover the intended survey area.	Point-and-click selection of waypoints Manual entry of waypoints Interactive editing of waypoints and lines Display waypoints and lines on GRASS display monitor Create GRASS site file showing waypoints Create GRASS vector file showing survey lines Create Maptech-compatible route file for helm guidance

Table 2.3 (continued). RTGIS Toolbox features.

Feature	Specifications
To assist in survey planning, a tool should allow the user to convert digital NOAA charts to GRASS data layers.	Import scanned NOAA chart images from proprietary sources Create GRASS xy raster layers Geographic registration of xy raster layers to UTM raster layers Mosaic separate chart images into complete chart raster
To assist in survey planning, a tool should allow the user to create a preliminary Digital Terrain Model from NOAA chart data layers.	Site digitization of NOAA chart soundings and contours Creation of site file from digit file Creation of raster data layer (DTM) from site data file
To assist in survey data acquisition, tools should provide data acquisition from a suite of serial data instruments	Modular program design to allow accommodation of numerous serial instruments: GPS navigation, LORAN navigation, magnetic heading, gyro heading, echosounder, ship speed, and water quality Simultaneous acquisition from asynchronous serial inputs. Each acquisition program runs as an independent process User configurable RS232 communication parameters
To assist in survey data acquisition, a tool should provide sidescan sonar data acquisition	Acquisition of digital data from SMS960 parallel port Conversion of parallel data to serial RS232 data Data logged in native format for future processing
To assist in survey data acquisition, a tool should provide multibeam data acquisition	Acquisition of SeaBeam 2100 data via ethernet
To assist in survey monitoring, a tool should show real time ship track	Ship track shown as vector Vector track layer created on-the-fly from raw navigation data Vector color changes with user input to show on or off survey line
To assist in survey monitoring, a tool should show real time swath coverage	Swath coverage shown as line perpendicular to ship course Swath width adjusted for water depth and fish altitude On-the-fly creation of port and starboard vector coverage data layers
To assist in survey monitoring, a tool should show real time waterfall sidescan	Grey scale color table Display every ping
To assist in survey monitoring, a tool should show real time waterfall multibeam	Multibeam bathymetry and sidescan shown User-selectable ping update interval User-selectable color tables
To assist in survey monitoring, a tool should show text updates of current log files	User-defined update interval Display most recent record for user quality assurance Warn user via text and sound if problem is encountered
To assist in survey monitoring, a tool should output data to a helm guidance system	Immediate rebroadcast of acquired position, heading, and depth data to helm guidance computer via serial line Minimal latency
To assist in survey monitoring, a tool should show data layer values at current ship position	User-selectable data layers User-defined time interval Graphical interface shows data layer values updated with current ship position
To assist in survey data reporting, a tool should output information about the collected survey data	Automated detection of new survey data Output text lists showing file name, type, size, and recording time
To assist in survey data processing, a tool should provide single beam bathymetry processing.	Programs to perform tidal correction, sound velocity correction, transducer depth offset, and merge with navigation. Procedure to create raster data layer from the site bathymetry points.

Table 2.3 (continued). RTGIS Toolbox feautures.

Feature	Specifications
To assist in survey data processing, a tool should provide sidescan sonar processing.	Complete geometric corrections: Water column removal, slant range correction Complete radiometric corrections: Destripping, beam angle and grazing angle corrections, image stretch and equalization Swath gridding and mosaicing
To assist in survey data processing, a tool should provide multibeam sonar processing.	Complete beam and navigation editing Sound velocity correction Swath gridding and mosaicing
To assist in data archiving, a tool should allow the user to copy, move, and delete GRASS database files.	Graphical User Interface to GRASS g.copy and g.remove commands
To assist in data archiving, a tool should allow the user to copy files to CDROM disks and then remove them from the fixed disk.	Interactive selection of file list for writing to CD

The RTGIS Toolbox described by Tables 2.1-2.3 can be visualized with the full system block diagram shown in Figure 2.2 below.

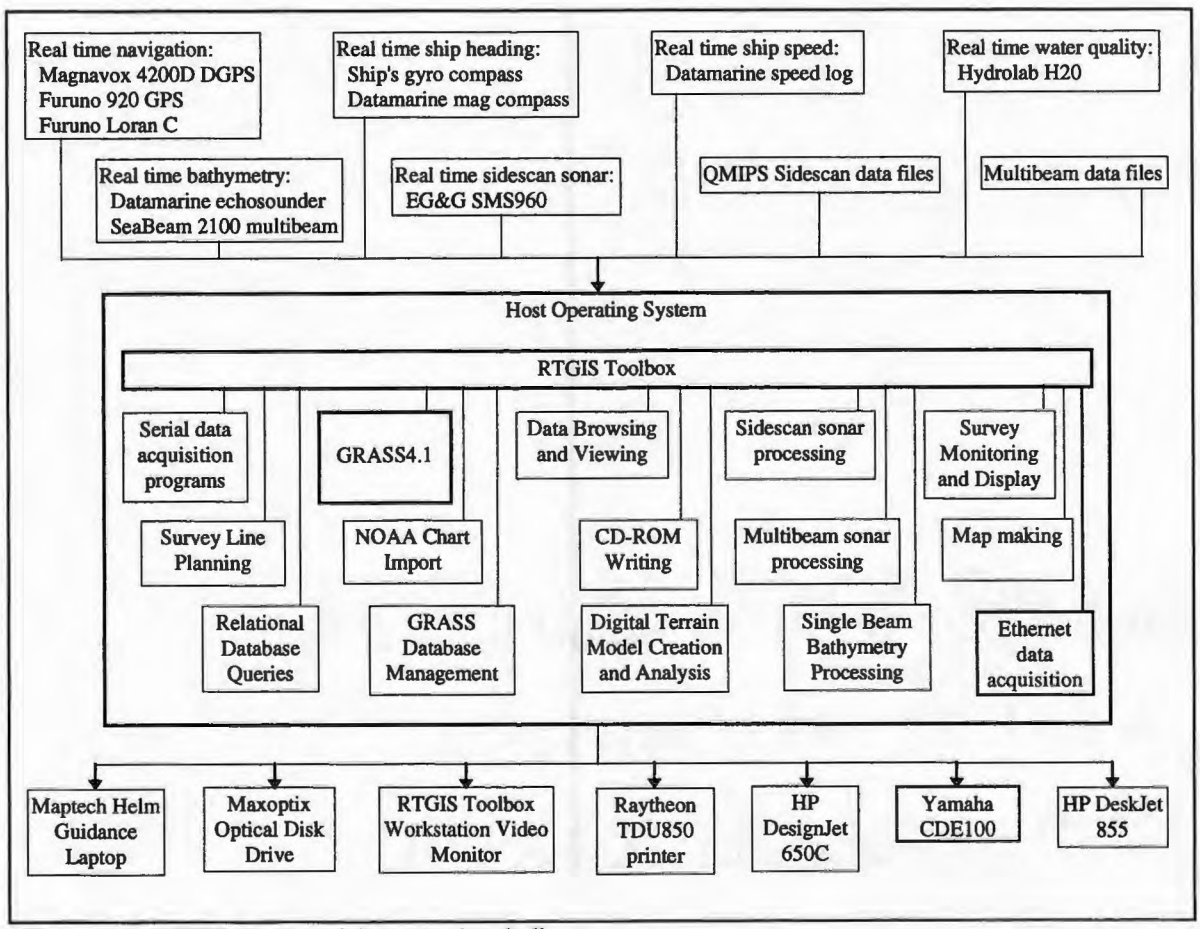


Figure 2.2. RTGIS Toolbox full system block diagram.

Finally, for the purpose of building the RTGIS Toolbox described by Figure 2.2 and Tables 2.1-2.3, the following sub-systems were identified:

1. Operating system / computer platform
2. Toolbox coordinator
3. GRASS4.1 GIS
4. Relational Database Management System
5. Historic data browsing / coverage display
6. Survey line planning
7. Digital Terrain Model creation and analysis
8. Interprocess Communication of real time modules
9. Serial data acquisition
10. Real time survey monitoring / data logging monitoring
11. Helm guidance
12. Data documentation and reporting
13. Hardcopy map output
14. Sidescan sonar data display and processing
15. Single beam bathymetry processing
16. Multibeam bathymetry processing

Each sub-system addresses an RTGIS Toolbox component or requirement. The RTGIS Toolbox was created by implementing each sub-system in a modular design approach. The remaining sections in this chapter discuss the development of each sub-system.

2.2 Operating System / Computer Platform

Linux was chosen as the RTGIS Toolbox target operating system. The target platform is an IBM-compatible PC with an Intel 80x86 processor based motherboard. Linux is a port of the UNIX operating system to the Intel 386 family of processors, including 486,

Pentium, and above. Linux was originally developed as a hobby project by Linus Torvalds at the University of Helsinki in Finland. Its development continues by many UNIX programmers and "wizards" across the Internet (Yggdrasil, 1994). Linux is a complete multitasking, multi-user operating system (just like any other version of UNIX), capable of running X Windows, TCP/IP networking, Emacs, UUCP, mail and news software, and an exhaustive list of other software (Yggdrasil, 1994). In addition, all source code for the Linux system, including the kernel, device drivers, libraries, user programs, and development tools, is in the public domain and freely available. Since Linux is a public domain port of UNIX, the GRASS community is very excited about its potential as the host operating system for GRASS GIS workstations. Linux provides the possibility of developing a completely open system, including the operating system. In addition, the ability to run all RTGIS Toolbox software on PC hardware provides a large potential user base and cuts the system cost by $10,000 - $15,000!

The Linux system was developed with portability in mind and it is mostly compatible with a number of UNIX standards on the source level, including IEEE POSIX.1, System V, and BSD features (Yggdrasil, 1994). This greatly facilitates porting software from other flavors of UNIX. In addition, benchmarks on x486 Linux systems have found them to be comparable to mid-range workstations from Sun Microsystems (Sun) and Digital Equipment Corporation (DEC) (Yggdrasil, 1994). Table 2.4 below shows some custom benchmark tests between the SGI IRIX and PC Linux systems. The benchmarks are not the typical rigorous tests of individual hardware subsystems (e.g., CPU benchmark tests). Rather, they are timed executions of user processes and are intended to provide a rough

indication of overall system performance. Each test was performed 5 times. The maximum and minimum times were discarded. The listed time is the average of the remaining three times.

Table 2.4. User benchmark tests between Linux and SGI workstations.

	SGI IRIX Workstation	PC Linux Workstation
Hardware		
CPU	100MHz IP22 R4000	133MHz Intel Pentium
RAM	32M byte	32M byte
Disk Controller	Internal SCSI	Adaptec 3940W PCI SCSI Adapter
Display Hardware	GU1-Extreme	ATI Mach 64 PCI
Test		
Time to compile PROJ4 source	163 sec.	73 sec.
Time to run r.out.ascii on raster data layer chart13223	17.6 sec.	77 sec.
Time to grid sidescan sonar swath 13-14.dat at 1m resolution	473 sec.	86 sec.

2.3 Toolbox Coordinator

A high priority of this work is to build a comprehensive "toolbox" from a suite of existing software. This implies that some entity will act as the toolbox, or container, which envelops all RTGIS Toolbox software components. The toolbox entity will be a software development system which creates graphical user interfaces which can provide front ends to existing software. Since there is a large amount of software to integrate into the RTGIS Toolbox, the toolbox software must be quick and easy to use and flexible. If the toolbox software is quick and easy to use, a maximum amount of time can be budgeted to porting software to Linux and creating interfaces to the GIS. If the toolbox software is flexible, it will allow integration of multiple, possibly dissimilar, software on the user level, rather than on the source code level which would require unwieldy amounts of code modification.

The toolbox software chosen was the Tcl/Tk system for graphical user interface development. Tcl (Ousterhaut, 1994) is a simple scripting language for controlling and extending applications; its name stands for "tool command language". Tcl provides generic programming facilities such as variables and loops and procedures. Furthermore, Tcl is embeddable. Its interpreter is a library of C procedures that can easily be incorporated into applications, and each application can extend the core Tcl features with additional commands for that application. One of the most useful extensions to Tcl is Tk, which is a toolkit for the X Window System (Ousterhaut, 1994). Tk extends the core Tcl facilities with commands for building user interfaces, so that one can construct Motif-like user interfaces by writing Tcl scripts instead of C code. Together, Tcl and Tk allow rapid application development. Compared to toolkits where you program in C, such as the Motif toolkit, there is much less to learn in order to use Tcl and Tk and much less code to write (Ousterhaut, 1994).

Tcl/Tk provides the means for quickly creating consistent graphical user interfaces to all RTGIS Toolbox component software. Figure 2.3 shows the RTGIS Toolbox (also referred to as the RTGIS Main Menu) created with Tcl/Tk. A series of menus and buttons allow the user to launch independent software with the perception of a single comprehensive software system. More complex Tcl/Tk interfaces allow the coordination of multiple command line programs to perform a specified task with the appearance of a single program (e.g., the Line Planning Tool in Section 2.7).

Figure 2.3. Tcl/Tk RTGIS Toolbox (also referred to as the RTGIS Main Menu).

The Tcl/Tk toolbox in Figure 2.3 is created by the Tcl script file RTGIS.tclsh located in the GRASS4.1 script directory. The toolbox contains 4 pull down menus and 3 command buttons. Figure 2.4 shows the 4 pull down menus and the Mapping Tools widget. The *SLS Tools* pull down menu, Figure 2.4(A), provides an interface to sidescan sonar tools. Table 2.5 lists the provided sidescan sonar capabilities. The *MB Tools* pull down menu, Figure 2.4(B), provides an interface to multibeam tools. Table 2.6 lists the provided multibeam sonar capabilities. The *Survey Tools* pull down menu, Figure 2.4(C), provides an interface to real time survey monitoring and data acquisition tools. Table 2.7 lists the provided survey utilities. The *GIS Tools* pull down menu, Figure 2.4(D), provides an interface to both GRASS4.1 command line programs and custom created tools. Table 2.8 lists the provided GIS capabilities. The *Mapping Tools* command button launches the custom created map making utility, Figure 2.4(E). The *Demo* command button launches an RTGIS Toolbox demo/tutorial. The *Quit* command button causes the Tcl/Tk RTGIS Main Menu to exit. Table 2.9 summarizes the source code written to create the full RTGIS Toolbox menu interface.

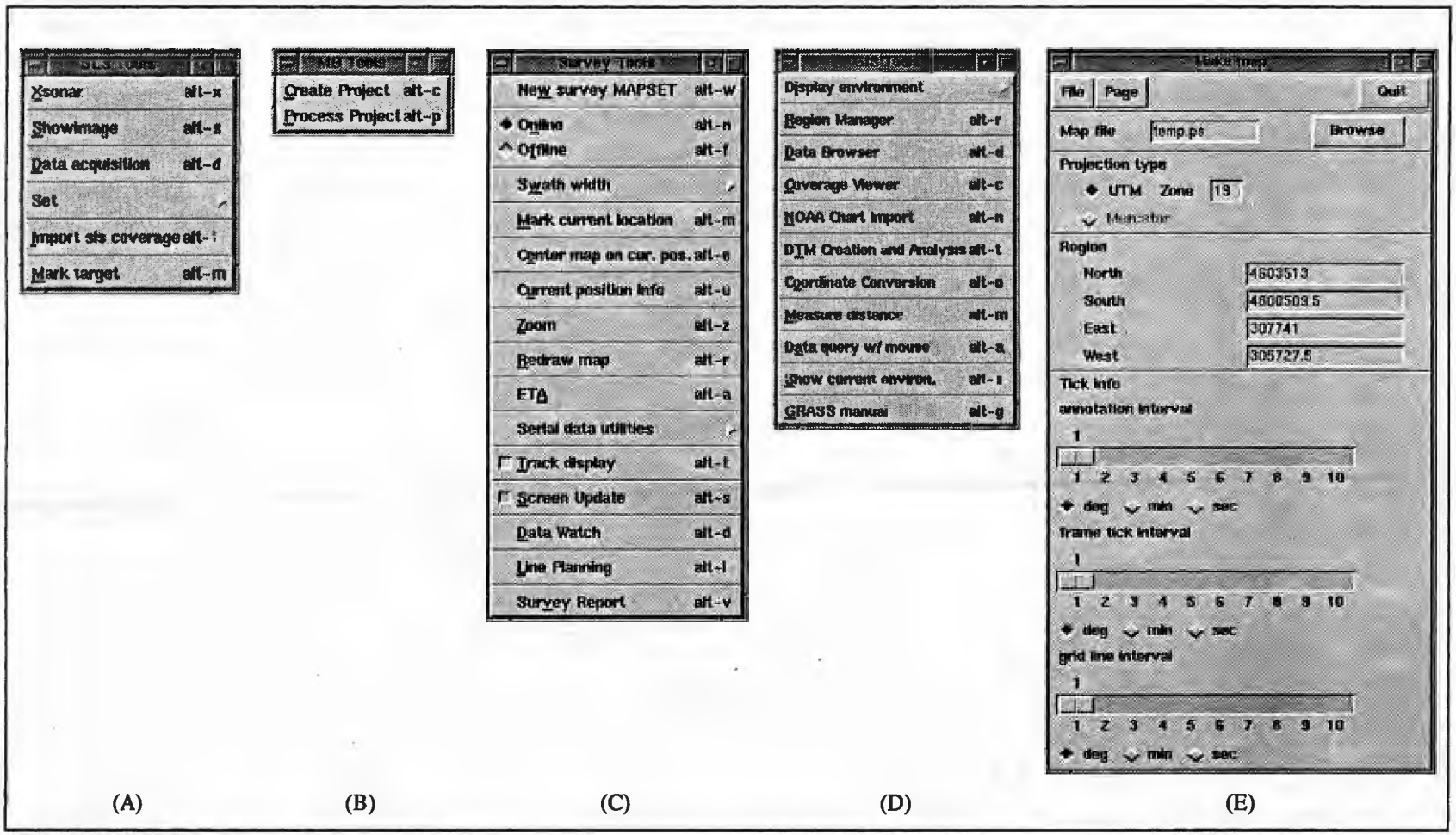


Figure 2.4. RTGIS top level pull down menus and widgets.
(A) *SLS Tools* pull down menu provides an interface to sidescan sonar tools
(B) *MB Tools* pull down menu provides an interface to multibeam processing tools
(C) *Survey Tools* pull down menu provides an interface to routines for survey operations
(D) *GIS Tools* pull down menu provides an interface to commonly used GIS functions
(E) *Make map* widget is the map making utility which is launched by the *Mapping Tools* button

Table 2.5. Sidescan tools available through the *SLS Tools* pull down menu.

Menu item	Programs interfaced to	Purpose
Xsonar	Xsonar	Complete sidescan sonar processing, gridding, and output to thermal display unit or geographic raster file
ShowImage	ShowImage	Sidescan sonar viewing and bottom tracking correction
Data acquisition	ser.SMS960 m.stop	Digital sidescan sonar data acquisition and logging
Set	m.setrange m.setlayback m.setfishdep m.setcableout	Setting various sidescan sonar survey parameters, such as range, swath width, and tow fish depth
Import sls data	f.xsonar2grass	Converting gridded output from Xsonar to a GRASS raster data layer
Mark target	m.ll2u slstarget d.icons	Marking sidescan sonar targets via a GRASS site file

Table 2.6. Multibeam tools available through the *MB Tools* pull down menu.

Menu Item	Programs interfaced to	Purpose
Create Project	UNIX commands	Set up a multibeam processing project file
Process Project	mbinfo mbedit mbnavedit mbm_plot mbfilter mbgrid	Complete multibeam sonar processing and gridding

Table 2.7. Survey tools available through the *Survey Tools* pull down menu.

Menu Item	Programs interfaced to	Purpose
New survey MAPSET	UNIX commands	Creates an empty database directory structure for a new survey
Online / Offline	m.setonline	Toggles whether the ship is currently running a survey line or not
Swath width	m.setrange	Sets the current sonar swath width
Mark current location	mark	Marks the current vessel position on the GRASS display monitor
Center map on cur. pos.	center.sh, g.gisenv, g.region, m.setenvfix, proj, newmap	Centers and redraws the GRASS display on the current vessel position
Current position info	info, g.gisenv, g.region, r.what, proj	Gets information about data layers at the current vessel position
Zoom	d.zoom.inout	Allow user to interactively zoom the display in or out
Redraw map	newmap, d.erase, d.rast, d.vect.des	Redraws the current GRASS monitor
ETA	eta	Computes estimated time of arrival to selected location
Serial data utilities	ser.setdev, ser.monitor, ser.MX4200D, ser.CT1GYRO, ser.DATAMARINE, ser.FRLS6000, ser.MAPTECH, ser.RS920, ser.SMS960	Starting and stopping serial data loggers, data monitors, and setting device communication parameters
Track display	d.nav	Draws real time ship track on GRASS display
Screen update	scr_update.sh, g.gisenv, d.graph, center.sh	Updates the GRASS display to center the ship location at a user specified interval

Table 2.7 (continued). Survey tools available through the *Survey Tools* pull down menu.

Menu Item	Programs interfaced to	Purpose
Line Planning	d.mon, d.vect, newmap, xedit, s.out.mpro, getll.sh, m.ll2u, m.u2ll, d.paint.labels, g.region, v.in.ascii	Comprehensive line planning utility

Table 2.8. GIS tools available through the *GIS Tools* pull down menu.

Menu Item	Programs interfaced to	Purpose
Display environ.	d.mon, xedit	GUI to starting, stop, and selecting GRASS display monitors
Region Manager	d.zoom.inout, g.region, d.mon	GUI to viewing and modifying the GRASS region
Data Browser	d.mon, d.erase, d.rast, d.vect, d.icons, gzip	GUI to displaying GRASS data layers
Coverage Viewer	d.mon, d.slscover, d.shiptrack, gzip	GUI to displaying raw data coverage
NOAA Chart Import	pcxtoppm, r.in.tiff, r.reclass, g.copy, d.rast, g.remove, d.mon, g.region, d.erase, r.mapcalc, run_ll2utm.tclsh, i.points, run_i.rectify2	GUI for converting digital NOAA charts to geographically registered GRASS raster data layers
DTM Creation and Analysis	d.mon, g.region, d.erase, v.digit, v.support, v.to.sites, filter_dtmsites, d.rast, s.surf.idw, r.colors	GUI for digitizing NOAA chart data layers and creating bathymetry raster data layers
Coordinate Conversion	m.ll2u, m.u2ll	GUI for converting coordinates between latitude/longitude and UTM
Measure distance	d.measure	Measure distance on GRASS display
Data query w/ mouse	d.info, d.where, proj, g.region, d.what	Gets information about data layers at the mouse selected location
Show current environ.	g.gisenv	Prints out the current GRASS environment variables
GRASS manual	g.manual	Runs g.manual, the GRASS manual, in a separate X terminal

Table 2.9. Source code written for the RTGIS toolbox menu interface.

Source File†	# lines	Description
RTGIS.tclsh	394	Creates the RTGIS Toolbox Main Menu as shown in Figure 2.3 and 2.4. Located in /usr/local/grass4.1/scripts/
convert_chart.tcl	644	Creates an interface for converting digital NOAA charts to GRASS raster layers.
coverage_viewer.tcl	355	Creates an interface for viewing raw data coverage.
data_browser.tcl	393	Creates an interface for viewing GRASS database data layers.
devices.tcl	66	Creates an interface for setting serial device communications parameters
dialogs.tcl	41	Contains several generic dialog widgets.
displayenv.tcl	50	Creates an interface for interacting with the GRASS display monitors
fileselect.tcl	280	Creates a file selection widget
getsitdesc.tcl	22	Creates a dialog for entering a comment when marking the current vessel position
importside_scan.tcl	34	Creates an interface for importing Xsonar sidescan grids as GRASS raster data layers

† All source files are Tcl scripts. All source files (except where noted) are located in the script directory /usr/local/grass4.1/scripts/tcl.

Table 2.9 (continued). Source code written for the RTGIS toolbox menu interface.

Source File†	# lines	Description
logging.tcl	268	Creates an interface for interacting with (starting and stopping) serial logging programs
logmonitor.tcl	61	Creates an interface for starting serial data log file monitor programs
mbm_destripess.tcl	181	Creates an interface for launching the MB-System program destripess on a set of multibeam data files
mbm_mbedit.tcl	128	Creates an interface for launching the MB-System program mbedit on a set of multibeam data files
mbm_mbinfo.tcl	228	Creates an interface for launching the MB-System program mbinfo on a set of multibeam data files
mbm_mbm_plot.tcl	348	Creates an interface for launching the MB-System program mbm_plot on a set of multibeam data files
mbm_mbnavedit.tcl	133	Creates an interface for launching the MB-System program mbnavedit on a set of multibeam data files
mbm_project.tcl	740	Creates an interface for creating and processing a multibeam project
mbm_util.tcl	334	Contains general support utilities for the multibeam processing interfaces
mkdataset.tcl	112	Creates default files and directories for a new survey MAPSET
region_manager.tcl	697	Creates an interface for viewing and modifying the GRASS region
run_demo.tclsh	1805	Runs the demo script
run_dtmtool.tclsh	358	Creates an interface for digitizing NOAA chart data layers and creating GRASS raster data layers
run_ll2utm.tclsh	204	Creates an interface for converting coordinates between lat/lon and UTM
run_map.tclsh	409	Creates the map-making interface
slsacquisition.tclsh	121	Creates the sidescan sonar data acquisition interface
slscableout.tcl	31	Creates a dialog box for entering the sidescan towfish cable out value
slsfishdepth.tcl	31	Creates a dialog box for entering the sidescan towfish fish depth value
slslayback.tcl	32	Creates a dialog box for entering the sidescan towfish layback value
slsrange.tcl	31	Creates a dialog box for entering the sidescan range value
slstarget.tcl	69	Creates an interface for marking a sidescan sonar target
survey_report.tcl	218	Creates an interface for viewing and printing a survey data report
trackplan.tcl	671	Creates an interface for planning a set of survey lines
userswath.tcl	30	Creates a dialog box for entering a swath width value

† All source files are Tcl scripts. All source files (except where noted) are located in the script directory /usr/local/grass4.1/scripts/tcl.

2.4 GRASS4.1 GIS

The GIS used as the basis of the RTGIS Toolbox is the Geographical Resources Analysis Support System version 4.1 (GRASS4.1). GRASS was first released in 1984 by the United States Army Corps of Engineers (USACE). GRASS is a public domain raster GIS, vector GIS, image processing system, and a graphics production system (Westervelt, 1991). GRASS is written mostly in C for the UNIX operating system (Westervelt, 1991). GRASS has been ported to a variety of computer systems ranging from 386 PCs to high-

end workstations and the only specific system requirement is that GRASS must run under UNIX (Westervelt, 1991). GRASS was originally developed to assist the USACE in complex land management and land use issues concerning US land-based construction operations. Since being released to the public domain, GRASS has been used by government, academic, commercial, and private groups around the world for diverse applications.

Because of its wide user base, GRASS has grown to a GIS suited for many diverse applications. Continued GRASS development has attempted to maintain the ability to build a GRASS-based system from freely available public domain software. The complete GRASS distribution includes user contributed programs addressing specific problems and applications and other public domain software packages, including the following:

- pbmplus which consists of over 100 image processing programs;
- Mapgen by Gerry Evenden of the USGS which is a vector plotting package supporting a large number of popular hardcopy plotting devices;
- RIM by the University of Washington, Seattle which is a relational database management system (Fox, 1990);
- and two coordinate transformation packages from the USGS, Proj4 (Evenden, 1990) and GCTP.

Continued GRASS development and release is coordinated by the GRASS Inter-Agency Steering Committee which represents users from many of GRASS's largest government users.

GRASS was chosen as the underlying GIS of the RTGIS Toolbox for the following reasons: GRASS is public domain software and freely available via the Internet; all GRASS source code is freely available; GRASS is a fully functional GIS including 3D visualization and volume analysis modules comparable to the leading commercial GISs; GRASS has a large and diverse user base which offers excellent user and programmer support via un-moderated electronic mailing lists; and GRASS's raster-based origins make it particularly good at dealing with much of the data collected by swath sonar systems. For these reasons, GRASS is particularly ideal for low budget GIS projects requiring modification and refinement of particular features.

The remainder of this section discusses the issues of porting GRASS4.1 to Linux and modifications made to the base GRASS4.1 system in order for it to be used as a real time tool.

2.4.1 Porting GRASS4.1 to Linux

Since the GRASS user community is very attracted to Linux as a free UNIX operating system, much work has already been done towards porting GRASS to Linux. In addition to the GRASS4.1 source code, pre-compiled Linux binaries are available via anonymous ftp from the CERL ftp server (ftp://moon.cecer.army.mil/pub/grass/grass4.1/release/binary/linux). Both the source code and pre-compiled binaries were obtained and installed for this work. The source code has provided a template for creating additional GRASS programs.

2.4.2 Extension to GRASS4.1 Database Structure

GRASS defines a very strict format for its database, with several hierarchical layers. These layers are directories and files within the UNIX hierarchical directory structure. The top level directory is known as GISDBASE. This is the location of all geographic data and its hierarchical structure. There may be multiple GISDBASEs, for instance, on other system hard disks, for database management purposes (Shapiro et al., 1993). For this thesis, however, all data resides within a single GISDBASE. Figure 2.5 shows a diagram of the GRASS database structure. Included are the database names and examples in parentheses. Refer to it during the following discussion.

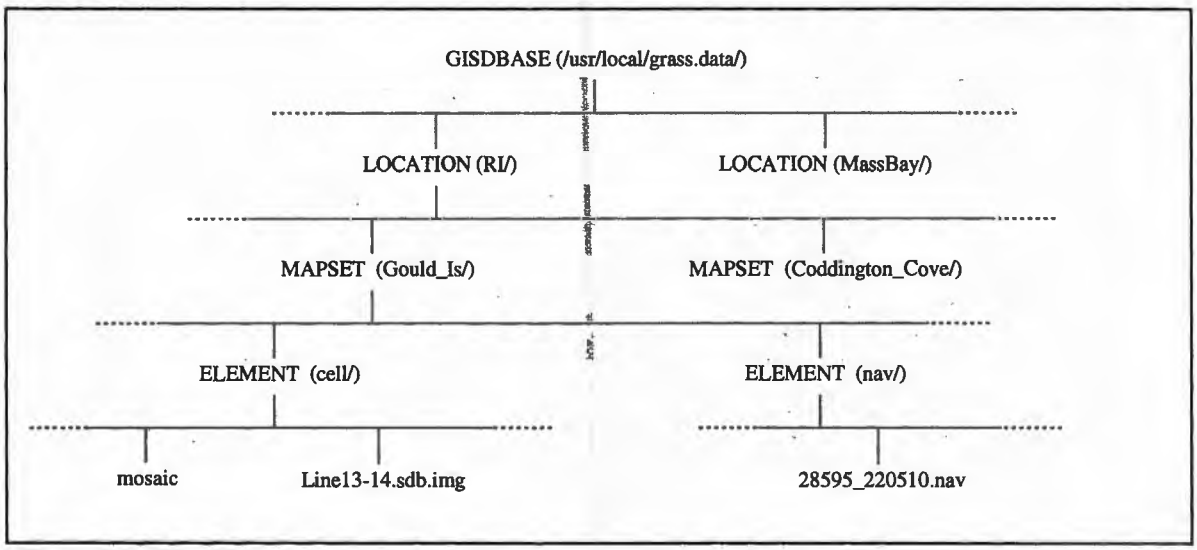


Figure 2.5. GRASS database structure.

Subdirectories below the GISDBASE are known as LOCATIONs. Each LOCATION is an independent database. LOCATIONs contain data which are related by their geographic locale. All GRASS database queries and modifications are made to a single LOCATION during a given GRASS session (Shapiro et al., 1993). It is not possible to simultaneously access multiple LOCATIONs. Subdirectories under any LOCATION are

known as MAPSETs. MAPSETs contain data that is located within the geographic locale of the LOCATION and is related by a particular theme. Modifications to the database can only be made in the current MAPSET. In addition, users may only select (and thus modify) a MAPSET that they own (i.e., have created) (Shapiro et al., 1993). However, data in all MAPSETs for a given LOCATION can be read by anyone (unless prevented by UNIX file permissions) (Shapiro et al., 1993).

MAPSETs contain directories, known as database ELEMENTs, which contain the actual data files which populate the GRASS database. ELEMENTs organize the data into groups of like format. For example, the *cell/* ELEMENT contains GRASS format raster data files and the *windows/* ELEMENT contains region files which may be used to set the current geographic region used by GRASS programs.

All real time extensions and other programs created by this thesis adhere to the GRASS database format. Survey data is logged to ELEMENTs of an appropriate MAPSET. MAPSETs are created for each survey location and may contain survey data from multiple days. The raw data file names contain the date and time of acquisition. Several new ELEMENTs were created for real time data and other data specific to the RTGIS Toolbox. Table 2.10 below contains a list of MAPSET ELEMENTs created for the RTGIS Toolbox. These new ELEMENTs have become a standard part of the RTGIS database. RTGIS programs which operate on the raw survey data expect the data to be located in the proper ELEMENT. Processed data is then written to the proper GRASS database ELEMENT. For example, the program sls.coverage will read raw sidescan data

from the *side_scan/* MAPSET ELEMENT, convert it to a GRASS-format raster layer, and write the converted data to the *cell/* MAPSET ELEMENT.

Table 2.10. GRASS4.1 MAPSET ELEMENTs created for new RTGIS Toolbox data sources.

ELEMENT Name	Data Format
NEWMAP	file containing instructions for redrawing the GRASS display being used for real time data display and update
ZOOMMAP	file containing instructions for drawing the small scale map when zooming out
nav/	raw ASCII navigation from ser.* nav loggers
heading/	raw ASCII heading from magnetic compass and ship's gyro
depth/	raw ASCII single beam bathymetry from ser.DM?? logger
side_scan/	raw and intermediate processed binary side scan sonar data files
multi_beam/	raw and intermediate processed binary multibeam sonar data files
ctd/	raw ASCII water quality data
debug/	program debugging output
devices/	default device serial communication parameters
maptech/	Maptech route files to be transferred to the Maptech helm guidance computer
speed/	raw ASCII hip speed from the Datamarine speed log
tides/	raw NOAA predicted tide tables used for processing bathymetry
svp/	sound velocity profiles

When creating a new survey MAPSET, the RTGIS Toolbox will automatically create all required MAPSET ELEMENTS. Table 2.11 below lists the source code written to support the GRASS database extensions described above.

Table 2.11. Source code written for GRASS database extensions.

Source File	# lines	Type	Description
mkdataset.tcl	109	Tcl script	Creates all ELEMENTs for a new survey MAPSET. Can be run interactively but normally launched from the RTGIS Toolbox Main Menu. Located in /usr/local/grass4.1/scripts/tcl/

2.4.3 Extension to GRASS4.1 Region Handling

GRASS can have only one geographic region active at a time. This limitation presents some problems. For example, all GRASS programs which require access to the geographic region must share the currently active region (including GRASS display programs). The implication is that two or more GRASS display monitors showing different regions (e.g., one window zoomed out to the entire survey region and another

zoomed into the current survey line) can not be drawn to at the same time without first going through the process of changing the current geographic region.

In order to solve this problem, the Region Manager Tool was created for interactively setting and modifying the current GRASS region. The Region Manager Tool, shown in Figure 2.6 below, provides an interface for setting the region manually, graphically, or from a database data file or region file. The significant feature of the Region Manager Tool is that when the Apply or OK button is selected, the selected region is not only updated as the current region, it is also written to a region file in the *windows/* MAPSET ELEMENT. The region file will be named for the currently active display monitor. RTGIS Toolbox programs which draw to the GRASS display monitor were modified to first check the *windows/* MAPSET ELEMENT for a region file by the same name as the current GRASS display monitor. If the file exists, the region is updated in the background before drawing takes place. This method allows the use of two separate GRASS display monitors showing two different regions. For example, if GRASS display monitors *x0* and *x1* are open and *x1* is the currently active display monitor and the region is modified with the Region Manager Tool, when the user selects the OK button, the current region is updated and a region file named **x1** is written to the *windows/* MAPSET ELEMENT. RTGIS Toolbox programs which draw to a GRASS display monitor will then use the **x1** region file for updating the GRASS region before any drawing is done. This feature is used by all RTGIS Toolbox tools, including real time survey tools, such as the shiptrack display, and planning tools, such as the Data Browser. However, this feature is only available through the RTGIS Toolbox. When using command line

programs, the user must still be conscious of the current GRASS display monitor and the current geographic region .

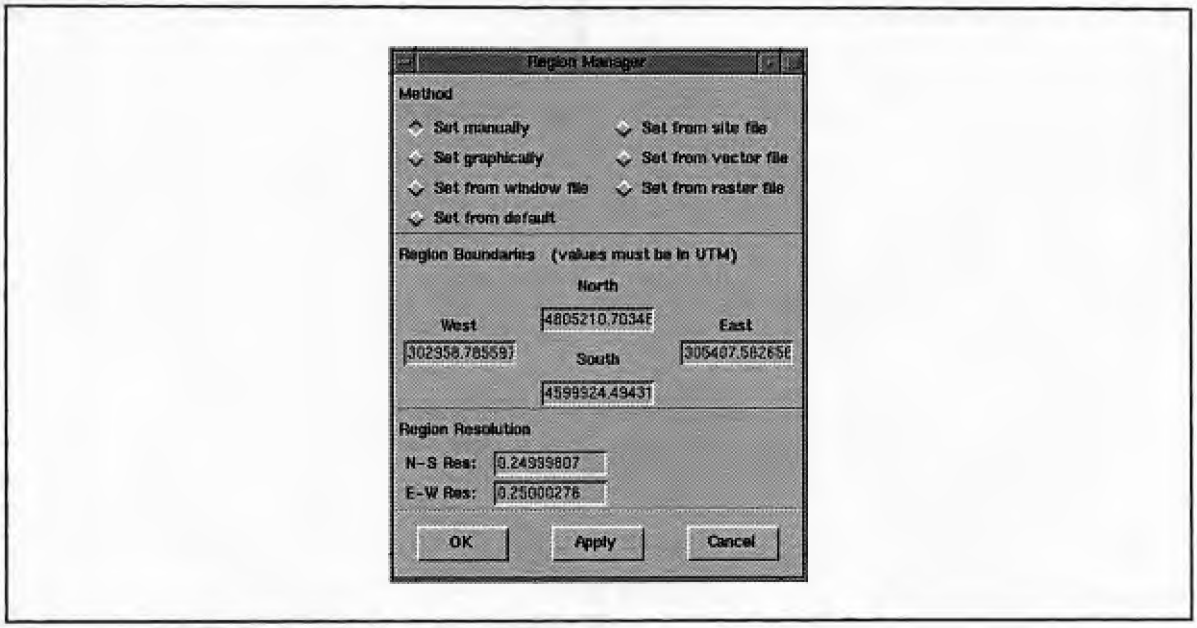


Figure 2.6. RTGIS Toolbox Region Manager Tool.

2.4.4 Extension to GRASS4.1 Library

The RTGIS Toolbox required several modifications to the standard GRASS4.1 software libraries. These include the addition of completely new libraries and bug fixes to existing libraries. GRASS4.1's general GIS library, *libgis.a*, contains functions used by other programs to access the GRASS environment variable file. These functions contain a feature which is deadly to real time operations; a *lack* of file locking. Without file locking, two or more real time processes will invariably attempt to write the GRASS environment file simultaneously. This will cause a fatal error and one or both processes will exit. In order to permit real time operations, GRASS's GIS library had to be upgraded to include semaphore file locking. A semaphore is a UNIX construct which

allows resources (such as files) to be shared by multiple processes by guaranteeing that only one process will have access to the resource at any time. The upgrade consists of the addition of two new GRASS libraries and the modification of one of the standard GRASS libraries. The first new library, *libGsemaphore.a*, provides functions callable by the GRASS library, *libgis.a*, which perform the semaphore file locking. The second new library, *libGerr.a*, provides error functions called by functions in the *libGsemaphore.a* library. The GRASS standard library, *libgis.a*, was then modified to take advantage of the semaphore functions in *libGsemaphore.a*. Specifically, the *libgis.a* source code file env.c was modified such that wherever a call is made which accesses the GRASS resource file, semaphore functions in *libGsemaphore.a* are called to lock the resource file via semaphores. Real time access of the GRASS resource file, its limitations, and the remedy provided by semaphores are discussed in greater detail in Section 2.9, *Interprocess Communication of Real Time Modules*. After the GIS library source code was upgraded and the library was re-compiled, all RTGIS modules were re-linked with the new library.

In addition to the real time upgrades, an error was found in libgis.a source file strip.c. This file contains a single function, G_strip(), required by libgis.a functions. The error occasionally caused processes to exit with a fatal system error. It has been fixed.

In order to differentiate original GRASS4.1 library source code from added code in files such as env.c and strip.c, a standard documentation procedure was adopted. At the beginning of each library source file modified, there is a comment block which begins

with the text string "OMDC RTGIS LIBRARY MOD." The comment block tells the author of the modifications and describes each modification via a numbered list. Comments are provided in the source file, such as "MOD 1," which can be used to locate the code for each modification described. Table 2.12 below summarizes the source code written and modified for GRASS4.1 library upgrades.

Table 2.12. Source code written and modified for GRASS4.1 library upgrade for real time modifications.

Source File	# lines♦	Language	Purpose
Gsemaphore.c	261(N)	C	Provides semaphore routines for semaphore file locking. This is new source code for the libGsemaphore.a library.
Gerr.c	380(N)	C	Provides error routines for Gsemaphore.c. This is the new source code for the libGerr.a library.
env.c	569(E) 76(N)	C	Source code for GRASS environment file access functions. Modified to use semaphore routines in libGsemaphore.a for file locking.
strip.c	47(E) 2(N)	C	Source code for utility function G_strip(). Modified to fix a core dump error.

♦ (N) = new code, (E) = existing code

2.5 Relational Database Management System

At the core of GRASS, and any GIS, is the geographic database. GRASS programs exist to display, analyze, manipulate, and process GRASS-format data contained in the GRASS database. However, GRASS does not contain relational database management programs for performing relational queries on its database. The GRASS4.1 distribution does include the RIM public domain Relational Database Management System from the University of Washington (Fox, 1990). RIM consists of a SQL-like command language and provides relational queries and report generation on a RIM database. A RIM database may be thought of as a collection of one or more tables. These tables (also called relations) consist of rows (tuples) and columns (attributes). The RIM database consists of all the column and table definitions, all of the table data (rows), and

supplementary information such as passwords. Figure 3.3 from the RIM User's Manual (Fox, 1990) is repeated here as Figure 2.7 below. It provides an example of a RIM relational database.

atomic_no.	symbol	resistivity	name
29	Cu	12.2	Copper
13	Al	12.9	Aluminum
26	Fe	20.3	Iron
-NA-	LB	-NA-	Bernstein
92	U	-MV-	Uranium

conductors table

id	name	position
5	John Jones	Tech 1
22	Jim Smith	Tech 2
35	Joe Jackson	Tech 1

Techs table

symbol	id	date	time	resistivity
Cu	5	88-01-21	08:10	11.9
Cu	5	88-01-21	10:32	12.4
Al	35	88-02-10	13:45	13.0
Al	35	88-02-11	09:34	14.3
Cu	22	88-02-22	08:48	12.5
Fe	5	88-03-04	15:33	19.4

measures table

Figure 2.7. Figure 3.3 from RIM User's Manual showing a sample RIM database.

In order for RIM to be useful for performing relational queries on the GRASS database, a filter was created to extract pertinent information from GRASS database files and organize it into RIM database tables. Figure 2.8 shows the tables created for representing the GRASS database in a RIM relational database.

The database was designed to test four specific types of relational queries. The first type of query interrogates the geographic location of the data files. It is used by several

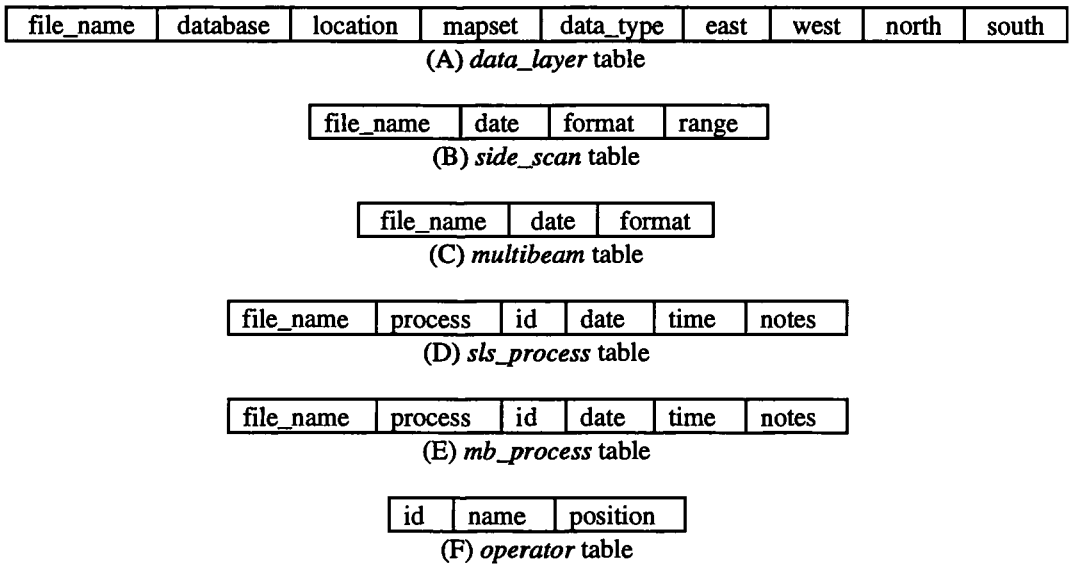


Figure 2.8. RIM tables created for relational database queries on GRASS database files.

programs for determining which data files are located within a specified geographic region. The *data_layer* table is the primary table used by this query. The second type of query interrogates raw data files for parameters regarding their acquisition. This query is used primarily for determining parameters regarding raw sidescan sonar and multibeam sonar data files. The *side_scan* and *multibeam* tables are the primary tables used by this query. The third type of query interrogates the processing history of raw data files. Like the previous query type, this is used primarily with sidescan sonar and multibeam sonar data. This query allows the user to see who has done what to each raw data file. The *sls_process*, *mb_process*, and *operator* tables are the primary tables used by this query. The fourth type of query utilizes all tables for user-defined report generation.

The RIM database is given the same name as the GISDBASE it describes. The database files are located in the GISDBASE directory. Figure 2.9 shows the RIM code used to create the database described in Figure 2.8. After the RIM database was created, a

```
define grass.data
columns
    file_name     text      var     format a32
    database      text      var     format a32
    location      text      var     format a32
    mapset        text      var     format a32
    data_type     text      var     format a12
    east          double            format f12.4
    west          double            format f12.4
    north         double            format f12.4
    south         double            format f12.4
    M_date        date              format 'yy-mm-dd'
    M_time        time              format 'hh:mm'
    format        text      var     format a32
    range         double            format f6.1
    process       text      var     format a32
    id            int               format i5
    notes         text      var     format a64
    name          text      var     format a32
    position      text      var     format a32
tables
    data_layers with file_name database location mapset data_type east west north south
    side_scan with file_name date format range
    multibeam with file_name date format
    sls_process with file_name process id date time notes
    mb_process with file_name process id date time notes
    operator with id name position
links
    M_F1 from file_name in data_layers to file_name in side_scan
    M_F2 from file_name in data_layers to file_name in multibeam
    M_F3 from file_name in side_scan to file_name in sls_process
    M_F4 from file_name in multibeam to file_name in mb_process
    M_I1 from id in sls_process to id in operator
    M_I2 from id in mb_process to id in operator
end
```

Figure 2.9. RIM code used to create the GRASS RIM relational database.

program was created to initially populate the database. The program, rim.update, is also used by the user to update the RIM database any time data is added to the GRASS database (i.e., the acquisition of raw survey data or the addition of fully processed data). Rim.update compiles a list of all LOCATIONs, MAPSETs, ELEMENTs, and data in the GISDBASE. It then compares that list to the files in the *data_layers* table of the RIM database. Each file missing from the RIM database is then added to it. In order for this database update method to be successful, rim.update needs to know all the tables and

columns in the RIM database, how to read each type of data file, and which columns apply to which data types. All this knowledge is coded into rim.update for the database described by Figure 2.8. However, each time a column or table is added to the RIM database, the system developer must update and re-compile the rim.update program to reflect the database additions.

Lastly, a Tcl/Tk interface was created for viewing the RIM database and printing reports. The RIM viewer simply reads the RIM database and presents a list of tables. All the row and column data for each table is printed in the text widget of the RIM viewer. The user may view any table by selecting the radiobutton to the left of the table name. The text widget will automatically scroll to the selected table. The user may also scroll through the tables manually with the text widget's scrollbar. Tables, rows, and columns are selected for printing by selecting them on the text widget. An entire table may be printed by selecting the table name. A specific row in a table may be printed by selecting the row. Printing of table rows may be limited to a subset of columns by selecting the column names instead of the table name. When all desired information is selected, pressing the *print* button will generate a report.

2.6 Historic Data Browsing / Coverage Display

This sub-system was conceived to assist in survey planning. The idea is to utilize the GIS's ability to overlay multiple data layers to show where data have been previously collected relative to the intended survey area. This need arose during a series of surveys near Coddington Cove, RI. Several cruises to the same area were performed on 3 non-

consecutive days over a period of several weeks. However, there was not time to fully process the data between cruises. The result was a frustrating lack of knowledge about where data had been previously collected. Displaying the ship navigation allowed us to at least know where the ship had been. But this was a crude and inefficient mechanism. It provided no insight to where sidescan sonar data had been collected and at what range. There was no coverage information. This problem was alleviated by building two tools. The first is a Tcl/Tk tool for selecting historic data layers for viewing in a GRASS display monitor. The second is a program to read through raw data files and produce GRASS data layers showing raw data coverage. Coverage display is provided for raw navigation and sidescan sonar in the form of QMIPS data files.

The Data Browser, shown in Figure 2.10, provides a GUI to the GRASS command line programs d.mon, d.erase, d.rast, d.vect, and d.sites, including all program options. Data Browser reads through each MAPSET in the user's SEARCH_PATH file and lists all raster, vector, and site files found. Note the notation used to list the files in Figure 2.10. The '@' symbol is used to indicate which MAPSET the data layer is from. This follows the GRASS standard for naming data layers. The user may then view any listed data layer by selecting it, setting the options, and selecting the *Show* button. Data Browser also allows any GRASS monitor to be drawn to and cleared. Note also that some files are compressed (files that end with a ".Z" or a ".gz" extension). Data Browser will uncompress, draw, and re-compress compressed data files automatically. This is a very powerful tool. It allows all data files to reside on the fixed disk in compressed format. This has the potential for saving large amounts of disk space. Data Browser will

seamlessly display compressed data with no need for the user to explicitly uncompres

Figure 2.11 shows an example GRASS display created with Data Browser. The user displayed the NOAA chart for the area, a previously processed sidescan mos bathymetry contours, coastline, and a series of core sites. Knowing where the sidesca seeing the bottom characteristics in its true geographic location along with the locat of previously collected cores provides an valuable tool for planning future core sites.

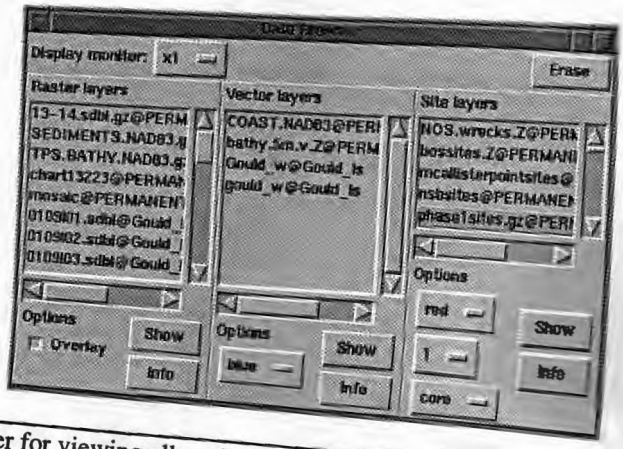


Figure 2.10. Data Browser for viewing all available raster, vector and site data layers interactively.

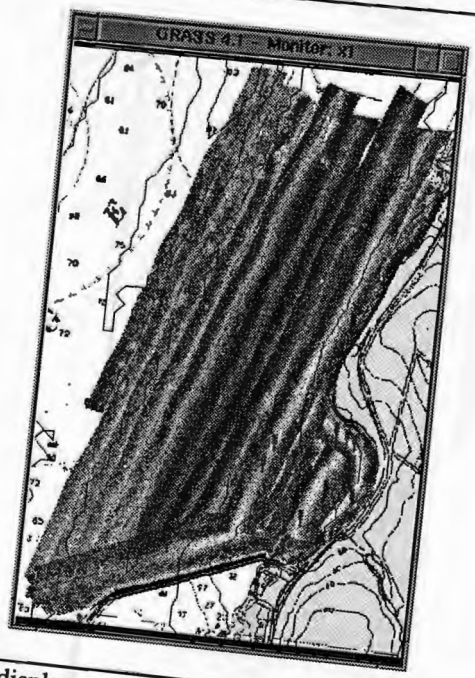


Figure 2.11. An example GRASS display created by Data Browser.

Coverage Viewer, shown in Figure 2.12, provides a GUI to the command line programs d.shiptrack and d.slscoverage. Coverage Viewer will read through each MAPSET in the user's SEARCH_PATH file and list all raw navigation, depth, and sidescan sonar files found. The file naming notation is the same as for Data Browser so the file name and MAPSET are specified in one name. Coverage Viewer is similar to Data Browser in both its interface and operation but different in the type of data file on which it operates. Like Data Browser, Coverage Viewer will display site, vector, and raster data on the GRASS display monitor. However, unlike Data Browser, Coverage Viewer creates those GRASS data layers from the raw data files on-the-fly. This is a very powerful feature. It allows immediate viewing of raw data coverage prior to full processing. Such displays are useful for preparing cruise reports and planning successive cruises based on such prior coverage. Like Data Browser, Coverage Viewer will also automatically decompress and re-compress compressed data files. Again, this means that large raster and ASCII data files may be stored on the fixed disk in compressed format, saving valuable disk space.

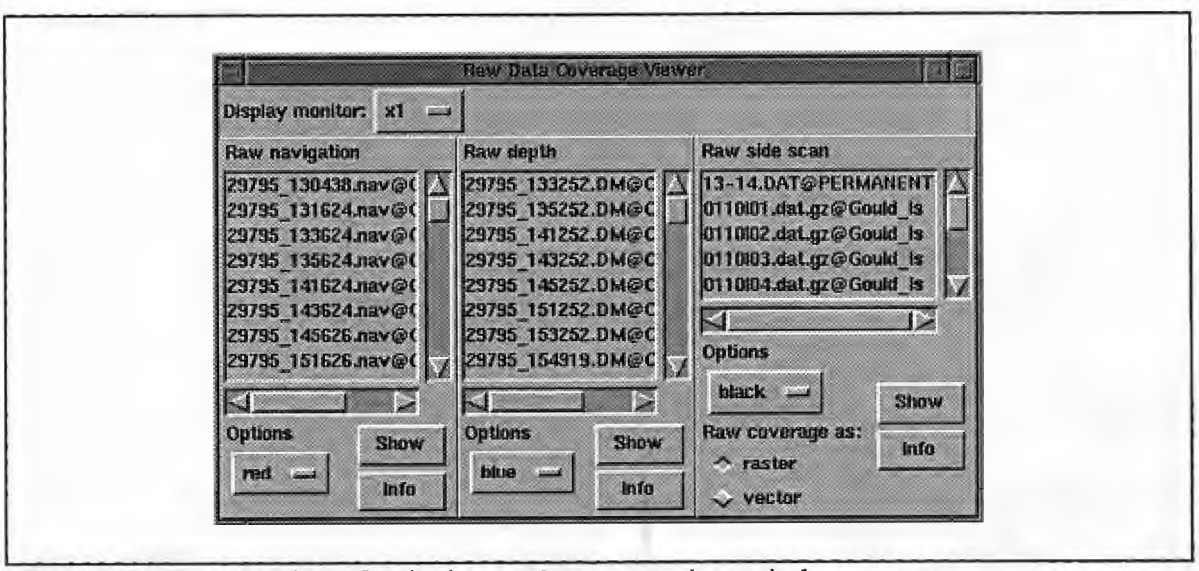


Figure 2.12. Coverage Viewer for viewing raw data coverage interactively.

Table 2.13. Source code written for Historic Data Browsing / Coverage Display sub-system.

Source File	# lines	Type	Description
data_browser.tcl	393	Tcl script	Generates the graphical user interface for selecting data for display. Located in /usr/local/grass4.1/scripts/tcl
coverage_viewer.tcl	355	Tcl script	Generates the graphical user interface for selecting the raw data for display Located in /usr/local/grass4.1/scripts/tcl
d.slscover	1517	C	Reads through raw QMIPS sidescan sonar files, creates raster and vector data layers which show the raw sonar coverage, then displays the raster and vector coverage Located in /usr/local/grass4.1/src.OMDC/display/d.slscover/cmd
d.shiptrack	469	C	Reads through raw navigation data files, creates vector data layer showing raw navigation, then displays the coverage. Located in /usr/local/grass4.1/src.OMDC/display/d.shiptrack/cmd

2.7 Survey Line Planning

To plan and execute an efficient survey it is often critical to know where one has previously acquired data. With that knowledge, it is equally important to communicate where one would like to acquire data next. This usually implies telling the survey vessel where to steer and is typically accomplished by laying out a series of lines which define the desired path of the survey vessel. By displaying the lines and the survey vessel's navigation on some form of helm display, we can best ensure that data will be collected in the proper location.

Several commercial helm guidance packages were evaluated, including Navigate! from Fair Tide Technologies, Maptech Professional from Resolution Mapping, and YoNav from the USGS. Both Navigate! and Maptech Professional offer some route planning capabilities but were obviously designed primarily for pleasure boaters. At the time these packages were reviewed, neither had a strong line planning interface nor did they have features typically attractive to survey planners. Below is a list of the line planning

features commonly desired by oceanographic and hydrographic surveyors compiled from personal experience.

1. Coastline backdrop
2. Bathymetry backdrop
3. NOAA chart backdrop
4. Cursor tracking with geographic position display
5. Manual waypoint entry
6. Point-and-click waypoint entry
7. Waypoint editing prior to selection
8. Waypoint deletion and automatic line adjustment
9. Automatic track reversal
10. Generation of a series of offset lines
11. Output to GIS data layers
12. Output to helm guidance software

Table 2.14 provides a checklist of the features above for each of the programs reviewed, including the resulting RTGIS Toolbox Line Planning Tool which will be discussed in more detail below.

Table 2.14. Line planning features attractive to oceanographic/hydrographic surveyors.

Feature	Navigate!	Maptech	YoNav	RTGIS
Coastline backdrop	✔		✔	✔
Bathymetry backdrop	✔		✔	✔
NOAA chart backdrop		✔		✔
Cursor tracking w/ geo display	✔	✔	✔	
Manual waypoint entry			✔	✔

Table 2.14 (continued). Line planning features attractive to oceanographic/hydrographic surveyors.

Feature	Navigate!	Maptech	YoNav	RTGIS
Point-and-click waypoint entry	✔	✔	✔	✔
Waypoint editing			✔	✔
Waypoint deletion & auto line adjust			✔	✔
Auto track reversal		✔	✔	✔
Generation of offset lines			✔	✔
Output to GIS				✔
Output to helm guidance software	✔	✔	✔	✔

Examining Table 2.14, YoNav, written by John Gann at the USGS (Gann, 1992), appears to rank highly. Indeed, YoNav was found to have an ideal line planning tool. The problem is that YoNav is not a UNIX software and could not be integrated into the Toolbox in its current form. Therefore, a Tcl/Tk script was created to emulate the line planning functionality of YoNav. By interfacing to the GIS, this new line planning utility offers even more features. Figure 2.13 shows the created Line Planning Tool.

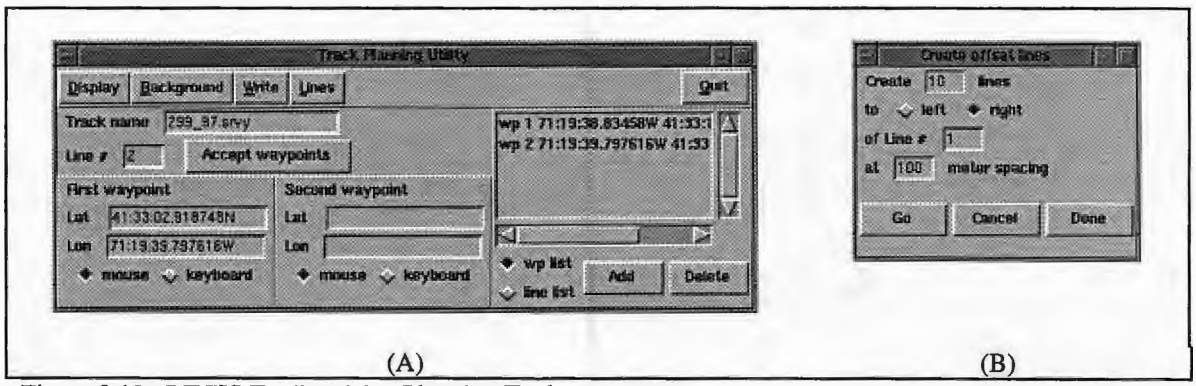


Figure 2.13. RTGIS Toolbox Line Planning Tool.
(A) the main window showing menu, waypoint entry interface, and waypoint/line list
(B) the widget for generating a series of regularly spaced lines offset from a base line

2.8 DTM Creation and Analysis

DTM Basics

Perhaps the most common tool used when planning coastal surveys is the marine chart. The marine chart provides an excellent visual aid for planning survey lines and locating

the survey vessel relative to the planned survey location. In fact, many surveys, even today, are laid out in pencil on a paper chart. Survey lines are then followed by careful positioning relative to landmarks indicated on the chart, and some assistance from a GPS or Loran C navigation receiver. The marine chart shows large scale geologic features (e.g., land masses and bodies of water), aids to navigation, and water depths. The water depths are often the most important and relevant piece of information to survey planning. Water depth imposes the most immediate restrictions on surveys and must be carefully considered during survey planning. However, it is often very difficult to look at a chart and see large scale trends in the bathymetry and the relationships between finer features.

A DTM provides the basis on which to build more in-depth investigations (Hall, 1996, Doytsher and Hall, 1997). The availability of a Digital Terrain Model (DTM) of the survey area can assist in extracting information from the water depths and produce a better survey plan. Used most simply, the DTM can be displayed and viewed to provide foresight into the survey area. But through digital techniques, the DTM can provide numerous new capabilities to the survey planner. The RTGIS Toolbox uses the DTM to obtain preliminary foresight into the survey environment and assist in automated survey optimization. Routines are provided for survey optimization via the following means:

1. Slope calculation and analysis to suggest the most efficient survey orientation
2. Minimum depth analysis to highlight areas of concern to towed bodies

3. Using tow speed and sonar range user inputs, a maximum cable out coverage is created that can be referenced during the survey to predict impending collisions between the towfish and the bottom

NOAA Chart Import

Before DTM analysis may be performed for survey planning, a DTM must be obtained or created. This is most often not a trivial task, especially for seafloor mapping surveys for which the primary objective is bathymetry data acquisition. Populating the database with a DTM often requires time consuming data acquisition or data reduction. However, for coastal surveys, we may take advantage of the work already completed by the National Ocean Service of NOAA. The analog data on printed NOAA charts may provide the ability to create a DTM if the chart can be digitized. A procedure was developed for creating first approximation DTMs from digital NOAA charts. Figure 2.14 shows the Digital NOAA Chart Import Tool. The Digital NOAA Chart Import Tool provides an interface to numerous command line programs, including GRASS programs, pbmplus programs, and shell scripts and completely automates the conversion of digital NOAA chart images to GRASS raster data layers. The conversion process includes geographic rectification and transfer of the raster chart into the permanent GRASS database.

The Digital NOAA Chart Import Tool allows selection of the digital chart type and the NOAA chart number. The type is either Maptech digital images from scanned paper charts or BBA digital images from scanned chart films. The chart number is selected from a chart database on CD-ROM. After the chart type and number have been selected,

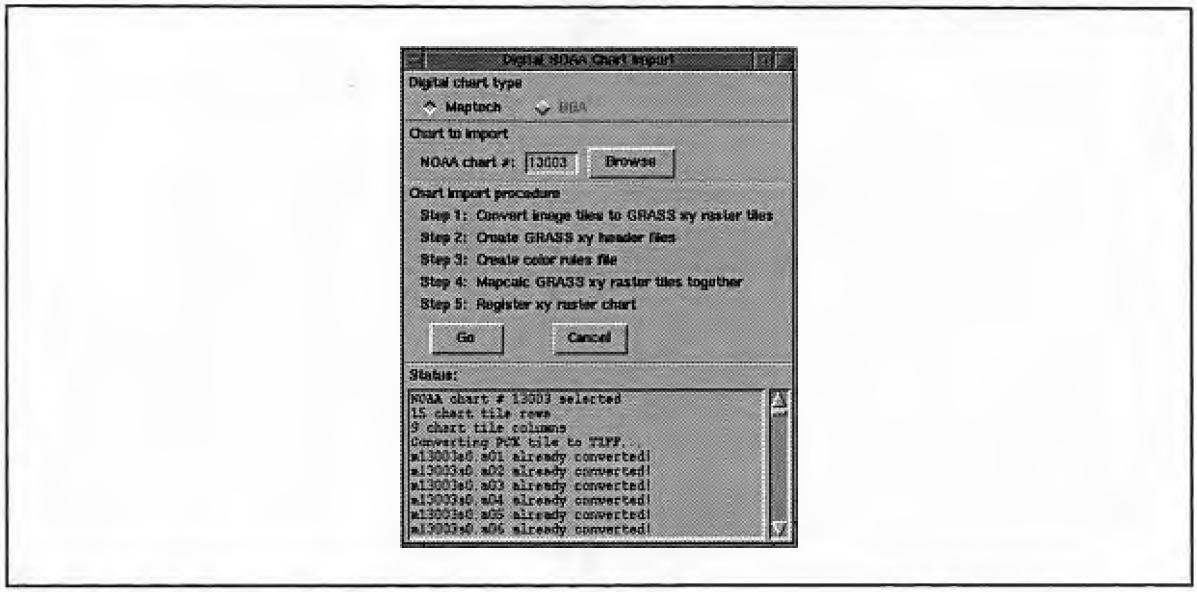


Figure 2.14. RTGIS Toolbox NOAA Chart Import Tool.

selecting the Go button will begin automatic chart import. Steps 1-3 will run without user intervention. Step 1 converts the digital chart image files to GRASS XY raster files. Step 2 calculates the GRASS XY header file values for each chart image tile so that they may be properly assembled into a single, full chart image. Step 3 constructs a color table for the full chart image from the color tables of each chart image tile. Step 4 then begins the chart tile assembly procedure. A GRASS display monitor is used to display the assembly progress. First all tiles in each column are combined one-by-one. Then the columns are combined one-by-one to produce the final chart raster layer in XY units. Step 5 requires user intervention for geographically rectifying the chart image. This is accomplished via the GRASS imagery programs i.points and i.rectify2. In order to do this, the user must have some knowledge of the chart's geographic boundaries. However, if they are not known, they can be read from the chart image using the GRASS image registration display. The rectification procedure performed by i.rectify2 constructs a third

order transformation matrix based on points registered by i.points which is then used to compute the geographically rectified image.

DTM Creation

The DTM Creation Tool, shown in Figure 2.15, is configured to work with the NOAA chart raster data layers and walks the user through the complete digitization procedure.

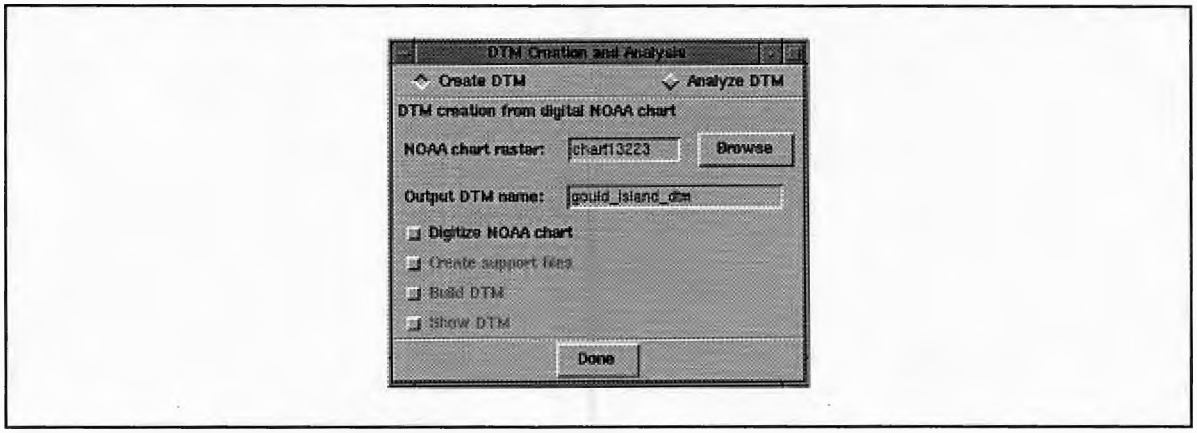


Figure 2.15. DTM creation of NOAA chart data layers.

The DTM Creation Tool allows the user to select the NOAA chart to digitize and input a DTM file name. The user is then guided through the chart digitization procedure using the GRASS digitizing program v.digit. V.digit allows the user to extract the depth data from the chart by digitizing the chart soundings and contours and applying labels (i.e., the depth values). The DTM Creation Tool will then automatically create required support files for the DTM, build the DTM, and display the resulting DTM to the user.

DTM Analysis

After a DTM has been created, there are several options available for DTM analysis. Slope and aspect data layers can be created and displayed. A procedure has been written

to analyze the slope and aspect layers to determine optimum survey line orientations based on two separate criterion; minimum cross-swath slope and minimum along-swath slope. The former criterion results in the smallest backscatter intensity differential between the port and starboard channels along each survey line, the later results in less concern over adjustment of towfish altitude over a particular survey line.

Another process performs a hypsographic analysis of the DTM and creates data layers which identify areas of depth less than a user-defined percentage of the total depths. This data layer can then be used as a danger reminder during survey operations.

The last DTM analysis procedure uses user-defined tow speed and sonar range to compute a raster data layer showing optimal cable out. For this purpose, a fish altitude of 10% of the sonar range is considered optimal. The cable out value will be measured from the water surface. The resulting data layer can be used for adjusting the length of towfish cable below the water surface and warning of impending tow fish grounding.

2.9 Interprocess Communication of Real Time Modules

Early in the development of the serial data acquisition routines (discussed in the next section), fundamental shortcomings were discovered in the GRASS system which prevented efficient real time operations. GRASS provides communication between its various programs through a disk resource file which defines the GRASS environment via a series of variables, similar to UNIX environment variables. When a GRASS program desires to set or query the state of the GRASS environment, it does so through several

GRASS library routines, such as G_setenv() and G_getenv(). These routines open the GRASS resource file, write or read an environment variable, then close the resource file. During initial test cruises this method was used to provide interprocess communication (IPC) of real time programs. This is fine for typical GRASS use; post processing and display during which only one single program accesses the resource file. However, the GRASS library routines provide no mechanism for coordinating access to the resource file. During real time operations there are many processes (data loggers, data monitors, data displays, etc) running concurrently which make numerous asynchronous reads and writes to the GRASS resource file. There is no guarantee that two or more processes will not read and write the resource file simultaneously. Under certain circumstances (e.g., two processes writing to the file simultaneously), such uncoordinated file operations can cause processes to crash and exit. Indeed, during preliminary testing aboard the Ocean Engineering Department's Research Vessel *CT-1*, the serial data acquisition programs frequently and randomly crashed for this reason. In addition, IPC via a disk file is simply too slow for real time programs which must repeatedly update certain variables and keep pace with data input rates. A far more efficient system was designed for coordinating real time programs which consists of the following three components:

1. Semaphore file locking
2. IPC via Berkeley sockets
3. A coordinating entity for all RTGIS Toolbox real time programs

The first component provides a way of controlling access to the GRASS resource file. There are occasionally instances in the real time processes when it is more desirable to

use the resource file (e.g., setting a variable which only needs to be set once at startup). For this reason, use of the resource file for IPC was not abandoned completely. In order to guarantee that processes would not crash by simultaneously writing the resource file, the GRASS library functions which access the resource file were modified to use semaphore file locking. Semaphores are a UNIX kernel synchronization primitive (Stevens, 1990). As a form of IPC, they are intended to let multiple processes synchronize their operations (Stevens, 1990). Since semaphores are used to provide resource synchronization between different processes, the actual semaphore value is stored in the operating system kernel, rather than the program's own memory space (Stevens, 1990). Semaphore file locking guarantees that only a single process will access the GRASS resource file at any one time, thus eliminating the failures observed in the early development of the serial data acquisition programs.

The second component provides a data (or message) passing mechanism for processes. Instead of arbitrarily updating the GRASS environment each time a change occurs, processes can respond only when requested to do so. This messaging interface was implemented using the Data Transfer Mechanism from the National Center for Supercomputing Applications (NCSA). Data Transfer Mechanism provides a simplified interface to Berkeley sockets. Sockets are an application programming interface (API) to the network communication protocols (Stevens, 1990). They provide a means for processes to communicate over the network; a network I/O facility analogous in some ways to file I/O. Communicating via sockets, rather than the GRASS resource file,

provides faster and more reliable IPC and also makes the linkages between real time programs much more clear.

The third component is simply a C header file, included by all real time programs, which defines the ports to use for real time message passing between the real time programs. The header file is placed in a common source directory (/usr/local/grass4.1/src.OMDC/include/) which all real time extension programs can include. The header file provides an easy way of coordinating and maintaining the IPC for all RTGIS real time programs. Appendix B shows the RTGIS Toolbox IPC header file.

Consider the case of the Magnavox 4200D differential GPS navigation acquisition program (ser.MX4200D) and the real time ship track display program (d.nav). Figure 2.16 below shows a cartoon depicting how these two programs communicate with each other and the GRASS environment using the system just described. The cartoon is a mixture of flowchart and block diagram and shows a portion of the main loop of each program which concerns the SENDNAV request from d.nav and the SENDNAV reply message from ser.MX4200D. When ser.MX4200D starts, it sets the name of the current navigation log file in the GRASS resource file using a call to G_setenv(), which accesses the file using semaphore file locking. Likewise, when d.nav starts, it checks the name of the current navigation log file in the GRASS resource file using a call to G_getenv(), which also accesses the file using semaphore file locking. The main program loop of d.nav sends a SENDNAV request to ser.MX4200D, whose request port is defined in ports.h, and waits for a SENDNAV message reply. The main loop of ser.MX4200D

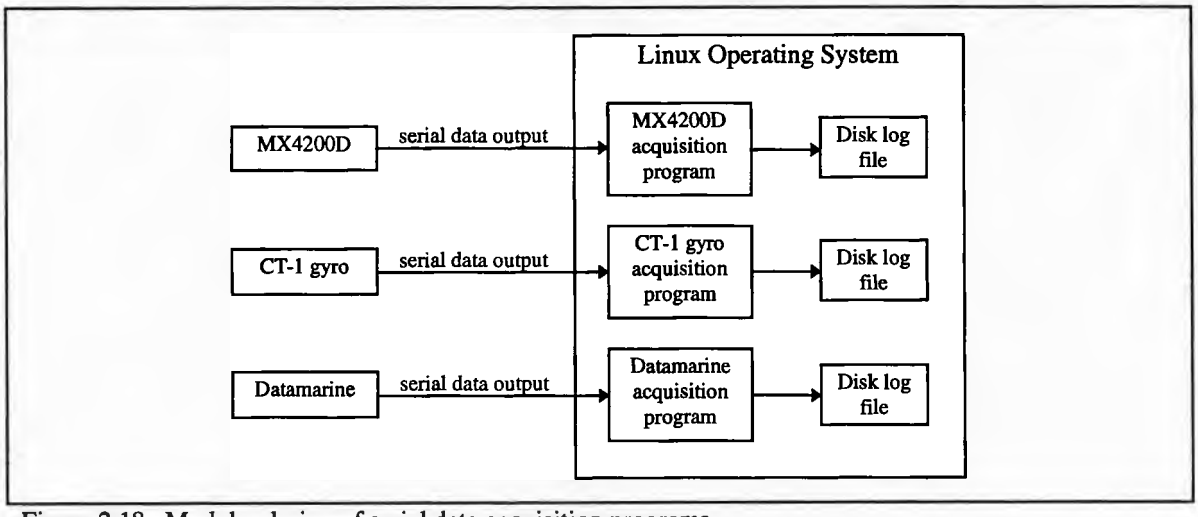


Figure 2.18. Modular design of serial data acquisition programs.

In order to obtain a useful real time system from the block diagram in Figure 2.18 above, each serial data acquisition program interacts with other programs via Data Transfer Mechanism messages, (described in Section 2.9). Each acquisition program has a pre-defined set of known requests and a unique Data Transfer Mechanism request port through which requests for data and other messages are sent by other programs. All requests and ports for each serial data acquisition program are defined in the ports.h header file. Ports.h is included in Appendix C. As an example, the table below summarizes the requests associated with the Magnavox 4200D differential GPS serial data acquisition program. Similar tables for each serial data acquisition program are included in Appendix C.

Table 2.15. ser.MX4200D real time request signals.

Request ID	Request Meaning	Reply Format	Programs requested by
MX4200D_SENDNAV	Reply with position fix	"int char[]" int: Request ID char[]: postition in OMDC nav log record format	d.nav
MX4200D_SENDCOG	Reply with course over ground	"int double" int: Request ID double: COG value	none this version

Table 2.15 (continued). ser.MX4200D real time request signals.

Request ID	Request Meaning	Reply Format	Programs requested by
MX4200D_SENDSOG	Reply with speed over ground	"int double" int: Request ID double: SOG value	none this version
MX4200D_SENDFILE	Reply with name of log file	"int char[]" int: Request ID char[]: log file name	d.nav
MX4200D_SETSWATH	Set swath width in log file	(none)	m.sls.setswath
MX4200D_SETSLSRNG	Set sonar range in log file	(none)	m.sls.setrange
MX4200D_SETDEPTH			
MX4200D_SETONLINE	Set on/off survey line flag in log file	(none)	m.sls.setonline
MX4200D_STOPLOG	Stop logging and exit	(none)	m.stop

All messages except the stop message (discussed below), consist of two parts; a header value and a data value. The header value is always the request ID. The data value depends on the request ID but always takes one of two forms depending on whether the message requests data to be sent or requests that a variable be set. For messages requesting data, the data part is the Data Transfer Mechanism port to which to reply. The data part of the reply message is the data value requested. For messages requesting that a variable be set, the data part is the value to which the variable should be set. There is no reply in response to a set message.

Most real time programs, including serial data acquisition and display programs, will attempt to continue running until explicitly asked to stop. This eliminates the need for complex algorithms in each real time program for determining when input has stopped or when the user "most likely" intended to stop. A simple and straightforward mechanism was designed for stopping real time programs. Each real time program has as one of its pre-defined requests, a stop request. If the process receives a stop request, it will stop itself and exit normally, doing any cleanup required. There is no validity checking because the stop request can come from only one source; the program m.stop. M.stop

controls stopping of all real time programs which include a stop request. M.stop requires a single argument indicating which process to stop. Through its interface to the ports.h header file, m.stop knows exactly the Data Transfer Mechanism port to which to send the stop request. M.stop can be run interactively by the user on the command line, however, it was designed to be launched by buttons in the RTGIS Toolbox.

For each serial data acquisition program, there is a corresponding instrument and monitor program. Most acquisition programs also have an associated simulator program and display program. The instrument is the oceanographic instrument whose output will be acquired via the serial port hardware. The simulator program creates data messages in the same format generated by the instrument and outputs them on a user-defined serial port. The simulator programs are used for demonstrating real time capabilities in the absence of direct connections to instruments. The display program displays the logged data to the user. The display could be text or graphical and there could be more than one display program per serial data acquisition program. The monitor program monitors the data logging process. At a user-specified interval, the monitor program checks that data is being updated in the current log file. If not, the user is warned by text warnings and audible beeps. Table 2.16 below summarizes the serial data acquisition programs written and associated instrument, simulator program, display program, and monitor program.

Table 2.16. Serial data logging programs.

Logging Program	Instrument	Simulator Program	Display Program	Monitor Program
ser.MX4200D	Magnavox 4200D differential GPS	sim.MX4200D	d.nav	ser.monitor
ser.CT1GYRO	RV CT-1 gyro compass	sim.CT1GYRO	(none)	ser.monitor
ser.DATAMARINE	Datamarine speed log, compass, and echosounder	sim.DATAMARINE	(none)	ser.monitor
ser.FRLS6000	Furuno LL-90 Loran C	(none)	d.nav	ser.monitor
ser.HYDROLAB	Hydrolab H-20 water quality instrument	sim.HYDROI.AB	(none)	ser.monitor

Table 2.16 (continued). Serial data logging programs.

Logging Program	Instrument	Simulator Program	Display Program	Monitor Program
ser.MAPTECH	Maptech helm guidance	sim.MAPTECH	Maptech	ser.monitor
ser.RS920	Raytheon RS920 GPS	(none)	d.nav	ser.monitor
ser.SMS960	EG&G SMS960 digital sidescan	sim.SMS960	ShowImage	ser.monitor

2.11 Real Time Survey Monitoring / Data Logging Monitoring

There are many programs which assist in the task of survey monitoring and data logging monitoring. The survey monitoring sub-system includes display programs which show real time ship track, real time swath coverage, real time side scan sonar, and real time multibeam sonar. The data logging monitoring sub-system includes text windows which show real time ASCII data from log files. The data logging monitoring sub-system design has paralleled the design of the loggers themselves. To each data logger is attached a logging monitor which runs as a separate process from other logging monitors. This is done to isolate system failures. If one data logging monitor fails, it will not affect the operation of other data logging monitors. Table 2.17 contains a summary of the programs written for the Real Time Survey Monitoring and Data Logging Monitoring sub-systems.

Table 2.17. Programs written for the Survey Monitoring/Data Logging Monitoring module.

Program	Type	Location†	Lines*	Description
d.nav	C program	$GIS/src.OMDC/ display/d.nav	510	Draws ship track on GRASS display monitor by updating the current navigation in real time
m.setonline	C program	$GIS/src.OMDC/ misc/m.setonline	66	Sets the online/offline flag used by ser.MX4200D and d.nav to indicate whether the ship is currently on a survey line or not
d.vect.des	C program	$GIS/src.OMDC/ display/d.vect.des	349	Draws vector data layers by downsampling the number of arc segments based on user input. Provides faster redrawing.
ShowImage (display.c)	C program	/usr/local/xsonar/ showimage	307(E) 192(N)	Modifications to display.c permit the ShowImage display to be used for real time waterfall display of sidescan sonar data.
v.in.mpro	C program	$GIS/src.OMDC/ mapdev/v.in.mpro	110	Imports route files from the Maptech helm guidance software as GRASS vector files.
ser.monitor	C program	$GIS/src.OMDC/ serial/ser.monitor	444	Monitors ASCII data log files

† $GIS = /usr/local/grass4.1 * (E) = Existing, (N) = New

2.12 Helm Guidance

While much effort was devoted to investigating the issue of helm guidance; it was out of the scope of this project to develop a complete helm guidance system. Early in this work, three separate commercial helm guidance systems were tested and evaluated; Maptech from Resolution Mapping, Navigate! from Fairtide Technologies, and YoNav from the USGS. Maptech was preferred because it provided scanned NOAA chart overlay and was able to update the display (including redrawing the raster chart) very quickly. Maptech was installed on a laptop computer which is placed at the survey vessel's helm, in easy view of the vessel operator.

Maptech has a well organized display; with the NOAA chart and ship position shown on the center of the screen along with survey lines and a side panel which may be used to show useful ancillary inputs, such as heading and water depth. However, the laptop computer has only two serial ports. Since one is reserved for output to an autopilot there can only be one data input to the helm guidance; position, course over ground, and speed over ground from an navigation instrument. Since it was desired to input other data (e.g., gyro and magnetic heading and depth), a system was developed which augments the Maptech helm guidance display. All data are first acquired by the RTGIS Toolbox workstation. A program was written which rebroadcasts all appropriate messages back out to the Maptech laptop. Figure 2.19 below shows a block diagram for this system.

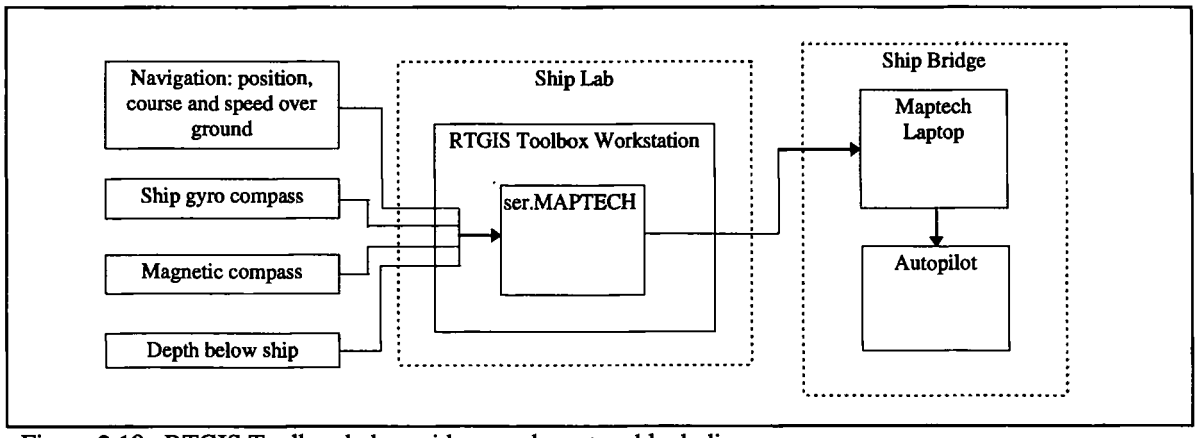


Figure 2.19. RTGIS Toolbox helm guidance sub-system block diagram.

Whenever the navigation acquisition program receives a position message, it will check if the ser.MAPTECH request port is open. If it is, the received message will be sent to ser.MAPTECH. Ser.MAPTECH will then immediately rebroadcast the message on a separate serial port to the Maptech laptop. A similar procedure is followed by the ship gyro acquisition program, the magnetic heading acquisition program, and the depth acquisition program. Using this method, data are acquired by the RTGIS Toolbox and rebroadcast to the Maptech laptop with a latency of less than the update rate of the data inputs. All desired data inputs can now be sent to the Maptech helm guidance laptop on a single serial port, allowing the other serial port to be used for interfacing to an autopilot.

2.13 Data Documentation and Reporting

There are three forms of data documentation and reporting. The first form involves generating text reports of collected survey data. When a new survey is begun, a program will automatically run in the background and catalog the existing MAPSET data files. When a survey is complete, the user may request that a survey report be generated. A

process will compare the current MAPSET data files to the list made before the start of the survey. The survey report will consists of a text list of the new data files added to the MAPSET, including the file name, type, size, and recording time.

The second form of data documentation and reporting involves generating first cut maps showing raw data coverage. The Hardcopy Map Output sub-system, discussed in the next section, is used to generate these maps. In general, map output is distinguished as a Data Documentation and Reporting task when the mapped data consists of primarily uncorrected data for the purpose of reporting on collected survey data rather than creating final presentation maps.

The third form of data documentation and reporting provides an interface to the RIM relational database management system. The RIM database may be queried and relational reports may be generated.

2.14 Hardcopy Map Output

Hardcopy map output is an essential part of the RTGIS Toolbox product. It provides the means for communicating and sharing the survey data with others who do not have access to the RTGIS Toolbox workstation. Along with the digital raw and processed survey data, hardcopy maps comprise the deliverables to survey clients. Two hardcopy output devices were used in this thesis for the purpose of map output; the HP DeskJet 850 printer and the HP DesignJet 650C large format plotter.

A two-component system was created for providing high quality maps from user-defined combinations of survey data and historic data. The first component is a Postscript map production system, the second is a Postscript output system. The Postscript map production system consists of a Tcl/Tk interface (called the Map Making Tool) which orchestrates the use of a GRASS display monitor, GRASS display programs, GRASS Postscript commands, the Generic Mapping Tools (GMT), and Ghostview for the purpose of creating a final map in Postscript format. A block diagram of this system is shown in Figure 2.20 below.

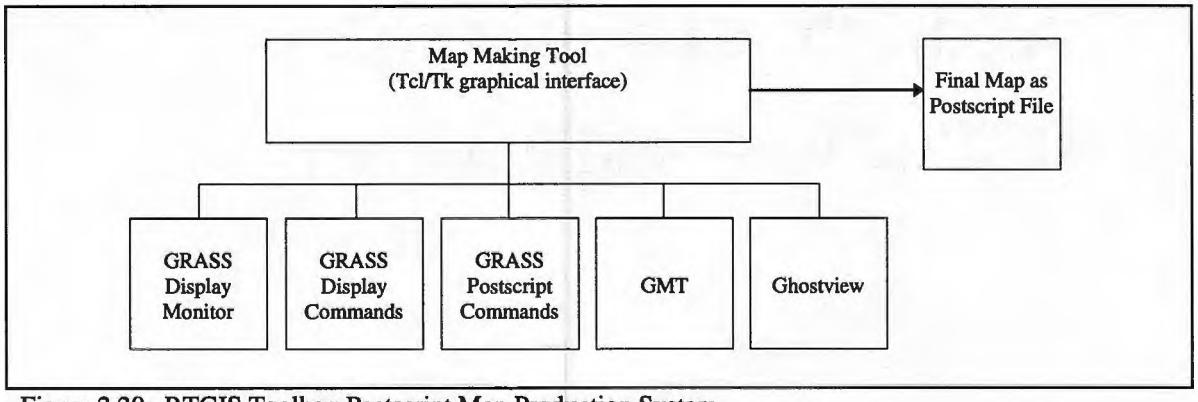


Figure 2.20. RTGIS Toolbox Postscript Map Production System.

GRASS provides a program (ps.map) for creating Postscript output describing the current GRASS display monitor. This means that a Postscript image of raster, vector, and site data layers may be produced. However, GRASS does not provide more advanced map making features, such as creating geographic grids, borders, labels, and titles. GMT, a Postscript mapping system created at Lamont-Doherty Earth Observatory (Wessel and Smith, 1995), provides the more advanced map making features along with the ability to import raster, vector, and site data for mapping. The Map Making Tool, shown in Figure

2.4(E), allows the user to select a GRASS display monitor in which to setup data for mapping. The user may then select raster, vector, and site data to display in the display monitor via the Map Making menu. When the user is satisfied with the GRASS display, a Postscript file is generated via ps.map. GMT is then used to import the Postscript map from GRASS and attach grids, borders, labels, etc. The user may determine when the map is complete via inspection with Ghostview. The resulting Postscript file is then printed via the Postscript output system.

The Postscript output system consists of a custom script, rasterize.sh, Ghostscript, the UNIX lp print spooler, and the printing or plotting device. Rasterize.sh is included in Appendix C. A block diagram of the Postscript output system is show in Figure 2.21 below.

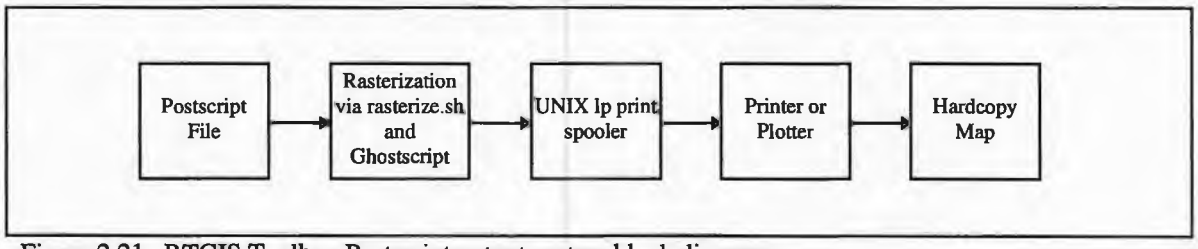


Figure 2.21. RTGIS Toolbox Postscript output system block diagram.

The inputs to rasterize.sh include the page size and the device to which to print. The purpose of rasterize.sh is to convert the Postscript image to the native format of the target printer or plotter. This system can be used to print any Postscript image on any size page to any device supported by Ghostscript. The devices supported by Ghostscript are included in Appendix C. This system provides a very flexible and robust printing capability and is referred to later when printing Postscript multibeam data.

2.15 Sidescan Sonar Data Acquisition, Display, and Processing

Sidescan sonar basic theory

Traditional sidescan sonar instruments are towed behind a survey vessel with a tow cable, relatively close to the seafloor (an altitude of approximately 5%-15% of the sonar range is optimal). The towed sidescan body is called the towfish. The sidescan sonar towfish has two transducers mounted longitudinally on either side of its body which are directed to the side and slightly downward. The transducers transmit acoustic energy into the water in a beam which is very narrow along track (parallel to the tow direction) and very wide across track (perpendicular to the tow direction). The transducers also receive the acoustic energy which returns from the seafloor. For a given frequency and angle of insonification, the magnitude of the returned acoustic energy (commonly referred to as backscattered energy) is a function of the seafloor material type, the particle size, and the micro-relief, or "roughness" of the seafloor, with roughness and incidence angle playing the major role (Urick, 1983). In general, the image of acoustic backscatter which results does not necessarily quantify the aforementioned seafloor properties (Byrne, 1990). More detailed discussions of sidescan sonar operation theory can be found in Flemming (1976), Leenhardt (1974), Byrne (1990), and Clapp (1994). The sidescan sonar system, along with a bathymetric system, is of vital importance to seafloor mapping operations.

Sidescan sonar history

Sidescan sonars have been extensively used since the 1960's for locating objects of interest on the seafloor and for surveying the geologic features of the seafloor (Flemming,

1976). These imaging sonar systems have seen increasing use through the 1970's and 1980's with increasing interest in the geological characteristics of the seafloor from both the oil industry and academia (Tyce, 1986, Flemming, 1976). Two factors in the 1990's opened up sidescan sonar technology to new markets and contributed to the continued proliferation of sidescan sonar technology. The first was the opening of foreign markets to seafloor mapping technology as second world countries, particularly those in the Middle East and Far East, have begun to recognize and invest in the need for surveying their waters for both navigational and resource management purposes. Much of this recognition has resulted directly from the United Nations' continuously evolving Law of the Sea Agreement which governs national and international use of the ocean for numerous purposes, including commercial. The second was the development of powerful personal computer (PC) systems which could provide the processing and display capabilities required for sonar acquisition and processing. Numerous small businesses have begun developing and marketing compact, high quality, low cost sonar mapping systems on desktop PCs.

Sidescan sonar equipment

The sidescan sonar surveying equipment used by the URI Ocean Engineering Department includes the EG&G Model 960 Seafloor Mapping System (SMS960) and an EG&G Model 272-TD Towfish. Analog echo levels are filtered in the 272-TD and converted to decibels. A time varied gain is applied in the towfish to correct for spreading loss, absorption loss, and the effect of increasing insonification area with range. The filtered, corrected analog levels are then transmitted up the tow cable to the SMS960 where they

are digitized and displayed. The A/D conversion produces a 6-bit logarithmic word, resulting in a dynamic range of 64 decibels. Flag bits are attached to distinguish status data from port and starboard data, resulting in an 8-bit word per sample. The digitized data are then multiplexed with a 32 word status header for each scan line of sonar data. The status header contains date and time and may contain speed, heading, and position if supplied to the SMS960. The SMS960 produces a true plan view, or map of the seafloor on an dry electrostatic paper record by tracking the bottom in the data. The record is geometrically corrected for slant range and vessel speed and radiometrically corrected for transducer beam pattern and power loss due to grazing angle variations. Gain and contrast controls on the SMS960 may be used to enhance the paper record. The raw data produced by the digitization procedure is also sent out the SMS960's Tape Write port. The data sent to the Tape Write port is not effected by any processing by the SMS960 or adjustments made by the operator. This is a desirable feature if the raw data is to be post-processed by other systems or replayed by the SMS960 (Byrne, 1994).

The need for digital data acquisition

The current need for real time sidescan sonar data acquisition at the Ocean Engineering Department arose during several sidescan sonar surveys aboard the *CT-1*, for which there was no satisfactory mechanism for acquiring real time digital sidescan sonar data. Historically, sidescan sonar systems have consisted of either 1) analog display and recording or 2) digital data collection on mainframe or workstation computers with paper recording devices for display (Byrne, 1990). Today, however, the availability of low cost, high speed PCs with sophisticated display and data storage hardware allow for the

development of a system which can greatly exceed the capabilities of past units (Clapp, 1994). Real-time display of digital data and the ability to store accurate data for later post processing via digital techniques results in better scientific interpretation and archiving of data (Clapp, 1994).

Previous developments at the OMDC

The need for acquiring sidescan sonar data digitally has been addressed previously at the OMDC by Byrne (1990) and Clapp (1994). The system designed and built by Byrne utilized the SMS960, the 272-TD towfish, and a VAX computer. The VAX acquired the digital data output from the SMS960 Tape Write port, logged the data, and displayed the seafloor coverage in real time. Software running on the VAX was used to post-process the acquired data; including geometric and radiometric corrections and swath mosaicing to produce a sidescan sonar map of an area of the seafloor.

The system designed and built by Clapp bypassed the SMS960, acquired the analog data from the 272-TD towfish directly on a PC running the DOS operating system. The system's primary component is a custom-built printed circuit board which is installed in an ISA slot of the PC. The board controls triggering of the 272-TD towfish transmit pulse, digitizes the analog return, applying appropriate filtering and gain, and provides the digitized data directly to logging and display software running on the PC. The result is a compact, low cost sonar system on a PC (Clapp, 1994). This system is significant because it opens sidescan technology to large potential markets where the use of traditional systems has not been feasible (Clapp, 1994). However, Clapp's system is

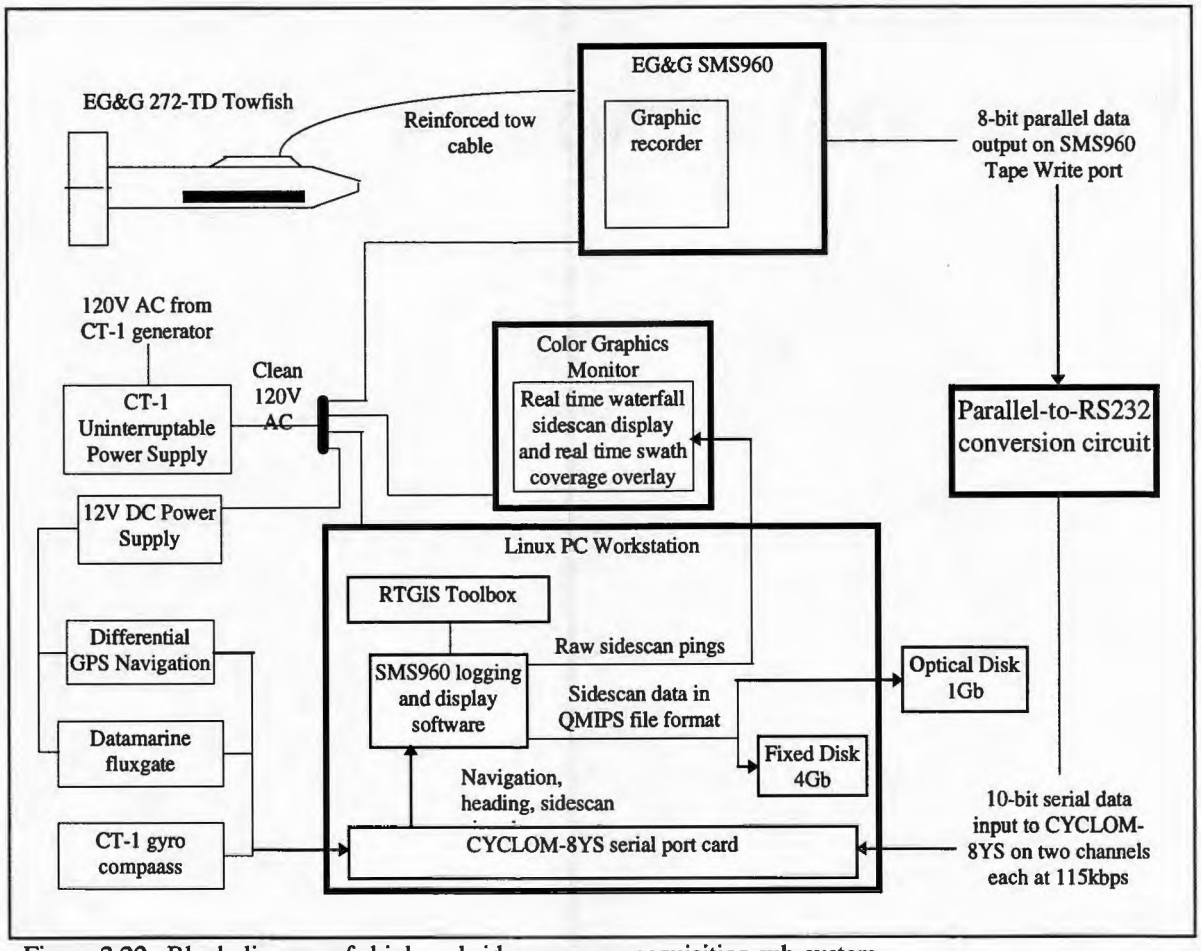


Figure 2.22. Block diagram of shipboard sidescan sonar acquisition sub-system.

during hydrographic surveying. Therefore, the paper record was not discarded by this system. Digital sidescan data from the Tape Write port is converted to RS232 serial data and logged and displayed on the Linux PC workstation via its high speed serial ports. Navigation and heading data is also logged via the serial ports. The SMS960 acquisition and display software merges the sidescan data with the navigation and heading data and logs the data in QMIPS file format to either the fixed disk or the optical disk. The raw ping data is simultaneously displayed on the Linux workstation monitor in waterfall format. All hardware is powered from the ship's 120V AC UPS to provide reliable power and protect the data logging from ship generator failures.

The primary component of the acquisition hardware is the parallel-to-RS232 conversion hardware which consists of an electronic device designed to accommodate the CYCLOM-8YS serial port expansion card. The SMS960 Tape Write port transmits data at a rate of 12kHz. The data consists of 8 parallel bits. In order to keep up with the data input rate, the parallel-to-RS232 conversion electronics must output each serial word within 83.3 micro seconds (the inverse of 12kHz). Each serial word consists of the 8 bits from the SMS960 plus 1 start bit and 1 stop bit tacked on by the conversion electronics to create RS232 output. Thus 10 bits must be output every 83.3us, resulting in a required serial bit rate of at least 120kbps. However, the CYCLOM-8YS provides a maximum input bit rate of 115kbps in its Extended Baud Rate Mode (Cyclades, 1995). Since the serial ports can not operate at a high enough speed to read the SMS960 parallel output, the parallel-to-RS232 conversion electronics contain two FIFO memory buffers and output two serial channels each at a rate of 115kbps. By outputting at the slower speed supported by the CYCLOM-8YS but outputting simultaneously on two channels, the SMS960 data rate can be accommodated. Figure 2.23 below shows a block diagram of the parallel-to-RS232 conversion electronics.

Referring to Figure 2.23, the 8 data lines and 1 strobe line from the SMS960 Tape Write port are open collector outputs and must be pulled up to 5 volts in order to produce the 0-5V logic levels. The 8 data lines are loaded into the FIFOs which provide buffering so that the output rate can be slower that the input rate.

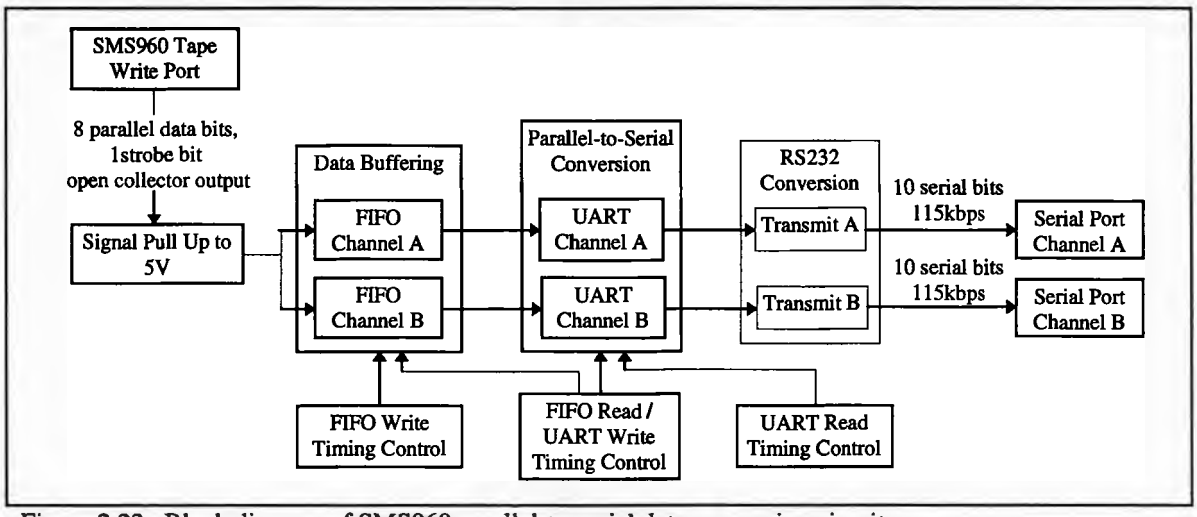


Figure 2.23. Block diagram of SMS960 parallel-to-serial data conversion circuit.

The FIFO Write Timing Control consists of some logic interfaced with the FIFOs to determine to which FIFO to pass the SMS960 strobe. When a FIFO becomes full, the strobe is blocked from it and toggled to pass through to the other FIFO. Data from the SMS960 is loaded only into one FIFO at a time, determined by the one which receives the SMS960 12kHz strobe. Since each FIFO is being filled at a 50% duty cycle, they only need to be emptied at half the rate at which they are being filled.

The FIFO Read/UART Write Timing Control consists of a timer circuit oscillating at approximately 9.5kHz and some logic which interfaces with the FIFOs. The 9.5kHz frequency was chosen because it is a convenient frequency between 6kHz and 11.5kHz; 6kHz is the minimum speed at which data must be read from the FIFOs in order to keep up with the 12kHz rate at which data is written into the FIFOs and 11.5kHz is the maximum frequency at which data can be read from the FIFOs into the UARTs in order for the UARTs to transmit the 10 bits per serial word at 115kbps. The logic involved

determines when the FIFOs are empty. When a FIFO is empty, its corresponding UART is not clocked since there is no data to load.

When an 8-bit word is loaded into a UART, it is immediately transmitted out the UART's serial output line. The UART Read Timing Control consists of a free-running 1.84MHz oscillator. The 1.84MHz frequency is determined by the UART specification. The UART transmitter must be clocked at 16 times the desired output bit rate (16x115000 = 1.84M). The serial output from the UARTs is input into the RS232 transmitter which converts the 0-5V logic levels to ±15V levels and adds a start bit and stop bit. The RS232 output from the RS232 converter is supplied on two DB9 connectors.

The complete circuit diagram for the parallel-to-RS232 conversion hardware is shown in Figure 2.24 below. The entire circuit is powered by a single 9 volt plug-in power supply. The power circuitry is shown separately in Figure 2.25.

The schematic in Figure 2.24 follows the discussion of the circuit block diagram above. The AM7203-25 ICs provide 2048 9-bit words of FIFO memory. The HD-6402 ICs perform the parallel-to-serial conversion of the 8-bit SMS960 words and tack on the start and stop bits. The MAX232 IC provides dual RS232 transmitters which convert the 0-5V logic levels to ±15V. Abbreviated spec sheets for all ICs used are provided in Appendix D. There are, however, two aspects of the circuit which deserve additional discussion; the FIFO Write Timing Control and the FIFO Read / UART Write Timing Control.

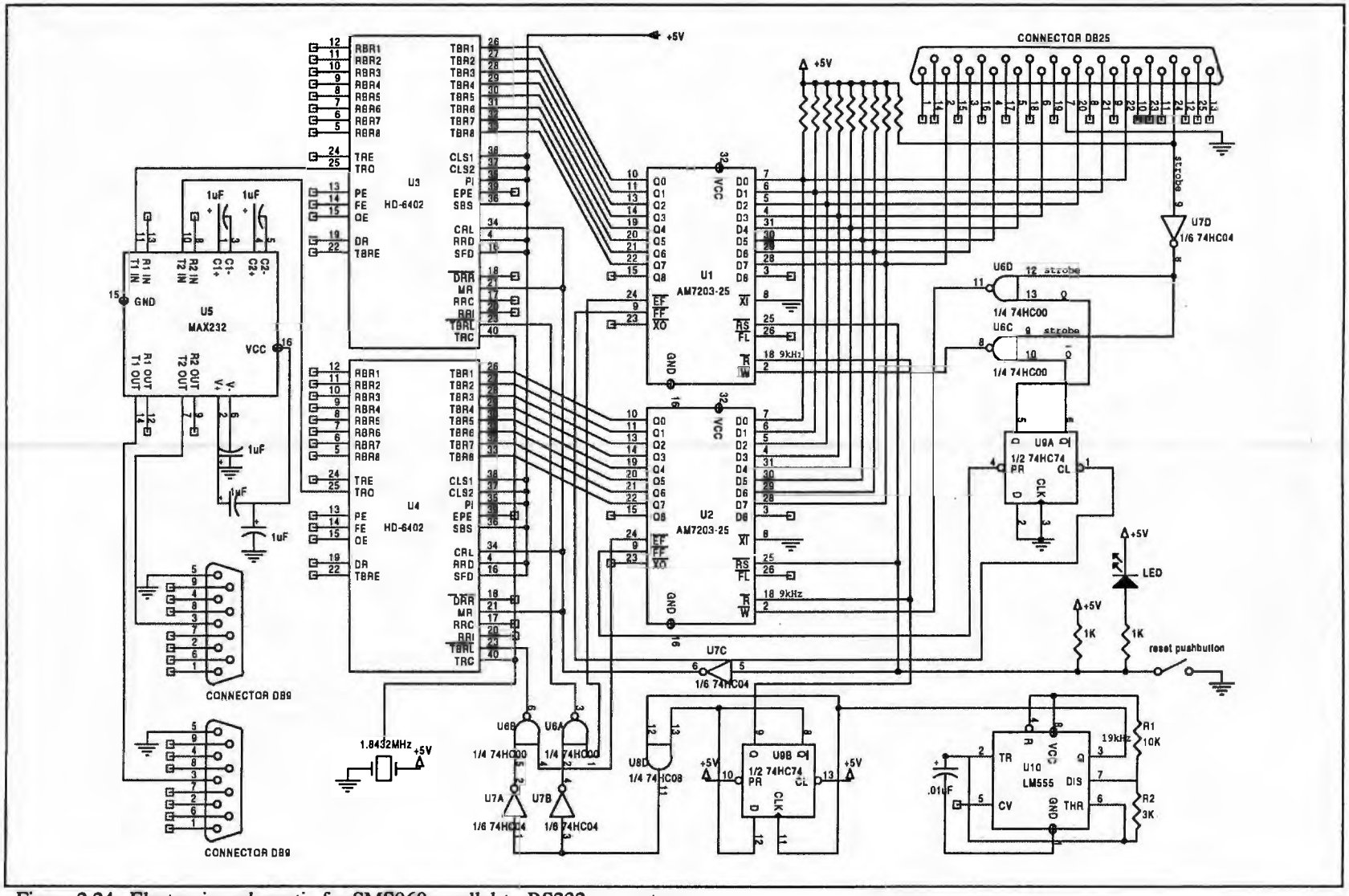


Figure 2.24. Electronics schematic for SMS960 parallel-to-RS232 converter.

The FIFO Write Timing Control determines which FIFO the SMS960 strobe is passed to. The SMS960 data bits appear at the inputs of both FIFOs. However, the data is loaded only into the FIFO which receives the SMS960 strobe. The FIFO Write Timing Control ensures that only one FIFO is strobed at a time. This is done by tying the inverted SMS960 strobe to one input on the two NAND gates U6C and U6D (each is ¼ of a 74HCT00 IC). The other input to the NAND gates comes from either the Q or Q' output from flipflop U9A. The CLK and D inputs to the flipflop are grounded since it will be used only in preset/clear mode. The CL (clear) input to the flipflop is tied to the FF' line of FIFO U1. When FIFO U1 becomes full, its FF' line goes LOW. This brings CL to LOW and sets Q LOW. Setting Q LOW will guarantee that the output from NAND gate U6D stays HIGH; the strobe is thus inhibited from passing to the W' line on FIFO U1 and data is no longer loaded into U1. At the same time, bringing CL LOW will set Q' HIGH. Setting Q' HIGH will allow the inverted strobe to pass through NAND gate U6C to the W' line of FIFO U2 as the original negative-true strobe generated by the SMS960. Data is now loaded into FIFO U2 until it becomes full. When FIFO U2 becomes full, its FF' line is brought LOW. Since it is tied to the PR (preset) input of flipflop U9A, PR is brought LOW; thus setting Q HIGH and allowing consecutive strobes to pass through U6D to FIFO U1 and setting Q' LOW and inhibiting consecutive strobes from passing through to FIFO U2. This behavior will continue, toggling which FIFO is being filled after one becomes full. Figure 2.25 shows the timing diagram for the case of normal loading of a FIFO U1. Figure 2.26 shows the timing diagram for the case when a full condition is generated on FIFO U1.

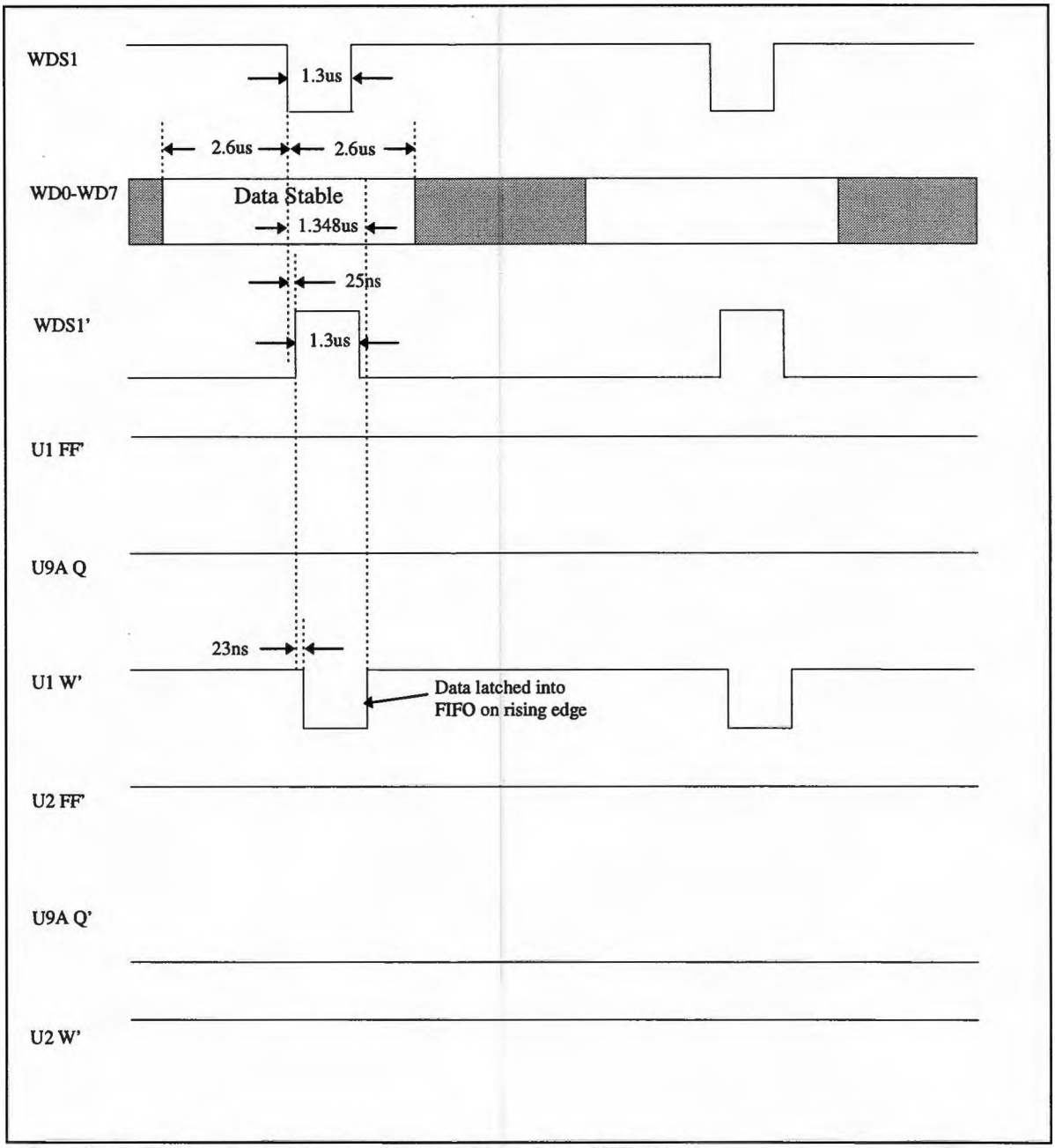


Figure 2.25. Timing diagram for normal (not full) loading of FIFO U1.

There are two important issues to note from Figure 2.25 above. The first is that while U9A Q' is LOW, U2 W' will remain HIGH and no data is loaded into FIFO U2. The second is that while U9A Q is HIGH, U1 W' is strobed with a delay of only 48ns

(guaranteed maximum propagation delay), well within the duration during which the SMS960 data lines are stable, and data is loaded into FIFO U1.

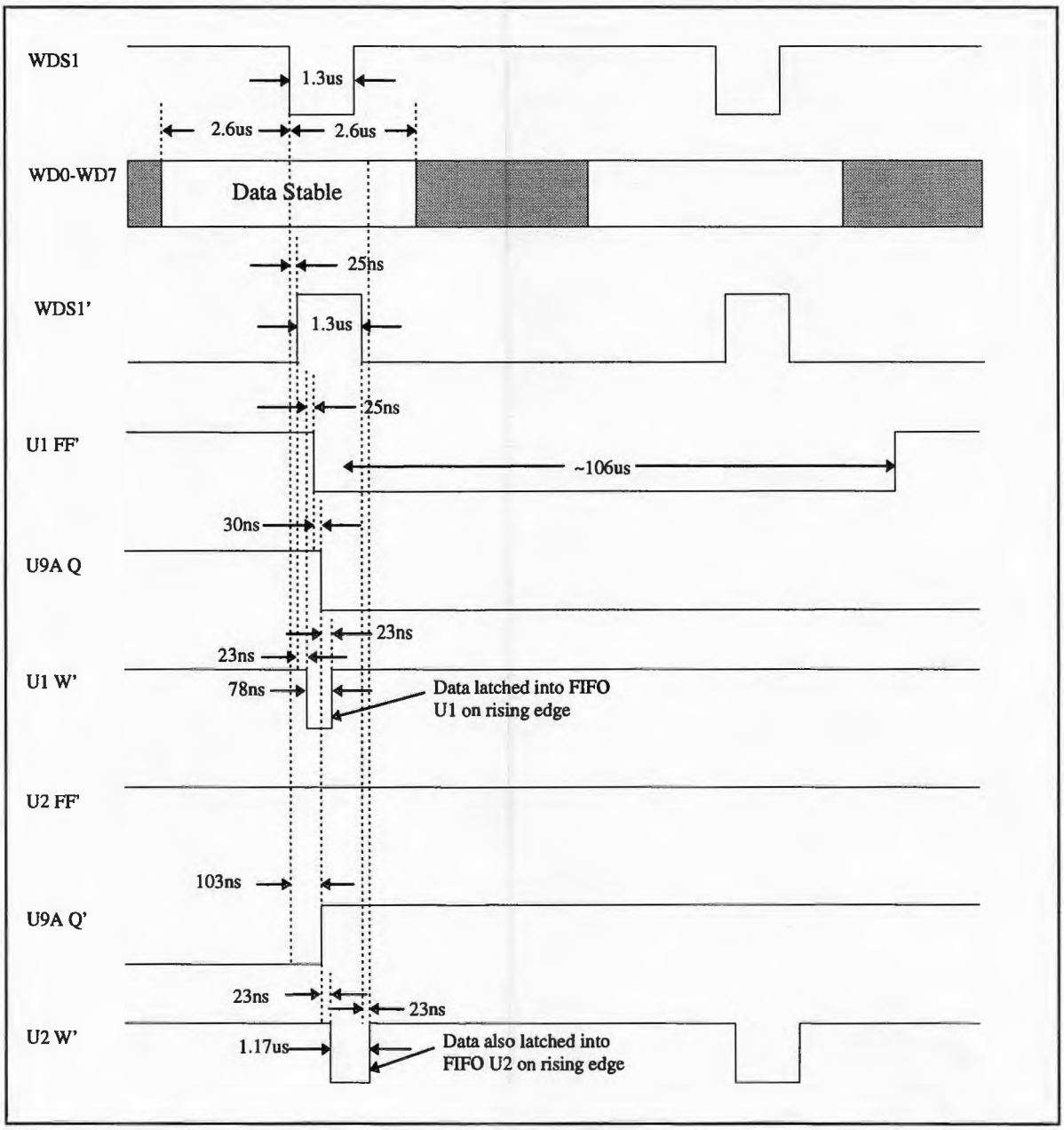


Figure 2.26. Timing diagram for case of full condition on FIFO U1.

Figure 2.26 above shows an interesting timing feature. Since the Q and Q' outputs from U9A toggle very quickly after a full condition is generated by a FIFO (on the order of 10ns while the SMS960 strobe is ~1,300ns), the last strobe on FIFO U1 will also be

generated on FIFO U2. The result is a data byte which will be repeated (i.e., read twice). This will occur each time the strobe is toggled between FIFOs and means that the last byte loaded into each FIFO will also be the first byte loaded into the next FIFO. This behavior is absolutely repeatable and is easily accounted for in the acquisition software by discarding the first character read from each serial port.

The FIFO Read / UART Write Timing Control determines when data is read out of the FIFO and written into the UART. The UART writes must be timed synchronously with the FIFO reads to ensure that data is loaded into the UART while valid data appears on the FIFOs' outputs or else the UARTs will pass invalid serial data to the computer. There are two problems to address. The first is that the UART must not be loaded when its corresponding FIFO is empty. The second is that the FIFO outputs are only stable during the LOW level of R'; the data is not stable on the falling or rising edge of R'. The latter problem is addressed by generating a clock pulse with the 555 U10 which is twice the desired FIFO read frequency (19kHz = 2 X 9.5kHz). The 555 output is then divided in half by using flipflop U9B in "divide-by-two" mode. The Q output of flipflop U9B is the 9.5kHz clock pulse which generates FIFO reads. The FIFO read signal does not need to be concerned with the FIFO empty condition. The FIFO will automatically ignore its R' input when it is empty. By ANDing the 19kHz output of the 555 with the Q' output of the flipflop U9B, a clock pulse is generated on the output of AND gate U8D which is at the same frequency of the FIFO read pulse (9.5kHz) but has an active pulse width exactly half that of the FIFO read pulse. The rising edge of this pulse, which loads the UART transmitter buffer register, occurs approximately 79ns after the falling edge of R'. This

ensures that the FIFO output will be loaded into the UART while the data is stable (during the FIFO R' LOW level). Figure 2.27 shows a timing diagram for the case of normal (i.e., FIFO not empty) FIFO read and UART write for U1 and U3. FIFO U2 and UART U4 are timed similarly.

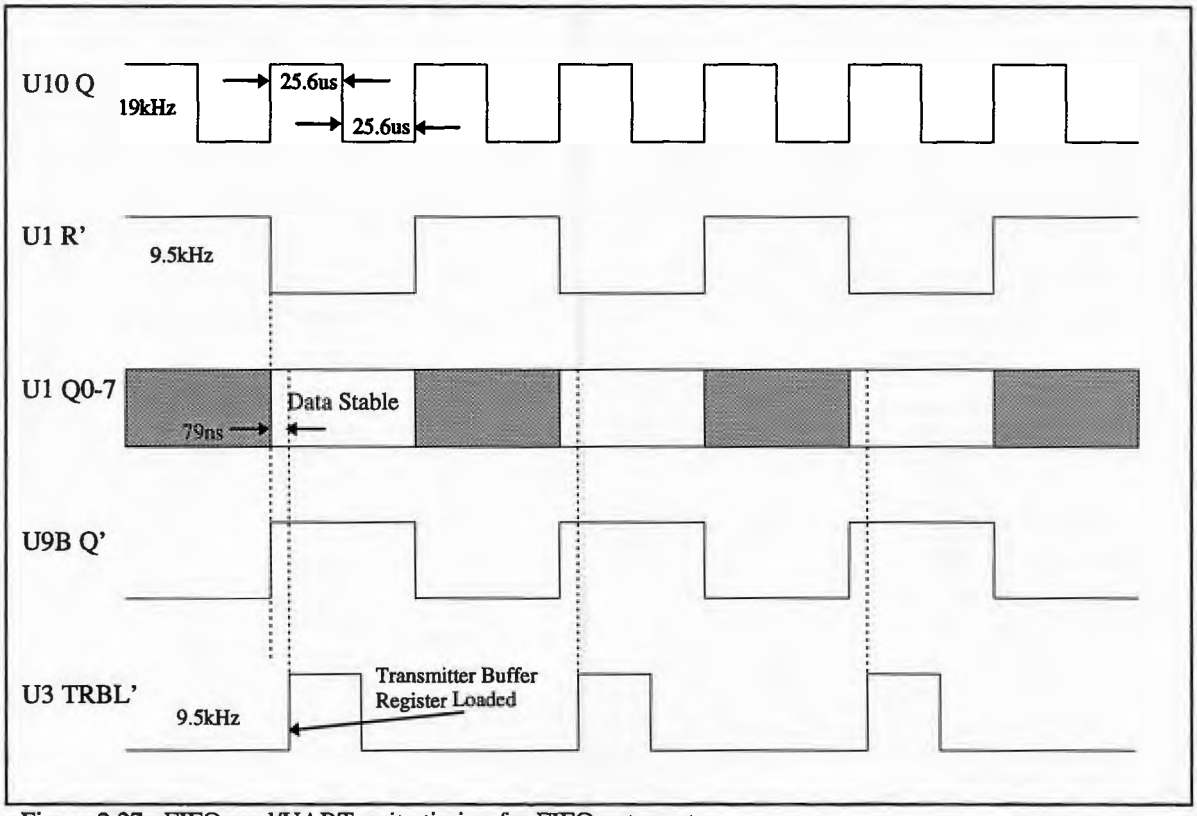


Figure 2.27. FIFO read/UART write timing for FIFO not empty.

In figure 2.27 above, the timing of the inverting buffers U7A and U7B and NAND gates U6A and U6B have not been shown. These elements provide logic for controlling when the UARTs are loaded in the exact same manner as was described for controlling the SMS960 strobe into the FIFOs above. The case for U1 and U3 will be discussed here, and the same will apply for U2 and U4. The output from AND gate U8D is inverted and input to one input on NAND gate U6A. The other input to NAND gate U6A is tied to the EF' output of FIFO U1. When U1 is not empty, EF' is HIGH and the clock signal input

to U6A is inverted and passed through to the output of U6A. U3 TRBL' is thus pulsed and data is loaded into the UART. When U1 becomes empty, EF' goes LOW and the output from U6A will be HIGH, inhibiting the 9.5kHz clock signal from loading the UART. Figure 2.28 below shows a timing diagram for the case when FIFO U1 becomes empty.

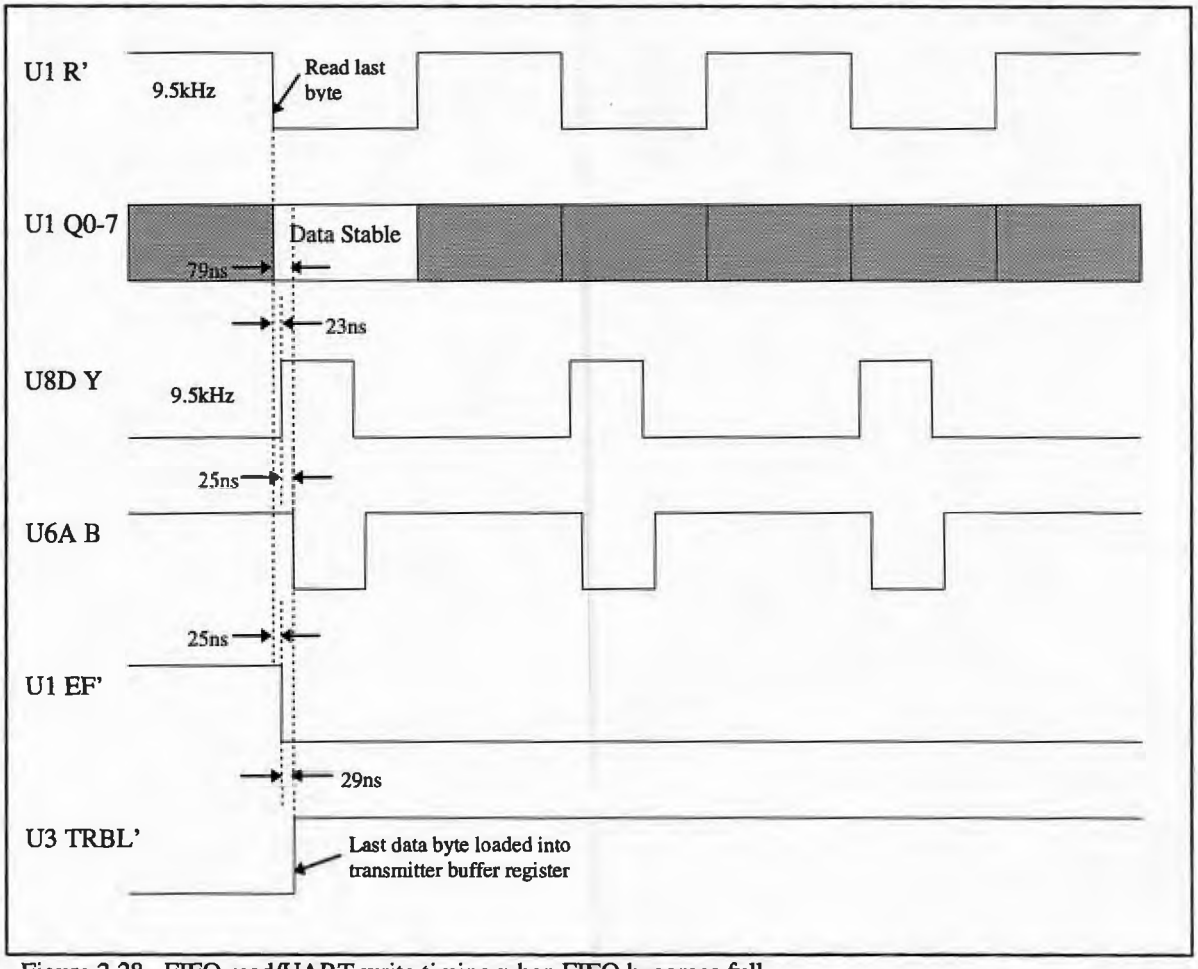


Figure 2.28. FIFO read/UART write timing when FIFO becomes full.

Power to the circuit shown in Figure 2.24 is provided by a 7805 5V voltage regulator. The 7805 provides reliable 5 volt power from a 9 volt plug-in power supply. The 5 volt power supply is turned on and off by a switch. An LED is provided as a convenience to indicate that the power is turned on. Figure 2.29 below shows the power circuitry. The

power circuit is mounted to the same circuit board as the previous circuitry. Several labeled test points are provided on the circuit board for testing and debugging purposes, including all SMS960 data lines and strobe, the FIFO W' inputs, the 19kHz 555 clock signal, the 9.5kHz FIFO R' inputs, the UART TRBL' inputs, the UART TRO outputs, the MAX232 T1 and T2 output, the +5V supply and ground.

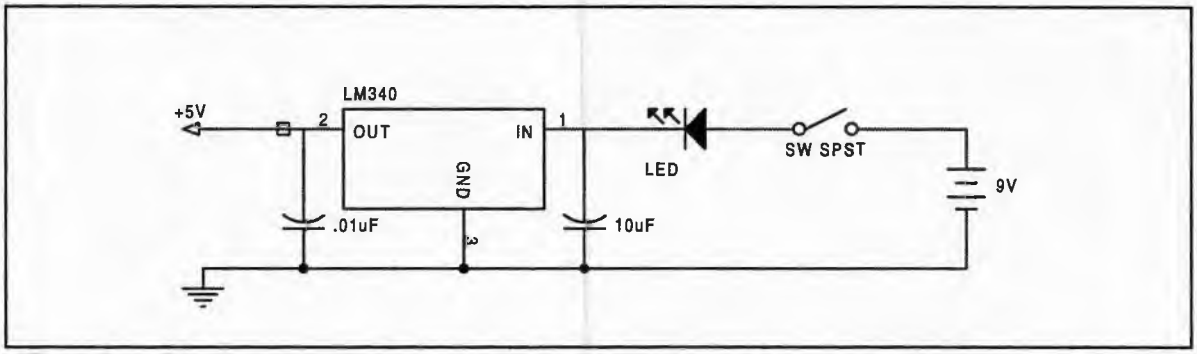


Figure 2.29. Parallel-to-RS232 conversion board power circuitry.

The circuit described above is assembled in a plastic housing with DB25 female input connector and two DB9 female output connectors. The output connectors are labeled Channel A and Channel B. Special cables have been made for connecting the DB9F outputs to the RJ-12 serial ports on the RTGIS Toolbox workstation. The two outputs from the SMS960 parallel-to-RS232 converter may be connected to any of the CYCLOM-8YS serial ports. However, as discussed below, the acquisition software must know where each channel is connected. Figure 2.30 below shows a block diagram of the parallel-to-RS232 converter and its interface to the sidescan hardware and acquisition computer.

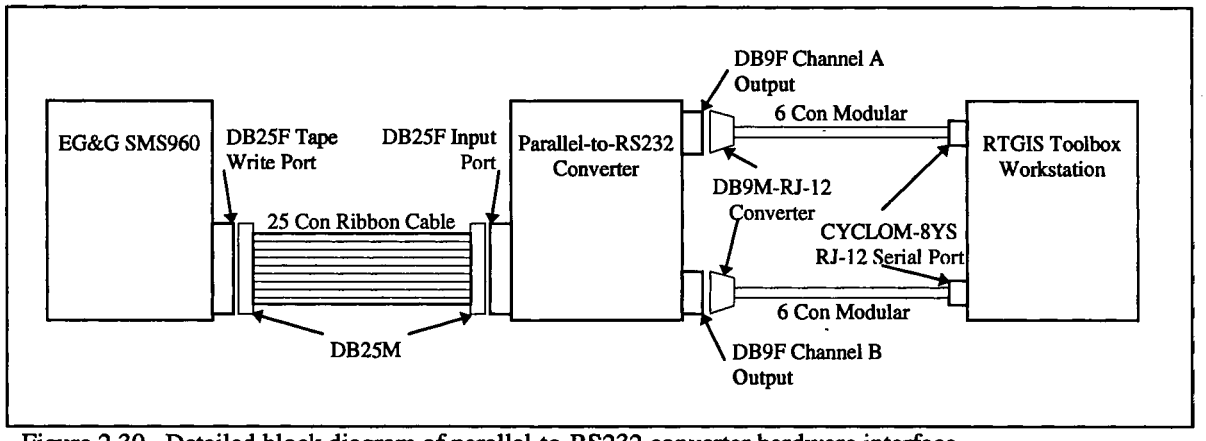


Figure 2.30. Detailed block diagram of parallel-to-RS232 converter hardware interface.

The new data acquisition system - Software

Acquisition Software

Digital sidescan sonar data from the SMS960 parallel-to-RS232 converter is acquired via a serial data logging program similar to the ser.* data acquisition programs discussed in Section 2.10. The program, ser.SMS960, contains several modifications from the standard ser.* program flow due to the need to read a single data source on two separate serial ports. The data from the two serial ports must then be reconstructed into a single contiguous data stream. Ser.SMS960 takes two arguments, the serial port to which the parallel-to-RS232 converter Channel A is connected and the serial port to which Channel B is connected. Ser.SMS960 does not need to know any other RS232 communications parameters because they are specific to the parallel-to-RS232 converter built and are hard-coded into the program (i.e., 115200kbps, 8 data bits, 1 start bit, 1 stop bit, no parity). Ser.SMS960 will synchronize itself to Channel A by waiting for Channel B to

become full, then emptying it, and then beginning to read channels alternately starting with Channel A. After ser.SMS960 successfully reads 25 pings to verify that a consistent data stream is being received, it will begin logging and displaying ping data.

At 100 meter range, the SMS960 digital data consists of 32 bytes of status data followed by 1768 bytes of alternating port and starboard ping data. Status bytes are differentiated from data by a HIGH value in bit number 7. The status data, which consists of time, date, speed, and heading, if provided, from the SMS960 front panel display, is discarded. Time, date, and navigation data (including speed, heading, and water depth) are added by ser.SMS960 through its interface to the navigation data acquisition program ser.MX4200D. Port data is differentiated from starboard data by a HIGH value in bit number 6. The remaining 6 bits (8 bits minus 1 status bit flag minus 1 port bit flag) indicate the backscatter intensity in decibels. The backscatter data ranges from 0-63dB, for a dynamic range of 64dB. Before the digital sidescan data is logged or displayed, it is converted from decibel values to greyshade values of 0-255.

Two methods are provided for converting the decibel data from the SMS960 to greyshade values. The first is a linear mapping of the backscatter pressures over the 0-255 grey values. The second is a linear mapping of the decibel values over the 0-255 grey values, which results in a logarithmic mapping of the backscatter pressures. The latter mapping method is quite straightforward. The decibel values range from 0-63; 0 being the weakest possible return and 63 the strongest. Therefore, 0 decibel is mapped to 0 grey (black) and

63 decibel is mapped to 255 grey (white). For any decibel value dB_n, the corresponding grey value $grey_n$, is determined from the following equation.

$$grey_n = grey_{max} - \frac{(grey_{max} - grey_{min})(dB_{max} - dB_n)}{(dB_{max} - dB_{min})} \quad \text{eq. 1}$$

where

$$grey_{max} = 255 \quad \text{eq. 2}$$

$$grey_{min} = 0 \quad \text{eq. 3}$$

$$dB_{max} = 63 \quad \text{eq. 4}$$

$$dB_{min} = 0 \quad \text{eq. 5}$$

Substituting equations 2-5, equation 1 reduces to the following.

$$grey_n = 255 - (255 - 4.05\ dB_n) \quad \text{eq. 6}$$

Equation 6 is used by ser.SMS960 to logarithmically map the backscatter pressures over 0-255 grey. The former mapping is slightly more complicated because the backscatter intensities must be calculated from the decibel notation. To start, for any backscatter pressure P_n, the corresponding grey value $grey_n$, is given by the following equation.

$$grey_n = grey_{max} - \frac{(grey_{max} - grey_{min})(P_{max} - P_n)}{(P_{max} - P_{min})} \quad \text{eq. 7}$$

The P_{min} and P_{max} values are computed using the decibel notation,

$$dB = 20 \log_{10} (P_n/P_{ref}) \quad \text{eq. 8}$$

where P_{ref} is 1uPa. For any decibel value dB_n, the backscatter pressure P_n, is given by the following equation.

$$P_n = P_{ref}\, 10^{(dBn/10)} \qquad \text{eq. 9}$$

Substituting equation 4 and 5, respectively, the following values are obtained.

$$P_{max} = 0.0014 \qquad \text{eq. 10}$$

$$P_{min} = 0.000001 \qquad \text{eq. 11}$$

Substituting equations 9-11, equation 7 reduces to the following:

$$grey_n = 255 - (255.18 - 0.18 \bullet 10^{(dBn/20)}) \qquad \text{eq. 12}$$

Equation 12 is used by ser.SMS960 to linearly map the backscatter pressures over 0-255 shades of grey.

After the SMS960 decibel output is converted to grayshade, the ping data is logged in QMIPS file format and displayed. The QMIPS files may be post-processed as discussed below.

Display Software

ShowImage, a sidescan sonar viewing and image manipulation program developed by William Danforth at the USGS, Woods Hole (Danforth, 1996), has been modified for use as the real time sidescan sonar display. Modifications to the X Window and Motif code in ShowImage permitted ShowImage to be used for real time waterfall display of the SMS960 digital data. A button, labeled "Real Time", has been added to the ShowImage File menu. Selecting the Real Time button will put ShowImage in real time mode. In real time mode, ShowImage will open a DTM request port and listen for requests to

display pings. When ser.SMS960 determines that ShowImage has opened its request port, it will send ping display requests. A ping display request to ShowImage consists of a header and ping data. The header contains two values; an integer indicating the number of pings sent and an integer indicating the number of bytes per ping. The data portion of the message contains pings of data, transmitted consecutively as separate DTM data sets. When ShowImage receives ping data, it adds the new data to a memory display buffer and then shows it in the ShowImage display window. By adding the new pings to the top of the memory buffer and discarding those which fall off the bottom, a waterfall display of sidescan sonar data is created.

Processing Software

Xsonar, developed by William Danforth at the USGS, Woods Hole (Danforth, 1996), is used to perform complete sidescan sonar swath processing. Xsonar is a comprehensive sidescan sonar data display and processing system. Xsonar uses the ShowImage application for display of raw and processed sonar data. While Xsonar and ShowImage are separate applications and may be used independently of each other, the functions in each program are complemented and enhanced by the other and both are necessary for the complete processing scheme.

Xsonar allows for the input of several sidescan sonar file formats; processes the sonar navigation; performs slant range, destripping, and beam angle corrections; produces histogram displays; and allows image processing functions such as linear stretching and histogram equalization. After complete swath processing, Xsonar will produce gridded

sidescan data in UTM projection. The gridded data may be output in raw raster format or TDU-850 thermal printer format. The TDU-850 output may be printed for hardcopy output. The raw raster format contains a 72-byte geographic header and may be imported to the GRASS GIS for additional processing. Figure 2.31 shows the Xsonar main menu.

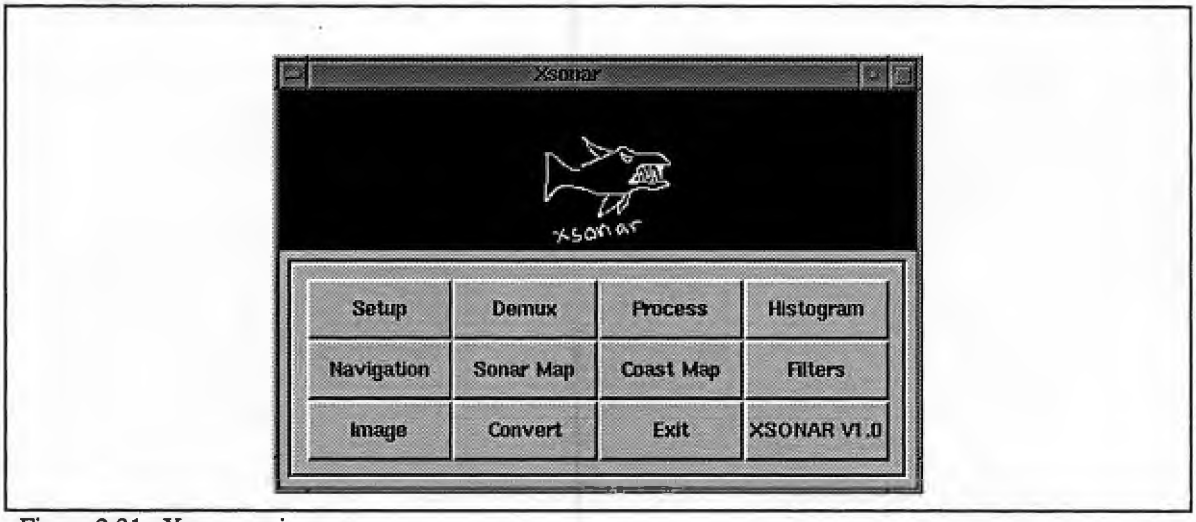


Figure 2.31. Xsonar main menu.

Figure 2.32 below shows the main Xsonar setup windows. At (A) is the File Parameter Setup Window which provides an interface for setting a number of parameters regarding the input file, the output grid, and the processing steps. At (B) is the File Input Format Selection Window which is obtained by the selecting the Convert button on the Xsonar main menu. This interface is used primarily for converting MIPS files (the format historically used by the OMDC) to Xsonar format.

A complete description and tutorial of Xsonar is provided by Danforth (1996). Below is a discussion of some of the highlights of Xsonar sidescan sonar processing and the interface created to the RTGIS Toolbox.

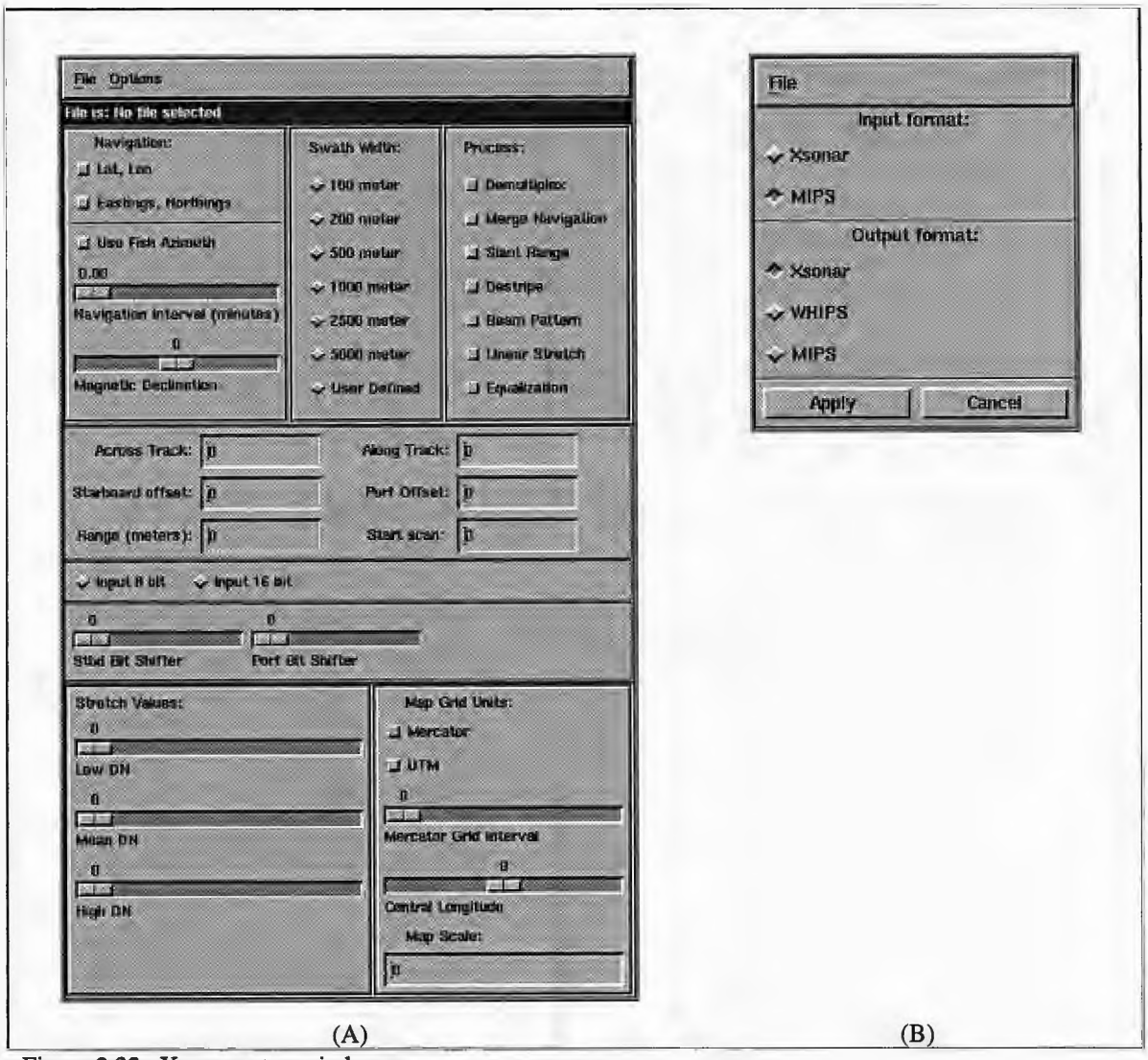


Figure 2.32. Xsonar setup windows.

Navigation Processing

Xsonar provides interactive navigation viewing and editing. When converting an input QMIPS sidescan sonar data file to Xsonar format, Xsonar will read the QMIPS navigation and create an ASCII navigation file. Duplicate and spurious navigation fixes in the ASCII navigation file are flagged and may be selected for deletion. Towfish layback and depth may be entered for application during the navigation processing. The

edited navigation file is then filtered and smoothed using a cubic spline algorithm and re-merged with the Xsonar format file.

Backscatter Data Processing

Xsonar provides complete processing of the backscatter data in a sidescan sonar file. The first step is to verify and/or correct the bottom tracking and then perform the slant range to ground range correction. Figure 2.33 below shows the ShowImage display window (Figure 2.33(A)) and ShowImage zoom window (Figure 2.33(B)). The zoom window is used for interactive, mouse-driven bottom tracking viewing and correcting. The ShowImage display window is used for viewing the sidescan sonar data at any intermediate or final processing step. Figure 2.33(A) shows a raw file before the slant range correction has been performed.

After water column removal and slant range correction, destriping is performed to fill in gaps between pings, most often caused by severe towfish heading or pitch changes. A beam angle correction is then performed to remove backscatter intensity differences which are caused by the angle at which the acoustic signal strikes the bottom due to bottom slope and not due to real geologic features. The sonar hardware also attempts to correct for this anomaly and the correction is included in the raw data. The correction is "hardcoded" into the sonar electronics and makes numerous assumptions. How well the correction works for a particular swath depends on many factors, including towing configuration and environmental parameters. The degree to which beam angle corrections must "adjust" the data varies significantly from swath to swath. To

accommodate this feature, Xsonar includes an averaging factor which determines the relative magnitude of the beam angle correction and also provides the ability to correct the port and starboard channels to different magnitudes.

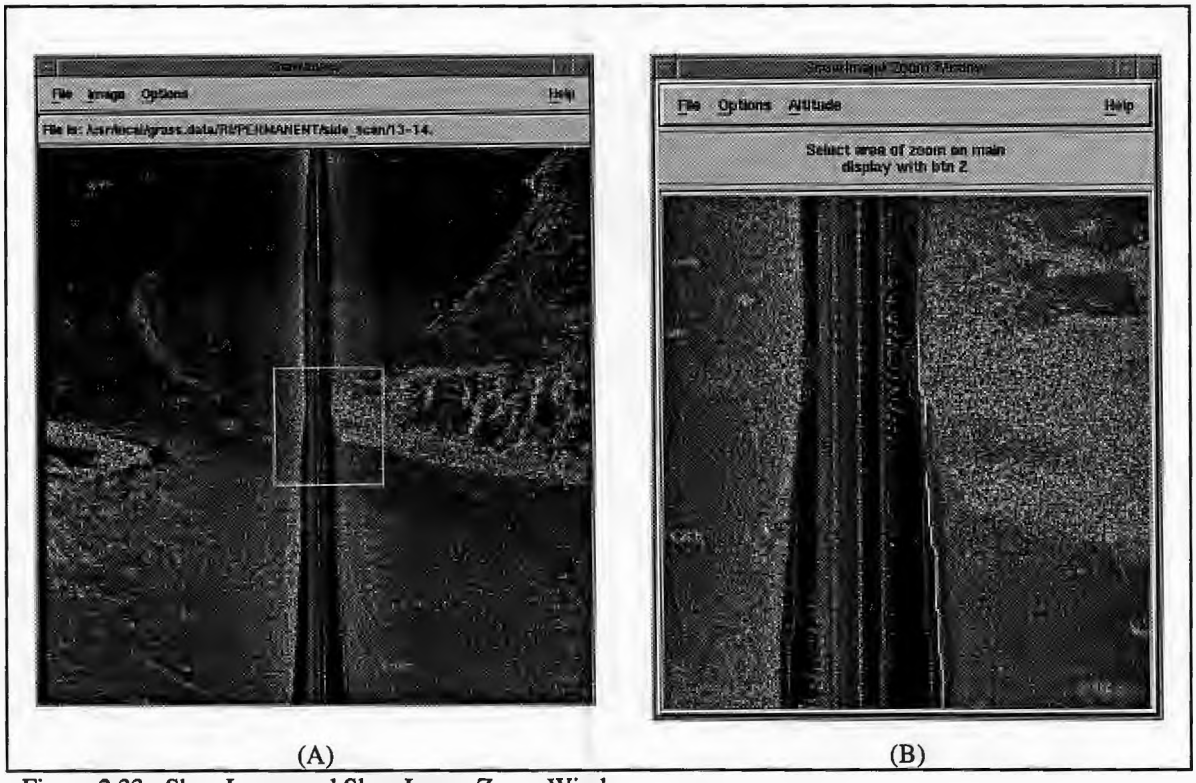


Figure 2.33. ShowImage and ShowImage Zoom Window.

After geometric and radiometric corrections have been made, several image processing functions are included for further swath processing. Linear stretching and equalization of the swath imagery may be performed. The Filters button on the Xsonar main menu provides an interface to performing low pass, high pass, median, and other filtering of the swath data. A median filter may be used to remove any speckle noise from the swath imagery.

Geographic Gridding

When all swath processing is complete, Xsonar will produce grids of the sonar data on the UTM projection. The user may define the extent of the swath to grid, the grid resolution, and the output grid type. Two grid types are used by the RTGIS Toolbox. The first is TDU-850 format grids. These grids are binary raster files which may be dumped to the TDU-850 thermal printer connected to the RTGIS Toolbox workstation parallel port and provide a means for obtaining immediate sidescan swath hardcopy.

The second grid type consists of binary raster data with a geographic header attached. The header indicates the grid's geographic extent and resolution. The header data is used by RTGIS Toolbox tools to import the Xsonar raster as a GRASS raster data layer. Figure 2.34 below shows the Xsonar raster import utility which interfaces to the command line program f.xsonar2grass.

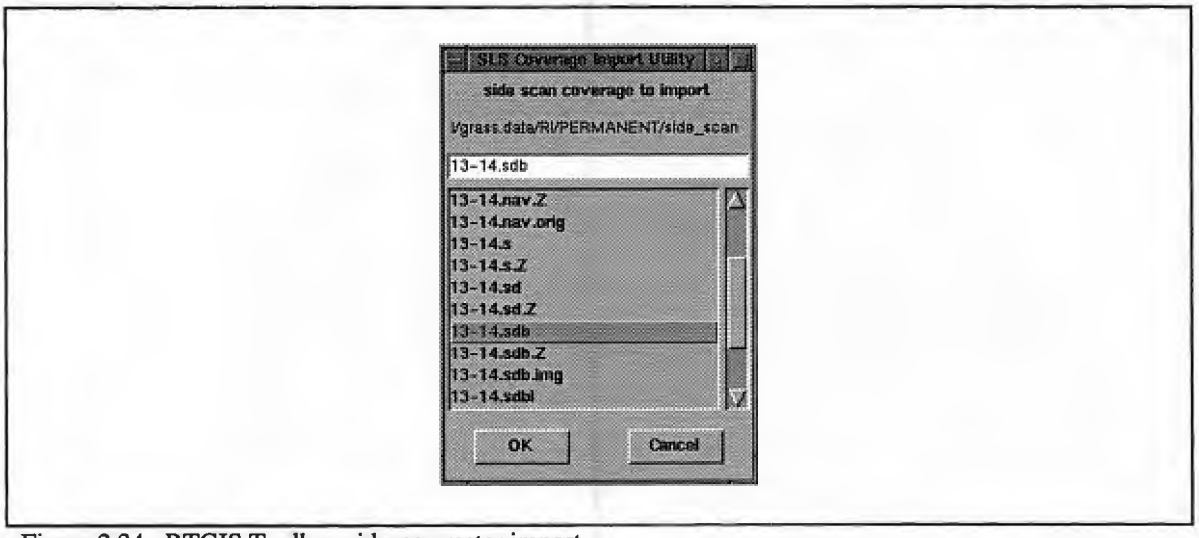


Figure 2.34. RTGIS Toolbox sidescan raster import.

Mosaicing

Mosaicing of swath data is performed in the GIS by GRASS commands. Several scripts have been created for combining swaths using different overlay formulas in r.mapcalc. The mosaic procedure consists of combining swaths in a swath-by-swath fashion until all swaths have been added to the mosaic. The mosaic may be viewed at any scale during the mosaicing procedure to ensure that swaths are being properly located. GRASS programs provide translation and rubbersheeting of the swath data if necessary.

2.16 Single Beam Bathymetry Data Processing

A number of programs were written for processing single beam bathymetry data. The Single Beam Tool provides a graphical interface to these programs. The input is depth files logged by the RTGIS which contain time-stamped depths. The output is a fully processed GRASS raster bathymetry data layer. The processing steps are as follows:

1. Filter navigation
2. Apply transducer depth offset
3. Apply sound velocity correction
4. Apply tidal correction
5. Merge bathymetry with navigation
6. Produce grid

Filter Navigation

A navigation filtering program was written to identify and remove artifacts in the navigation data resulting from spurious data points, GPS wandering, and temporary losses of differential corrections to the GPS data. The program, m.navfilter, expects as input either the name of a navigation file to filter or a Julian day for which all navigation files will be filtered together. The output is a navigation file of the same format as the input but with filtered values and a ".fix" extension appended.

Transducer Offset

This correction consists of applying a single constant offset to the depth data. The depth data is referenced to the transducer (i.e., the depth values are a measure of the distance of water between the transducer and the seafloor). In order to obtain true depths from the water surface to the seafloor, the depth of the transducer below the water surface must be added to each depth value. Figure 2.35 below shows the geometry involved.

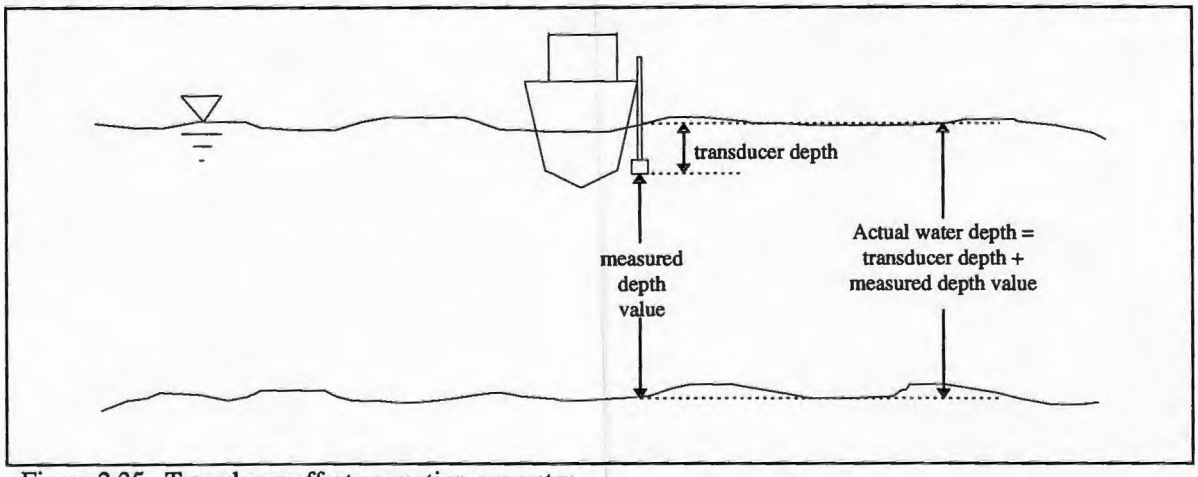


Figure 2.35. Transducer offset correction geometry.

The program m.xducerdep performs this correction. The inputs to m.xducerdep include the transducer depth, the units of the transducer depth (feet or meters), and the file to correct or a Julian day's worth of files to simultaneously correct. The output is a depth file of the same format containing corrected depths. The output file name will be that of the input file with a ".x" extension appended.

In general, horizontal offsets must be made to the navigation data to account for the position of the transducer relative to the navigation antenna. However, the standard shipboard survey configuration for this thesis mounted the GPS navigation antenna directly to the pole to which the transducer was mounted. Thus the horizontal position from the navigation receiver was also the horizontal position of the transducer and such a correction utility was not provided as part of this work.

Sound Velocity Correction

This correction uses Hydrolab H-20 log files to compute a sound velocity profile (SVP). The H-20 log files contain time-stamped records of salinity in parts per thousand, temperature in degrees Celsius, and depth in meters read from markings on the H-20 instrument cable. The Hydrolab H-20 uses the following formula to compute salinity in units of parts per thousand from conductivity measurements.

$$S = 5.9950(10^{-8})C^4 - 2.3120(10^{-5})C^3 + 3.4346(10^{-3})C^2 + 5.3532(10^{-1})C - 1.5494(10^{-2}) \quad \text{Eq. (13)}$$

This relationship is based on salinity data from the USGS Water Supply Paper No. 2311 (Hydrolab Corp., 1991). The computed SVP is then used to correct the water depths. The program which performs this correction, m.velcor, provides three different methods

for computing the SVP using the Hydrolab H-20 temperature and salinity data and logged depth. The three methods include the Leroy, Medwin, and Mackenzie sound velocity expressions listed by Urick (1983). These three expressions, respectively, are given by the equations below.

$$c = 1492.9 + 3(T - 10) - 6\times10^{-3}(T - 10)^2 - 4\times10^{-2}(T - 18)^2 + 1.2(S - 35) - 10^{-2}(T - 18)(S - 35) + D/61 \quad \text{Eq. (14)}$$

$$c = 1449.2 + 4.6T - 5.5\times10^{-2}T^2 + 2.9\times10^{-4}T^3 + (1.34 - 10^{-2}T)(S - 35) + 1.6\times10^{-2}D \quad \text{Eq. (15)}$$

$$c = 1448.96 + 4.591T - 5.304\times10^{-2}T^2 + 2.374\times10^{-4}T^3 + 1.340(S - 35) + 1.630\times10^{-2}D + 1.675\times10^{-7}D^2 - 1.025\times10^{-2}T(S - 35) - 7.139\times10^{-13}TD^3 \quad \text{Eq. (16)}$$

Where D is the depth in meters, S is the salinity in parts per thousand, and T is the temperature in degrees Celsius. The user selects the desired computation method via command line arguments. M.velcor computes the SVP internally and writes it out as an ASCII text file to the svp/ MAPSET ELEMENT and a Postscript plot to the psmap/ MAPSET ELEMENT. Figure 2.36 on the following page shows a sample SVP Postscript plot output by m.velcor. The filled triangles represent the computed sound velocity for the measured depths. The profile line is computed as a cubic spline interpolation between the sampled points. The "Figure" caption at the bottom of the plot is intentionally printed without a number so that the plot can be included as a figure in a report and numbered appropriately.

Once the SVP has been calculated, the total travel time, Δt_{TOTAL}, for each sample is calculated from the depth measurement, d, using the following formula (also used by the Datamarine echo sounder electronics to compute the depth value).

$$\Delta t_{TOTAL} = d / 1500 m/s \qquad \text{Eq. (17)}$$

The SVP, which is a function of depth, is then divided into a discreet number of intervals on which the sound velocity is assumed constant and is computed from the SVP using cubic spline interpolation between the original sample points. The interval distance is

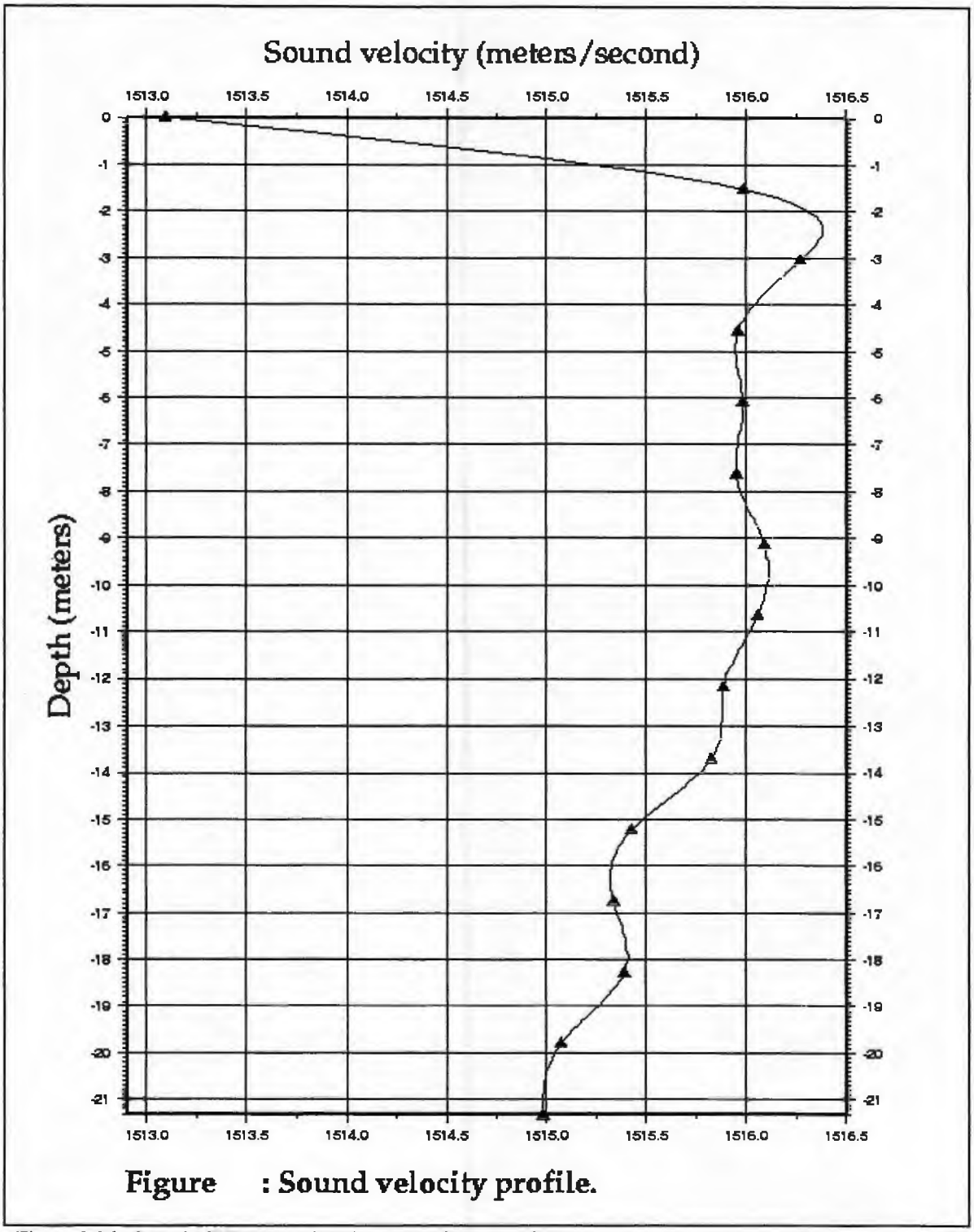


Figure 2.36. Sample SVP Postscript plot output from m.velcor.

determined by the user and entered as one of the inputs to m.svpcor. The travel time through each depth interval is computed and added together until the total travel time for the current depth sample is reached. The depth interval at which the total travel time is obtained is the true depth value. The output is a depth file of the same format with depth values corrected for sound speed variations. The output file name will be the same as that of the input file name with a ".s" extension appended.

Tidal Correction

This correction uses NOAA format tide tables available via anonymous FTP to correct the depth values for the tidal variation about the mean lower low water (MLLW) level. The NOAA tide data is given at a temporal resolution of 6 minutes. The tidal elevation for each depth value is determined using cubic spline interpolation of the 6 minute tide data. Inputs to the program m.tidecor include the NOAA tide table file name and the depth filename to process or a Julian day for which to process all depth files. The output is a depth file of the same format with depth values corrected to MLLW. The output file name will be the same as that of the input file name with a ".t" extension appended.

Merge Bathymetry With Navigation

The fully corrected echo sounder data contains only depth values and time stamps. In order to locate the depth values accurately, they must be merged with navigation via the time values. Since the echo sounder data is received at 5 samples per second and navigation at only 1 sample per second, positions must be interpolated from the navigation fixes. The program m.regbathy does this using a simple linear interpolation.

For fixes separated by only 1 second, a linear interpolation has proven to be sufficient. The output from this program is a GRASS site file containing UTM positions and depth values.

Gridding

The final step in single beam bathymetry processing is to create a raster bathymetry data layer from the site samples. This is accomplished via the GRASS program s.to.rast which converts GRASS site files to GRASS raster files. S.to.rast uses nearest neighbor weighting to compute cell values for the output raster. Inputs to s.to.rast include the number of neighbors to include in cell computations. The resolution of the output raster will be that of the current geographic region. For example, if the Region Manager Tool (Section 2.4.3) is used to set the resolution of the current geographic region to 10 meters, s.to.rast will create an output raster with a grid cell size of 10 meters.

2.17 Multibeam Sonar Data Acquisition, Display, and Processing

Multibeam sonar basic theory

100% seafloor coverage is now perceived as part of the hydrographic requirement for safety of navigation. Swath bathymetric surveys in the deep sea have demonstrated that this technique has the potential to solve the problem of incomplete coverage that plague all single beam sounding surveys. Shallow-water swath bathymetry provides a possible solution to this demand (Clarke, 1994). In order to accommodate the growing demand and need for swath bathymetry (multibeam) surveying for coastal waters, the RTGIS

Toolbox has been equipped with several multibeam sonar data acquisition, display, and processing tools.

RTGIS Toolbox Multibeam Tools

There are two separate sub-systems which contribute to the multibeam functionality of this thesis. The two sub-systems are differentiated by their availability; one is public domain and freely available with the RTGIS Toolbox distribution, the other is proprietary software of SeaBeam Instruments, Inc. and available only through SeaBeam licensing. The public domain sub-system consists of the MB-System multibeam processing system (Caress and Chayes, 1996) and provides complete post-processing of multibeam sonar data in numerous file formats. The SeaBeam proprietary sub-system consists of a subset of the SeaView™ software suite and provides additional post-processing techniques, along with real time multibeam data acquisition, display, and processing.

MB-System

MB-System includes tools for modeling water sound velocity profiles, calculating multibeam bathymetry from travel time values by raytracing through a water sound velocity profile, interactive and automatic editing of multibeam bathymetry, as well as a variety of tools for the manipulation and display of multibeam data (Caress and Chayes, 1996). A modular input/output library allows MB-System programs to access and manipulate data in any of a number of supported swath-mapping sonar data formats (Caress and Chayes, 1996). The MB-System software package consists of more that 20 programs which manipulate, translate, process, list, or display swath-mapped sonar data.

The source code is freely available from the Lamont-Doherty Earth Observatory via anonymous FTP (Caress and Chayes, 1996).

Within the MB-System sub-system of the RTGIS Toolbox, three tools are provided in addition to the MB-System software. The MB Project Tool, the MB Operator Tool, and the MB Process Tool. These tools are shown in Figure 2.37 below.

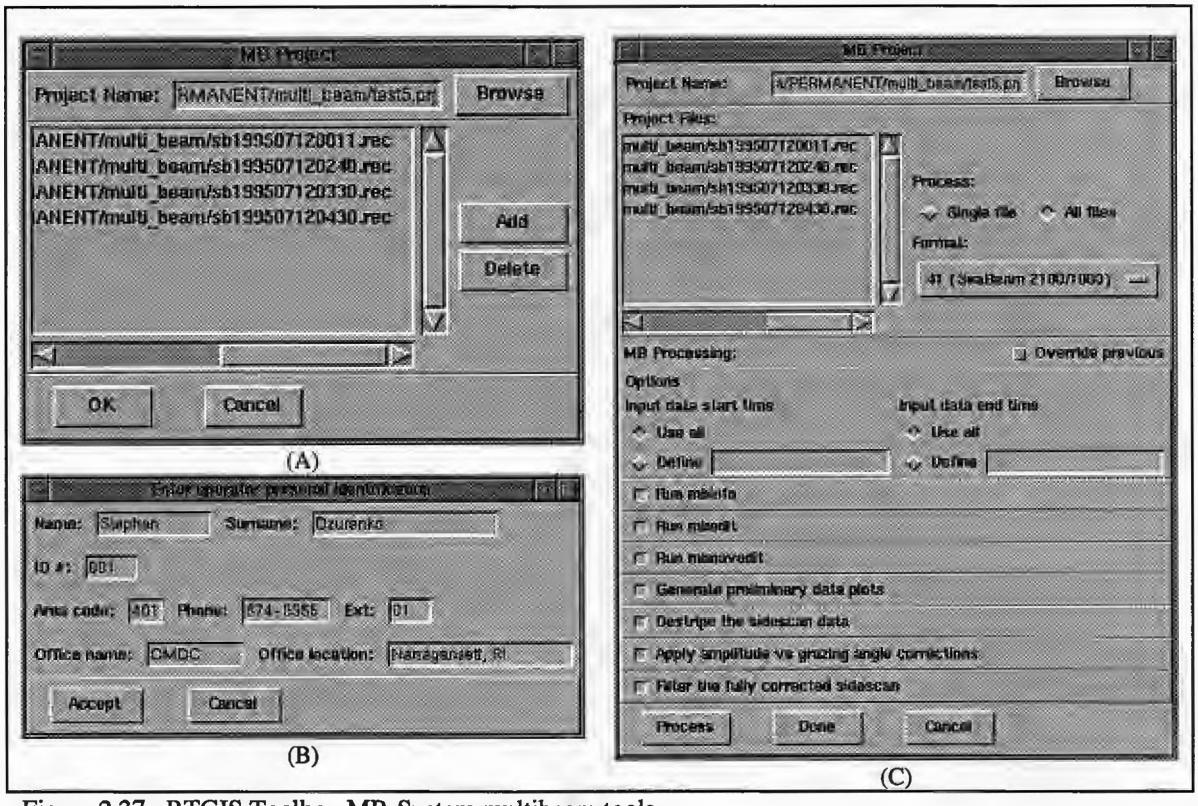


Figure 2.37. RTGIS Toolbox MB-System multibeam tools.
(A) MB Project Tool, (B) MB Operator Tool, (C) MB Process Tool

The MB Project Tool, Figure 2.37(A), and MB Operator Tool, Figure 2.37(B), provide added convenience for the hydrographic data processor. The MB Project Tool assists the hydrographer in keeping track of multibeam processing by organizing all the data files to be processed into a single multibeam project. MB Project allows the user to select a project name and add and delete data files. MB Project then creates a ".prj" project file in

which is recorded all data files included in the project and the processing history for all data files. By default, the project file and all data files are stored in the multibeam/ ELEMENT of the current GRASS MAPSET. The MB Process Tool, Figure 2.37(C), controls interactive processing of a multibeam project. Before MB Process can start, the user is required to fill in all fields of the MB Operator Tool. MB Operator assists in hydrographic processing by tracking the individuals who have processed files in a project. Each processing step in MB Process is recorded in the project file along with the operator's personal information. Once MB Operator is completed, MB Process allows the user to select the project to process and whether to process a single file at a time or all at once. The user may select the data file format and set common options. Then the user is guided through the complete processing; from generating file stats to importing grids into the GIS. Interfaces are provided to numerous MB System commands, including mbnavedit, the interactive navigation editor, and mbedit, the interactive beam editor. Mbnavedit and mbedit are shown in Figure 2.38 below.

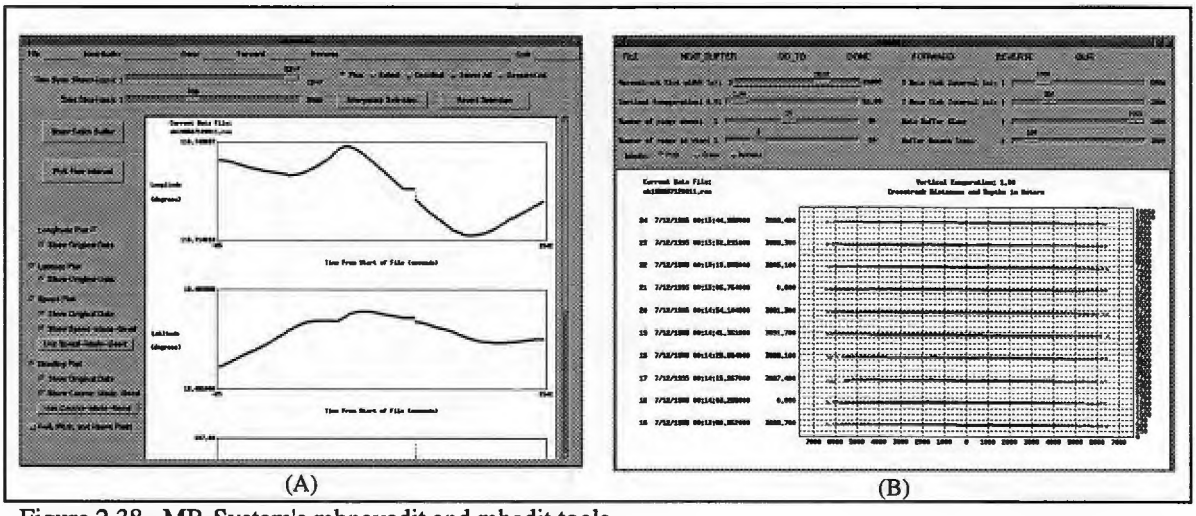


Figure 2.38. MB-System's mbnavedit and mbedit tools.

Mbnavedit displays the latitude, longitude, heading, and speed values. The user may inspect the data on a sample-by-sample basis and flag spurious values to ignore in processing. Mbnavedit will interpolate new values for samples that have been flagged. Mbedit displays the depth values for all samples in every ping. The user may scroll through all pings in a data file and inspect the depth values on a beam-by-beam basis and flag spurious values to ignore during subsequent processing. After filtering the multibeam data through mbnavedit and mbedit, mbm_plot is used with the Postscript output system discussed in Section 2.14 to produce preliminary hardcopy output.

SeaView

Multibeam sonar data acquisition is provided for using Seabeam 2000 series sonars and the a subset of the proprietary SeaView™ software system. Those components ported include SeaLogger, SeaSwath, and SeaPatch. Figure 2.39 on the following page shows the real time data acquisition configuration.

SeaLogger, Figure 2.39(A), is a multibeam acquisition and logging tool. When multibeam surveying is to begin, SeaLogger is started. SeaLogger will then listen for data on a user configurable ethernet socket. Once data is received, the user may begin logging the data to file. SeaSwath consists of a real time waterfall display of the multibeam data being logged by SeaLogger. Both bathymetry and sidescan can be selected for display. Figure 2.39(A) shows SeaLogger as it is logging data via ethernet. Figure 2.39(B) shows SeaSwath in operation with waterfall bathymetry on the left and waterfall sidescan on the

Figure 2.39. SeaView's SeaLogger and SeaSwath tools.

right. Both tools show current ping stats at the lower left. A block diagram of the multibeam acquisition system is shown in Figure 2.40 on the following page. Also on the following page, Table 2.18 summarizes the software which pertains to the multibeam sonar data acquisition, display, and processing sub-system.

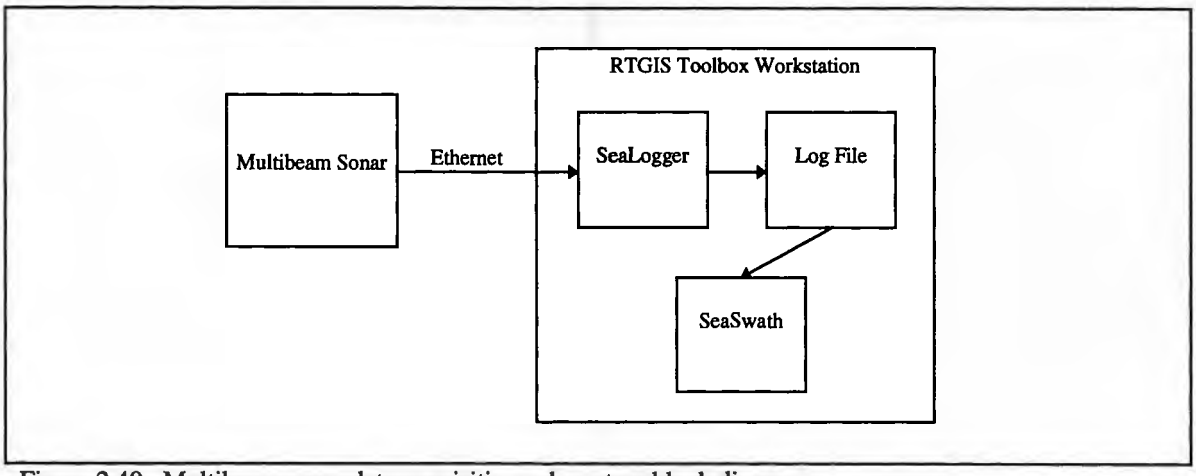


Figure 2.40. Multibeam sonar data acquisition sub-system block diagram.

Table 2.18. Software components of the RTGIS Toolbox multibeam sonar data acquisition, display, and processing sub-system.

Software	Type	Location	Lines	Description
MB-System	C software system	/usr/local/mbsystem/	>1000's	Multibeam processing system
G_mb_project_edit	Tcl process	$GIS/scripts/tcl/mb m_project.tcl	233	Controls creation and editing of RTGIS MB-System Project files
G_mb_project_process	Tcl process	$GIS/scripts/tcl/mb m_project.tcl	364	Controls MB-System processing of the multibeam data files in a project file
G_mbinfo	Tcl process	$GIS/scripts/tcl/ mbm_mbinfo.tcl	228	Controls mbinfo processing for a multibeam project
G_mbnavedit	Tcl process	$GIS/scripts/tcl/ mbm_mbnavedit.tcl	134	Controls mbnavedit processing of the selected multibeam files
G_mbedit	Tcl process	$GIS/scripts/tcl/ mbm_mbedit.tcl	129	Controls mbedit processing for a multibeam project
G_mbmplot_prelim	Tcl process	$GIS/scripts/tcl/mb m_mbm_plot.tcl	356	Controls running mbm_plot on the selected multibeam files to produce first cut data plots
G_mbfilter_destripe	Tcl process	$GIS/scripts/tcl/mb m_destripess.tcl	183	Controls destriping of side scan data in the currently selected multibeam files
G_mb_get_options	Tcl process	$GIS/scripts/tcl/mb m_util.tcl	58	Constructs and returns the options list for the given tcl process
G_mb_get_operator	Tcl process	$GIS/scripts/tcl/mb m_util.tcl	195	Controls initializing of the current processing operator's personal information
G_mb_update_projectfile	Tcl process	$GIS/scripts/tcl/mb m_util.tcl	78	Controls updating of the project file when a processing step is completed
SeaLogger	C software system	/usr/local/sealogger	>1000	SeaBeam 2100 Multibeam sonar data acquisition and logging system
SeaSwath	C software system	/usr/local/seaswath	>1000	Real time SeaBeam 2100 multibeam sonar data display

3. Oceanographic Surveying

Chapter 2 detailed the RTGIS Toolbox system design and implementation. In this chapter, use of the complete RTGIS Toolbox for an actual survey will be described. Just as Chapter 2 was divided into subsections detailing the various RTGIS Toolbox sub-systems, this chapter will be divided into subsections detailing the phases of the oceanographic survey and how RTGIS Toolbox components play a role in each phase.

3.1 Pre-Cruise Planning

The pre-cruise planning phase must be performed before the survey vessel begins the survey. This requirement results from the fact that the primary objective of the pre-cruise planning is to determine where the survey vessel should go. Indeed, all survey planning tools have been designed to aid in determining the best survey route. How far in advance of the survey the pre-cruise planning should begin is entirely dependent on the nature of the cruise. For small cruises performed a day at a time in coastal waters where the prior history of the survey area is well known, pre-cruise planning can be completed the morning of or day before the cruise. The Line Planning Tool, along with NOAA chart overlay, provides ample means for planning a survey in such a scenario. For larger surveys in unfamiliar areas, pre-cruise planning should be started a week or more in advance. In this scenario, full advantage may be taken of all the RTGIS Toolbox tools for advanced survey planning.

The first step in pre-cruise planning is to determine the approximate extent of the intended survey area. The estimates should be liberal to include some of the surrounding area. For the demo survey, bathymetry and sidescan data will be collected from an area between two prior surveys in Narragansett Bay. One survey was in Coddington Cove in the Northern East Passage and the other was around Gould Island, just west of Coddingtion Cove in the East Passage. Once the general survey area has been determined, the RTGIS Toolbox is started. From the GIS Tools menu on the RTGIS Main Menu, the Display Monitor Tool, the Region Manager Tool, the Data Browser, and the Coverage Viewer are started. The Display Monitor Tool is used to start and select GRASS display monitors for viewing data. The Region Manager Tool is used to interactively set and modify the desired geographic region. The Data Browser is used to list and select GRASS database files for display. The Coverage Viewer is used to display coverage of raw, unprocessed data files. Figure 3.1 shows a typical screen setting up for survey planning.

The System Toolchest, Figure 3.1(A), contains a button labeled RTGIS. This button is used to start the RTGIS Toolbox. The RTGIS Toolbox is shown at Figure 3.1(B). The Display Monitor Tool, Figure 3.1(C), was used to start the GRASS display monitor shown at Figure 3.1(D). The Region Manager Tool, Figure 3.1(E), was used to set and modify the geographic region until a satisfactory region is obtained. The Data Browser, Figure 3.1(F), was used to display existing database files, such as the NOAA chart, the coastline, and the Coddington Cove mosaic. The Coverage Viewer, Figure 3.1(G), was used to display unprocessed data coverage. The sidescan data from the Gould Island

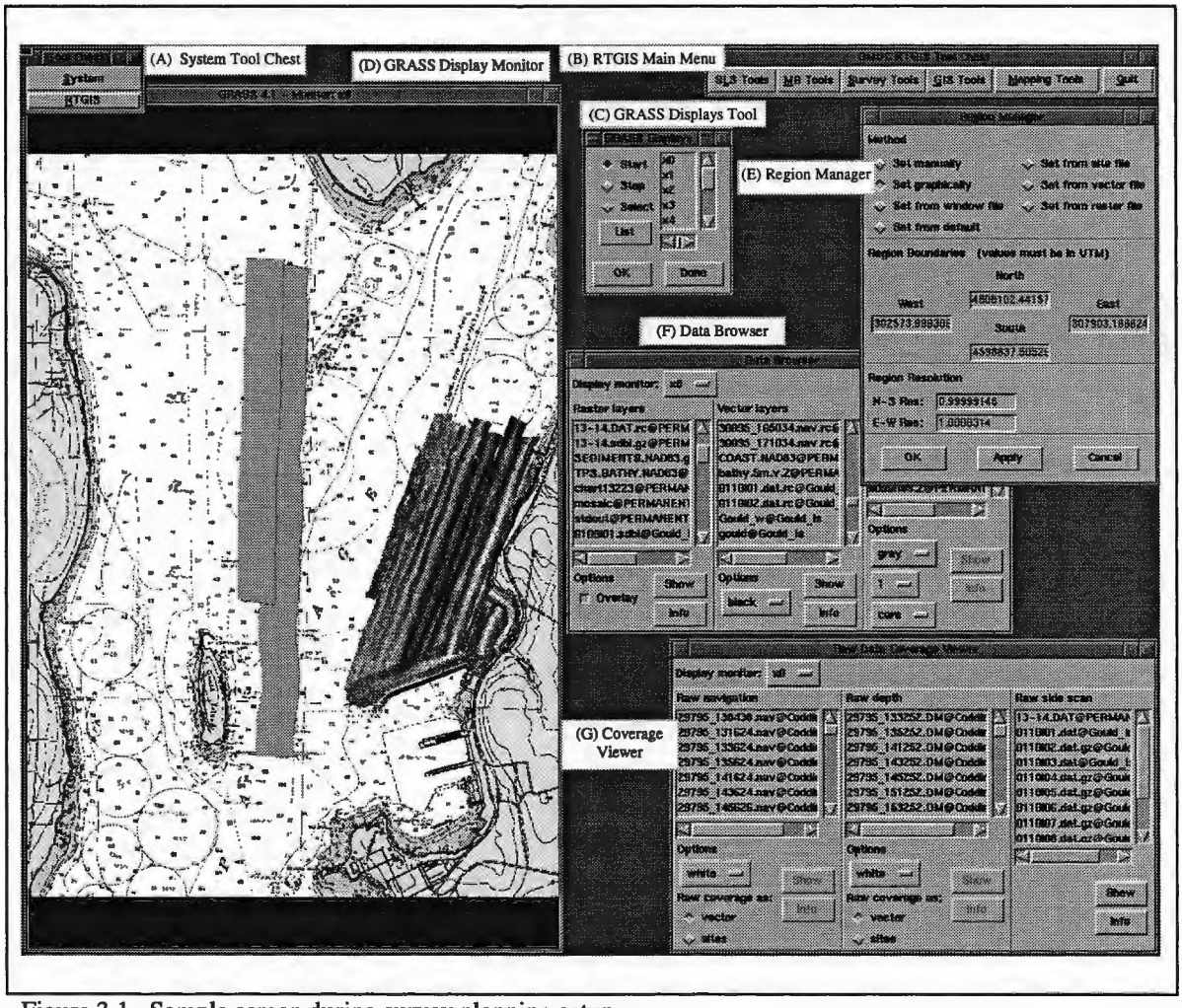


Figure 3.1. Sample screen during survey planning setup.

survey has not yet been processed to a mosaic. However, the Coverage Viewer will display the raw coverage to give a good indication of the location of collected data so that the next survey may be planned without gaps in the data or redundant.

With the geographic display shown in Figure 3.1(D), it is easy to lay out a series of lines to accurately define the survey. The Line Planning Tool is used to interactively create the survey lines. Figure 3.2 shows a typical screen during line planning.

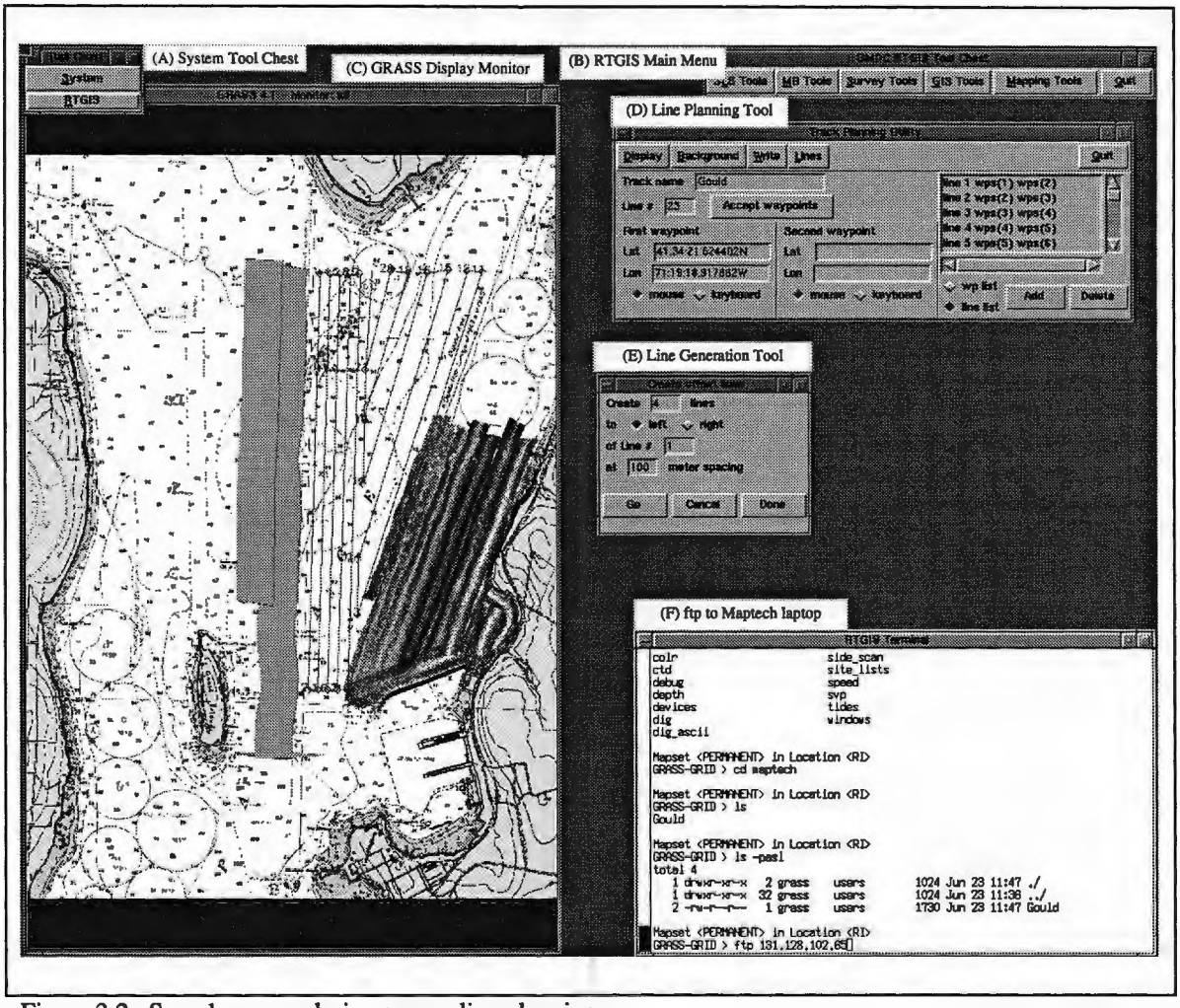


Figure 3.2. Sample screen during survey line planning.

The Line Planning Tool, Figure 3.2(D), was used to create the survey lines shown in the GRASS display monitor at Figure 3.2(C). An initial line was created by point-and-click with the mouse. The Offset Line Tool, Figure 3.2(E), was then used to generate a series of offset lines parallel to the first line. The line spacing used for the offset lines is determined from the desired sidescan sonar range and desired overlap. With a desired range of 100 meters and overlap of at least 100%, the spacing between lines should be no greater than 100 meters. Menu options in the Line Planning Tool allow the user to convert the planned survey lines to GRASS site and vector database files. The GRASS

vector file is then converted to a Maptech Professional format route file. The Maptech route file is then transported to the Maptech helm guidance computer via ethernet ftp, as shown in the GRASS terminal window at Figure 3.2(F). The ship operator then steers the survey lines via the helm guidance software.

The procedure described above and depicted in Figures 3.1 and 3.2 is the simplest form of survey planning offered by the RTGIS Toolbox; survey line layout on top of reference data layers. The procedure is greatly simplified by having access to the digital NOAA chart data layer. However, the RTGIS Toolbox provides a method to import a scanned NOAA chart for areas for which there is not already one in the GRASS database. In addition, the RTGIS Toolbox provides a tool for creating a preliminary Digital Terrain Model (DTM) from the imported NOAA chart. From the DTM, several advanced survey planning features are provided. Figure 3.3 shows a typical screen during the NOAA chart import procedure.

The NOAA Chart Import Tool, Figure 3.3(D), provides automated import of scanned NOAA charts on CDROM to a GRASS raster database file; it was used to import the NOAA chart shown in Figure 3.1(D) and Figure 3.2(C). The chart import procedure automatically creates a GRASS image file from separate image tiles and interactively lets the user geographically rectify the chart image. Figure 3.3(F) shows the i.points menu. I.points is a GRASS program that allows registration of points on an image for later image processing. The GRASS display monitor, Figure 3.3(C), shows the registration window. The user is registering the point at the intersection of the lat/lon grid lines

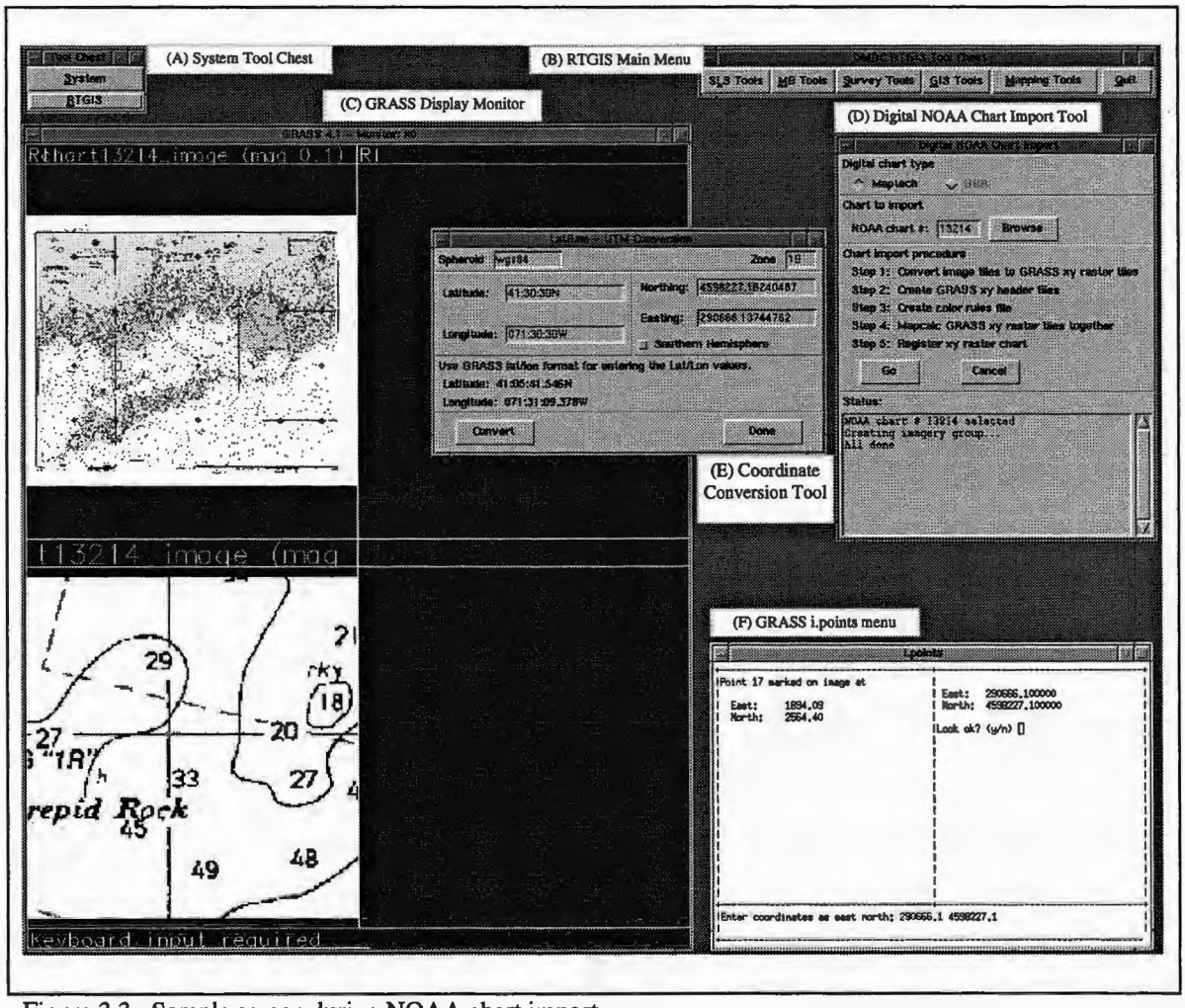


Figure 3.3. Sample screen during NOAA chart import.

shown. The Lat/Lon - UTM Tool, Figure 3.3(E), assists the user in converting the chart grid lat/lons to UTM (the units required by i.points). After a sufficient number of points have been registered, the image is rectified. The result is a geographically true GRASS raster layer in the UTM projection.

The DTM Creation Tool, Figure 3.4(D), allows the user to interactively digitize soundings on the NOAA chart (Figure 3.4(C)) and then automatically creates a DTM.

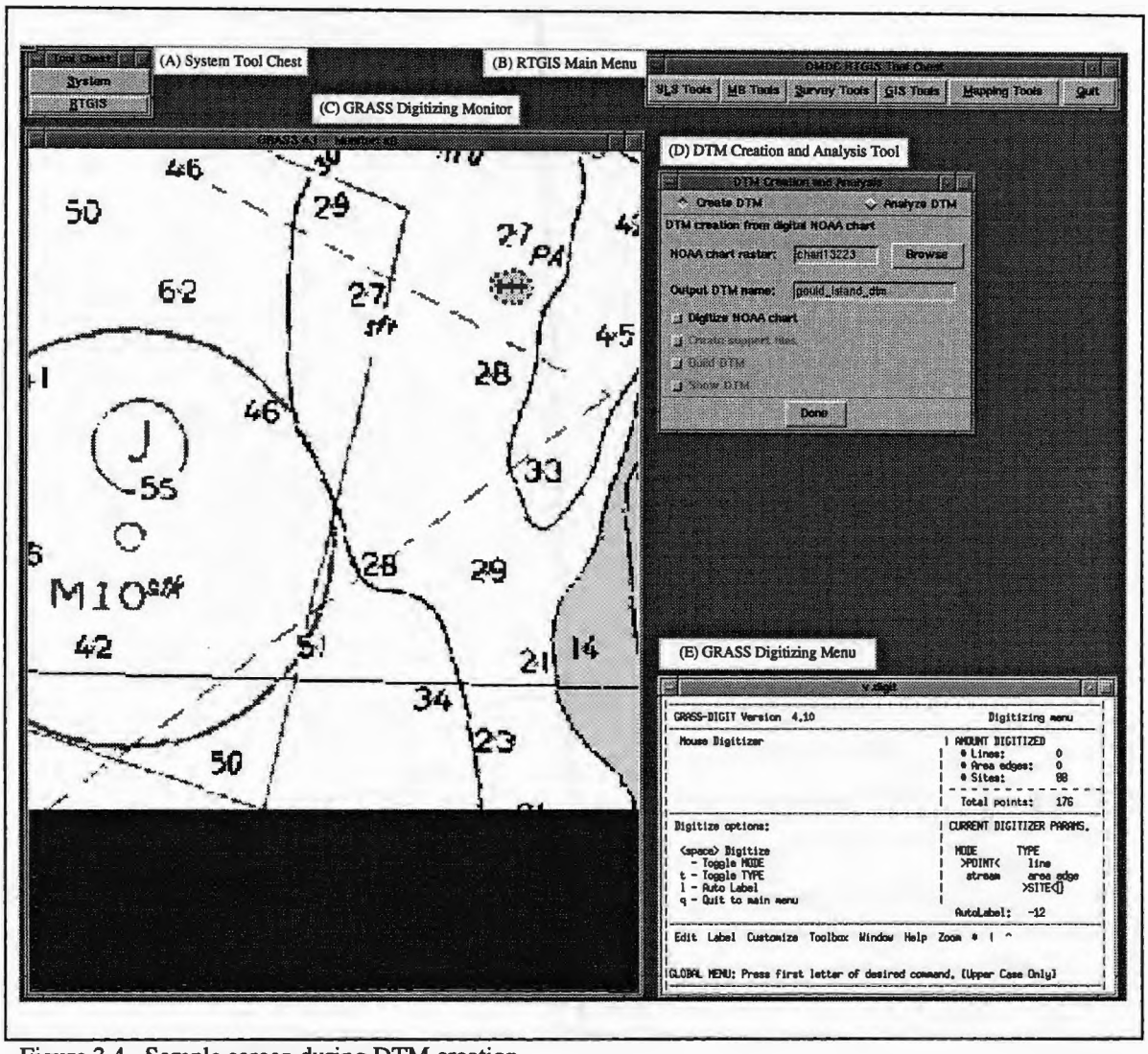


Figure 3.4. Sample screen during DTM creation.

The DTM Creation Tool is designed specifically to work with NOAA chart rasters. The first step in creating a DTM is to digitize chart soundings and contours. Figure 3.4(E) shows the GRASS digitizing menu. Figure 3.4(C) shows the digitizing display with a number of soundings digitized; digitized locations are indicated by a small colored "x" and, optionally, the digitized label input by the user. After digitizing is complete, the DTM Creation Tool builds all required support files automatically, then builds the DTM and finally shows the DTM as a 2D raster data layer.

The DTM Creation Tool is also used for interactive analysis of the DTM to provide further insights for optimized survey planning. The simplest form of DTM analysis is 3D viewing of the DTM. For most small surveys in restricted geographic regions, simply viewing the DTM provides the survey planner with enough insight for advanced survey planning techniques, such as pre-determining towfish depth and cable-out requirements and re-orienting the survey lines for optimized coverage. However, tools are provided to assist in these procedures. The DTM Analysis Tool is used for slope and aspect analysis of the DTM. Slope and aspect data can suggest an optimal orientation for survey lines based on user parameters. The DTM Analysis Tool can also compute tow fish depth and cable out raster data layers based on user input. These data layers are used in the planning phase to determine restrictions on the towfish depth and in the data acquisition phase to warn the user of impending tow fish grounding.

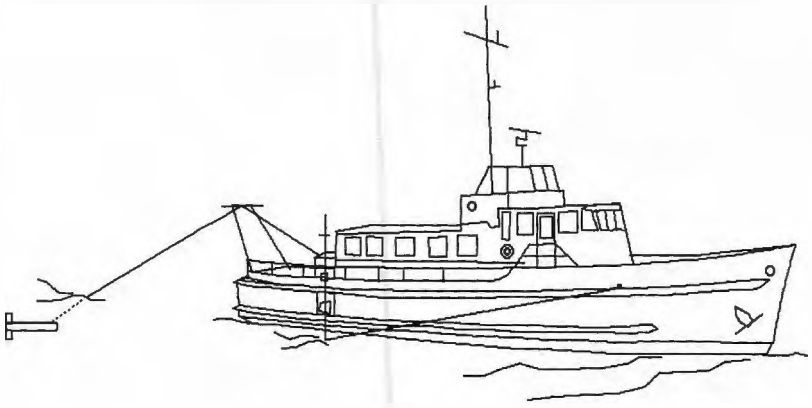
3.2 Data Acquisition

The data acquisition phase involves going aboard ship and carrying out the planned survey. This means collecting data. For this thesis, surveys were conducted aboard the Ocean Engineering Department's Research Vessel *CT-1*. The EG&G 272-TD towfish is used for sidescan sonar mapping. The sidescan sonar data from the 272-TD is displayed as a paper record on the EG&G SMS960 Seafloor Mapping System. The SMS960 outputs digital data that is acquired and displayed on the RTGIS Toolbox workstation. The Datamarine Link 6000 is used to acquire single beam bathymetry data, ship speed through the water, and magnetic heading. The RTGIS Toolbox workstation also collects

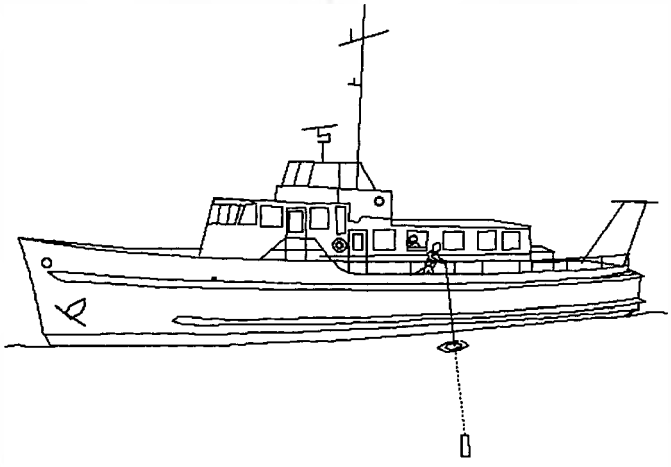
the ASCII output from the *CT-1* gyro compass, the *CT-1* Furuno GPS, and the Magnavox 4200D differential GPS. CTD casts are made with the Hydrolab H-20 water quality instrument in order to compute a sound velocity profile for correcting the bathymetry measurements. The *CT-1* has been previously outfitted by OMDC personnel with the proper cabling and mounting equipment to accommodate all aforementioned instruments. The configuration of these instruments aboard the *CT-1* during survey operations is shown in Figure 3.5 on the following page.

Figure 3.5(A) shows underway data acquisition. Navigation antennas are mounted to the CT-1 superstructure. The sidescan towfish is towed either directly behind the ship using the ship's A-frame or off the side using a davit. The single beam echo sounder is mounted to a pole on the ship's starboard side. Figure 3.5(B) shows the water sampling data acquisition. The purpose of water sampling is to obtain conductivity and temperature data as a function of depth for creating sound velocity profiles. Typically, the ship is stopped at several locations so that sound velocity profiles may be computed at several locations to better characterize the survey region.

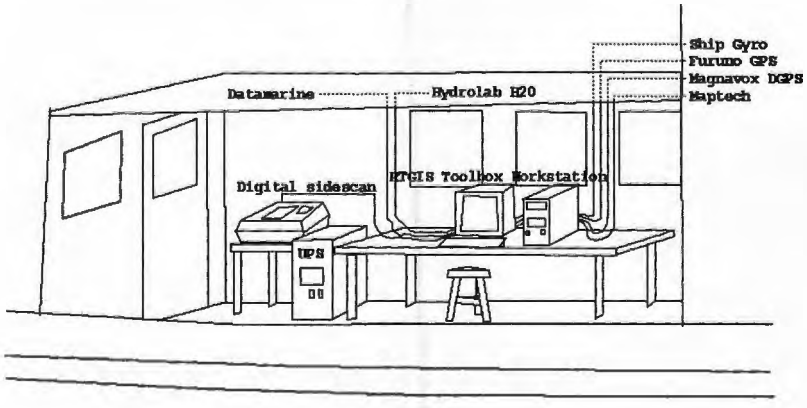
Figure 3.5(C) shows the lab setup during survey operations. All instruments are wired directly to the RTGIS Toolbox workstation and are connected to serial ports on the Cyclades CYCLOM-8YS serial port board. All instruments should be tested and logged at the dock, before the ship departs for the survey. Serial data logging is controlled by the Serial Data Logging Tool found in the Survey Tools menu button on the RTGIS Main Menu. Independent logging programs are provided for each instrument in order to



(A)



(B)



(C)

Figure 3.5. Shipboard data acquisition configuration.

isolate instrument logging failures. Serial data logging programs should be started when the equipment is turned on and not stopped until the vessel returns to port or when the instruments are turned off.

3.3 Survey Monitoring

The survey monitoring phase runs concurrent to the Data Acquisition phase. The objective of this phase is to monitor the progress of the data logging processes and correct any problems which occur. The RTGIS Toolbox contains several tools for this purpose. The simplest tool is the program ser.monitor. Ser.monitor is used to check the data file of a user-specified instrument at a user specified time interval. If the data file is updating (i.e., growing in size), ser.monitor assumes that the data logging is proceeding successfully and prints the latest record in the file to the terminal window so that the user may verify the validity of the logged data (for the case of readable ASCII data). If the data file is not updating, an audible warning is sounded and an error message is printed to the terminal window. The operator may then proceed to determine and correct the cause of the problem. Ser.monitor is started for each supported instrument via the Log Monitor Tool. A separate terminal window is started to run each instance of ser.monitor to avoid confusion of the ser.monitor text output. The ser.monitor terminals are typically oriented along the side or bottom of the RTGIS Toolbox workstation screen so that all log monitors may be easily checked visually by the operator in a single glance.

In addition to the simple log file check provided by ser.monitor, the RTGIS Toolbox provides a number of display programs for graphically tracking the survey progress and

data quality. Most obvious are the real time shiptrack display, the sidescan swath coverage display, and the waterfall sidescan display. The Shiptrack Tool will ask the user to which GRASS display monitors to display and the desired update interval. The shiptrack color will change based on whether the ship is currently on a survey line or in transit. The sidescan sonar swath coverage is displayed as a line perpendicular to the ship heading. Both the online/offline flag and the swath width are set by selection buttons in the Survey Tools menu. The Update Display on Current Position button in the Survey Tools menu will re-center the display window on the ship position whenever the ship leaves the window region. This is useful for keeping track of the ship. Viewing the raw sonar coverage is valuable for determining if there are any gaps in the sidescan sonar coverage. If there are, the operator can decide if it is necessary to re-run a portion of a line.

In addition to sidescan sonar coverage, the sonar data is displayed in waterfall fashion in a modified ShowImage window. By selecting the Realtime button on the ShowImage File menu, ShowImage will go into real time mode and wait for pings to be sent from the sidescan sonar logging program. This very conveniently allows the survey operator to view the sidescan sonar data without having to divide his time between the SMS960 display and the RTGIS Toolbox workstation display.

3.4 Post-Cruise Data Reporting

The objective of the post-cruise data reporting phase is to generate a report of the data collected, including text lists and map views. The report is generated aboard ship

immediately after data acquisition has been completed and the research vessel is en route to its home port. The reports are useful for presenting to the client an immediate summary of the survey's degree of success and the amount and location of data collected. The reports are also useful for the data processor. As data processing sometimes follows the data collection phase by more than a few days, the post-cruise reports can assist in reminding the operator of the data collected and its location in the GRASS database. The lists of data in the reports can also be used as processing checklists. The RTGIS Toolbox provides two tools for post-cruise data reporting. The first is the Data Reporting Tool, the other is the Map Making Tool.

The Data Reporting Tool compiles lists of survey data files collected on the current cruise. For each new survey data file it finds, the Data Reporting Tool will read through and generate some file statistics. Each data file and its stats are printed in the Data Reporting Tool's text window. When the reporting is complete, the text is saved and printed to a printer. The information files generated by the Data Reporting Tool can be used during the post processing phase for relational database management queries.

The Map Making Tool is started by clicking on the Mapping Tools button on the RTGIS Main Menu. The Map Making Tool is a general purpose map making utility which interfaces to the Generic Mapping Tools (GMT), GRASS mapping programs, and several OMDC programs. There are options for mapping raw data coverage. When the map is satisfactory, it is printed to either a 8.5x11 color printer or a color plotter in sizes up to Arch E (36" x 48").

3.5 Post Processing Data

The objective of the post processing phase is to apply all required corrections to the raw data and create final data layers to be added to the permanent GRASS database. Both the nature of processing and the type of GRASS database file generated varies with each type of raw data. There are 7 basic steps to processing the survey data; 1) sidescan sonar processing, 2) navigation filtering, 3) single beam bathymetry processing, 4) multibeam sonar processing, 5) map making, 6) RDBMS querying, and 7) GIS analysis. The order presented is not necessarily the order in which the processing must proceed, nor must each step be included (i.e., if no multibeam data was collected, there is no need for multibeam data processing).

The sidescan sonar processing is controlled by options in the SLS Tools menu of the RTGIS Main Menu. The first step in sidescan sonar processing is to start Xsonar and ShowImage via the SLS Tools menu. ShowImage is used for visual inspection of the swath data at any intermediate Xsonar processing step and for correcting the bottom tracking in the raw sonar file. Xsonar produces fully corrected sidescan sonar swaths as geographic grids. The grids are then imported as GRASS raster data layers via a custom created filter program. GRASS map calculation programs are then used to mosaic the individual swaths. The result is a fully corrected sidescan sonar mosaic.

The purpose of the navigation filtering is to remove any spurious positions and smooth over any possible dropouts in the differential signal. The navigation is later merged with

the time stamped single beam bathymetry data for geographic registration during the single beam bathymetry processing. The navigation is filtered via the navigation filtering program m.navfilter and a GRASS display monitor for viewing the raw and filtered navigation.

There are a number of steps required for the single beam bathymetry processing. The processing is controlled by the Bathymetry Processing Tool which provides an interface to a number of command line programs written for single beam bathymetry processing. Single beam bathymetry processing includes application of transducer depth offset, tidal corrections, and sound velocity corrections. The tidal corrections are performed using NOAA format tide tables freely available via anonymous FTP. The sound velocity corrections are performed by creating an SVP curve using collected temperature and salinity data. The corrected data is then merged with the filtered navigation to produce a GRASS site file of UTM-registered depth values. The GRASS site data is then converted to a GRASS raster data layer of bathymetry.

Multibeam sonar processing is accomplished by tools in the MB Tools menu of the RTGIS Main Menu. The first step is to create a multibeam project file using the MB Project Tool started by the MB Project button in the MB Tools menu. The project file identifies the multibeam sonar files to include in the processing and also records all processing steps performed on each multibeam sonar file. After the project file is created, the MB Processing Tool is started by the MB Process button in the MB Tools menu. Before the MB Processing Tool actually starts, an Operator Information Tool requires the

user to enter several pieces of personal information to assist in keeping track of the project's processing history. The MB Processing Tool controls processing of the multibeam files in the specified multibeam project. The multibeam files are processed using MB-System. The MB Processing Tool walks the user through the MB-System processing. When each swath is completely processed, the Import MB Tool is used to convert the gridded output from MB-System to a GRASS raster data file.

The Map Making Tool is used for creating final production maps of the survey data. Site, vector, and raster data layers may be added to a map. Text labels and borders may be added. The final map is output in Postscript format which is then rasterized and printed.

The Data Reporting Tool, discussed in section 3.4 for preparing post-cruise reports, may also used for making queries on GRASS database files in general. It can be used to determine which data files are within a certain geographic region.

Finally, the post processing can continue with additional analysis of the survey data using GRASS GIS analysis programs. For example, slope and aspect analysis of bathymetry raster data layers, map calculations to determine variances in multiple bathymetry rasters, volume analyses of bathymetry data layers, and image processing of sidescan mosaics.

3.6 Data Archiving

The objective of the data archiving phase is threefold: 1) to clean the GRASS database of any intermediate files resulting from the data processing phase, 2) to be sure that

processed data is properly located in the permanent GRASS database, and 3) to move selected files from the hard drive to permanent storage on CDROM disks. The RTGIS Toolbox provides two tools for the data archiving phase. The first tool, the Database Management Tool, addresses the first two objectives. The other tool, Xcdroast, addresses the third objective. The Database Management Tool is used to browse the GRASS database and delete all unnecessary data files produced as intermediate files during data processing. For example, the slant range corrected swath files (*.s files) produced by Xsonar and the transducer depth offset files (*.x files) produced during the single beam bathymetry processing. The Database Management Tool is also used to rename and copy fully processed survey data to the PERMANENT MAPSET if they are to become part of the permanent GRASS database. For example, copying the fully processed Coddington Cove mosaic GRASS raster file named "mosaic" in the Coddington_Cove MAPSET to the GRASS raster file named "Coddington_Cove_mosaic" in the PERMANENT MAPSET and then deleting the original "mosaic" GRASS raster.

The RTGIS Toolbox provides CD-ROM writing using the Xcdroast CD-ROM writing utility. Data files can be written to CD-ROM for several purposes. The CD-ROM can be used as a backup of intermediate processed data files. In this case, the data files should be written to CD-ROM with Xcdroast before they are deleted from the GRASS database with the Database Management Tool. For example, the operator may wish to write all raw survey data to CD-ROM immediately after the survey for backup purposes. The CD-ROM can also be used as off-line storage of GRASS database files which are useful but take up a large amount of disk space. For example, the fully processed mosaic from a

sidescan survey may be kept on the fixed disk, but the fully processed, individual swath files may be written to CD-ROM. The result is a compromise in which the individual swaths are readily accessible through the GRASS database structure but they do not take up unwieldy amounts of fixed disk space. The CD-ROM can also be used to create presentations of the survey data for clients. After all processing is complete, the raw data files and fully processed data files are written to CD-ROM and presented to the client.

station; coordinating data acquisition, display, and processing and survey planning, monitoring, and reporting. This is the next generation of oceanographic surveying.

Perhaps the most significant part of this work is the computer that the RTGIS Toolbox is running on. The Linux PC workstation is comparable in all respects to high-end desktop workstations from manufacturers such as SGI, Sun, and DEC. All software that has been ported to the Linux operating system is typically run by other commercial and academic groups on these commercial desktop workstations. The result in porting the software to Linux is a highly competitive product $10,000-$15,000 less expensive than competitive systems without a real performance compromise.

5. Recommendations For Future Work

The work presented by this thesis addressed solutions to a large number of problems. As a result, many areas have been identified which would benefit from additional work. There are a number of modifications and additions which can be made to add to the existing RTGIS Toolbox functionality. The GRASS GIS provides a large and diverse set of GIS analysis tools which can be applied singly or in combination to the oceanographic data for the purpose of geographic data analysis. However, to use the GRASS GIS to its full potential requires a knowledge of all that GRASS has to offer and the ability to combine numerous GRASS and UNIX command line programs. To assist in simplifying and automating geographic data analysis, additional GRASS4.1 image processing and data analysis routines can be integrated into the RTGIS graphical toolbox.

The RTGIS Toolbox would benefit from a more efficiently integrated helm guidance system, especially the ability to modify survey lines and update the helm display more quickly. Integration of an Electronic Chart Display and Information System (ECDIS) can be investigated. The RTGIS Toolbox workstation can be made to acquire and display radar on the GIS display. A very worthwhile effort would involve the investigation of having two RTGIS displays; one acting as the complete control RTGIS Toolbox and the other exported to the bridge with a simplified display showing just the survey lines, a line steering tool, and possibly an ECDIS display.

Future RTGIS Toolbox development should include the addition of other data sources and devices. While real time serial data acquisition was demonstrated for only a handful of sources, additional serial data sources can be easily added. Review of the ser.* program source code will provide a good starting point for such additions. Furthermore, data input through devices other than the serial port can be added. For instance, data logging via the parallel port is possible and direct device driving via serial, parallel, and network ports is also possible. Data I/O via these devices requires Linux kernel programming. For more information, review the RTGIS test programs in the $GIS/src.OMDC/parallel directory and visit the following websites: for parallel data I/O, http://www.lvr.com/parport.htm and http://www.torque.net/linux-pp.html and for Linux kernel programming, http://victoria.uci.agh.edu.pl/doc/khg/khg.html.

In addition, there are a number of avenues which can provide new topics of research, worthy of theses themselves, using the RTGIS Toolbox as a base. Integration of Digital Signal Processing technology into the Linux workstation could allow for investigation of acquisition of analog sidescan sonar data and/or multibeam sonar data. This would enable the RTGIS Toolbox to be compatible with any sidescan or multibeam sonar system which provides a raw analog output, thus greatly increasing the portability and commercial value of the RTGIS Toolbox. Digital signal processing techniques can be applied to the sonar data to create a sonar display and processing system complete with full real time image filtering and analysis.

New environmental applications could be explored. The potential of the technology demonstrated here is boundless. The GRASS4.1 suite of programs itself is expansive. By adding a suite of additional software applications, as this work did, the net functionality of the resulting system is exhaustive. However, this thesis only addressed the application of oceanographic survey monitoring and data acquisition and processing for a limited subset of data inputs with a definite focus on seafloor mapping. New data interfaces and new processing tools need to be investigated for new applications, such as coastal and wetlands management and biological resource management.

The RIM Relational Database Management System (RDBMS) was integrated into the RTGIS Toolbox to provide a few pre-defined relational queries. In order to facilitate report generation, a simple Tcl/Tk interface was created to read the RIM database and allow the user to select tables, columns, and rows for printing. While the objective of demonstrating the value of adding of a relational query interface was satisfied, only a limited subset of relational queries are provided and the RIM interface is limited in its ability to edit the relational database. The Tcl/Tk interface prototyped here can be extended to a full-blown RDBMS interface with screens for adding and changing tables, columns, and data. To this can be added a suite of interactive relational queries specific to oceanographic data. In addition, this would require great refinements of the RIM database along with more stringent metadata requirements in order for the relational database to be populated with more useful data.

The only limitation imposed by the Linux system is related to the graphics performance. Many of the software systems in the RTGIS Toolbox (e.g., GRASS, Xsonar, and MB-System) are very graphics intensive and were originally written for workstations with high-end graphics hardware capable of supporting the demanding drawing operations of these software. With PC CPUs now approaching clock speeds of 300-600MHz, it can be argued that the only remaining shortcoming of PC platforms compared to traditional workstation platforms is the graphics hardware. Single-board EISA and PCI video adapter boards simply do not pack the power of the integrated video hardware of traditional workstations. As a result, certain demanding graphics operations run much slower on Linux. However, industry has claimed that before 1997 is through, much more powerful PCI video cards will be available which provide large on-board video memory, double buffered displays, and hardware support for advanced graphics operations such as texturing, lighting, and depth buffering. Any future RTGIS Toolbox development on the Linux platform should investigate such a next generation video board.

References

Aronoff, S. Geographic Information Systems: A management Perspective. WDL Publications. Ottawa, Canada. (1989): 31-45.

Byrne, J. S. The Acquisition, Processing, and Display of Digital Side Scan Sonar Data. M.S. Thesis. The University of Rhode Island, Kingston, (1990). 87 p.

Caress, D. W. and D. N. Chayes. Improved Processing of Hydrosweep DS Multibeam Data on the R/V *Maurice Ewing*. Marine Geophysical Researches, 18:631-650, (1996).

Clapp, D. L. The Design of a Sidescan Sonar Data Acquisition System. M.S. Thesis. The University of Rhode Island, Kingston, (1994). 114 p.

Clarke, J.E.H. Swath Bathymetry Multibeam Training Course Overview. Coastal Multibeam Hydrographic Surveys Course Book. US/Canada Hydrographic Commission Coastal Multibeam Working Group. 1994. 19 p.

Cyclades Corporation. Cyclom-Y User's Guide, Version 6.0. Fremont, CA. March, 1995. 33 p.

Danforth, W. W. Xsonar/ShowImage: A Complete Processing and Display System for Rapid Sidescan Sonar Processing. U.S. Geological Survey Open-File Report (in progress). 1996. 59 p.

Doytsher, Y. and J. K. Hall. Gridded affine transformation and rubber-sheeting algorithm with FORTRAN program for calibrating scanned hydrographic survey maps. Computers and Geosciences. 1997. (in press).

Evenden, G. I. Cartographic Projection Procedures for the UNIX Environment - A User's Manual. U.S. Geological Survey Open-File Report 90-284. 1990. 64 p.

Flemming, B. Side-Scan Sonar: A Practical Guide. International Hydrographic Review, LIII:65-92. (1976).

Fox, Jim. RIM User's Manual. University Computing Services, University of Washington. 1990. 98 p.

Gann, J. T. YoNav: Your Own Integrated Navigation System for DOS Platforms. U.S. Geological Survey Open-File Report 92-565. 1992. 62 p.

Guptill, S. C. "Evaluating Geographic Information Systems Technology." Photogrammetric Engineering and Remote Sensing. 55.11 (1989): 1583-1587.

Hall, J. K. Digital topography and bathymetry of the area of the Dead Sea Depression. Tectonophysics 266 (1996): 177-185.

Hatcher Jr., G. A. A Geographic Information System as a Data Management Tool for Seafloor Mapping. M. S. Thesis. The University of Rhode Island, Kingston, (1993). 160 p.

Hydrolab Corp. H20 Multiparameter Water Quality Data Transmitter Operating Manual. Austin, TX. 1991. 63 p.

Leenhardt, O. Side Scanning Sonar - A Theoretical Study. International Hydrographic Review. LI:62-80. (1974).

Maguire, D. "An Overview and Definition of GIS." In: Maguire, D., M. Goodchild, and D. Rhind (eds.) Geographical Information Systems: Principles and applications. John Wiley and Sons, New York, NY. 1 (1991): 9-20.

Maher, R. V. "Applications of GIS to the Ocean Environment.. The Ocean - An International Workplace." Proceedings Oceans '87. 3 (1987): 1065-1067.

Merz, T. Ghostscript Manual. http://ftp.cs.wisc.edu/ghost/gs5man_e.pdf. 1997.

NCSA Data Transfer Mechanism Programming Manual Version 2.3. National Center for Supercomputing Applications. University of Illinois at Urbana-Champaign. 1992. 61 p.

Ousterhout, John K. Tcl and the Tk Toolkit. Addison-Wesley Publishing Company. Reading, MA. 1994. 439 p.

Shapiro, Michael, James Westervelt, Dave Gerdes, Marjorie Larson, and Kenneth R. Brownfield. GRASS4.1 Programmer's Manual. U.S. Army Construction Engineering Research Laboratory. 1993. 338 p.

Stevens, W. Richard. UNIX Network Programming. Published by Prentice-Hall, Inc. 1990. 748 p.

Terstriep, J. A. and D. E. Weber. The NCSA Data Transfer Mechanism Programming Manual. http://xtc.ncsa.uiuc.edu/DTM/Documentation/DTM2.4/dtm.tp.html. 1993.

Tyce, R. "Deep Seafloor Mapping Systems - A Review." Marine Technology Society Journal. 2.4 (1986): 4-16.

Tyce, R., Hatcher, G. A. "Real Time Geographic Information Systems For Survey Data Management." Proceedings, 22nd Joint Meeting of UJNR Sea-Bottom Surveys Panel. 22. (1993).

Urick, R. Principles of Underwater Sound, 2nd Edition. McGraw-Hill, New York. (1983). 417 p.

Wessel, P. and W. H. F. Smith. New Version of the Generic Mapping Tools Released. EOS Trans. *AGU* 76, 329. 1995.

Westervelt, J. Introduction to GRASS4.0. Central Washington University, Extended University Programs, Ellensburg, Washington 98926. (1991).

Yggdrasil Computing Incorporated. The Linux Bible, The GNU Testament - 2nd Edition. San Jose, CA. 1994. 173 p.

Appendix A: RTGIS Toolbox Software Component Table

The table below lists all software packages assembled to create the RTGIS Toolbox. A brief summary of each software is provided, along with Internet locations for obtaining information about the software and the software itself.

Package	Description	Source	Information	Restrictions
Linux	UNIX for Intel CPU architecture	ftp://ftp.cdrom.com/pub/linux/slackware	http://www.linux.org	Free public licensing with terms included
Xfree86	X windows for UNIX	ftp://ftp.xfree86.org	http://www.xfree86.org	Free public licensing with terms included
Tcl/Tk	X Windows GUI development system	ftp://ftp.smli.com/pub/tcl/	http://www.smli.com/research/tcl	Free public licensing with terms included
Ghostscript	Postscript interpreter	ftp://ftp.cs.wisc.edu/ghost/aladdin/	http://www.cs.wisc.edu/~ghost/index.html	Free public licensing with terms included
Ghostview	Postscript viewer	ftp://ftp.cs.wisc.edu/pub/X/	http://www.cs.wisc.edu/~ghost/index.html	Free public licensing with terms included
GRASS4.1	Geographic Information System	ftp://moon.cecer.army.mil/pub/grass/grass4.1/release/	http://www.cecer.army.mil/grass/GRASS.main.html	Public domain No license required
XSonar	Side scan sonar processing system	ftp://boomer.er.usgs.gov/pub/sonar/	ftp://boomer.er.usgs.gov/pub/sonar/xsonar.ps.Z	Public domain No license required
MB-System	Multibeam sonar processing system	ftp://lamont.ldeo.columbia.edu/pub/swath_data/	http://www.ldeo.columbia.edu/MB-System/MB-System.intro.html	Public domain No license required
GMT	Generic Mapping Tools	ftp://kiawe.soest.hawaii.edu/pub/gmt/	http://www.soest.hawaii.edu/wessel/gmt.html	Free public licensing with terms included
netCDF	Common data exchange utility	ftp://ftp.unidata.ucar.edu /pub/netcdf/	http://www.unidata.ucar.edu /packages/netcdf/	Free public licensing with terms included
DTM	Data Transfer Mechanism for Interprocess Communication	ftp://ftp.ncsa.uiuc.edu /DTM/	http://xtc.ncsa.uiuc.edu/DTM/Documentation/DTM2.4/dtm.tp.html	Public domain No license required
Motif 2.0	X Windows GUI programming system	Metrolink, Inc.	http://www.metrolink.com holly@metrolink.com	Proprietary $199 license
OpenGL	Graphics programming system	Metrolink, Inc.	http://www.metrolink.com holly@metrolink.com	Proprietary $199 license
Maptech Professional	Helm Guidance / Electronic Chart	Resolution Mapping	1-617-860-0430	Proprietary $500-1000 license
SeaView / SeaSurvey	Multibean collection and processing software	SeaBeam Instruments Corp.	mkt@seabeam.com 1-800-732-2326	Proprietary commercial licensing
RTGIS	Real Time Extensions to GRASS	URI Ocean Mapping Development Center	tyce@oce.uri.edu	Distribution limited to supporting members who can license

Appendix B: IPC of real time programs; ports.h and known DTM messages.

IPC of real time programs is accomplished via the Data Transfer Mechanism (DTM) from the National Center for Supercomputing Applications. DTM provides a simplified interface to Berkeley sockets for network communications. The ports.h header file, shown below, contains all the DTM port addresses and message IDs for all real time programs. Following the ports.h file are tables summarizing the DTM messages for each real time program.

```
/* DTM ports for Real Time IPC */
/* There can be 32 ports (in and out) per process */

#define MAX_PORTS     32

/* The Real Time .gisrc Setter */
#define RTENV_BASEPORT      0x1000
#define RTENV_INPORT        RTENV_BASEPORT + 0x01
#define RTENV_OUTPORT       RTENV_BASEPORT + 0x02

#define RTENV_GETENV        0x01
#define RTENV_SETENV        0x02

/* The MX4200D logger, ID is MX4200D */
#define MX4200D_BASEPORT            0x1020
#define MX4200D_REQUESTPORT         MX4200D_BASEPORT + 0x01

#define MX4200D_BASEREQUEST         0x1020
#define MX4200D_SENDNAV             MX4200D_BASEREQUEST + 0x01
#define MX4200D_SENDCOG             MX4200D_BASEREQUEST + 0x02
#define MX4200D_SENDSOG             MX4200D_BASEREQUEST + 0x03
#define MX4200D_SETSWATH            MX4200D_BASEREQUEST + 0x04
#define MX4200D_SETSLSRNG           MX4200D_BASEREQUEST + 0x05
#define MX4200D_SETDEPTH            MX4200D_BASEREQUEST + 0x06
#define MX4200D_SETONLINE           MX4200D_BASEREQUEST + 0x07
#define MX4200D_STOPLOG             MX4200D_BASEREQUEST + 0x08
#define MX4200D_SENDFILE            MX4200D_BASEREQUEST + 0x09

/* ShowImage, ID is SHOWIMAGE */
#define SHOWIMAGE_BASEPORT          0x1040
#define SHOWIMAGE_RTINPORT          SHOWIMAGE_BASEPORT + 0x01

/* d.nav, ID is DNAV */
```

```
#define DNAV_BASEPORT                0x1060
#define DNAV_INPORT                  DNAV_BASEPORT + 0x01

#define DNAV_BASEREQUEST             0x1060
#define DNAV_STOP                    DNAV_BASEREQUEST + 0x01

/* ser.monitor, ID is MON_?? (e.g., MON_MX4200D) */
#define MON_BASEPORT                 0x1080
#define MON_MX4200D_INPORT           MON_BASEPORT + 0x01
#define MON_CT1GYRO_INPORT           MON_BASEPORT + 0x02
#define MON_DATAMARINE_INPORT        MON_BASEPORT + 0x03
#define MON_HYDROLAB_INPORT          MON_BASEPORT + 0x04

#define MON_BASEREQUEST              0x1080
#define MON_STOP                     MON_BASEREQUEST + 0x01
```

Appendix C: Postscript output system; rasterize.sh and Ghostscript supported devices.

The two main components of the postscript output system are the shell script rasterize.sh and the program Ghostscript. Rasterize.sh accepts input to set the postscript output page size, gamma correction, and output device and then invokes Ghostscript to perform rasterization of the postscript file. Shown below is rasterize.sh and the devices supported by Ghostscript.

```
#!/bin/sh
# ------------------------------------------------------------
# file: rasterize.sh
# usage: rasterize.sh <infile> <outfile> <pagewidth> <pageheight>
#          <gamma> <device>
# ------------------------------------------------------------
# ------------------------------------------------------------
# Originally from:
# Battelle Memorial Institute
# Pacific Northwest Laboratory
#
# With many modifications from:
# Ocean Mapping Development Center
# University of Rhode Island
# ------------------------------------------------------------
# ------------------------------------------------------------
# Created June 13, 1996 by William A Perkins
# Last Change: Thu Jun 13 14:56:38: 1996 by William A Perkins <perkAyama.pnl.gov>
#          : Sat Mar 1 1997 by Steve Dzurenko <steve@ocean.oce.uri.edu>
# ------------------------------------------------------------

# set the files
file=$1
outfile=$2
device=$6

# let user know we got files ok
echo
echo Input file is $file
echo Output file is $outfile
echo Device is $device

# determine line number of %%EndComments line
line=`awk '{if (match($1,"%%EndComments")) exit} END {print NR}' $1`
nextline=`expr $line + 1`
echo %%EndComments at line $line
```

```
# set the page sizes
xsize=$3
ysize=$4
echo
echo X size is $xsize
echo Y size is $ysize
xdim=`echo $xsize\*72 | bc`
ydim=`echo $ysize\*72 | bc`

# set the gamma correction
gamma=$5

echo
echo Gamma correction is $gamma

echo
echo Rasterizing with Ghostscript ...
( \
        head -$line $file; \
        echo '{ '"$gamma" exp } dup dup currenttransfer setcolortransfer'; \
        echo '<</PageSize ["$xdim $ydim"]>> setpagedevice'; \
        tail +$nextline $file
) | \
        gs -q -dNOPAUSE -sDEVICE=$device -sOutputFile=$outfile -
```

rasterize.sh

```
#   Copyright (C) 1989, 1996 Aladdin Enterprises.  All rights reserved.
#
# This file is part of Aladdin Ghostscript.

# ------------------------------- Catalog ------------------------------- #

# It is possible to build configurations with an arbitrary collection of
# device drivers, although some drivers are supported only on a subset
# of the target platforms.  The currently available drivers are:

# MS-DOS displays (note: not usable with Desqview/X):
#  MS-DOS EGA and VGA:
#       ega     EGA (640x350, 16-color)
#       vga     VGA (640x480, 16-color)
#  MS-DOS SuperVGA:
# +     atiw    ATI Wonder SuperVGA, 256-color modes
# +     s3vga   SuperVGA with S3 86C911 chip (e.g., Diamond Stealth board)
#       svga16  Generic SuperVGA in 800x600, 16-color mode
#                 (replaces atiw16, tseng16, and tvga16 -- see below)
#       tseng   SuperVGA using Tseng Labs ET3000/4000 chips, 256-color modes
# +     tvga    Trident SuperVGA, 256-color modes
#  ****** NOTE: The vesa device does not work with the Watcom (32-bit MS-DOS)
#  ****** compiler or executable.
#       vesa    SuperVGA with VESA standard API driver
#  MS-DOS other:
#       bgi     Borland Graphics Interface (CGA)  [MS-DOS only]
# *     herc    Hercules Graphics display  [MS-DOS only]
```

```
# *       pe        Private Eye display
# Other displays:
#  MS Windows:
#       mswindll  Microsoft Windows 3.1 DLL  [MS Windows only]
#       mswinprn  Microsoft Windows 3.0, 3.1 DDB printer  [MS Windows only]
#       mswinpr2  Microsoft Windows 3.0, 3.1 DIB printer  [MS Windows only]
#  OS/2:
# *       os2pm   OS/2 Presentation Manager   [OS/2 only]
# *       os2dll    OS/2 DLL bitmap           [OS/2 only]
# *       os2prn   OS/2 printer               [OS/2 only]
#  Unix and VMS:
#  ****** NOTE: For direct frame buffer addressing under SCO Unix or Xenix,
#  ****** edit the definition of EGAVGA below.
# *       att3b1    AT&T 3b1/Unixpc monochrome display   [3b1 only]
# *       lvga256  Linux vgalib, 256-color VGA modes  [Linux only]
# *       sonyfb   Sony Microsystems monochrome display   [Sony only]
# *       sunview  SunView window system   [SunOS only]
# +       vgalib    Linux PC with VGALIB   [Linux only]
#       x11       X Windows version 11, release >=4   [Unix and VMS only]
#       x11alpha  X Windows masquerading as a device with alpha capability
#       x11cmyk  X Windows masquerading as a 1-bit-per-plane CMYK device
#       x11mono  X Windows masquerading as a black-and-white device
#  Platform-independent:
# *       sxlcrt    CRT sixels, e.g. for VT240-like terminals
# Printers:
# *       ap3250  Epson AP3250 printer
# *       appledmp  Apple Dot Matrix Printer (should also work with Imagewriter)
#       bj10e     Canon BubbleJet BJ10e
# *       bj200     Canon BubbleJet BJ200
# *       bjc600   Canon Color BubbleJet BJC-600, BJC-4000 and BJC-70
#         also good for Apple printers like the StyleWriter 2x00
# *       bjc800   Canon Color BubbleJet BJC-800
# *    ccr    CalComp Raster format
# *       cdeskjet  H-P DeskJet 500C with 1 bit/pixel color
# *       cdjcolor  H-P DeskJet 500C with 24 bit/pixel color and
#                 high-quality color (Floyd-Steinberg) dithering;
#                 also good for DeskJet 540C and Citizen Projet IIc (-r200x300)
# *       cdjmono  H-P DeskJet 500C printing black only;
#                 also good for DeskJet 510, 520, and 540C (black only)
# *       cdj500   H-P DeskJet 500C (same as cdjcolor)
# *       cdj550   H-P DeskJet 550C/560C
# *       cp50      Mitsubishi CP50 color printer
# *       declj250  alternate DEC LJ250 driver
# +       deskjet  H-P DeskJet and DeskJet Plus
#       djet500  H-P DeskJet 500
# *       djet500c  H-P DeskJet 500C alternate driver
#                 (does not work on 550C or 560C)
# *       dnj650c  H-P DesignJet 650C
#       epson     Epson-compatible dot matrix printers (9- or 24-pin)
# *       eps9mid  Epson-compatible 9-pin, interleaved lines
#                 (intermediate resolution)
# *       eps9high  Epson-compatible 9-pin, interleaved lines
#                 (triple resolution)
# *       epsonc   Epson LQ-2550 and Fujitsu 3400/2400/1200 color printers
```

```
# *       ibmpro  IBM 9-pin Proprinter
# *       imagen  Imagen ImPress printers
# *       iwhi    Apple Imagewriter in high-resolution mode
# *       iwlo    Apple Imagewriter in low-resolution mode
# *       iwlq    Apple Imagewriter LQ in 320 x 216 dpi mode
# *       jetp3852  IBM Jetprinter ink-jet color printer (Model #3852)
# +       laserjet  H-P LaserJet
# *       la50    DEC LA50 printer
# *       la70    DEC LA70 printer
# *       la70t   DEC LA70 printer with low-resolution text enhancement
# *       la75    DEC LA75 printer
# *       la75plus DEC LA75plus printer
# *       lbp8    Canon LBP-8II laser printer
# *       lips3   Canon LIPS III laser printer in English (CaPSL) mode
# *       ln03    DEC LN03 printer
# *       lj250   DEC LJ250 Companion color printer
# +       ljet2p  H-P LaserJet IId/IIp/III* with TIFF compression
# +       ljet3   H-P LaserJet III* with Delta Row compression
# +       ljet3d  H-P LaserJet IIID with duplex capability
# +       ljet4   H-P LaserJet 4 (defaults to 600 dpi)
# +       lj4dith  H-P LaserJet 4 with Floyd-Steinberg dithering
# +       ljetplus  H-P LaserJet Plus
# *       lp2563  H-P 2563B line printer
# *       m8510   C.Itoh M8510 printer
# *       necp6   NEC P6/P6+/P60 printers at 360 x 360 DPI resolution
# *       nwp533  Sony Microsystems NWP533 laser printer  [Sony only]
# *       oce9050  OCE 9050 printer
# *       oki182  Okidata MicroLine 182
# *       okiibm  Okidata MicroLine IBM-compatible printers
# *       paintjet  alternate H-P PaintJet color printer
# *       pj      H-P PaintJet XL driver
# *       pjetxl  alternate H-P PaintJet XL driver
# *       pjxl    H-P PaintJet XL color printer
# *       pjxl300  H-P PaintJet XL300 color printer;
#                 also good for PaintJet 1200C
# *       r4081   Ricoh 4081 laser printer
# *       sj48    StarJet 48 inkjet printer
# *       sparc   SPARCprinter
# *       st800   Epson Stylus 800 printer
# *    stcolor  Epson Stylus Color
# *       t4693d2  Tektronix 4693d color printer, 2 bits per R/G/B component
# *       t4693d4  Tektronix 4693d color printer, 4 bits per R/G/B component
# *       t4693d8  Tektronix 4693d color printer, 8 bits per R/G/B component
# *       tek4696  Tektronix 4695/4696 inkjet plotter
# *       xes     Xerox XES printers (2700, 3700, 4045, etc.)
# Fax systems:
# *       dfaxhigh  DigiBoard, Inc.'s DigiFAX software format (high resolution)
# *       dfaxlow  DigiFAX low (normal) resolution
# Fax file format:
#  ****** NOTE: all of these drivers adjust the page size to match
#  ****** one of the three CCITT standard sizes (U.S. letter with A4 width,
#  ****** A4, or B4).
#         faxg3    Group 3 fax, with EOLs but no header or EOD
#         faxg32d  Group 3 2-D fax, with EOLs but no header or EOD
```

```
#       faxg4    Group 4 fax, with EOLs but no header or EOD
#       tiffcrle  TIFF "CCITT RLE 1-dim" (= Group 3 fax with no EOLs)
#       tiffg3    TIFF Group 3 fax (with EOLs)
#       tiffg32d  TIFF Group 3 2-D fax
#       tiffg4    TIFF Group 4 fax
# Other file formats and devices:
#       bit       Plain bits, monochrome
#       bitrgb    Plain bits, RGB
#       bitcmyk   Plain bits, CMYK
#       bmpmono          Monochrome MS Windows .BMP file format
#       bmp16   4-bit (EGA/VGA) .BMP file format
#       bmp256 8-bit (256-color) .BMP file format
#       bmp16m24-bit .BMP file format
#       cgmmono  Monochrome (black-and-white) CGM -- LOW LEVEL OUTPUT ONLY
#       cgm8     8-bit (256-color) CGM -- DITTO
#       cgm24    24-bit color CGM -- DITTO
# *     cif      CIF file format for VLSI
#       miff24   ImageMagick MIFF format, 24-bit direct color, RLE compressed
# *     mgrmono  1-bit monochrome MGR devices
# *     mgrgray2  2-bit gray scale MGR devices
# *     mgrgray4  4-bit gray scale MGR devices
# *     mgrgray8  8-bit gray scale MGR devices
# *     mgr4     4-bit (VGA) color MGR devices
# *     mgr8     8-bit color MGR devices
#       pcxmono          PCX file format, monochrome (1-bit black and white)
#       pcxgray PCX file format, 8-bit gray scale
#       pcx16    PCX file format, 4-bit planar (EGA/VGA) color
#       pcx256   PCX file format, 8-bit chunky color
#       pcx24b   PCX file format, 24-bit color (3 8-bit planes)
#       pbm      Portable Bitmap (plain format)
#       pbmraw Portable Bitmap (raw format)
#       pgm      Portable Graymap (plain format)
#       pgmraw Portable Graymap (raw format)
#       pgnm     Portable Graymap (plain format), optimizing to PBM if possible
#       pgnmraw          Portable Graymap (raw format), optimizing to PBM if possible
#       pnm      Portable Pixmap (plain format) (RGB), optimizing to PGM or PBM
#                 if possible
#       pnmraw Portable Pixmap (raw format) (RGB), optimizing to PGM or PBM
#                 if possible
#       ppm      Portable Pixmap (plain format) (RGB)
#       ppmraw Portable Pixmap (raw format) (RGB)
#       pkm      Portable inKmap (plain format) (4-bit CMYK => RGB)
#       pkmraw Portable inKmap (raw format) (4-bit CMYK => RGB)
#       pngmono          Monochrome Portable Network Graphics (PNG)
#       pnggray 8-bit gray Portable Network Graphics (PNG)
#       png16    4-bit color Portable Network Graphics (PNG)
#       png256   8-bit color Portable Network Graphics (PNG)
#       png16m 24-bit color Portable Network Graphics (PNG)
#       psmono  PostScript (Level 1) monochrome image
#       sgirgb    SGI RGB pixmap format
#       tiff12nc  TIFF 12-bit RGB, no compression
#       tiff24nc  TIFF 24-bit RGB, no compression (NeXT standard format)
#       tifflzw   TIFF LZW (tag = 5) (monochrome)
#       tiffpack  TIFF PackBits (tag = 32773) (monochrome)
```

```
# User-contributed drivers marked with * require hardware or software
# that is not available to Aladdin Enterprises.  Please contact the
# original contributors, not Aladdin Enterprises, if you have questions.
# Contact information appears in the driver entry below.
#
# Drivers marked with a + are maintained by Aladdin Enterprises with
# the assistance of users, since Aladdin Enterprises doesn't have access to
# the hardware for these either.

# If you add drivers, it would be nice if you kept each list
# in alphabetical order.

# ---------------------------- End of catalog ---------------------------- #
```

Ghostscript supported devices

Appendix D: Spec sheets for ICs used in serial sidescan data acquisition

FINAL

Am7203A

High Density First-In First-Out (FIFO) 2048 x 9-Bit CMOS Memory

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- RAM based FIFO
- 2048 x 9 organization
- Cycle times of 25/35/45/65 ns for standard products
- Cycle times of 40/65 ns for APL products
- Asynchronous and simultaneous writes and reads
- Low power consumption
- Status flags—full, half-full, empty
- Retransmit capability
- Expandable in both width and depth
- Increased noise immunity for $\overline{XI}$—CMOS threshold
- Functional and pin compatible wi: dustry standard devices

GENERAL DESCRIPTION

The Am7203A is a RAM-based CMOS FIFO that is 2048 words deep with 9-bit wide words. It is expandable to any width and/or depth to create much larger FIFOs.

This FIFO can input and output data asynchronously and simultaneously at data rates from 0 to 40 MHz for Standard Products and 0 to 25 MHz for APL products. Status flags are provided to signify empty, full and half-full conditions. The capability also exists to retransmit data from the FIFO.

High-density FIFOs such as the Am7203A are useful in a wide range of applications. The ability to buffer large transfers of data and the rate adaption capabilities make the Am7203A useful in communication, image processing, mass storage, DSP, and printing systems.

BLOCK DIAGRAM

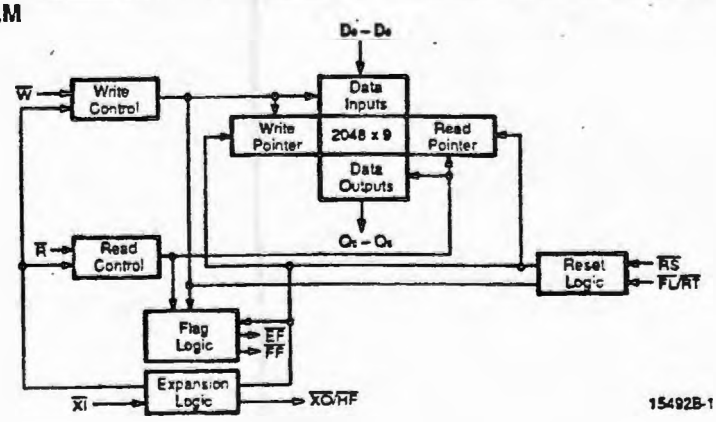
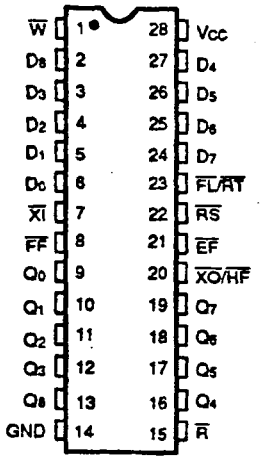


Figure 1.

AMD

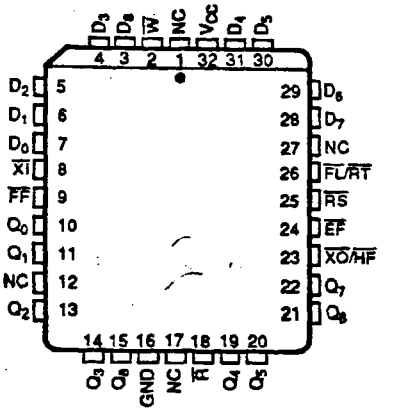
CONNECTION DIAGRAMS
Top View

DIPs*



15492B-2

PLCC*



15492B-3

Notes:

Pin 1 is marked for orientation for plastic packages.

*Pinout identical for both plastic and ceramic DIPs.

PIN DESCRIPTION

D_{0-8}

Data In (Inputs (9))

These nine pins are the data inputs to the FIFO.

$\overline{EF}$

Empty Flag (Output; Active LOW)

The HIGH state of the Empty Flag ($\overline{EF}$) indicates that the FIFO contains data to be read. The $\overline{EF}$ goes LOW when the read pointer is equal to the write pointer, indicating that the device is empty. $\overline{EF}$ LOW inhibits further Read operations.

The $\overline{EF}$ goes HIGH after the rising edge of Write ($\overline{W}$) during the first write cycle for an empty FIFO (See Figure 4). The $\overline{EF}$ goes LOW after the falling edge of Read ($\overline{R}$) during the read cycle which creates the empty condition.

During a Reset cycle, the $\overline{EF}$ is driven LOW (active).

$\overline{FF}$

Full Flag (Output; Active LOW)

The HIGH state of the Full Flag ($\overline{FF}$) indicates that the FIFO is capable of accepting data. The $\overline{FF}$ goes LOW when the write pointer is one location less than the read pointer, indicating that the device is full. $\overline{FF}$ LOW inhibits further Write operations.

The $\overline{FF}$ goes HIGH after the rising edge of Read ($\overline{R}$) during the first read cycle following a full condition (See Figure 6). The $\overline{FF}$ goes LOW after the falling edge of Write ($\overline{W}$) during the write cycle which creates the full condition.

During a Reset cycle, the $\overline{FF}$ is driven HIGH (inactive).

$\overline{FL}/\overline{RT}$

First Load/Retransmit (Input; Active LOW)

This is a dual purpose input, dependent upon whether the FIFO is in Single Device Mode or Depth-Expansion Mode.

This pin acts as a FIRST LOAD ($\overline{FL}$) pin when in the Depth-Expansion Mode. The device receiving data first will have the $\overline{FL}$ input tied LOW, while the remaining devices will have the $\overline{FL}$ pin tied HIGH. The states of the $\overline{FL}$ and Expansion In ($\overline{XI}$) pins are used to determine the FIFO's mode of operation, as shown in Tables 1 and 2.

This pin is used as the Retransmit ($\overline{RT}$) input during Single Device Mode. The device can be instructed to retransmit the previously written data when $\overline{RT}$ is pulsed LOW.

GND

Power Supply, Ground

This pin is the 0 V power supply for the FIFO.

NC

No Connect

These pins are not connected.

Q_{0-8}

Data Out (Outputs (9), Three State)

These nine pins are the data outputs for the FIFO. These pins are in a high impedance state whenever Read ($\overline{R}$) is HIGH.

$\overline{R}$

Read (Input; Active LOW)

The falling edge of Read ($\overline{R}$) initiates a read cycle, except when the device is empty, as indicated by the Empty Flag ($\overline{EF}$) being LOW. Valid data appears on the outputs (Q_{0-8}) after the falling edge of $\overline{R}$. After $\overline{R}$ goes HIGH, the Data Outputs (Q_{0-8}) will return to a high impedance condition.

$\overline{RS}$

Reset (Input; Active LOW)

The falling edge of Reset ($\overline{RS}$) is used to reset the FIFO. During Reset, both the read and write pointers are set to the first location in the FIFO. Since the reset cycle initializes the FIFO to an empty condition, the Empty Flag ($\overline{EF}$) is driven LOW (active), and both the Half-Full Flag ($\overline{HF}$) and Full Flag ($\overline{FF}$) are driven HIGH (inactive).

V_{CC}

Power Supply

This pin is the +5 V power supply for the FIFO.

$\overline{W}$

Write (Input; Active LOW)

The falling edge of Write ($\overline{W}$) initiates a write cycle, except when the device is full, as indicated by the Full Flag ($\overline{FF}$) being LOW. Data is latched into the FIFO on the rising edge of $\overline{W}$.

$\overline{XI}$

Expansion In (Input; Active LOW)

Expansion In ($\overline{XI}$) is grounded to indicate operation in the Single Device or Width-Expansion Modes. In Depth Expansion Mode, the $\overline{XI}$ pin is connected to the Expansion Out ($\overline{XO}$) pin of the previous device, except for the $\overline{XI}$ pin of the first device which is connected to the $\overline{XO}$ pin of the last FIFO.

This pin operates at CMOS logic levels, thus providing noise immunity between cascaded devices.

HARRIS SEMICONDUCTOR

CDP6402
CDP6402C

CMOS Universal Asynchronous Receiver/Transmitter (UART)

January 1992

Features

- Low Power CMOS Circuitry..... 7.5mW (Typ) at 3.2MHz (Max Freq.) at V_{DD} = 5V
- Baud Rate
  - DC to 200K Bits/s (Max) at..... 5V, 85°C
  - DC to 400K Bits/s (Max) at.....10V, 85°C
- 4V to 10.5 Operation
- Automatic Data Formatting and Status Generation
- Fully Programmable with Externally Selectable Word Length (5 - 8 Bits), Parity Inhibit, Even/Odd Parity, and 1, $1^1/_2$, or 2 Stop Bits
- Operating Temperature Range
  - CDP6402D, CD ..... -55°C to +125°C
  - CDP6402E, CE ..... -40°C to +85°C
- Replaces Industry Types IM6402 and HD6402

Description

The CDP6402 and CDP6402C are silicon gate CMOS Universal Asynchronous Receiver/Transmitter (UART) circuits for interfacing computers or microprocessors to asynchronous serial data channels. They are designed to provide the necessary formatting and control for interfacing between serial and parallel data channels. The receiver converts serial start, data, parity, and stop bits to parallel data verifying proper code transmission, parity and stop bits. The transmitter converts parallel data into serial form and automatically adds start parity and stop bits.

The data word can be 5, 6, 7 or 8 bits in length. Parity may be odd, even or inhibited. Stop bits can be 1, $1^1/_2$, or 2 (when transmitting 5 bit code).

The CDP6402 and CDP6402C can be used in a wide range of applications including modems, printers, peripherals, video terminals, remote data acquisition systems, and serial data links for distributed processing systems.

The CDP6402 and CDP6402C are functionally identical. They differ in that the CDP6402 has a recommended operating voltage range of 4V to 10.5V, and the CDP6402C has a recommended operating voltage range of 4V to 6.5V. Both types are supplied in 40 lead dual-in-line ceramic packages (D suffix), and 40 lead dual-in-line plastic packages (E suffix).

Ordering Information

PACKAGE	TEMPERATURE RANGE	5V/200K BAUD	10V/400K BAUD
Plastic DIP	-40°C to +85°C	CDP6402CE	CDP6402E
Burn-In		CDP6402CEX	- -
Ceramic DIP	-40°C to +85°C	CDP6402CD	CDP6402D
Burn-In		CDP6402CDX	CDP6402DX

Pinout

CDP6402, CDP6402C
TOP VIEW

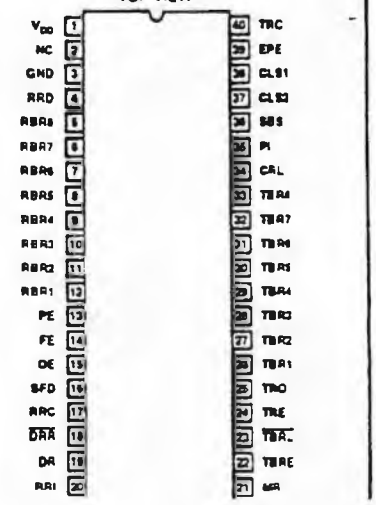


Table I - Control Word Function

CONTROL WORD					DATA BITS	PARITY BIT	STOP BIT(S)
CLS2	CLS1	PI	EPE	SBS			
L	L	L	L	L	5	ODD	1
L	L	L	L	H	5	ODD	1.5
L	L	L	H	L	5	EVEN	1
L	L	L	H	H	5	EVEN	1.5
L	L	H	X	L	5	DISABLED	1
L	L	H	X	H	5	DISABLED	1.5
L	H	L	L	L	6	ODD	1
L	H	L	L	H	6	ODD	2
L	H	L	H	L	6	EVEN	1
L	H	L	H	H	6	EVEN	2
L	H	H	X	L	6	DISABLED	1
L	H	H	X	H	6	DISABLED	2
H	L	L	L	L	7	ODD	1
H	L	L	L	H	7	ODD	2
H	L	L	H	L	7	EVEN	1
H	L	L	H	H	7	EVEN	2
H	L	H	X	L	7	DISABLED	1
H	L	H	X	H	7	DISABLED	2
H	H	L	L	L	8	ODD	1
H	H	L	L	H	8	ODD	2
H	H	L	H	L	8	EVEN	1
H	H	L	H	H	8	EVEN	2
H	H	H	X	L	8	DISABLED	1
H	H	H	X	H	8	DISABLED	2

X - Don't Care

Table II - Function Pin Definition

PIN	SYMBOL	DESCRIPTION
1	V_{DD}	Positive Power Supply
2	N/C	No Connection
3	GND	Ground (V_{SS})
4	RRD	A high level on RECEIVER REGISTER DISABLE forces the receiver holding register ouputs RBR1-RBR8 to a high impedance state
5	RBR8	The contents of the RECEIVER BUFFER REGISTER appear on these three-state outputs. Word formats less than 8 characters are right justified to RBR1.
6	RBR7	See Pin 5 - RBR8
7	RBR6	
8	RBR5	
9	RBR4	
10	RBR3	
11	RBR2	
12	RBR1	
13	PE	A high level on PARITY ERROR indicates that the received parity does not match parity programmed by control bits. The output is active until parity matches on a succeeding character. When parity is inhibited, this output is low
14	FE	A high level on FRAMING ERROR indicates the first stop bit was invalid. FE will stay active until the next valid character's stop bit is received
15	OE	A high level on OVERRUN ERROR indicates the data received flag was not cleared before the last character was transferred to the receiver buffer register. The Error is reset at the next character's stop bit if DRR has been performed (i.e., DRR, active low).
16	SFD	A high level on STATUS FLAGS DISABLE forces the outputs PE, FE, OE, DR, TBRE to a high impedance state.
17	RRC	The RECEIVER REGISTER CLOCK is 16X the receiver data rate.
18	$\overline{DRR}$	A low level on DATA RECEIVED RESET clears the data received output (DR), to a low level
19	DR	A high level on DATA RECEIVED indicates a character has been received and transferred to the receiver buffer register.
20	RRI	Serial data on RECEIVER REGISTER INPUT is clocked into the receiver register.
21	MR	A high level on MASTER RESET (MR) clears PE, FE, OE and DR, and sets TRE, TBRE, and TRO. TRE is actually set on the first rising edge of TRC after MR goes high. MR should be strobed after power-up.
22	TBRE	A high level on TRANSMITTER BUFFER REGISTER EMPTY indicates the transmitter buffer register has transferred its data to the transmitter register and is ready for new data.

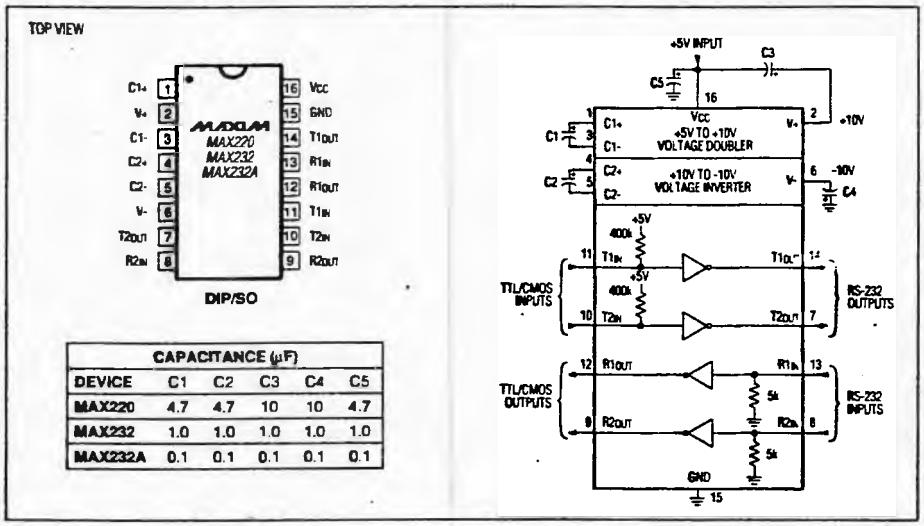
Table II - Function Pin Definition (Cont'd)

PIN	SYMBOL	DESCRIPTION
23	$\overline{TBRL}$	A low level on TRANSMITTER BUFFER REGISTER LOAD transfers data from inputs TBR1-TBR8 into the transmitter buffer register. A low to high transition on $\overline{TBRL}$ requests data transfer to the transmitter register. If the transmitter register is busy, transfer is automatically delayed so that the two characters are transmitted end to end.
24	TRE	A high level on TRANSMITTER REGISTER EMPTY indicates completed transmission of a character including stop bits.
25	TRO	Character data, start data and stop bits appear serially at the TRANSMITTER REGISTER OUTPUT.
26	TBR1	Character data is loaded into the TRANSMITTER BUFFER REGISTER via inputs TBR1-TBR8. For character formats less than 8-bits, the TBR8, 7, and 6 inputs are ignored corresponding to the programmed word length
27	TBR2	See Pin 26 - TBR1
28	TBR3	
29	TBR4	
30	TBR5	
31	TBR6	
32	TBR7	
33	TBR8	

PIN	SYMBOL	DESCRIPTION
34	CRL	A high level on CONTROL REGISTER LOAD loads the control register.
35	PI*	A high level on PARITY INHIBIT inhibits parity generation, parity checking and forces PE output low
36	SBS*	A high level on STOP BIT SELECT selects 1.5 stop bits for a 5 character format and 2 stop bits for other lengths
37	CLS2*	These inputs program the CHARACTER LENGTH SELECTED (CLS1 low CLS2 low 5-bits) (CLS1 high CLS2 low 6-bits) (CLS1 low CLS2 high 7-bits) (CLS1 high CLS2 high 8-bits)
38	CLS1*	See Pin 37 - CLS2
39	EPE*	When PI is low, a high level on EVEN PARITY ENABLE generates and checks even parity. A low level selects odd parity
40	TRC	The TRANSMITTER REGISTER CLOCK is 16X the transmit data rate

*See Table I (Control Word Function)

+5V-Powered, Multi-Channel RS-232 Drivers/Receivers



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 5. MAX220/232/232A Pin Configuration and Typical Operating Circuit

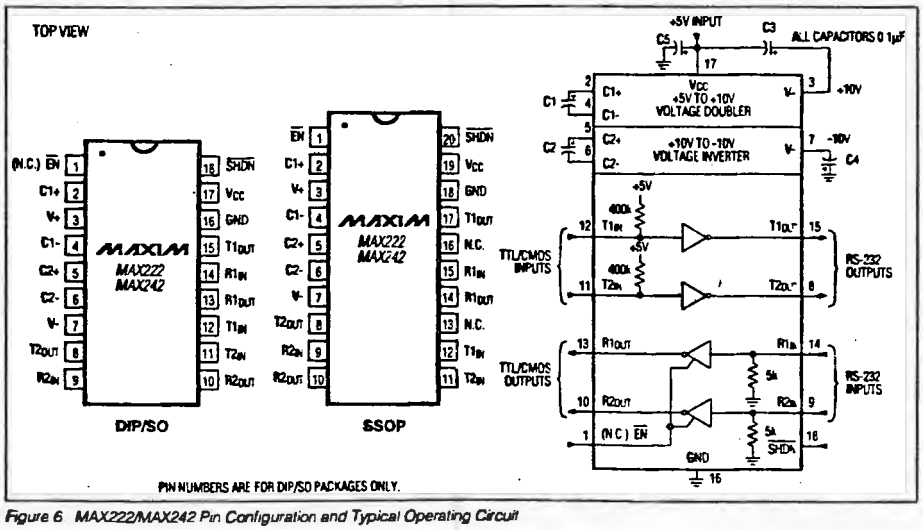


Figure 6. MAX222/MAX242 Pin Configuration and Typical Operating Circuit

Appendix E: RTGIS User's / Programmer's Manual

This appendix contains an attachment of the RTGIS User's / Programmer's Manual. The RTGIS User's / Programmer's Manual is a self-contained document. It is intended to accompany all RTGIS installations. It provides a brief overview of the RTGIS and complete instructions for using each RTGIS module. Along with the user instructions for each module are notes for programmers to assist in adding to and modifying the code. An index is provided with the User's / Programmer's Manual for convenient searching of all RTGIS modules and processes.

RTGIS Toolbox User's / Programmer's Manual

A guide to using the RTGIS Toolbox tools for survey operations including notes for application developers.

Ocean Mapping Development Center
University of Rhode Island
Narragansett, RI 02882
USA

Table of Contents

1. Introduction

The RTGIS Toolbox is a comprehensive software system for assisting in all phases of a seafloor mapping survey. This manual provides both user and programmer information regarding the RTGIS Toolbox.

This manual is divided into 9 sections. Section 1, "Introduction," this section, provides a short introduction to the manual. Section 2, "Installation," provides installation instructions for installing the RTGIS software from the RTGIS Distribution CD. Section 3, "System Tasks," describes some system tasks which are occasionally necessary for proper RTGIS Toolbox operation. Section 4, "RTGIS Toolbox Basics," discusses some basic operations which are performed repetitively during many other RTGIS Toolbox procedures. Section 5, "Pre-cruise Planning," describes RTGIS Toolbox tools provided for assisting in survey planning. Section 6, "Data Acquisition," describes RTGIS Toolbox tools provided for assisting with survey data acquisition. Section 7, "Survey Monitoring," describes RTGIS Toolbox tools provided for monitoring the progress of a survey, including vessel position and data logging. Section 8, "Post-cruise Data Reporting," describes RTGIS Toolbox tools provided to assist in reporting on the data collected during a survey. Section 9, "Post Processing Data," describes the tools provided for processing survey data.

Each of sections 3-9 includes user information and detailed step-by-step instructions for common tasks. Where appropriate, some sections include notes to the programmer regarding RTGIS Toolbox development. The purpose of the Programmer's Notes is to assist the application developer in adding to and modifying the RTGIS Toolbox.

2. Installation

2.1 A Few Words of Introduction

This section is meant to be a companion to the install script included in the top level directory of the RTGIS Distribution CD. The installation script is contained in the file named "install."

Please read this section carefully before running the install script.

To install the RTGIS, the installer must be familiar with UNIX, be comfortable typing UNIX commands in a shell window, and know how to use the UNIX man utility. This installation guide is not meant to be a UNIX tutorial nor a C programming and compiling guide. It will be assumed that the installer understands instructions concerning UNIX commands and tools.

A copy of this section, "Installation," along with the installation instructions for each of the separate software installation instructions has been collated for convenience and can be found in the /INSTALL subdirectory on the installation CD. The following installation files can be found in the /INSTALL directory:

Installation file	For software module	Original location
RTGIS.INSTALL	Real Time Geographic Information System Toolbox	/INSTALL/RTGIS.INSTALL
installGuide.ps	GRASS4.1	/grass4.1/documents/ installGuide.ps.Z
README.Xsonar	Xsonar	/xsonar/README
README.GMT	GMT	/gmt/README
README.tcl	Tcl	/tcl7.4/README
README.tk	Tk	/tk4.0/README

This section will not attempt to rewrite or repeat the existing installation instructions listed above for the various RTGIS components. Where appropriate, the installation notes contained in this section will refer the user to one of the other installation guides listed above.

2.2 RTGIS Development and Operational Theory

As may be obvious, the core of the Real Time Geographic Information System is a Geographic Information System. The GIS selected as the core of the RTGIS was the GRASS GIS. Hence, the development of the RTGIS has closely modeled that of GRASS. GRASS consists of a collection of software tools for manipulating and analyzing geographic data. All of the tools are self-contained modules which may be run from the command line, or a windowing user interface, independent of GRASS itself.

Likewise, the RTGIS consists of a collection of separate software tools which can be run independently of the RTGIS.

What binds the separate modules of GRASS and RTGIS together is the UNIX environment. By taking advantage of the existing UNIX environment and UNIX Operating System tools, a series of seemingly separate software applications are made to work together as a single definable system.

2.3 Platform Support

The initial RTGIS development was done on a Silicon Graphics Indigo2 Workstation. The prototype system was then ported to a PC Linux Workstation.

The install script and the sources located on the RTGIS Distribution CD support IRIX5.2 (and higher) and Linux2.0.0 (and higher). However, pre-compiled binaries are provided for Linux only.

All RTGIS software can be (and has been) compiled and run on most all UNIX operating systems, including DEC Ultrix, SUN OS, SUN Solaris, SGI IRIX, and Linux. If the installer wishes to install the RTGIS on an unsupported operating system, it is assumed that he or she has the knowledge and know-how to do so, as operating systems other that IRIX and Linux are unsupported and untested by the OMDC.

2.4 Before you begin

There are three simple items which will effect the operation of the RTGIS and must be addressed before continuing with the installation.

1. Decide who will "own" RTGIS files. GRASS is sometimes picky regarding by whom it is being executed. Choose a single user who will use the RTGIS system, this is usually a user called "grass". If necessary, ask your system administrator to create user grass for you.

   Note: Do NOT let root own grass.

2. Decide where you want the RTGIS system to "live." You must choose a top level directory which will contain all of the RTGIS software.

   If at all possible, it is suggested to install the RTGIS software under the /usr/local directory. This is the directory used by the OMDC and will result in the easiest installation. Throughout the remainder of these installation notes, /usr/local will be referred to as the top level directory for examples.

3. Some of the RTGIS components may be run from the distribution CDROM, but others may not. If you are concerned about disk space, you may customize the installation, leaving some components to run off the CDROM. Following is a list of the requirements for each component.

Component	CD	Local
GRASS4.1	X	X
Xsonar	X	X
GMT	X	X
Tcl	X	X
Tk	X	X

Although the various components indeed may be run from the CD, these implementations are not supported by this RTGIS Release. It is recommended that all component executables be installed on the host computer.

2.5 Installation Hints

Before embarking out on your own and customizing the RTGIS system and the interface between its components, it is highly recommended that the installer adhere to some "suggested" guidelines. They are

as follows:

1. Create a user named "grass" to own the RTGIS installation.

2. Install the RTGIS to the /usr/local directory.

3. Give user grass write permission for the /usr/local directory.

4. Install all RTGIS system component sources via the provided install script.

5. When possible, use the pre-compiled binaries provided on the CD rather than allowing the install script to compile the sources for you.

If these simple guidelines are followed, the installation should proceed perfectly when installing on the supported operating systems. Of course, the installer may choose to ignore all of the above. In that case, it is assumed that he or she knows what he or she is doing.

Deviating from the suggested guidelines above will surely create additional porting issues. And render the provided installation script virtually useless.

2.6 Installing the RTGIS Distribution

Installation of the RTGIS system can be quite involved. It encompasses the installation of at least 5 separate software systems and requires configuration of the UNIX environment to make those separate systems work together. In order to make the installation bearable for non UNIX experts, an installation script has been created.

The installation script is an sh shell script. It will set environment variables, make directories, and install files based on user input. After each prompt for user input, the available options will be listed in brackets. Where appropriate, the option in CAPITALS indicates the default, which may be selected simply by pressing <ENTER>.

Below are step-by-step instructions for installing the RTGIS Distribution Software on IRIX and Linux workstations via the install script. Installers very familiar with UNIX, software installation, and shell programming are welcome to read through the install script and possibly make customizations.

1. Mount the RTGIS Distribution CD.

   If you are unsure how to do this, consult with your system administrator or your operating system documentation. Usually, you must be root to mount the CD.

   For example, type one of the following commands at the root prompt (#) to mount the CD on Linux and IRIX, respectively.

   (for Linux)
   # mount -t iso9660 /dev/hdc /cdrom

   (for IRIX)
   # mount -t iso9660 -r /dev/scsi/sc1d1l0 /CDROM

2. Copy the install script to your home directory.

It is assumed at this point that you have followed the guidelines in section 2.5 above and are now working as user grass.

For example, at the grass prompt (denoted by the % symbol), type the following:

% cp /cdrom/install ~/install

3. Change directories to your home directory.

   % cd

4. Run the install script.

   % ./install

5. After reading the startup message, press ENTER if you agree to continue.

6. Next, the install script will determine your platform operating system.

   IRIX and Linux are recognized and supported. If you are installing on a different OS, you will be warned that the install may not proceed properly. You will then be asked to continue.

   Press ENTER to continue if all is OK.

7. Next, you are warned that the CD must be mounted for the installation to proceed.

   In the event that this is not the first installation and the install script was not just previously copied from the CD, be sure the CD is mounted before continuing.

   Answer yes or no.

8. Enter the CD mount directory.

   On Linux systems, this is typically /cdrom. On IRIX systems, it is typically /CDROM.

   After you enter the CD mount directory, the script will verify the RTGIS CD.

9. Next, enter the RTGIS top level directory.

   You should accept the default here. Although you are provided with an option, be warned that there are many dependencies on the /usr/local top level directory in this release of the RTGIS. Providing a different top level directory will most likely prevent some components from installing properly.

   Press ENTER to accept the default.

10. You will then be warned that some directories will be created for the RTGIS software.

    Press ENTER to make the directories and continue.

    The script will then report the status of the directory creation.

11. Next, the script will determine how much disk space you have available on your system.

This feature is provided for future support of a more customizing installation script. For this RTGIS release, the disk space is not monitored during the installation. It is assumed that you have enough disk space for the full RTGIS installation, about 350 megabytes.

Press ENTER to continue.

12. Next, you are prompted for the components you would like to install.

    You should accept the defaults. Accepting the defaults will allow the script to install all RTGIS source code and the pre-compiled RTGIS binaries on your system. This installation method will provide the greatest ease and success.

    Press ENTER to accept all the defaults.

13. The install script will then review your selections.

    If everything appears OK, press ENTER.

    If you made and error, you may answer no and restart the installation script.

14. The installation of RTGIS files will then begin. This can take up to 20 minutes, depending on which components you have selected to install.

15. After all RTGIS files have been copied onto your system, the install script will offer to automatically compile each source code component you have installed.

    If installing on Linux, you should accept the defaults here to use the pre-compiled binaries rather than recompiling. If installing on IRIX, you should choose to compile all software.

    The compiling and installation can be rather lengthy. If you are compiling GRASS, it would be safe to take a 1 hour lunch at this point.

16. When installation is complete, the install script will request to add some lines to your default login file.

    If this is your first installation, you should answer yes to add the lines to your default login file. If you have performed a previous installation, there is no need to add the lines again.

17. Finally, the script will ask you if you would like to run a small test of the newly installed software. If this is your first installation, you are encouraged to answer yes.

    The test will simply start up the RTGIS system via the rtgrass4.1 script. You should see a GRASS console window appear along with a horizontal menu bar at the top right of your monitor.

    The test will then open a GRASS monitor to test the display and draw some data layers to test the data interface. You should see a colorful bathymetry raster layer appear on the GRASS monitor. This will quickly be overlayed by a NOAA chart raster layer, a side scan mosaic raster layer, and coastline and contoured bathymetry vector layers.

    The test will then start up both Xsonar and ShowImage.

    If any of the above modules did not appear during the test, the installation did not proceed correctly. You will have to go to the corresponding component directoy and install the component manually by reading the accompanying installation documents.

The RTGIS installation is complete. I hope you enjoy the software!

2.7 Demo

A demo procedure is included with the RTGIS Toolbox which walks the user through several tutorials using various RTGIS Toolbox tools. It is suggested that new users run the Demo. The Demo is started by selecting the Demo button on the RTGIS Main Menu. The RTGIS Toolbox User's Manual must be used with the Demo. Section 4.1 of the User's Manual describes how to start the RTGIS Toolbox. Section 4.2 describes how to start the Demo. Step-by-step instructions are provided throughout the remainder of the manual for the Demo procedures. Most Demo procedures will refer directly to an appropriate section of the User's Manual.

2.8 Troubleshooting and Support

Manual updates and technical support for the RTGIS Toolbox distribution is provided by the Ocean Mapping Development Center of the University of Rhode Island.

Questions and comments should be directed to Dr. Robert Tyce at the Department of Ocean Engineering, 401-874-6139 or tyce@ocean.oce.uri.edu.

2.9 Programmer's Notes

While the install script should theoretically work for SGI IRIX, it has only been tested and verified for Linux. The only part of the install script which should potentially present problems for operating systems other than Linux is the automatic compilation which assumes that certain include files, libraries, and utilities exist. If any of these items do not exist on the target system, certain parts of the compilation will almost certainly fail. In such cases, the install script must be made more bulletproof or the installer may separately troubleshoot and build the software system in question.

The remainder of this section contains a listing of the install script.

```
#!/usr/bin/bash
# This little sh shell script will attempt to install the
# RTGIS system based on user input.
# Written in a series of iterations by Stephen Dzurenko
# sometime in 1996.

# Utility functions follow, the main body is at the end

stty erase '^H'

#************************************************************
# GetReturn
# Prompts the user the hit the ENTER key to continue
#************************************************************

GetReturn ()
{
  echo
  echo Press ENTER to continue ...
```

```
 read RETURN
}


#************************************************************
# ContinueYorN
# Asks the user if we should continue the installation
# procedure or not, quits if no, returns if yes
#************************************************************

ContinueYorN ()
{
 echo
 echo Do you want to continue? [Y/n]
 read ans
 case "$ans" in
        y | Y | yes | YES | Yes | "")
                # return
                echo
                ;;
        n | N | no | NO | No)
                echo Abnormal termination.  Installation not complete.
                exit
                ;;
        *)
                echo Invalid response.
                exit
                ;;
 esac
}


#********************************************************************
# WakeUpCd
# Function spins up the CD.  On certain Linux boxes, it seems that
# if the CD-ROM drive is accessed from an "sleeping" state, it
# fails.  This will get it going.
#********************************************************************

WakeUpCd ()
{
 $LS $CDROMDIR >/dev/null
}


#************************************************************
# PrintInitialMsg
# Prints a start up greeting and message
#************************************************************

PrintInitialMsg ()
{
 echo
 echo
 echo Welcome to the RTGIS BETA Release Installation Script.
```

```
 echo
 echo After asking you for some input and making a few system
 echo checks, I will attempt to install the RTGIS System on
 echo your computer.
 echo
 echo Please be sure to read the README_INSTALL file first.
 echo It will explain exactly what the installation script
 echo will do and explain the choices you must make.
}


#***************************************************************************
# GetArch
# Function gets the current system architecture
#***************************************************************************

GetArch ()
{
 # We only support IRIX and Linux
 UNAME="/bin/uname"
 if [ ! -r $UNAME ]
 then
        UNAME="/sbin/uname"
 fi

 ARCH=`$UNAME`

 case "$ARCH" in
        Linux)
                # Linux supported, continue
                echo
                echo $ARCH architecture confirmed.
                GetReturn
                ;;
        IRIX)
                # IRIX supported, continue
                echo
                echo $ARCH architecture confirmed.
                GetReturn
                ;;
        *)
                # warn user that binaries are only supported for
                # Linux and IRIX
                echo
                echo WARNING: This seems to be a [$ARCH] machine.
                echo RTGIS only supports Linux and IRIX architectures.
                echo There is a high probability that the installation will fail.
                ContinueYorN
                ;;
 esac
}


#***********************************************************
```

```
# CheckNeededUtilities
# Checks for some common UNIX commands used by this script.
# Attempts to locate any not found in the default location.
#************************************************************

CheckNeededUtilities ()
{
 # The utilities I will need for the install
 case "$ARCH" in
        Linux)
                AWK="/usr/bin/awk"
                DF="/bin/df"
                CAT="/bin/cat"
                CP="/bin/cp"
                GREP="/usr/bin/grep"
                LS="/bin/ls"
                MKDIR="/bin/mkdir"
                WHOAMI="/usr/bin/whoami"
                TAR="/bin/tar"
                ;;
        *)
                AWK="/usr/bin/awk"
                DF="/sbin/df"
                CAT="/sbin/cat"
                CP="/bin/cp"
                GREP="/sbin/grep"
                LS="/sbin/ls"
                MKDIR="/sbin/mkdir"
                WHOAMI="/usr/bin/whoami"
                TAR="/sbin/tar"
                ;;
 esac

 FILES="$AWK $DF $CAT $CP $GREP $LS $MKDIR $WHOAMI $TAR"
 MISSING=0
 for FILE in $FILES
 do
        if [ ! -r $FILE ]
        then
                # attempt to find the file

                echo "$FILE: missing.  Installation may not proceed properly!"
                MISSING=1
        fi
 done
 if [ $MISSING -eq 1 ]
 then
        GetReturn
 fi
}


#************************************************************
# CheckCdMounted
# Checks to that the Distribution CD is mounted,
```

```
# prompts the user to do so if it is not
#************************************************************

CheckCdMounted ()
{
 echo In order to install the RTGIS software, the RTGIS
 echo Distribution CD must be mounted on your system.
 echo
 echo You may need Super User priveleges to mount the CD.  If
 echo you are not sure how to do this, contact your System
 echo Administrator.
 echo
 echo If YOU are not the System Administrator of the
 echo RTGIS target host, you should contact your System
 echo Administrator to discuss disk space limitations and
 echo possibly adding a new user to "own" the RTGIS software.
 echo
 echo Note:  If you intend to run some RTGIS components from
 echo the CD, you must mount the RTGIS Distribution CD at the
 echo same directory location for the installation as it will
 echo be mounted for running the RTGIS.  This is because the
 echo installation will set several environment variables based
 echo on the mount location of the RTGIS Distribution CD.  When
 echo you run the RTGIS, the CD will be expected to be at the
 echo same location.
 echo
 echo Is the RTGIS Distribution CD mounted? [Y/n]

 read ans
 case "$ans" in
        y | Y | yes | YES | Yes | "")
                # proceed
                echo
                ;;
        n | N | no | NO | No)
                echo
                echo Please try again after you have mounted the RTGIS Distribution CD.
                echo Abnormal termination.  Installation not complete.
                exit
                ;;
        *)
                echo Invalid response.
                exit
                ;;
 esac
}

#************************************************************************
# GetCdDir
# Function prompts the user to enter the mount location of the CD
# Assumes /CDROM as the default
#************************************************************************

GetCdDir ()
{
```

```
 echo
 echo What directory is the RTGIS CDROM mounted on? [/CDROM]
 read ans
 if [ "$ans" = "" ]
 then
        CDROMDIR=/CDROM
 else
        CDROMDIR=$ans
 fi

 # Make sure the CDROMDIR is valid by checking for the existence of a
 # couple directories
 WakeUpCd
 if test ! -d $CDROMDIR/grass4.1 && test ! -d $CDROMDIR/xsonar
 then
        echo
        echo I don\'t find the correct files in $CDROMDIR.
        echo Make sure you have the RTGIS Distribution CD mounted
        echo and you have entered the correct directory then try again.
        echo
        echo Abnormal termination.  Installation not complete.
        exit
 else
        # proceed
        echo
        echo RTGIS Distribution CD mount directory $CDROMDIR confirmed.
 fi
}

#*********************************************************************
# GetTopLevel
# Function prompts the user for the top level install directory
# Assumes /usr/local as the default
#*********************************************************************

GetTopLevel ()
{
 # Set the default top level directory
 RTGISROOTDIRDEFAULT=/usr/local
 RTGISROOTDIR=$RTGISROOTDIRDEFAULT

 # Get the top level directory from the user
 echo
 echo Please enter the RTGIS top level directory [$RTGISROOTDIR]:
 read RTGISROOTDIR
 if [ "$RTGISROOTDIR" = "" ]
 then
        RTGISROOTDIR=$RTGISROOTDIRDEFAULT
 fi
}


#*********************************************************************
# SetComponentDirs
# Function creates the component subdirectories based on the top level dir
```

```
#*********************************************************************

SetComponentDirs ()
{
 # set the component directories as subdirectories of the
 # top level directory
 echo
 echo I will now create the following component subdirectories:
 echo For GRASS4.1:  [$RTGISROOTDIR/grass4.1]
 echo For the RTGIS database:  [$RTGISROOTDIR/grass.data]
 echo For XSonar:  [$RTGISROOTDIR/xsonar]
 echo For GMT:  [$RTGISROOTDIR/gmt]
 echo For Tcl:  [$RTGISROOTDIR/tcl7.4]
 echo For Tk:  [$RTGISROOTDIR/tk4.0]

 echo
 echo Does everything look o.k.? [Y/n]
 read ans
 if [  ! "$ans" = "" ]
 then
        echo Abnormal exit.  Installation not complete.
        exit
 else
        GIS=$RTGISROOTDIR/grass4.1
        GISDBASE=$RTGISROOTDIR/grass.data
        BINDIR=$RTGISROOTDIR/bin
        LIBDIR=$RTGISROOTDIR/lib
        XSONARBINDIR=$BINDIR
        XSONARLIBDIR=$LIBDIR
        XSONARSRCDIR=$RTGISROOTDIR/xsonar
        GMTBINDIR=$BINDIR
        GMTLIBDIR=$LIBDIR
        GMTSRCDIR=$RTGISROOTDIR/gmt
        TCLTKBINDIR=$BINDIR
        TCLTKLIBDIR=$LIBDIR
        TCLSRCDIR=$RTGISROOTDIR/tcl7.4
        TKSRCDIR=$RTGISROOTDIR/tk4.0
 fi
}


#*********************************************************************
# MakeAllDirs
# Function to make the top level directory and all component sub dirs
#*********************************************************************

MakeAllDirs ()
{
 # Make the top level directory if it does not exist
 FAILED=0
 if test ! -d $RTGISROOTDIR
 then
        $MKDIR $RTGISROOTDIR
        if test ! -d $RTGISROOTDIR
        then
```

```
                echo Failed to make $RTGISROOTDIR
                FAILED=1
        else
                echo Made $RTGISROOTDIR
        fi
else
        echo $RTGISROOTDIR already exists
fi

# Now make all the component directories as needed
if test ! -d $GIS
then
        $MKDIR $GIS
        if test ! -d $GIS
        then
                echo Failed to make $GIS
                FAILED=1
        else
                echo Made $GIS
        fi
else
        echo $GIS already exists
fi

if test ! -d $GISDBASE
then
        $MKDIR $GISDBASE
        if test ! -d $GISDBASE
        then
                echo Failed to make $GISDBASE
                FAILED=1
        else
                echo Made $GISDBASE
        fi
else
        echo $GISDBASE already exists
fi

if test ! -d $BINDIR
then
        $MKDIR $BINDIR
        if test ! -d $BINDIR
        then
                echo Failed to make $BINDIR
                FAILED=1
        else
                echo Made $BINDIR
        fi
else
        echo $BINDIR already exists
fi

if test ! -d $LIBDIR
then
    $MKDIR $LIBDIR
```

```
        if test ! -d $LIBDIR
        then
                echo Failed to make $LIBDIR
                FAILED=1
        else
        echo Made $LIBDIR
        fi
else
    echo $LIBDIR already exists
fi

if test ! -d $XSONARSRCDIR
then
        $MKDIR $XSONARSRCDIR
        if test ! -d $XSONARSRCDIR
        then
                echo Failed to make $XSONARSRCDIR
                FAILED=1
        else
                echo Made $XSONARSRCDIR
        fi
else
        echo $XSONARSRCDIR already exists
fi

if test ! -d $GMTSRCDIR
then
        $MKDIR $GMTSRCDIR
        if test ! -d $GMTSRCDIR
        then
                echo Failed to make $GMTSRCDIR
                FAILED=1
        else
                echo Made $GMTSRCDIR
        fi
else
        echo $GMTSRCDIR already exists
fi

if test ! -d $TCLSRCDIR
then
        $MKDIR $TCLSRCDIR
        if test ! -d $TCLSRCDIR
        then
                echo Failed to make $TCLSRCDIR
                FAILED=1
        else
                echo Made $TCLSRCDIR
        fi
else
        echo $TCLSRCDIR already exists
fi

if test ! -d $TKSRCDIR
then
```

```
        $MKDIR $TKSRCDIR
        if test ! -d $TKSRCDIR
        then
                echo Failed to make $TKSRCDIR
                FAILED=1
        else
                echo Made $TKSRCDIR
        fi
 else
        echo $TKSRCDIR already exists
 fi

 if [ $FAILED -eq 1 ]
 then
        echo
        echo At least one RTGIS directory was not able to be created.
        echo Most likely the installation will fail.
        echo It is recommended that you quit the install and fix the problem.
        ContinueYorN
 fi
}



#*****************************************************************************
# GetTotalDiskSpace
# Function probes the local system to see how much disk space is available
# before starting RTGIS installation
#*****************************************************************************

GetTotalDiskSpace ()
{
 case "$ARCH" in
        Linux)
                TOTALDISKAVAIL=`$DF -k $RTGISROOTDIR | $AWK '{print $4}'`
                TOTALDISKAVAIL=`echo $TOTALDISKAVAIL | $AWK '{print $2}'`
                echo
                echo Total disk space available at start of installation\: $TOTALDISKAVAIL KB
                GetReturn
                ;;
        IRIX)
                TOTALDISKAVAIL=`$DF -k $RTGISROOTDIR | $AWK '{print $5}'`
                TOTALDISKAVAIL=`echo $TOTALDISKAVAIL | $AWK '{print $2}'`
                echo
                echo Total disk space available at start of installation\: $TOTALDISKAVAIL KB
                GetReturn
                ;;
        *)
                echo
                echo I don\'t know how to get the disk space available for
                echo this machine.  You will have to keep track of available
                echo disk space yourself
                echo
                ContinueYorN
 esac
```

```
}

#*****************************************************************************
# GetComponentsToInstall
# Function prompts user for the components to be installed
#*****************************************************************************

GetComponentsToInstall ()
{
 # Find out what the user wants to install and where
 echo
 echo Please indicate which components you want installed to your local disk

 echo Should I install the GRASS4.1 binaries? [y/N]
 read INSTALLGRASSBIN
 case "$INSTALLGRASSBIN" in
        y | Y | yes | YES | Yes)
                INSTALLGRASSBIN=y
                ;;
        *)
                INSTALLGRASSBIN=n
                ;;
 esac

 echo Should I install the GRASS4.1 source? [Y/n]
 read INSTALLGRASSSRC
 case "$INSTALLGRASSSRC" in
        n | N | no | NO | No)
                INSTALLGRASSSRC=n
                ;;
        *)
                INSTALLGRASSSRC=y
                ;;
 esac

 echo Should I install the XSonar binaries? [y/N]
 read INSTALLXSONARBIN
 case "$INSTALLXSONARBIN" in
        y | Y | yes | YES | Yes)
                INSTALLXSONARBIN=y
                ;;
        *)
                INSTALLXSONARBIN=n
                ;;
 esac

 echo Should I install the XSonar source? [Y/n]
 read INSTALLXSONARSRC
 case "$INSTALLXSONARSRC" in
        n | N | no | NO | No)
                INSTALLXSONARSRC=n
                ;;
        *)
                INSTALLXSONARSRC=y
```

```
                ;;
esac

echo Should I install the GMT binaries? [y/N]
read INSTALLGMTBIN
case "$INSTALLGMTBIN" in
        y | Y | yes | YES | Yes)
                INSTALLGMTBIN=y
                ;;
        *)
                INSTALLGMTBIN=n
                ;;
esac

echo Should I install the GMT source? [Y/n]
read INSTALLGMTSRC
case "$INSTALLGMTSRC" in
        n | N | no | NO | No)
                INSTALLGMTSRC=n
                ;;
        *)
                INSTALLGMTSRC=y
                ;;
esac

echo Should I install the Tcl/Tk libraries? [y/N]
read INSTALLTCLTKLIB
case "$INSTALLTCLTKLIB" in
        y | Y | yes | YES | Yes)
                INSTALLTCLTKLIB=y
                ;;
        *)
                INSTALLTCLTKLIB=n
                ;;
esac

echo Should I install the Tcl/Tk source? [Y/n]
read INSTALLTCLTKSRC
case "$INSTALLTCLTKSRC" in
        n | N | no | NO | No)
                INSTALLTCLTKSRC=n
                ;;
        *)
                INSTALLTCLTKSRC=y
                ;;
esac

echo Should I install the RTGIS sample data base? [Y/n]
read INSTALLDATABASE
case "$INSTALLDATABASE" in
        n | N | no | NO | No)
                INSTALLDATABASE=n
                ;;
        *)
                INSTALLDATABASE=y
```

```
                ;;
esac

# Now recap for the user
echo
echo You have selected to install the following components:
if [ "$INSTALLGRASSBIN" = "y" ]
then
        echo GRASS4.1 binaries
fi
if [ "$INSTALLGRASSSRC" = "y" ]
then
        echo GRASS4.1 source
fi
if [ "$INSTALLXSONARBIN" = "y" ]
then
        echo XSonar binaries
fi
if [ "$INSTALLXSONARSRC" = "y" ]
then
        echo XSonar source
fi
if [ "$INSTALLGMTBIN" = "y" ]
then
        echo GMT binaries
fi
if [ "$INSTALLGMTSRC" = "y" ]
then
        echo GMT source
fi
if [ "$INSTALLTCLTKLIB" = "y" ]
then
        echo Tcl/Tk libraries
fi
if [ "$INSTALLTCLTKSRC" = "y" ]
then
        echo Tcl/Tk source
fi
if [ "$INSTALLDATABASE" = "y" ]
then
        echo RTGIS sample database
fi


echo
echo And leave the following components on the RTGIS CD:
if [ "$INSTALLGRASSBIN" = "n" ]
then
    echo GRASS4.1 binaries
fi
if [ "$INSTALLGRASSSRC" = "n" ]
then
    echo GRASS4.1 source
fi
if [ "$INSTALLXSONARBIN" = "n" ]
```

```
then
    echo XSonar binaries
fi
if [ "$INSTALLXSONARSRC" = "n" ]
then
    echo XSonar source
fi
if [ "$INSTALLGMTBIN" = "n" ]
then
    echo GMT binaries
fi
if [ "$INSTALLGMTSRC" = "n" ]
then
    echo GMT source
fi
if [ "$INSTALLTCLTKLIB" = "n" ]
then
    echo Tcl/Tk libraries
fi
if [ "$INSTALLTCLTKSRC" = "n" ]
then
    echo Tcl/Tk source
fi
if [ "$INSTALLDATABASE" = "n" ]
then
        echo RTGIS sample database
fi

echo
echo Does everything look o.k.? [Y/n]
read ans
case "$ans" in
        y | Y | yes | YES | Yes | "")
                # continue
                echo
                ;;
        *)
        echo Abnormal exit.  Installation not complete.
        exit
                ;;
esac
}


#*********************************************************************
# InstallRtgisFiles
# Function copies the requested RTGIS components to the user's fixed
# disk
#*********************************************************************

InstallRtgisFiles ()
{
 echo
 echo RTGIS installation begins...
```

```
if [ "$INSTALLGRASSBIN" = "y" ]
then
    echo
        echo Copying GRASS binaries...
        echo
        cd $GIS
        WakeUpCd
        cd $CDROMDIR/grass4.1
        $TAR cf - bin | (cd $GIS; $TAR xf -)
        $TAR cf - driver | (cd $GIS; $TAR xf -)
        $TAR cf - etc | (cd $GIS; $TAR xf -)
        $TAR cf - man | (cd $GIS; $TAR xf -)
        $TAR cf - scripts | (cd $GIS; $TAR xf -)
        $TAR cf - locks | (cd $GIS; $TAR xf -)
        $TAR cf - fonts | (cd $GIS; $TAR xf -)
        $CP grass4.1 $GIS/grass4.1
        $CP rtgrass4.1 $GIS/rtgrass4.1
        $CP xgrass4.1 $GIS/xgrass4.1
        $CP .grassrc ~/.grassrc
        $CP .grassrc_lock ~/.grassrc_lock
        $CP .grass.cshrc ~/.grass.cshrc
        $CP .grass.cshrc.rt ~/.grass.cshrc.rt
        echo
        echo GRASS4.1 executables and support files installed!

        GRASSBINDIR=$GIS/grass4.1
else
        GRASSBINDIR=$CDROMDIR/grass4.1
fi

if [ "$INSTALLGRASSSRC" = "y" ]
then
        echo
    echo Copying GRASS source files...
        echo
        WakeUpCd
        cd $CDROMDIR/grass4.1
        $TAR cf - man | (cd $GIS; $TAR xf -)
        echo ...
        $TAR cf - src | (cd $GIS; $TAR xf -)
        echo ...
        $TAR cf - src.OMDC | (cd $GIS; $TAR xf -)
        echo ...
        $TAR cf - src.alpha | (cd $GIS; $TAR xf -)
        echo ...
        $TAR cf - src.contrib | (cd $GIS; $TAR xf -)
        echo ...
        $TAR cf - src.related | (cd $GIS; $TAR xf -)
        $TAR cf - scripts | (cd $GIS; $TAR xf -)
        $CP gmake4.1 $GIS/gmake4.1
        $CP gmake4.1.Linux $GIS/gmake4.1.Linux
        $CP gmake4.1.IRIX $GIS/gmake4.1.IRIX
        $CP grass4.1 $GIS/grass4.1
        $CP rtgrass4.1 $GIS/rtgrass4.1
        $CP xgrass4.1 $GIS/xgrass4.1
```

```
        $CP .grassrc ~/.grassrc
        $CP .grassrc_lock ~/.grassrc_lock
        $CP .grass.cshrc ~/.grass.cshrc
        $CP .grass.cshrc.rt ~/.grass.cshrc.rt
        $CP .grass.cshrc.default ~/.grass.cshrc.default
        $CP .grass.bashrc.rt ~/.grass.bashrc.rt
        $CP .grass.bashrc.default ~/.grass.bashrc.default
        WakeUpCd
        cd $CDROMDIR
        $TAR cf - tcl | (cd $RTGISROOTDIR; $TAR xf -)
        echo
        echo GRASS4.1 source files installed!
fi

if [ "$INSTALLXSONARBIN" = "y" ]
then
    echo
        echo Copying XSonar binaries...
        echo
        WakeUpCd
        cd $CDROMDIR/xsonar
        $CP xsonar/xsonar $XSONARBINDIR/xsonar
        $CP showimage/showimage $XSONARBINDIR/showimage
        $CP usgsmap/libmap.a $XSONARLIBDIR/libmap.a
        $CP xsonar/XSonar.ad ~/XSonar
        $CP showimage/ShowImage.ad ~/ShowImage
        echo
        echo XSonar executable and support files installed!
else
        # set things up to run from the CD
        WakeUpCd
        cd $CDROMDIR/xsonar
        $CP xsonar/XSonar.ad ~/XSonar
        $CP showimage/ShowImage.ad ~/ShowImage
        XSONARBINDIR=$CDROMDIR/xsonar/xsonar
        SHOWIMAGEBINDIR=$CDROMDIR/xsonar/showimage
        XSONARLIBDIR=$CDROMDIR/xsonar/usgsmap
fi

if [ "$INSTALLXSONARSRC" = "y" ]
then
    echo
        echo Copying XSonar source files...
        echo
        WakeUpCd
        cd $CDROMDIR/xsonar
        $TAR cf - showimage | (cd $XSONARSRCDIR; $TAR xf -)
        echo ...
        $TAR cf - usgsmap | (cd $XSONARSRCDIR; $TAR xf -)
        echo ...
        $TAR cf - xsonar | (cd $XSONARSRCDIR; $TAR xf -)
        $CP Gmakefile $XSONARSRCDIR/Gmakefile
        $CP Makefile $XSONARSRCDIR/Makefile
        $CP README $XSONARSRCDIR/README
        WakeUpCd
```

```
        cd $CDROMDIR/xsonar
        $CP -f xsonar.c $XSONARSRCDIR/xsonar/xsonar.c
        echo
        echo XSonar source files installed!
fi

if [ "$INSTALLGMTBIN" = "y" ]
then
    echo
        echo Copying GMT binaries...
        echo
        WakeUpCd
        cd $CDROMDIR/gmt
        $TAR cf - bin | (cd $GMTBINDIR; $TAR xf -)
        $CP lib/libgmt.a $GMTLIBDIR/libgmt.a
        $CP lib/libnetcdf.a $GMTLIBDIR/libnetcdf.a
        echo
        echo GMT executables and support files installed!
fi

if [ "$INSTALLGMTSRC" = "y" ]
then
    echo
        echo Copying GMT source files...
        echo
        WakeUpCd
        cd $CDROMDIR/gmt
        $TAR cf - src | (cd $GMTSRCDIR; $TAR xf -)
        echo ...
        $TAR cf - netcdf-232pl2 | (cd $GMTSRCDIR; $TAR xf -)
        echo ...
        $TAR cf - lib | (cd $GMTSRCDIR; $TAR xf -)
        echo ...
        $TAR cf - include | (cd $GMTSRCDIR; $TAR xf -)
        $TAR cf - examples | (cd $GMTSRCDIR; $TAR xf -)
        $CP CHANGES $GMTSRCDIR/CHANGES
        $CP README $GMTSRCDIR/README
        $CP README.GMT $GMTSRCDIR/README.GMT
        $CP install_gmt $GMTSRCDIR/install_gmt
        echo
        echo GMT source files installed!
fi

if [ "$INSTALLTCLTKLIB" = "y" ]
then
    echo
        echo Copying Tcl/Tk binaries...
        echo
        WakeUpCd
        cd $CDROMDIR/tcl7.4
        $CP tclsh $BINDIR/tclsh
        $CP tclsh7.4 $BINDIR/tclsh7.4
        $CP libtcl7.4.a $LIBDIR/libtcl7.4.a
        cd $CDROMDIR/tk4.0
        $CP wish $BINDIR/wish
```

```
        $CP wish4.0 $BINDIR/wish4.0
        $CP libtk4.0.a $LIBDIR/libtk4.0.a
        echo
        echo Tcl/Tk binaries and libraries installed!
else
        TCLBINDIR=$CDROMDIR/tcl7.4
        TKBINDIR=$CDROMDIR/tk4.0
fi

if [ "$INSTALLTCLTKSRC" = "y" ]
then
    echo
        echo Copying Tcl/Tk source files...
        echo
        WakeUpCd
        cd $CDROMDIR
        $TAR cf - tcl7.4 | (cd $RTGISROOTDIR; $TAR xf -)
        echo ...
        WakeUpCd
        cd $CDROMDIR
        $TAR cf - tk4.0 | (cd $RTGISROOTDIR; $TAR xf -)
        echo
        echo Tcl/Tk source files installed!
fi

if [ "$INSTALLDATABASE" = "y" ]
then
        echo
        echo Copying sample database files ...
        WakeUpCd
        cd $CDROMDIR
        $TAR cf - grass.data | (cd $RTGISROOTDIR; $TAR xf -)
        echo
        echo RTGIS sample database installed!
fi
}


#*********************************************************************
# CompileRtgisSource
# Function determines if we are installing source code and run the
# appropriate routines to set the system up for RTGIS compiliing if
# we are
#*********************************************************************

CompileRtgisSource ()
{
 if [ "$INSTALLGRASSSRC" = "y" ]
 then
        echo
        echo You have installed the Grass4.1 source code.
        echo Would you like me to automatically compile it for you
        echo now and install the binaries? [Y/n]
        read ans
        case "$ans" in
```

```
        n | N | no | NO | No)
                echo
                echo Grass4.1 source not compiled.
                echo
                ;;
        *)
                echo
                echo Building Grass4.1 binaries ...
                cd $GIS
                case "$ARCH" in
                        Linux)
                                $CP $GIS/gmake4.1.Linux $GIS/gmake4.1
                                $CP $GIS/src/CMD/GISGEN.ibm.linux
$GIS/src/CMD/GISGEN
                                $CP $GIS/src/CMD/generic/GISGEN.sh.Linux
$GIS/src/CMD/generic/GISGEN.sh
                                $CP $GIS/src/CMD/MAKELINKS.ibm.linux
$GIS/src/CMD/MAKELINKS
                                ;;
                        *)
                                $CP $GIS/gmake4.1.IRIX $GIS/gmake4.1
                                $CP $GIS/src/CMD/GISGEN.sgi.irix
$GIS/src/CMD/GISGEN
                                $CP $GIS/src/CMD/generic/GISGEN.sh.IRIX
$GIS/src/CMD/generic/GISGEN.sh
                                $CP $GIS/src/CMD/MAKELINKS.sgi.irix
$GIS/src/CMD/MAKELINKS
                                ;;
                esac
                cd $GIS/src/CMD
                GISGEN

                # install the Xdrivers
                WakeUpCd
                cd $CDROMDIR/grass4.1/etc
                case "$ARCH" in
                        Linux)
                                $CP moncap.Linux $GIS/etc/monitorcap
                                ;;
                        *)
                                $CP moncap.IRIX $GIS/etc/monitorcap
                                ;;
                esac
                cd $GIS/src/display/devices/XDRIVER/lib
                gmake4.1
                cd $GIS/src/display/devices/XDRIVER/XDRIVER
                gmake4.1

                cd $GIS/src/CMD
                MAKELINKS

                # copy the scripts back over, as the grass compile
                # overwrites the scripts directory
                WakeUpCd
                cd $CDROMDIR/grass4.1
```

```
                $TAR cf - scripts | (cd $GIS; $TAR xf -)
                # copy over GIS.sh again, which gets rewritten by
                # the install
                WakeUpCd
                cd $$CDROMDIR/grass4.1/etc
                $CP GIS.sh $GIS/etc/GIS.sh
                echo
                echo Cleaning up ...
                cd $GIS
                find src* -name '*.o' -print | xargs rm -f
        esac

fi

if [ "$INSTALLXSONARSRC" = "y" ]
then
        echo
        echo You have installed the Xsonar source code.
        echo Would you like me to automatically compile it for you
        echo now and install the binaries? [Y/n]
        read ans
        case "$ans" in
                n | N | no | NO | No)
                        echo
                        echo Xsonar source not compiled.
                        echo
                        ;;
                *)
                        echo
                        echo Building Xsonar binaries ...
                        # we need GRASS gmake4.1 to build xsonar and
                        # showimage.  If not installing GRASS, make sure
                        # we can find it.
                        if [ "$INSTALLGRASSSRC" = "n" ]
                        then
                                case "$ARCH" in
                                        Linux)
                                                $CP $GIS/gmake4.1.Linux $GIS/gmake4.1
                                                ;;
                                        *)
                                                $CP $GIS/gmake4.1.IRIX $GIS/gmake4.1
                                                ;;
                                esac
                        fi

                        cd $XSONARSRCDIR/usgsmap
                        make
                        makelib libmap.a
                        $CP libmap.a $RTGISROOTDIR/lib/libmap.a
                        \rm -f *.o

                        cd $XSONARSRCDIR/xsonar
                        case "$ARCH" in
                                Linux)
                                        $CP Gmakefile.Linux Gmakefile
```

```
                        ;;
                *)
                        $CP Gmakefile.IRIX Gmakefile
                        ;;
        esac
        $GIS/gmake4.1
        $CP xsonar $BINDIR/xsonar
        $CP XSonar.ad ~/XSonar
        \rm -f *.o

        cd $XSONARSRCDIR/showimage
        case "$ARCH" in
                Linux)
                        $CP Gmakefile.Linux Gmakefile
                        ;;
                *)
                        $CP Gmakefile.IRIX Gmakefile
                        ;;
        esac
        $GIS/gmake4.1
        $CP showimage $BINDIR/showimage
        $CP ShowImage.ad ~/ShowImage
        \rm -f *.o
        ;;
    esac
    echo
    echo XSonar installation complete!
fi

if [ "$INSTALLGMTSRC" = "y" ]
then
    echo
    echo You have installed the GMT source code.
    echo Would you like me to automatically compile it for you
    echo now and install the executables? [Y/n]
    read ans
    case "$ans" in
        n | N | no | NO | No)
            echo
            echo GMT source not compiled.
            echo
            ;;
        *)
            echo
            echo It is highly likely that you will see a large amount of
            echo warning messages during the building of GMT binaries.
            echo These warnings may safely be ignored.
            GetReturn
            echo
            echo Installing netcdf library ...
            cd $GMTSRCDIR/netcdf-232pl2
            case "$ARCH" in
                Linux)
                    echo
                    echo Configuring for Linux make ...
```

```
                CC=gcc FC=\"\" OS=Linux ./configure
                ;;
        IRIX)
                echo
                echo Configuring for IRIX make ...
                CC=cc FC=\"\" OS=IRIX ./configure
                ;;
esac
echo
echo Compiling netcdf ...
make all >log 2>&1
make test
make install
make clean
cd ..
echo
echo Copying gmt support files to /usr/local/gmt/lib ...
if test ! -d /usr/local/gmt
then
        $MKDIR /usr/local/gmt
fi
if test ! -d /usr/local/gmt/lib
then
        $MKDIR /usr/local/gmt/lib
        $CP $GMTSRCDIR/lib/\* /usr/local/gmt/lib
fi
cd $GMTSRCDIR/src
case "$ARCH" in
        Linux)
                $CP Makefile.Linux Makefile
                ;;
        IRIX)
                $CP Makefile.IRIX Makefile
                ;;
esac
echo
echo Building GMT binaries ...
make all
echo
echo Installing GMT binaries ...
make install
echo
echo Installing GMT man pages ...
if test ! -d $RTGISROOTDIR/man
then
        $MKDIR $RTGISROOTDIR/man
fi
if test ! -d $RTGISROOTDIR/man/man1
then
        $MKDIR $RTGISROOTDIR/man/man1
fi
make man
make man_install
make clean
echo
```

```
                        echo Testing GMT installation ...
                        cd $GMTSRCDIR/examples/ex14
                        if test -e example_14.ps
                        then
                                \\rm -f example_14.ps
                        fi
                        job14
                        if test -e example_14.ps
                        then
                                echo
                                echo GMT installation complete!
                                GetReturn
                        else
                                echo
                                echo GMT installation failed.  Please follow installation
                                echo notes to install manually.
                                GetReturn
                        fi
                        ;;
        esac
fi

if [ "$INSTALLTCLTKSRC" = "y" ]
then
        echo
        echo You have installed the Tcl/Tk source code.
        echo Would you like me to automatically compile it for you
        echo now and install the binaries? [Y/n]
        read ans
        case "$ans" in
                n | N | no | NO | No)
                        echo
                        echo Tcl/Tk source not compiled.
                        ;;
                *)
                        echo
                        echo Configuring Tcl ...
                        cd $TCLSRCDIR
                        ./configure --prefix /usr/local
                        echo
                        echo Building Tcl binaries ...
                        make
                        make install
                        make test

                        echo
                        echo Configuring Tk ...
                        cd $TKSRCDIR
                        ./configure --prefix /usr/local
                        echo
                        echo Building Tk binaries ...
                        make
                        make install
                        echo
                        echo I will now test the Tk installation.  You should see
```

```
                        echo many widgets flying across your screen during the test.
                        echo If you return to this installation script, the test
                        echo was successful.  Otherwise, you will have to review
                        echo the Tk installation notes for troubleshooting.
                        GetReturn
                        make test

                        cd $BINDIR
                        ln -s wish4.0 wish
                        ln -s tclsh7.4 tclsh

                        echo
                        echo Tcl/Tk installation complete!
                        GetReturn
                        ;;
        esac
 fi
}


#*********************************************************************
# PrintEndMsg
# Function prints the end of installation message
#*********************************************************************

PrintEndMsg ()
{
 # RTGIS installation has completed successfully
 echo
 echo Successful RTGIS installation is complete.  I will now
 echo do some clean up.
}


#*********************************************************************
# UpdateLoginFile
# Function adds necessary lines to the user's .login file
#*********************************************************************

UpdateLoginFile ()
{
 echo
 echo I need to update your login file so that necessary RTGIS
 echo environment variables get set each time you login.
 echo The following lines will be appended to your login file:
 echo
 case "$ARCH" in
        Linux)
                echo export GIS=\"$GIS\"
                echo export GISDBASE=\"$GISDBASE\"
                echo export GISBASE=\"$GIS\"
                echo export GISRC=\"$HOME/.grassrc\"
                echo export PATH=\"\$PATH:.:$GIS:$BINDIR:$LIBDIR\"
                echo export TCL_LIBRARY=\"$LIBDIR/tcl7.4\"
```

```
                echo
                ;;
        *)
                echo setenv GIS $GIS
                echo setenv GISDBASE $GISDBASE
                echo setenv GISBASE $GIS
                echo setenv GISRC $HOME/.grassrc
                echo setenv SG3D_WIDTH 750
                echo setenv SG3D_HEIGHT 600
                echo setenv PATH \$PATH:.:$GIS:$BINDIR:$LIBDIR
                echo setenv TCL_LIBRARY $LIBDIR/tcl7.4
                echo
                ;;
esac

echo
echo May I add these lines to your login file? [Y/n]
echo Note\: If I don\'t, you MUST add them yourself later.
read ans
case "$ans" in
        n | N | no | NO | No | "")
                echo
                echo OK, but remember to add these lines to your login file later!
                ;;
        *)
                case "$ARCH" in
                        Linux)
                                $CP ~/.bash_profile ~/.bash_profile.pre_rtgis
                                echo export GIS=\"$GIS\" >> ~/.bash_profile
                                echo export GISDBASE=\"$GISDBASE\" >> ~/.bash_profile
                                echo export GISBASE=\"$GIS\" >> ~/.bash_profile
                                echo export GISRC=\"$HOME/.grassrc\" >> ~/.bash_profile
                                echo export PATH=\"\$PATH:.:$GIS:$BINDIR:$LIBDIR\" >>
~/.bash_profile
                                echo export TCL_LIBRARY=\"$LIBDIR/tcl7.4\" >> ~/.bash_profile
                                echo
                                echo I added the above lines to your .bash_profile file and copied the
                                echo original to the file .bash_profile.pre_rtgis.
                                ;;
                        IRIX)
                                cp ~/.login ~/.login.pre_rtgis
                                echo setenv GIS $GIS >> ~/.login
                                echo setenv GISDBASE $GISDBASE >> ~/.login
                                echo setenv GISBASE $GIS >> ~/.login
                                echo setenv GISRC $HOME/.grassrc >> ~/.login
                                echo setenv SG3D_WIDTH 750 >> ~/.login
                                echo setenv SG3D_HEIGHT 600 >> ~/.login
                                echo setenv PATH \$PATH:.:$GIS:$BINDIR:$LIBDIR >> ~/.login
                                echo setenv TCL_LIBRARY $LIBDIR/tcl7.4 >> ~/.login
                                echo
                                echo I added the above lines to your .login file and copied the
                                echo original to the file .login.pre_rtgis.
                                ;;
                esac
                ;;
```

```
 esac
}

#*************************************************************************
# TestRtgis
# Function runs a script to perform a short test, more a demostration,
# of the newly installed RTGIS
#*************************************************************************

TestRtgis ()
{
 echo
 echo If you would like, I will now test your RTGIS installation.
 echo Would you like me to run a simple test of your RTGIS? [Y/n]
 read ans
 case "$ans" in
        n | N | no | NO | No)
                echo
                echo Then we are all done.
                echo Bye-bye!
                echo
                echo Normal termination.
                exit
                ;;
        *)
                # set up a simple display of some data
                echo
                echo Uncompressing some nice data ...
                echo
                cd $GISDBASE/RI/PERMANENT/cell
                uncompress chart13223.Z mosaic.Z TPS.BATHY.NAD83.Z
                cd $GISDBASE/RI/PERMANENT/dig
                uncompress bathy.5m.v.Z
                cd $RTGISROOTDIR
                echo
                echo Starting rtgrass4.1 ...
                xterm -e rtgrass4.1 &
                echo
                echo After positioning the rtgrass4.1 console window,
                echo be sure that the MAPSET is set to PERMANENT,
                echo the LOCATION is set to RI, and the database
                echo is set to $GISDBASE.
                echo Then press the \<ESC\> key in the rtgrass4.1
                echo console window.
                GetReturn
                echo
                echo After the Real Time extensions Tool Bar appears,
                echo press the \<ENTER\> key in the rtgrass4.1 console
                echo window to enter the GRASS prompt.
                GetReturn
                echo
                echo We are now ready for RTGIS processing.
                $GIS/bin/d.mon start=x0 &
                echo
```

```
                echo Please resize and reposition the GRASS monitor
                GetReturn
                $GIS/bin/d.mon select=x0
                $GIS/bin/g.region region=demo
                $GIS/bin/d.erase
                $GIS/bin/d.rast -o TPS.BATHY.NAD83
                $GIS/bin/d.rast -o chart13223
                $GIS/bin/d.rast -o mosaic
                $GIS/bin/d.vect map=COAST.NAD83 color=red
                $GIS/bin/d.vect map=bathy.5m.v color=blue
                xsonar &
                echo
                echo Please position the XSonar window
                GetReturn
                showimage &
                echo
                echo Please position the Showimage window
                GetReturn
                echo
                echo That\'s all for now!
                ;;
 esac
}


#*****************************
# Main Body
#*****************************

PrintInitialMsg
ContinueYorN
GetArch
CheckNeededUtilities
CheckCdMounted
GetCdDir
GetTopLevel
SetComponentDirs
MakeAllDirs
GetTotalDiskSpace
GetComponentsToInstall
InstallRtgisFiles
CompileRtgisSource
PrintEndMsg
UpdateLoginFile
TestRtgis
exit
```

3. System Tasks

There are several tasks which must often be performed prior to using the RTGIS Toolbox but which are imperative to efficient RTGIS Toolbox use. These tasks are typically System Administration tasks, such as mounting CD-ROM disks and formatting new survey data optical disks, and require super-user privileges. The step-by-step instructions in this section apply to the Linux operating system. The equivalents for other UNIX operating systems should be similar but may differ slightly in the command names and options. Be sure to review the system man page for all listed commands.

3.1 CD-ROM Disks

The CD-ROM drive is often used for chart import and data archiving. Before the CD-ROM disk may be used, it must be mounted. Also, the CD-ROM disk must be unmounted before it is physically removed from the CD-ROM drive.

3.1.1 Mounting a CD-ROM disk

1. Insert the CD-ROM disk into the CD-ROM drive.

2. Login as root or switch to the root user in a shell window using the following command:

   ```
   su
   ```

   You will be prompted for the root password. Enter it.

3. Mount the CD-ROM disk using the following command:

   ```
   mount -t iso9660 -r /dev/hdc /cdrom
   ```

   The -t option specifies the file system type on the CD-ROM. Most CD-ROMs are written in the iso9660 format so that they may be shared among platforms that support the iso9660 file system standard. If your CD-ROM has been recorded in a different native file system format, replace iso9660 with the appropriate file system type.

   The -r option indicates that the disk should be mounted read only. Some systems require this option to be given when mounting a CD-ROM disk.

   The /dev/hdc argument specifies the CD-ROM device special file. This will most likely be different for different platforms and system configurations. Make sure you specify the correct device special file for your CD-ROM drive.

   The /cdrom argument specifies the directory at which you want the CD-ROM disk mounted. This is typically the /cdrom (/CDROM on some systems) or /mnt directory. The mount directory can be any directory and is sometimes a data base directory. For example, a CD-ROM disk containing digital chart data is often mounted at a chart data base directory rather than at the /cdrom directory.

Note: While it is good practice to limit disk mounting capability to the super-user, regular users may be given mount permission on certain drives (such as the CD-ROM drive) by appropriately setting the permissions on the drive's device special files. The decision to allow user mounts is left up to the System Administrator.

Typical errors resulting from mounting the CD-ROM are due to specifying the wrong file system type, the wrong device special file, or an inappropriate mount directory. If an error results, begin debugging via the following steps. Check the CD-ROM file system type (it should be stated on the disk or on accompanying documentation) and be sure that you have specified it properly in the mount command. Double check the CD-ROM drive's device special file. This can usually be determined via the fdisk command with the list option (/sbin/fdisk -l) which will list all drive devices and their partition tables. Lastly, check that the mount directory exists, has proper permissions, and that no shells are currently using the directory as the current directory.

3.1.2 Un-mounting a CD-ROM disk

The CD-ROM disk **must** be unmounted before it is physically removed from the drive.

1. Login as root or switch to the root user in a shell window using the following command:

   ```
   su
   ```

   You will be prompted for the root password. Enter it.

2. Unmount the CD-ROM disk using the following command:

   ```
   umount /cdrom
   ```

   When unmounting a disk, you may do it by specifying the mount directory alone.

   The same notes and common errors from the Mounting a CD-ROM section apply here. A very common error is to attempt to unmount a mount directory which is the current working directory. Change directories to a directory other than the mount directory and retry the umount command.

3. Remove the CD-ROM disk from the CD-ROM drive.

3.2 Optical Disks

The RTGIS Toolbox workstation is commonly equipped with an optical disk drive. The drive can accommodate 1.2 gigabyte optical disks. The optical disks are useful as survey data disks, general off-line data storage, and extra disk and swap space. Before an optical disk can be used, it must be partitioned, a file system must be written to it, and it must be mounted.

3.2.1 Partitioning an optical disk

THIS PROCEDURE WILL ERASE ALL DATA ON THE DISK.

BE SURE YOU ARE USING AN EMPTY DISK.

THIS OPERATION APPLIES TO ONLY ONE SIDE OF THE DISK (THE SIDE THAT IS UP).

1. Login as root or switch to the root user in a shell window using the following command:

   ```
   su
   ```

You will be prompted for the root password. Enter it.

2. Insert the new optical disk to be partitioned into the optical disk drive.

   See the warnings above. All data on the disk will be erased during the partitioning procedure.

3. View the current disk device and partition table by entering the following command:

   ```
   /sbin/fdisk -l
   ```

   Note the optical disk drive's device special file in the output. You will need this later.

4. Start the fdisk utility to edit the optical disk's partition table by entering the following command:

   ```
   /sbin/fdisk /dev/sdb
   ```

   The /dev/sdb argument is the optical disk drive's device special file name determined in step 3. This name will be different for different platforms and system configurations. Be sure you use the correct device special file name.

5. Enter `m` at the fdisk prompt and familiarize yourself with the available commands.

6. Enter `n` at the fdisk prompt to create a new partition on the optical disk.

   This procedure applies to creating a data disk containing a single primary partition. To create a user disk with multiple primary and extended partitions, read the fdisk man page and consult the Linux System Administrator's Guide.

7. Enter `p` at the fdisk prompt to create a primary partition.

8. Enter `1` at the fdisk prompt to created primary partition 1.

9. Enter `1` to start the partition at the first disk cylinder.

10. Enter the last cylinder number (as shown on the fdisk command line) to use the entire disk for the partition.

11. Enter `p` at the fdisk prompt to view the new partition table.

    Be sure the newly created partition is listed.

12. Enter `w` at the fdisk prompt to save the partition table and exit fdisk.

The disk is now ready to hold a file system on partition 1.

Note: This procedure only needs to be done once. After data has been written to the disk, the data may be accessed by simply mounting the disk.

3.2.2 Making a file system on an optical disk

THIS PROCEDURE WILL ERASE ALL DATA ON THE DISK.

BE SURE YOU ARE USING AN EMPTY DISK.

THIS OPERATION APPLIES TO ONLY ONE SIDE OF THE DISK (THE SIDE THAT IS UP).

1. Login as root or switch to the root user in a shell window using the following command:

   ```
   su
   ```

   You will be prompted for the root password. Enter it.

2. Insert the partitioned optical disk into the optical disk drive.

   See the warnings above. All data on the disk will be erased during the file system writing procedure.

3. View the partition table using the following command:

   ```
   /sbin/fdisk -l
   ```

4. Note the partitions to which are to be written a file system.

5. Make a file system on the partition using the following command:

   ```
   /sbin/mkfs -t ext2 /dev/sdb1
   ```

   The -t option specifies the type of file system to be written to the partition. Ext2 is the Linux native file system type. To create a file system of a different type, refer to the mkfs man page and consult the Linux System Administrator's Guide.

   The /dev/sdb1 argument specifies the disk and partition to which to write the file system. This command follows from the optical disk partitioning procedure above where /dev/sdb1 indicates primary partition 1 on the optical disk drive. If you are writing file systems to a different partition table, be sure to use the proper partition names as listed by the fdisk command in step 3.

The disk is now ready to be mounted and used as a normal disk drive.

Note: This procedure only needs to be done once. After data has been written to the disk, the data may be accessed by simply mounting the disk.

3.2.3 Mounting an optical disk drive

In order to mount an optical disk drive, it must contain a valid file system created by the optical disk partitioning and file system writing procedures above. The disk side that is up will be the accessed side. The drive can only access one side at a time. To access the other side, the current side must first be unmounted, then the disk may be physically flipped and the other side may be mounted.

1. Insert the optical disk into the optical disk drive.

2. Login as root or switch to the root user in a shell window using the following command:

   ```
   su
   ```

   You will be prompted for the root password. Enter it.

3. Mount the optical disk using the following command:

   ```
   mount -t ext2 /dev/sdb1 /mnt
   ```

The -t option specifies the file system type on the optical disk partition. This example is specific to the Linux native file system format, ext2. To mount an optical disk partition containing a different file system type, review the mount man page and consult the Linux System Administrator's Guide.

The /dev/sdb1 argument is the device special file name for the optical disk file system to be mounted. This will most likely be different for different platforms and system configurations. Make sure you specify the correct device special file for your optical disk drive and partition.

The /mnt argument specifies the directory at which you want the optical disk mounted. This is typically the /mnt directory but can be any directory and is sometimes a data base directory. For example, an optical disk partition containing raw side scan data is often mounted at a side scan data base directory rather than at the /mnt directory.

Note: While it is good practice to limit disk mounting capability to the super-user, regular users may be given mount permission on certain drives by appropriately setting the permissions on the drive's device special files. The decision to allow user mounts is left up to the System Administrator.

Typical errors resulting from mounting optical disk partitions are due to specifying the wrong file system type, the wrong device special file, or an inappropriate mount directory. If an error results, begin debugging via the following steps. Check the optical disk partition file system type (using the `/sbin/fdisk -l` command) and be sure that you have specified it properly in the mount command. Double check the optical disk partition's device special file. This can usually be determined via the fdisk command with the list option (/sbin/fdisk -l) which will list all drive devices and their partition tables. Lastly, check that the mount directory exists, has proper permissions, and that no shells are currently using the directory as the current directory.

3.2.4 Un-mounting an optical disk drive

The optical disk **must** be unmounted before it is physically removed from the drive.

1. Login as root or switch to the root user in a shell window using the following command:

   ```
   su
   ```

   You will be prompted for the root password. Enter it.

2. Unmount the optical disk partition using the following command:

   ```
   umount /mnt
   ```

   When unmounting a disk, you may do it by specifying the mount directory alone.

   The same notes and common errors from the "Mounting an optical disk drive" section apply here. A very common error is to attempt to unmount a mount directory which is the current working directory. Change directories to a directory other than the mount directory and retry the umount command.

3. Remove the optical disk from the optical disk drive.

4. RTGIS Toolbox Basics

There are several RTGIS Toolbox procedures that are performed very frequently for multiple survey phases. For example, starting a GRASS display monitor and adjusting the geographic region. These procedures are not specific to any phase of the survey and are grouped here as RTGIS Toolbox Basics.

4.1 Starting and Stopping the RTGIS Toolbox

The RTGIS Toolbox and all component software is owned by the user grass. GRASS's environment has been customized for running the RTGIS Toolbox. In order to use the RTGIS Toolbox, the user must login as user *grass*.

4.1.1 Starting the RTGIS Toolbox

1. Boot the RTGIS Toolbox workstation.

   If the RTGIS workstation is a dual operating system platform running the LILO boot loader, such as those used by OMDC installations, at the LILO prompt, press and release the left <Shift> key once. Then type `linux<Enter>`. This will cause the Linux operating system to boot.

2. At the login screen, enter 'grass' for the user.

3. When prompted for the password, enter 'rt_grass'

   A grass user session will start and the command prompt will appear.

3. Start X Window by entering the following command:

   ```
   startx
   ```

   The X Window system will start. At the upper left corner of the desktop will be a System Tool Chest. One of the menu options will be "RTGIS".

4. Select the **RTGIS** button on the System Tool Chest menu.

   The outline of a shell window will appear.

5. Click the left mouse button on the desktop. This will place the upper left corner of the shell window.

   The shell window runs the rtgrass program. It loads some parameters specific to real time operations and then starts GRASS. The shell window is known as the GRASS terminal window.

6. Complete the GRASS LOCATION, MAPSET, and DATABASE entry fields in the GRASS terminal window.

   If this GRASS session is being started for a new survey project, determine an appropriate name for the survey. The survey name should be entered as the MAPSET in the proper LOCATION.

7. Hit the <Esc> key.

GRASS will be started in the GRASS terminal window and the RTGIS Main Menu will appear at the upper right of the screen.

If a new MAPSET has been entered, GRASS will ask if the MAPSET should be created. Respond in the affirmative.

4.1.2 Stopping the RTGIS Toolbox

When quitting and RTGIS Toolbox session, the GRASS terminal window and the RTGIS Main Menu must both be stopped separately.

1. Select the **Quit** button on the RTGIS Main Menu.

   This will terminate the RTGIS Main Menu and any open RTGIS Toolbox tools.

2. Exit GRASS by entering the following command in the GRASS terminal window:

   ```
   exit
   ```

   If GRASS prompts for the MAPSET to be deleted, respond with 'n' for "no". This is the default. If GRASS prompts for interactive deletion of database files, respond with 'n' for "no". This is the default.

4.1.3 Programmer's Notes

The RTGIS Toolbox system developer may chose to allow any user to run RTGIS. This will require careful attention to the UNIX file permissions on all executable programs and data files. Presumably, user grass will still own all the RTGIS Toolbox system software. However, other users may use the RTGIS system software if they are given the proper permissions.

In addition, if a user other than grass is to run RTGIS Toolbox software, the user's environment must be modified to resemble the grass user environment.

4.2 RTGIS Toolbox Demo

The purpose of the RTGIS Toolbox Demo is to introduce a new user to the capabilities of the RTGIS Toolbox by guiding him or her through several RTGIS Toolbox procedures. The Demo must be used along with the RTGIS Toolbox User's Manual. The User's Manual provides step-by-step instructions for performing the procedures presented by the Demo.

4.2.1 Using the RTGIS Toolbox Demo

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.

2. Select the **Demo** menu button on the RTGIS Main Menu.

   Use the left mouse button. A Decision Dialog box will appear and ask if you really want to start the Demo.

3. Select the **Yes** button on the Decision Dialog box.

   The Demo interface will appear and you will be presented with the first page of the Demo.

   The Demo interface consists of three frames. The top frame, the title frame, shows the title of the current Demo section. The bottom frame, the action frame, contains action buttons for interacting with the Demo. The middle frame, the task frame, contains the text for the current Demo task.

   There are 5 buttons in the action frame. The **Done** button, in the middle, is used to quit the Demo at any time. The remaining 4 buttons are used to step through the Demo. The double arrow buttons (<< and >>) step to the previous and next Demo section, respectively. The single arrow buttons (< and >) step to the previous and next task, respectively. For each task, only the valid action buttons will be active.

   Within the task frame, the first line is printed in blue and gives the name of the current task. There may be multiple pages for a given task. Black text is used for discussions. Red text indicates a specific action that the user should perform.

You are now ready to use the RTGIS Toolbox Demo. The Demo is most effective when used in its entirety, in the order presented. The Demo may be performed in multiple sessions. Use the arrow buttons in the action frame to step to the proper demo task.

4.3 GRASS Display Monitor Management

The GRASS display monitor is the display entity used for display of all geographic data, including imported survey data. Since it is the "window" into the GRASS database, it is an integral component of all RTGIS Toolbox GIS and survey operations. A graphical user interface has been written for managing GRASS display monitors. It is called the GRASS Displays Tool.

4.3.1 Starting the GRASS Displays Tool

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.
2. Select the **GIS Tools** pull down menu.

   Use the left mouse button or the <alt>g hot key.

3. Select the **Display environ.** cascade menu.

   Use the left mouse button or the i hot key.

4. Select the **Start-stop-select displays** button.

   Use the left mouse button or the s hot key. The GRASS Displays Tool will appear on your screen.

5. When finished using the GRASS Displays Tool, quit it by selecting the **Done** button.

4.3.2 Starting a GRASS display monitor

1. Start the GRASS Displays Tool

2. Select the display monitor(s) to start from the listbox in the GRASS Displays Tool.

3. Select the **Start** checkbutton.

4. Select the **OK** button.

   The selected monitor will start and appear at the upper left corner of the screen.

5. When finished using the GRASS Displays Tool, quit it by selecting the **Done** button.

4.3.3 Selecting a GRASS display monitor

1. Start the GRASS Displays Tool

2. Select the display monitor to become the currently active GRASS display monitor from the listbox in the GRASS Displays Tool.

3. Select the **Select** checkbutton

4. Select the OK button.

   The selected GRASS monitor will become the current GRASS monitor which all following display commands will reference until another GRASS display monitor is selected, either via the command line, another GRASS program, or the GRASS Displays Tool.

5. When finished using the GRASS Displays Tool, quit it by selecting the **Done** button.

4.3.4 Stopping a GRASS display monitor

1. Start the GRASS Displays Tool.

2. Select the GRASS display monitor to be stoped from the listbox in the GRASS Displays Tool.

3. Select the **Stop** checkbutton.

4. Select the OK button.

   The selected GRASS display monitor will be terminated and will disappear from the screen.

5. When finished using the GRASS Displays Tool, quit it by selecting the **Done** button.

4.3.5 Listing the status of the GRASS display monitors

1. Start the GRASS Displays Tool.

2. Select the **List** button.

   A text list will be printed to the GRASS terminal window showing the status of each available GRASS display monitor.

3. When finished using the GRASS Displays Tool, quit it by selecting the **Done** button.

4.4 GRASS Region Management

GRASS defines the geographic characteristics of its display monitors by a "region." The GRASS region contains the geographic boundary (north, south, east, and west values), the resolution (both north-south and east-west), and projection information. GRASS can have only one region active at a time. A consequence of this is frequent region changing by the GRASS user. To simplify GRASS region modification, a graphical user interface was created for selecting and modifying the GRASS region. The interface is called the Region Manager Tool.

4.4.1 Starting the Region Manager Tool

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.
2. Select the **GIS Tools** pull down menu.

   Use the left mouse button or the <alt>g hot key.
3. Select the **Region Manager** button.

   Use the left mouse button to the r hot key.

4.4.2 Setting the current GRASS region manually

1. Start the Region Manager Tool.
2. Select the **Set manually** check button.
3. Select the entry field to edit.

   The **North**, **South**, **East**, **West**, **N-S Res**, and **E-W Res** values may be edited. After editing a value, hit the <Enter> key. This will cause the other values to be adjusted, if needed, to reflect the change. For example, if the North value is changed, the N-S Res value will be automatically updated to reflect the change.
4. When the region has been satisfactorily set, select the **Apply** button to apply the changes and leave the Region Manager Tool open for further region changes or select the **OK** button to apply the changes and close the Region Manager Tool.

   Note: At any time, the **Cancel** button may be selected to close the Region Manager Tool and restore the GRASS region to its value upon starting the Region Manager Tool.

4.4.3 Setting the current GRASS region graphically

1. Start the Region Manager Tool.
2. Select the **Set graphically** check button.

   This will launch the d.zoom.inout program and allow interactive zooming of the current GRASS region. A dialog box will appear at the lower left corner of the GRASS display monitor asking if you want to zoom in or zoom out.

3. Select the "zoom in" box to zoom in or the "zoom out" button to zoom out.

   If you choose to zoom in, you must define the new region on the current GRASS display monitor. If you choose to zoom out, a small map will be drawn in a box at the upper right corner of the current GRASS display monitor. The map will be in the MAPSET's default region. You must define the new region on this map. The new region is defined by specifying the corners of a rubber band box.

4. Place the mouse cursor at the lower left corner of the new region and click the left mouse button once.

   If you make an error, simply reselect the point using the same process.

5. Move the mouse cursor to the upper right corner of the new region and click the right mouse button once.

   As you drag the mouse cursor to the upper right corner, you should see a rubber band box following the cursor. After selecting the upper right corner of the new region, a dialog box will appear at the lower left of the GRASS display monitor.

   If the region has been properly outlined, select the YES box to accept the region.

   If an error has been made in outlining the region, select the NO box to try again.

6. When the region has been satisfactorily set, select the **Apply** button to apply the changes and leave the Region Manager Tool open for further region changes or select the **OK** button to apply the changes and close the Region Manager Tool.

   Note: At any time, the **Cancel** button may be selected to close the Region Manager Tool and restore the GRASS region to its value upon starting the Region Manager Tool.

4.4.4 Setting the current region from the default region

1. Start the Region Manager Tool.

2. Select the **Set from default** check button.

   The default region values will appear in the entry boxes and may then be edited manually or accepted.

3. When the region has been satisfactorily set, select the **Apply** button to apply the changes and leave the Region Manager Tool open for further region changes or select the **OK** button to apply the changes and close the Region Manager Tool.

   Note: At any time, the **Cancel** button may be selected to close the Region Manager Tool and restore the GRASS region to its value upon starting the Region Manager Tool.

4.4.5 Setting the current region from a window file

A window file is a database file in the windows subdirectory of the current GRASS MAPSET. The window file is a text file which contains the projection, boundary, and resolution information to completely define the GRASS region.

1. Start the Region Manager Tool.
2. Select the **Set from window file** button.

   A file selection dialog will appear.
3. Select the window file from which to set the GRASS region.
4. Select the **OK** button on the file selection dialog.
5. When the region has been satisfactorily set, select the **Apply** button to apply the changes and leave the Region Manager Tool open for further region changes or select the **OK** button to apply the changes and close the Region Manager Tool.

   Note: At any time, the **Cancel** button may be selected to close the Region Manager Tool and restore the GRASS region to its value upon starting the Region Manager Tool.

4.4.6 Setting the current region from a site data file

A site data file is GRASS-format data file containing scalar information (an x coordinate, a y coordinate, and a value). For example, sediment core locations. When setting the region from a site data file, the region boundaries will be set to the maximum and minimum x and y coordinates from the chosen site data file.

1. Start the Region Manager Tool.
2. Select the **Set from site file** button.

   A file selection dialog will appear.
3. Select the site file from which to set the GRASS region.
4. Select the **OK** button on the file selection dialog.
5. When the region has been satisfactorily set, select the **Apply** button to apply the changes and leave the Region Manager Tool open for further region changes or select the **OK** button to apply the changes and close the Region Manager Tool.

   Note: At any time, the **Cancel** button may be selected to close the Region Manager Tool and restore the GRASS region to its value upon starting the Region Manager Tool.

4.4.7 Setting the current region from a vector data file

A vector data file is a GRASS-format data file which contains vector information (lists of x,y coordinate pairs which define the vertices of lines). For example, a coast line. When setting the region from a vector data file, the region boundaries will be set to the maximum and minimum x and y coordinates from the chosen vector data file.

1. Start the Region Manager Tool.
2. Select the **Set from vector file** button.

   A file selection dialog will appear.
3. Select the vector file from which to set the GRASS region.
4. Select the **OK** button on the file selection dialog.
5. When the region has been satisfactorily set, select the **Apply** button to apply the changes and leave the Region Manager Tool open for further region changes or select the **OK** button to apply the changes and close the Region Manager Tool.

   Note: At any time, the **Cancel** button may be selected to close the Region Manager Tool and restore the GRASS region to its value upon starting the Region Manager Tool.

4.4.8 Setting the current region from a raster data file

A raster data file is a GRASS-format data file which contains raster information (a 2-dimensional matrix of data values geographically located by and accompanying header file). For example, a sidescan sonar mosaic. When setting the region from a raster data file, the region boundaries and resolution will be set to that defined by the chosen raster file's header file.

1. Start the Region Manager Tool.
2. Select the **Set from raster file** button.

   A file selection dialog will appear.
3. Select the raster file from which to set the GRASS region.
4. Select the **OK** button on the file selection dialog.
5. When the region has been satisfactorily set, select the **Apply** button to apply the changes and leave the Region Manager Tool open for further region changes or select the **OK** button to apply the changes and close the Region Manager Tool.

   Note: At any time, the **Cancel** button may be selected to close the Region Manager Tool and restore the GRASS region to its value upon starting the Region Manager Tool.

5. Pre-Cruise Planning

This section describes the RTGIS Toolbox tools which assist in pre-cruise planning.

5.1 Data Browser

The Data Browser provides a graphical user interface for selecting GRASS data layers for display. GRASS site, vector, and raster files may be displayed. Although presented here as a survey planning tool, the Data Browser may be used at any time to display data layers to a GRASS display monitor.

5.1.1 Starting Data Browser

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.
2. Select the **GIS Tools** menu button on the RTGIS Main Menu.
3. Select the **Data Browser** menu button.

   The Data Browser tool will start.

5.1.2 Selecting a GRASS display monitor using Data Browser

The Data Browser interface allows you to select the GRASS dislay monitor on which data will be displayed.

1. If Data Browser is not already running, start Data Browser.
2. Use the left mouse button to select the **Display monitor:** option menu tab.

   A list of GRASS display monitors will appear.
3. Use the left mouse button to select the desired GRASS display monitor.

   The selected GRASS display monitor will become the current GRASS display montior used in subsequent drawing commands.

5.1.3 Erasing the selected GRASS display monitor using Data Browser

The Data Browser interface allows you to erase the current GRASS dislay monitor.

1. If Data Browser is not already running, start Data Browser.
2. Use the left mouse button to select the **Erase** button.

   The current GRASS display monitor will be erased.

5.1.4 Drawing a GRASS raster data layer

1. If Data Browser is not already running, start Data Browser.

2. Select a display monitor (section 5.1.2), only if you wish to draw to a GRASS display monitor other than the current display monitor.

3. Use the left mouse button to scroll through the **Raster layers** list at the left of the Data Browser.

4. When you find the raster file to display, select it with the left mouse button.

5. Set the Overlay option button.

   If the selected raster file is to be overlayed on top of other data, this button should be selected so that it is highlighted yellow. Overlay is the default and is most often the desired raster drawing mode.

6. Select the **Show** button beneath the **Raster layers** list at the left of the Data Browser.

   The selected raster file will be displayed.

5.1.5 Drawing a GRASS vector data layer

1. If Data Browser is not already running, start Data Browser.

2. Select a display monitor (section 5.1.2), only if you wish to draw to a GRASS display monitor other than the current display monitor.

3. Use the left mouse button to scroll through the **Vector layers** list at the middle of the Data Browser.

4. When you find the vector file to display, select it with the left mouse button.

5. Use the left mouse button to select the color option menu tab.

   A list of color names will appear. The color defines the color in which the vector data will be drawn.

6. Use the left mouse button to select the desired color for the selected vector data.

7. Select the **Show** button beneath the **Vector layers** list at the middle of the Data Browser.

   The selected vector file will be displayed in the selected color.

5.1.6 Drawing a GRASS sites data layer

1. If Data Browser is not already running, start Data Browser.

2. Select a display monitor (section 5.1.2), only if you wish to draw to a GRASS display monitor other than the current display monitor.

3. Use the left mouse button to scroll through the **Site layers** list at the right of the Data Browser.

4. When you find the site file to display, select it with the left mouse button.

5. Use the left mouse button to select the color option menu tab in the site **Options** frame.

   A list of color names will appear. The color name defines the color in which the site data will be drawn.

6. Use the left mouse button to select the desired color for the selected site data.

7. Use the left mouse button to select the size option menu tab in the site **Options** frame.

   A list of numbers will appear. The number defines the size of the site symbol. A value of 1-3 is usually large enough.

8. Use the left mouse button to select the desired size for the selected site data.

9. Use the left mouse button to select the symbol option menu tab in the site **Options** frame.

   A list of symbol names will appear. The symbol name defines the symbol in which the site data will be drawn.

10. Use the left mouse button to select the desired symbol for the selected site data.

11. Select the **Show** button beneath the **Site layers** list at the right of the Data Browser.

    The selected site file will be displayed.

5.1.7 User's Notes

Data Browser simply lists the contents of the cell, dig, and site_lists directories without doing a verification of the validity of the files it finds. Some files listed may not actually be GRASS data layers if the database is not a strict GRASS database. Selecting such files for viewing will result in errors.

Data Browser can only read MAPSETs owned by the current user. If there appear to be missing data layers, they are not in a MAPSET that you own.

Data Browser does not actually check for file compression, it just looks for a ".Z" or ".gz" extension and assumes the file is appropriately compressed. Attempting to uncompress a file that is not compressed will result in errors.

Data Browser does not check that the selected GRASS monitor has been started. Attempting to display to a monitor that has not been started will result in an error.

5.2 Coverage Viewer

The Coverage Viewer interface and operation is very similar to Data Browser. However, it is significantly unique to the Data Browser. Coverage Viewer operates on raw survey data files. It will read through raw navigation, depth, and sidescan sonar files and show the raw data coverage. In addition, GRASS raster, vector, and site data files are created on-the-fly which show the raw data coverage.

Coverage Viewer is very useful for viewing raw data coverage and planning surveys during daily cruise operations when there may not be sufficient time for fully processing the data between cruises.

5.2.1 To start Coverage Viewer

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.
2. Select the **GIS Tools** menu button on the RTGIS Main Menu.
3. Select the **Coverage Viewer** menu button.

   The Coverage Viewer tool will start.

5.2.2 User's Notes

Coverage Viewer does not have its own interface for selecting the current GRASS display monitor. This should be done with Data Browser. If a GRASS display monitor has not been selected with Data Browser or the d.mon command line program, an error will result when attempting to view data coverage.

5.2.3 Programmer's Notes

The Tcl/Tk script coverage_viewer.tcl provides the interface for this module. The sidescan coverage widgets provide an interface to the C program d.slscover. The procedure for creating vector files from GRASS library routines and then raster files from the newly created vector files is quite obscure. Refer to the code in the d.slscover source file draw.c to view the working procedure. It must follow these steps:

1. Open the vector file using G_open_vect_new
2. Initialize the Points structure
3. Create an array of x y points defining a line
4. Convert the x y points array to a Points structure
5. Write the line to the file
6. When all lines have been written to the file, close the vector file using G_vect_close
7. Create the dig_att file with the lines and areas to convert to raster
8. Run v.build to create the dig_plus file. This file is needed in order for the raster to be created properly.
9. Run v.to.rast. This can be done using the v.to.rast program via a system call or by calling the vect_to_rast function by including the source code from the v.to.rast program.

5.3 Coordinate Conversion

The GRASS display commands and analysis commands consistently support only one projection; Universal Transverse Mercator (UTM). In dealing with survey data collected aboard ship using GPS navigation, the majority of our positioning data is in some form of latitude/longitude pairs. Hence, there is often the need to convert between coordinate systems. Most programs do the conversion automatically. However, it is often useful to be able to manually convert a position. For this reason, the RTGIS Toolbox contains a Coordinate Conversion tool which allows the user to convert between latitude/longitude coordinates and UTM easting/northing coordinates.

5.3.1 Using the Coordinate Conversion tool

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.

2. Select the **GIS Tools** menu button on the RTGIS Toolbox Main Menu.

3. Select the **Coordinate Conversion** menu button.

   Use the left mouse button or the o hot key. The **Lat/lon - UTM Conversion** tool will start.

4. Enter the desired spheroid if different than the default of wgs84.

   This must be a name supported by GRASS. When working with GPS data and NOAA chart data, wgs84 is most often the proper spheroid.

5. Enter the UTM zone number if different that the default of 19.

6. Enter the x and y values of the position to convert.

   The conversion may go either way; from lat/lon to UTM or from UTM to lat/lon.

   Lat/lon values must be entered in GRASS lat/lon notation. An example is given on the **Lat/lon - UTM Conversion** tool. Easting/Northing pairs must be entered as floating point values of meters.

7. If the position is located in the southern hemisphere, select the **Southern Hemisphere** check button with the left mouse button.

   The check box should become highlighted yellow.

8. Select the **Convert** button.

   The converted coordinates will appear in the appropriate boxes.

   Hint: If you must use the converted values in another program currently awaiting input in another window, you may use the left mouse button to highlight the value in the appropriate text box. This will load the data into the copy buffer. Then use the middle mouse button (the left and right mouse buttons pressed simultaneously on two-button mice) to paste the highlighted value in the other window. This saves the inconvenience of having to type large amounts of numbers and possibly making errors.

9. Repeat steps 5-7 for each position to convert.

10. When done, select the **Done** button to dismiss the **Lat/lon - UTM Conversion** widget.

5.4 Line Planning

The RTGIS Toolbox Line Planning tool allows the user to create a set of lines which define the desired vessel path. There is a special tool for generating a series of equally-spaced parallel lines for running swath mapping survey lines. When generating suvey lines, keep in mind that you are not just laying out the swath mapping lines, but the entire survey path. This means that you must include lines which define the transit path between the main data collecting survey lines.

5.4.1 Creating a new set of survey lines

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.

2. Select the **Survey Tools** menu button on the RTGIS Main Menu.

3. Select the **Line Planning** menu button.

   The **Line Planning Utility** will start.

4. Move the mouse cursor so that it is in the **Track name** text entry box.

   The text entry box will become highlighted and a cursor will appear.

5. Type in the desired track name, press <Enter> when finished.

   The **Line #** text entry box will become highlighted.

6. Type in the number 1, press <Enter>.

   The First waypoint **Lat** text entry box will become active.

7. Enter the first waypoint.

   This can be done manually if you have exact coordinates for the waypoint. The waypoint's latitude and longitude values must be entered in GRASS lat/lon notation. See the Coordinate Conversion tool in section 5.5 if you do not know this notation.

   To enter the waypoint graphically, select the **mouse** check button in the **First waypoint** frame. Then click the left mouse button once in the GRASS display monitor at the desired location of the waypoint. The coordinates of the selected point will be entered into the **First waypoint** frame.

8 Choose the method of entry for the second waypoint by using the left mouse button to select the **mouse** or **keyboard** check button in the **Second waypoint** frame.

9. Enter the second waypoint using the same procedure as in step 7.

10. Select the **Accept waypoints** button.

    The waypoints will be added to the waypoint list at the right of the **Track Planning Utility**. The **line #** becomes incremented by 1 and the second waypoint of the previous line becomes the first waypoint of the new line.

11. Select the **Write** menu button on the **Track Planning Utility** menu.

12. Select the **Write waypoints to GRASS site file** menu button.

13. Select the **Display** menu button on the **Track Planning Utility** menu.

14. Select the **View current waypoints** menu button.

    The current waypoints will be marked on the GRASS display monitor with red diamonds. Waypoint numbers will also be shown.

15. Continue entering line waypoints until you have entered all needed lines.

    Note: The lines should define the survey vessel's complete path, not just the main data collecting lines. It is for this reason that when the **Accept waypoints** button is chosen, the first waypoint for the next line is automatically set to the second waypoint of the previous line. After the first line has been entered, you should only have to enter the second waypoint for the remaining lines.

    Note: The **Accept waypoint** button must be selected to register each line.

16. After all waypoints have been created, repeat steps 11 and 12.

    All waypoints will now be displayed on the current GRASS display monitor.

17. Select the **Write** menu button on the **Track Planning Utility** menu.

18. Select the **Write waypoints to GRASS vector file** menu button.

19. Select the **Display** menu button on the **Track Planning Utility** menu.

20. Select the **View track** menu button.

    The created track will be displayed on the current GRASS display monitor in red.

21. When the track has been successfully made, select the **Write** menu button on the **Track Planning Utility**.

22. Select the **Write waypoints to Maptech ASCII file** menu button.

    The track will be written to a file in the maptech directory of the current GRASS MAPSET. This file can be ftped or copied to the Maptech helm guidance computer. Follow the Maptech documentation for importing the file as a Maptech route.

5.4.2 Creating a series of equally-spaced parallel survey lines

A survey plan often consists of a series of equally-spaced parallel lines. Such a set of survey lines is easily generated by the RTGIS Toolbox.

1. Follow steps 1-10 in section 5.6.1 to define the first survey line.

   This line will be the base line from which the parallel lines will be generated. In this case, be sure to select the waypoints in the order in which they are to be run.

2. Select the **Lines** menu button on the **Track Planning Utility** menu.

3. Select the **Create offset** button.

4. After the **Create** label, enter the number of parallel lines to generate.

5. Use the left mouse button to select the **left** check button or the **right** check button to determine the side of the base line to which the lines should be created.

   In chosing the side (left or right) remember that the orientation of the line proceeds from waypoint 1 to waypoint 2, as you previously selected them.

6. After the **Line #** label, enter the line number which will be the base line.

   This is usually line number **1**, but may be any line number previously created.

7. After the **at** label, enter the line spacing.

   This should be a value in meters.

8. Select the **Go** button to generate the waypoints.

9. After the waypoints have been generated, follow steps 18-24 from section 5.6.1 to view and output the created track.

5.4.3 Refreshing the line planning display monitor

During the line planning process, several mistakes may be made before you arrive at the proper set of lines. The result may be a clutered display monitor with unwanted sets of previous tracks and waypoints. The Track Planning Utility provides the ability to clear and redraw the current GRASS display monitor.

1. Select the **Background** menu button on the **Track Planning Utility** menu.

2. Select the **Refresh** menu button.

   If this fails, or you do not like the data layers that are drawn by default, select the **Background** menu button again and then select the **Edit** menu button.

   The outline of an xedit window will appear. Place the window. Enter the set of GRASS drawing commands to indicate the data layers you would like drawn when refreshing. For example, to draw the NOAA chart 13223, enter the following line:

   d.rast -o chart13223

   Select the **Save** button on the xedit window. Select the **Quit** button on the xedit window. Then refresh the GRASS display monitor again.

6. Data Acquisition

The RTGIS Toolbox performs the majority of its data acquisition via serial interfaces to devices. The sole exception is multibeam sonar which is acquired via ethernet network. Standard graphical user interfaces are provided for performing data logging tasks.

The procedures outlined in this section are true to their description; they provide only data acquistion. If you surveyed with the tools in this section only, you would have to take it on faith that valid data was actually being logged. Clearly, such a practice is not acceptable. To conduct an efficient survey, the tools presented here must be run along with the tools presented in the next section, Survey Monitoring. The survey monitoring tools provide text and graphical displays of the data being logged for the purpose of quality assurance and quality control.

6.1 Preparing for data Acquisition

Before beginning data acquisition, there are two general tasks to accomplish. The first is to prepare all data collection instruments and connect their outputs to the RTGIS Toolbox workstation. The second is to prepare the RTGIS Toolbox software for data acquisition. This User's Manual only discusses the later task. It is assumed that all instruments to be logged (e.g., sidescan sonar system and navigation receiver) have been properly configured and connected to the RTGIS Toolbox workstation using the appropriate operating manuals supplied with each instrument.

6.1.1 Creating a new survey mapset

The GRASS "MAPSET" is a directory in the GRASS database which groups thematic data. The MAPSET is beneath the LOCATION in the GRASS database hierarchy. The LOCATION is, in turn, beneath the DATABASE (the top level directory in the GRASS database hierarchy). The RTGIS Toolbox uses the MAPSET level of the GRASS database to group data from a particular survey or survey location. Hence, when beginning a new survey, or the first data collection cruise in a survey areay, you must create a new GRASS MAPSET. This is done when the RTGIS Toolbox is first started.

1. Start the RTGIS Toolbox.

   The RTGIS Terminal window will start. In the terminal window, there are three input lines which tell GRASS which LOCATION, MAPSET, and DATABASE to use. The top input line is the LOCATION line. This line is usually left the same for regional surveys.

2. Press the <Enter> key to skip to the next input line.

   This input line, the second, is the MAPSET line.

3. Enter a descriptive name for the new survey MAPSET.

   For example, if you are surveying around an island called Dutch Island, it would make sense to call the new MAPSET "dutch." If you are conducting the survey for the Department of Ocean Protection, you may decide to call the new MAPSET "DOP_dutch." If you are a new RTGIS Toolbox user and this is a practice MAPSET, you may call the new MAPSET by your own name. The MAPSET may be given any name but should be descriptive of its contents.

Note: When entering names for the MAPSET, LOCATION, and DATABASE, if the new name is shorter than the old one, you may notice extra text left after your new name. You must use the **space bar** to "white out" the left over text.

The last line is the DATABASE name and is rarely changed.

4. Press the <Esc> key to accept the changes and continue.

   You will be asked if you would like to create the new MAPSET.

5. Press the 'y' key and then <Enter> to create the new MASPET.

   Later, when you exit the RTGIS Toolbox, you will be aksed if the new MAPSET should be saved. You should answer yes.

6.1.2 Creating needed files and directories

After creating a new survey MAPSET and before collecting data, certain default files and directories must be created in the MAPSET. A tool will automatically create all required files and directories for you.

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.

2. Select the **Survey Tools** menu button on the RTGIS Main Menu.

   Use the left mouse button or the <alt>s hot key. The Survey Tools pull down menu will appear.

3. Select the **New survey MAPSET** menu button.

   All required files and directories will be made. This only needs to be done once for each new MAPSET.

   Note: If you are not sure if you need to do this, it is not harmful to run again. Existing files and directories will not be modified.

6.2 Navigation Acquisition

The RTGIS Toolbox provides for navigation acquisition via several devices. Position, course-over-ground, and speed-over-ground are acquired via a serial interface to the Magnavox 4200D differential GPS receiver. Gyro heading is acquired via a serial interface to the ship's gyro compass output. Magnetic heading and speed through the water are acquired via a serial interface to the Datamarine Link 6000 magnetic compass and paddle wheel transducer, respectively.

6.2.1 Starting Magnavox 4200D DGPS data acquisition

This procedure assumes that the Magnavox 4200D is turned on, operating properly, and the NMEA output port has been properly connected to one of the serial ports on the RTGIS Toolbox workstation.

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.

2. Select the **Survey Tools** menu button on the RTGIS Main Menu.

Use the left mouse button or the <alt>s hot key. The Survey Tools pull down menu will appear.

3. Select the **Serial data utilities** cascade menu

4. Select the **Set device communications** menu button.

   Use the left mouse button or the e hot key. The setdevices tool will appear.

   Note: Steps 4 - 18 only need to be performed if you add a new device or you change the port that a device is usually connected to. The device communications are stored in a defaults file so that they do not need to be reset for each survey if they are used in identical configurations.

5. Use the left mouse button to select **MX4200D** from the device list on the setdevices tool.

   The name MX4200D will appear in the device text box.

6. Use the left mouse button to select the **port #:** text entry box.

   The text entry box will become highlighted and a cursor will appear in the box.

7. Enter the port number to which the Magnavox 4200D NMEA output port is connected.

8. Use the left mouse button to select the **baud:** text entry box.

   The text entry box will become highlighted and a cursor will appear in the box.

9. Enter the baud rate of the serial output.

   This should be 4800 for the Magnavox 4200D.

10. Use the left mouse button to select the **data bits:** text entry box.

    The text entry box will become highlighted and a cursor will appear in the box.

11. Enter the number of data bits in the serial output.

    This should be 8 for the Magnavox 4200D.

12. Use the left mouse button to select the **stop bits:** text entry box.

    The text entry box will become highlighted and a cursor will appear in the box.

13. Enter the number of stop bits in the serial output.

    This should be 1 for the Magnavox 4200D.

14. Use the left mouse button to select the **parity:** text entry box.

    The text entry box will become highlighted and a cursor will appear in the box.

15. Enter the parity mode for the serial output.

    Valid values are O (odd parity), E (even parity), or N (no parity). This should be N for the Magnavox 4200D.

16. Select the **Set** button to accept the communications parameters for the Magnavox 4200D.

17. Select the **Done** button to dismiss the setdevices tool.

18. Select the **Survey Tools** menu button on the RTGIS Main Menu.

19. Select the **Serial data utilities** cascade menu.

20 Select the **Start/stop logging** menu button.

    The Start/stop logging tool will appear.

21. Use the left mouse button to select **Magnavox 4200D DGPS** in the device list.

    Magnavox 4200D DGPS will appear in the device text box.

22. Select the **Start** button to start logging.

    The Magnavox 4200D input will be logged to files in the nav directory of the current GRASS MAPSET. The name of the log file will be printed to the RTGIS terminal window. Log files will be closed and new ones opened by default every 20 minutes.

23. Select the **Done** button to dismiss the Start/stop logging tool.

6.2.2 Starting *CT-1* ship gyro compass data acquisition

This procedure assumes that the *CT-1* gyro compass is turned on, operating properly, and its ASCII output has been properly connected to one of the serial ports on the RTGIS Toolbox workstation.

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.

2. Select the **Survey Tools** menu button on the RTGIS Main Menu.

   Use the left mouse button or the <alt>s hot key. The Survey Tools pull down menu will appear.

3. Select the **Serial data utilities** cascade menu

4. Select the **Set device communications** menu button.

   Use the left mouse button or the e hot key. The setdevices tool will appear.

   Note: Steps 4 - 18 only need to be performed if you add a new device or you change the port that a device is usually connected to. The device communications are stored in a defaults file so that they do not need to be reset for each survey if they are used in identical configurations.

5. Use the left mouse button to select **CT1GYRO** from the device list on the setdevices tool.

   The name CT1GYRO will appear in the device text box.

6. Use the left mouse button to select the **port #:** text entry box.

   The text entry box will become highlighted and a cursor will appear in the box.

7. Enter the port number to which the *CT-1* ship gyro output is connected.

8. Use the left mouse button to select the **baud:** text entry box.

   The text entry box will become highlighted and a cursor will appear in the box.

9. Enter the baud rate of the serial output.

   This should be 9600 for the *CT-1* ship gyro compass.

10. Use the left mouse button to select the **data bits:** text entry box.

    The text entry box will become highlighted and a cursor will appear in the box.

11. Enter the number of data bits in the serial output.

    This should be 8 for the *CT-1* ship gyro compass.

12. Use the left mouse button to select the **stop bits:** text entry box.

    The text entry box will become highlighted and a cursor will appear in the box.

13. Enter the number of stop bits in the serial output.

    This should be 1 for the *CT-1* ship gyro compass.

14. Use the left mouse button to select the **parity:** text entry box.

    The text entry box will become highlighted and a cursor will appear in the box.

15. Enter the parity mode for the serial output.

    Valid values are O (odd parity), E (even parity), or N (no parity). This should be N for the *CT-1* ship gyro compass.

16. Select the **Set** button to accept the communications parameters for the *CT-1* ship gyro compass.

17. Select the **Done** button to dismiss the setdevices tool.

18. Select the **Survey Tools** menu button on the RTGIS Main Menu.

19. Select the **Serial data utilities** cascade menu.

20 Select the **Start/stop logging** menu button.

   The Start/stop logging tool will appear.

21. Use the left mouse button to select **CT1 gyro compass** in the device list.

    CT1 gyro compass will appear in the device text box.

22. Select the **Start** button to start logging.

The *CT-1* ship gyro compass input will be logged to files in the heading directory of the current GRASS MAPSET. The name of the log file will be printed to the RTGIS terminal window. Log files will be closed and new ones opened by default every 20 minutes.

23. Select the **Done** button to dismiss the Start/stop logging tool.

6.2.3 Stopping Magnavox 4200D DGPS data acquisition

This procedure assumes that Magnavox 4200D DGPS logging has been started by the procedure in section 6.2.1 and is still running.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.
2. Select the **Serial data utilities** cascade menu.
3. Select the **Start/stop logging** menu button.

   The Start/stop logging tool will appear.

4. Use the left mouse button to select **Magnavox 4200D DGPS** in the device list.

   Magnavox 4200D DGPS will appear in the device text box.

5. Select the **Stop** button to stop logging.

   Magnavox 4200D DGPS logging will be stopped.

6. Select the **Done** button to dismiss the Start/stop logging tool.

6.2.4 Stopping *CT-1* ship gyro compass data acquisition

This procedure assumes that *CT-1* ship gyro compass logging has been started by the procedure in section 6.2.2 and is still running.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.
2. Select the **Serial data utilities** cascade menu.
3. Select the **Start/stop logging** menu button.

   The Start/stop logging tool will appear.

4. Use the left mouse button to select **CT1 gyro compass** in the device list.

   CT1 gyro compass will appear in the device text box.

5. Select the **Stop** button to stop logging.

   CT1 gyro compass logging will be stopped.

6. Select the **Done** button to dismiss the Start/stop logging tool.

6.3 Single Beam Bathymetry Data Acquisition

The RTGIS Toolbox provides single beam bathymetry data acquisition via a serial interface to the Datamarine Link 6000 depth transducer.

6.3.1 Starting single beam bathymetry data acquisition

This procedure assumes that the Datamarine Link6000 is turned on, the depth transducer connected and operating properly, and the NMEA output port has been properly connected to one of the serial ports on the RTGIS Toolbox workstation.

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.
2. Select the **Survey Tools** menu button on the RTGIS Main Menu.

   Use the left mouse button or the <alt>s hot key. The Survey Tools pull down menu will appear.
3. Select the **Serial data utilities** cascade menu
4. Select the **Set device communications** menu button.

   Use the left mouse button or the e hot key. The setdevices tool will appear.

   Note: Steps 4 - 18 only need to be performed if you add a new device or you change the port that a device is usually connected to. The device communications are stored in a defaults file so that they do not need to be reset for each survey if they are used in identical configurations.
5. Use the left mouse button to select **DMDEPTH** from the device list on the setdevices tool.

   The name DMDEPTH will appear in the device text box.
6. Use the left mouse button to select the **port #:** text entry box.

   The text entry box will become highlighted and a cursor will appear in the box.
7. Enter the port number to which the Datamrine Link6000 output port is connected.
8. Use the left mouse button to select the **baud:** text entry box.

   The text entry box will become highlighted and a cursor will appear in the box.
9. Enter the baud rate of the serial output.

   This should be 4800 for the Datamrine Link6000.
10. Use the left mouse button to select the **data bits:** text entry box.

    The text entry box will become highlighted and a cursor will appear in the box.
11. Enter the number of data bits in the serial output.

    This should be 8 for the Datamrine Link6000.

12. Use the left mouse button to select the **stop bits:** text entry box.

    The text entry box will become highlighted and a cursor will appear in the box.

13. Enter the number of stop bits in the serial output.

    This should be 1 for the Datamrine Link6000.

14. Use the left mouse button to select the **parity:** text entry box.

    The text entry box will become highlighted and a cursor will appear in the box.

15. Enter the parity mode for the serial output.

    Valid values are O (odd parity), E (even parity), or N (no parity). This should be N for the Datamrine Link6000.

16. Select the **Set** button to accept the communications parameters for the Datamrine Link6000 depth input.

17. Select the **Done** button to dismiss the setdevices tool.

18. Select the **Survey Tools** menu button on the RTGIS Main Menu.

19. Select the **Serial data utilities** cascade menu.

20 Select the **Start/stop logging** menu button.

    The Start/stop logging tool will appear.

21. Use the left mouse button to select **Datamarine Link6000 depth** in the device list.

    Datamarine Link6000 depth will appear in the device text box.

22. Select the **Start** button to start logging.

    The Datamarine Link6000 depth input will be logged to files in the depth directory of the current GRASS MAPSET. The name of the log file will be printed to the RTGIS terminal window. Log files will be closed and new ones opened by default every 20 minutes.

23. Select the **Done** button to dismiss the Start/stop logging tool.

6.3.2 Stopping single beam bathymetry data acquisition

This procedure assumes that single beam bathymetry logging has been started by the procedure in section 6.2.1 and is still running.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.

2. Select the **Serial data utilities** cascade menu.

3. Select the **Start/stop logging** menu button.

The Start/stop logging tool will appear.

4. Use the left mouse button to select **Datamarine Link6000 depth** in the device list.

   Datamarine Link6000 depth will appear in the device text box.

5. Select the **Stop** button to stop logging.

   Datamarine Link6000 depth logging will be stopped.

6. Select the **Done** button to dismiss the Start/stop logging tool.

6.4 Sidescan Sonar Data Acquisition

The RTGIS Toolbox provides digital acquisition of sidescan sonar data via a serial interface to the EG&G SMS960. The serial sidescan sonar data acquisition software is accessed through a separate interface from the previous serially acquired data. The purpose of this is to provide a general purpose sidescan sonar acquisition utility which may

6.4.1 Starting digital sidescan sonar acquisition

This procedure assumes that the EG&G 272 sidescan sonar towfish is connected to the SMS960 and operating properly. It also assumes that the SMS960 Tape Write Port has been connected to the RTGIS SMS960 parallel-to-RS232 interface input and Channel A and B from the parallel-to-RS232 interface have been properly connected to RTGIS Toolbox serial ports.

1. If the RTGIS Toolbox is not already running, start the RTGIS Toolbox.

2. Select the **SLS Tools** menu button on the RTGIS Main Menu.

   The SLS Tools pull down menu will appear.

3. Select the **Data acquisition** menu button.

   The SLS acquisition setup tool will start.

4. Place the mouse cursor over the **System Type** option menu tab and click the left mouse button once.

   The list of supported systems will appear. Currently, only one system is supported; EG&G SMS960 -OMDC.

5. Place the mouse cursor over the **EG&G SMS960 - OMDC** menu option and click the left mouse button once.

   The setup interface for this system type will appear in the SLS acquisition setup tool.

   For the EG&G SMS960 - OMDC system type, the only setup options are the serial port numbers to which the parallel-to-RS232 interface's Channels A and B are connected.

6. Use the left mouse button to select the **Channel A serial port no.** text entry box.

   The text entry box will become highlighted and a cursor will appear in the box.

7. Enter the serial port number to which Channel A is connected.

8. Use the left mouse button to select the **Channel B serial port no.** text entry box.

   The text entry box will become highlighted and a cursor will appear in the box.

9. Enter the serial port number to which Channel B is connected.

10. Select the **Begin** button.

    Sidescan sonar acquisition will begin. The name of the output file will be printed to the RTIGS terminal window. The acquisition software will first verify that data is being received. Messages will be printed to the RTGIS terminal window.

    The SLS acquisition setup tool will not close. Rather, it will iconify itself and be placed near the upper right corner of the RTGIS Toolbox desktop. The same interface must be used later to stop the sidescan acquisition.

6.4.2 Stopping digital sidescan sonar acquisition

This procedure assumes that digital sidescan logging has been started by the procedure described in section 6.3.1 and is still running.

1. Double click the left mouse button on the SLS acquisition setup window icon.

   It should be located near the upper right corner of the RTGIS Toolbox desktop. You may need to move some windows to find it.

2. Select the **Stop** button.

   Digital sidescan sonar data acquisition will stop.

   If you are prepared to run the next line, leave the SLS acquisition setup tool open and simply select the Begin button to start logging the next swath.

3. When logging is complete, select the **Cancel** button to dismiss the SLS acquisition setup tool.

6.5 Multibeam Sonar Data Acquisition

Multibeam sonar data acquisition is accomplished via SeaBeam Instruments, Inc.'s proprietary SeaView™ system which currently supports the SeaBeam 2100 multibeam sonar system. With a SeaBeam 2100 sonar system, or equivalent, connected to the RTGIS Toolbox workstation via ethernet, follow the procedure below for data acquisition.

6.5.1 Logging and displaying SeaBeam 2100 multibeam data

This section does not discuss the hardware configuration and connection of the SeaBeam 2100 system. It is assumed that a SeaBeam 2100 system is properly connected to the RTGIS Toolbox workstation and ready to transmit data.

1. Make sure the SeaBeam 2100 system is configured to output on ethernet port 5209.

2. Start SeaLogger via the RTGIS Toolbox (MB Tools - SeaView - SeaLogger).

   Several status messages should appear. The last should read as follows:

   SeaLogger/SB2100 network port ready...

   If the SeaLogger message window does not display this message, there is a network error that must be debugged.

3. Select "Setup" on the SeaLogger main menu bar.

   The setup widget will appear.

4. Enter 5209 in the port field.

5. Enter 500 in the pings field.

6. Select OK on the setup widget.

7. Select "Save preferences" in the SeaLogger File menu.

8. Select the Start button at the lower part of the SeaLogger window.

9. Start the multibeam system and check for messages in the SeaLogger window indicating that data is being received.

   If data is being received, go on to the next step. If data is not being received, check that the SeaBeam 2100 system is properly working and connected to the RTGIS Toolbox workstation. When the problem is fixed, go on to the next step.

10. Click on the Start button at the lower part of the SeaLogger window (yes, again).

    After a few seconds, the following message should appear in the SeaLogger status window:

    ???

    This means that data is being logged.

11. Start SeaSwath via the RTGIS Toolbox (MB Tools - SeaView - SeaSwath)

12. Select "Real time File" in the SeaSwath File menu.

13. Use the file browser to select the log file indicated in the SeaLogger status window.

    The data being logged by SeaLogger should begin waterfalling in the SeaSwath display.

7. Survey Monitoring

The RTGIS Toolbox provides several tools for monitoring the progress of data logging and the overall progress of a survey. These tools include a helm guidance interface, text displays of logged data, and graphical displays of logged data.

7.1 Helm Guidance Interface

The RTGIS Toolbox provides helm guidance by interfacing to an existing helm guidance software running on a computer at the survey vessel's bridge. The software is called Maptech. The RTGIS Toolbox helm guidance interface consists of a program which rebroadcasts logged navigation and depth data on one of the RTGIS Toolbox workstation serial ports. By connecting that serial port to the Maptech computer, Maptech receives the the current navigation and depth inputs.

7.1.1 Starting the helm guidance interface

This procedure assumes that the Maptech software is running and awaiting input. The COM1 serial port on the Maptech computer should be connected to serial port 5 on the RTGIS Toolbox workstation. To change the default output port from port 5, the com file in the devices directory of the current MAPSET must be edited manually.

1. If navigation is not already being logged, start navigation data acquisition by the procedure in section 6.2.1.

   Heading from the ship gyro, and heading and depth from the Datamarine Link6000 will also be rebroadcast if they are being logged. As a minimum, navigation is required to be logged.

2. Select the **Survey Tools** menu button on the RTGIS Main Menu.

3. Select the **Serial data utilities** cascade menu button.

4. Select the **Output to Maptech** menu button.

   The RTGIS will rebroadcast navigation from the Magnavox 4200D, heading from the ship's gyro compass, and heading and depth from the Datamarine Link6000 to the Maptech helm guidance software. Use Maptech to verify that data is being received.

7.1.2 Stopping the helm guidance interface

This procedure assumes that the RTGIS Toolbox is currently rebroadcasting serial data to the Maptech helm guidance software.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.

2. Select the **Serial data utilities** cascade menu button.

   The **Output to Maptech** check button should be highlighted red.

3. Select the **Output to Maptech** menu button.

This will deselect the **Output to Maptech** check button and stop the output to the Maptech helm guidance software.

7.2 Monitoring Log Files

The RTGIS Toolbox provides a program to monitor each serial data acquisition program. The monitor program periodically checks the log file associated with the acquisition program to verify that it is updating and displays the most current record from the file. This provides a quick visual check of the data logging quality.

7.2.1 Starting log monitors

The following procedure assumes that the associated data acquisition program has been started.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.

2. Select the **Serial data utilities** cascade menu button.

3. Select the **Log monitors** menu button.

   The **Start log monitors** tool will start.

4. Use the left mouse button to select the device to monitor.

   For example, to monitor the Magnavox 4200D DGPS, select **Maganvox 4200D DGPS** from the device list box on the Start log monitors tool. Magnavox 4200D DGPS will appear in the device list box.

5. Use the left mouse button to select the monitor interval by adjusting the **Monitor interval** slider.

   The value is in seconds. This value determines the time interval between checking the log file and displaying its most recent record. A value of 5 is typical.

6. Select the **Start** button.

   A yellow xterm window will appear at the lower left corner of the screen. The log file monitor program will run in this xterm window. The most recent record from the file will be printed to this window. If the monitor program detects a problem, text warnings and audible beeps will be printed to the xterm window.

7. If this was not the first log monitor started, move the xterm window so that it does not obscure other xterm windows for previously started log monitors.

   The log monitor xterm windows are usually positioned across the bottom of the screen so that all log file monitors may be inspected in a single glance.

8. Repeat steps 4-7 for each device that is currently being logged.

9. When finished starting log monitors, select the **Done** button to dismiss the **Start log monitors** tool.

Log monitors will automatically stop when logging of the associated instrument has been stopped.

7.3 Monitoring Vessel Navigation

The RTGIS Toolbox provides tools to assist in always knowing the vessel's location. These include displaying the ship track on the current GRASS display monitor, updating the GRASS display monitor on the current vessel position, and marking the current position.

7.3.1 Starting ship track display

This procedure assumes that navigation is being logged as described in section 6.2.1, a GRASS display monitor is running and selected, and all desired background data layers have been drawn to the GRASS display monitor.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.

2. Select the **Track display** menu button.

   Use the left mouse button or the t hot key.

   The ship track display program will start. You may not see the ship track if the vessel's current navigation is outside the current geographic region. In this case, you may center the GRASS display on the current vessel location using the procedure in section 7.3.3 below.

7.3.2 Stopping ship track display

This procedure assumes that the RTGIS Toolbox is currently running the ship track display.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.

   The **Track display** check button should be highlighted red.

3. Select the **Track display** menu button.

   This will deselect the **Track display** check button and stop the ship track program.

7.3.3 Centering the GRASS display monitor on the vessel's current position

Occasionally, you may find that the vessel's position is outside of the current GRASS region. The result is that the ship track does not appear on the GRASS display monitor and you are unsure of your current location.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.

2. Select the **Center map on cur. pos.** menu button.

   Use the left mouse button or the e hot key.

   The GRASS region will update itself so that the vessel's current position is at the center. The GRASS display monitor will then be redrawn.

7.3.4 Marking the current vessel position on the GRASS display monitor

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.
2. Select the **Mark current location** menu button.

   The **Site annotaion** widget will appear.
3. Enter a comment or press the <Enter> key for no comment.
4. Press the **OK** button or the <Enter> key to accept the comment.

   The current vessel location will be shown on the GRASS display monitor with a vessel icon and the corresponding position and comment will be stored in the GRASS site file *Mark.pts*.

7.4 Monitoring Sidescan Sonar Data Acquisition

The sidescan sonar log monitor is the real time waterfall display. The displayed data is read from the sidescan log file and thus provides visual assurance that valid data is being logged.

7.4.1 Starting real time waterfall sidescan display

This procedure assumes that sidescan data is being logged as described in section 6.4.1.

1. Select the **SLS Tools** menu button on the RTGIS Main Menu.
2. Select the **Showimage** menu button.

   The Showimage application will start.
3. Select the **File** menu button on the Showimage main menu.
4. Select the **Realtime Button**.

   Sidescan sonar data should begin waterfalling in the Showimage display window. The waterfall display should only be run while sidescan data is being logged. When sidescan sonar data logging is stopped, the real time waterfall display should be stopped using the procedure in section 7.4.2.

7.4.2 Stopping real time waterfall sidescan sonar display

This procedure assumes that waterfall sidescan sonar display has been started as described in section 7.4.1 and is still running. The real time display should be stopped while sidescan is not being logged.

1. Select the **File** menu button on the Showimage main menu.
2. Select the **Realtime Button**.

   Sidescan sonar data should stop waterfalling in the Showimage display window.

3. If another swath is planned, you may leave Showimage open and start the display using section 7.4.1 when sidescan sonar logging has restarted.

   If there is to be no more sidescan sonar logging, select the **File** menu button on the Showimage main menu. Then select the **Exit** menu button. The Showimage application will be terminated.

7.5 Other Survey Monitoring Tools

The RTGIS Toolbox contains several additional tools which assist in visualizing the survey data and survey location.

7.5.1 Setting the online/offline flag

To assist in determining whether the survey vessel is currently on a survey line or in transit to the next line, the color of the ship track can be changed by setting the online/offline flag. This feature is of particular use in post-processing operations for retracing the survey.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.

2. If the vessel is currently on a survey line, use the left mouse button to select the **online** check button. If the vessel is currently in transit to a survey line, use the left mouse button to select the **offline** check button.

   When online, the ship track will be drawn in blue.

   When offline, the ship track will be drawn in red.

7.5.2 Setting the swath mapping system swath width

During swath mapping operations (e.g., sidescan sonar surveying), the swath width can be displayed as scaled lines perpendicular to the ship track.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.

2. Select the **Swath width** cascade menu.

3. Select the swath mapping system swath width.

   While the **online** flag is set, the swath will be displayed on the GRASS display.

7.5.3 Zooming the GRASS display monitor in and out

At any time during the survey, the region shown on the GRASS display monitor may be made larger or smaller.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.

2. Select the **Zoom** menu button.

This will launch the zoom program.

3. Follow steps 3-5 in section 4.4.3 to define the new region.

7.5.4 Redrawing the GRASS display monitor

At any time, the GRASS display monitor may be redrawn.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.
2. Select the **Redraw** menu button.

   The current GRASS display monitor will be erased and redrawn with a set of default data layers. To change the data layers displayed on redraws, follow the procedure in section 7.5.5 below.

7.5.5 Changing the data layers displayed on redraws

The commands for redrawing the GRASS display monitor duing survey operations are stored in the NEWMAP file in the current GRASS MAPSET directory. The commands must be valid GRASS display commands (e.g., d.rast, d.vect, d.sites, etc.). Consult the GRASS User's Manual or use the GRASS manual program (section 7.5.8) for the GRASS display commands formats.

During surveying, the RTGIS Toolbox will continuously update the NEWMAP file so that previous lines of ship track and swath coverage will be displayed on redraws.

1. Select the **GIS Tools** menu button on the RTGIS Main Menu.
2. Select the **Display environ.** cascade menu button.
3. Select the **Change coverages displayed** menu button.

   The outline of an xedit window will appear.
4. Place the xedit window.
5. Edit the display commands so that the proper GRASS data files are drawn on redraws.

   Consult with the GRASS User's Manual or section 7.5.8 below for command formats.
6. Select the **Save** button on the xedit window.
7. Select the **Quit** button on the xedit window.

   You are now ready to redraw with your new set of data layers.

7.5.6 Saving logging settings

The RTGIS Toolbox contains a feature to safeguard against system failures. The commands for all running serial data loggers, log file monitors, and the ship track display can be stored and quickly restarted following a system failure.

1. After all serial data logging and display programs have been started, select the **Survey Tools** menu button on the RTGIS Toolbox Main Menu.

2. Select the **Serial data utilities** menu button.

3. Select the **Save Settings** menu button.

   The settings will be stored in the SETTINGS file in the current GRASS MAPSET. Along with the serial data logging settings will be stored commands to reset the GRASS display monitor and region.

   A dialog box will inform you which serial data logging commands have been saved.

4. Select the **OK** button to dismiss the dialog box.

7.5.7 Restoring logging settings

This procedure assumes that logging settings have been previously saved by the procedure described in section 7.5.6.

1. If a system failure has been encountered, exit the RTGIS Toolbox (if it has not already exited on its own).

2. Restart the RTGIS Toolbox.

3. Select the **Survey Tools** menu button on the RTGIS Main Menu.

4. Select the **Serial data utilities** menu button.

5. Select the **Restore Settings** menu button.

   The serial data logging and display commands previously saved will be restarted. You will be back to the data acquisition configuration that existed before the system failure.

   Note: Interactive applications, such as Showimage, will have to be restarted manually.

   A dialog box will appear telling you all the commands which have been restarted.

6. Select the **OK** button to dismiss the dialog box.

7.5.8 Reviewing GRASS command formats

It is often necessary to check the exact command line format of a GRASS command. For example, when editing the default data layers displayed on redraws, you must enter the exact GRASS display commands to draw the desired data layers.

1. Select the **GIS Tools** menu button on the RTGIS Main Menu.

2. Select the **GRASS manual** menu button.

   A green xterm window will start up and run the GRASS manual program.

3. Enter the command name for which you require information.

8. Post-Cruise Data Reporting

Post-cruise data reporting consists of generating text lists of the newly collected survey data and creating maps showing survey data coverage on top of background data layers.

8.1.1 Generating text reports of new survey data

Upon startup, the RTGIS Toolbox generates a list of all database files. At the completion of a survey, a tool will generate another file list. By comparing the two lists, a report of new survey data can be generated.

1. Select the **Survey Tools** menu button on the RTGIS Main Menu.

2. Select the **Survey Report** menu button.

   The **Survey Report** widget will start and present a file list of new survey data files. The list will include the date, time, and size of each file.

3. Press the **Print** button to print the report.

   The report will be printed to the HP DeskJet 855Cse printer on 8.5 x 11 pages. If the printer is not connected to the RTGIS Toolbox workstation and turned on, the report will not be printed.

4. When finished viewing the survey data report, select the **Done** button to dismiss the **Survey Report** widget.

9. Post Processing Data

9.1 Multibeam Sonar Data Processing

9.1.1 MB-System Documentation

The MB-System multibeam sonar data processing system is very well documented by a series of on-line UNIX man pages. There are a number of programs and options not included by the RTGIS Toolbox and the multibeam data processor should be aware of the full functionality of MB-System. Man pages for several of the more commonly used MB-System commands are included here for quick reference.

9.1.2 mbsystem man page

mbsystem(l) MB-System 4.4 mbsystem(l)

NAME

mbsystem - A set of utilities for manipulating and processing multibeam bathymetry data.

VERSION

Version 4.4

INTRODUCTION

The MB-System is a software package consisting of programs which manipulate, process, list, or display multibeam bathymetry, amplitude, and sidescan data which has been developed at the Lamont-Doherty Earth Observatory (L-DEO). The heart of the system is an input/output library called MBIO which allows programs to transparently work with any of a number of supported multibeam data formats. This approach has allowed the creation of "generic" utilities which can be applied in a uniform manner to multibeam data from a variety of sources.

The development of MBIO and its associated programs began in 1990 as part of ongoing research at L-DEO involving multibeam bathymetry data. Development was accelerated in 1991 as part of the effort to support the Atlas Hydrosweep DS multibeam sonar on L-DEO's ship, the R/V Maurice Ewing. The National Science Foundation provided support in 1993 and 1994 to improve and extend MB-System so as to provide a standard "generic" set of tools for processing and display of multibeam data that could be used by the U.S. academic community. SeaBeam Instruments and Antarctic Support Associates provided additional support in 1994 for the development of MB-System, with particular emphasis on capabilities related to the new SeaBeam 2100 series of sonars.

The National Science Foundation has now funded a five year effort to maintain and further develop MB-System; this funding is being augmented by additional efforts at SeaBeam Instruments which relate to SeaBeam products but are nonetheless released to public domain distribution.

The source code for MB-System is written in C and should be portable to a wide variety of Unix platforms. Graphical utilities are based on the Motif widget set. To date, MB-System has been successfully installed on Sun workstations running under SunOS4.1 and Solaris, Silicon Graphics (SGI) workstations running under IRIX, Hewlett Packard workstations, and 486 PC's running under the Lynx operating system.

AUTHORSHIP

David W. Caress (caress@lamont.ldeo.columbia.edu)
Dale N. Chayes (dale@lamont.ldeo.columbia.edu)
Lamont-Doherty Earth Observatory
Palisades, NY 10964

LIST OF MB-SYSTEM PROGRAMS AND MACROS

See the individual manual pages for detailed information about specific programs. See the manual page for MBIO for information about the i/o library and the multibeam data formats supported by MBIO.

These are the current MB-system programs:

hsdump: Lists contents of the various sorts of data records in Hydrosweep DS data.

mbanglecorrect: Apply a grazing angle correction to beam amplitude or sidescan data.

mbbackangle: Generates a table of the average amplitude or sidescan values as a function of the grazing angle with the seafloor.

mbbath: Generates bathymetry from travel times in multibeam data.

mbclean: Identifies and flags bad beams in multibeam bathymetry data.

mbcontour: Generate GMT compatible Postscript color swath contour plots.

mbcopy: Copy multibeam data files.

mbcut: Removes data from portions of swath as specified by the user.

mbdefaults: Set or list default mbio parameters for reading and writing multibeam data

mbedit: Interactive editor used to flag bad beams in multibeam bathymetry data.

mbfilter: Apply some simple filter functions to sidescan, beam amplitude, or bathymetry data.

mbformat: List information about multibeam data formats supported by the MBIO library.

mbgetmask: Extract list of flagged or edited beams from a multibeam data file.

mbgrid: Grid bathymetry, amplitude, and sidescan data from multibeam data files.

mbhistogram: Obtain histogram of bathymetry, amplitude, or sidescan data from multibeam data files.

mbinfo: Output some basic statistics of multibeam data files.

mblevitus: Create a water velocity profile
which is representative of the mean
annual water column for a specified
1 degree by 1 degree region.
mblist: List data in multibeam data files.
mbmask: Apply editing information obtained
from one file with mbgetmask
to another file.
mbmerge: Merges multibeam data with new
navigation.
mbnavedit: Interactive editor used to fix
problems with navigation in
multibeam data files.
mbps: Simple perspective views of swath
bathymetry in Postscript.
mbrollbias: Evaluate the roll bias of a
multibeam system using two pieces
of coincident bathymetry data
collected with opposing ship headings.
mbswath: Generate GMT compatible Postscript
color and color shaded relief swath plots.
mbtide: Corrects multibeam bathymetry
for tide data.
mbunclean: Unflags edited beams in multibeam
bathymetry data.
mbvelocitytool: Interactive program for
modeling the affect of the water
velocity profile on multibeam
bathymetry calculations.

Macros are programs or shellscripts which make use of programs from the MB-system and other software packages to accomplish common tasks easily. These are the current MB-system macros:
mbmfmtvel: Scans a Hydrosweep DS data file
and outputs a formatted table of
the mean water velocity and surface
water velocity values used in
processing that data.
mbm_dslnavfix: Reads a WHOI DSL AMS-120 processed
navigation file containing UTM northings
and eastings and outputs a navigation file
containing longitude and latitude which is
suitable for use with mbmerge.
mbm_grdplot: Reads a GMT GRD grid file and
writes a shellscript which will
generate a GMT map of the data.
mbm_grd3dplot: Reads a GMT GRD grid file and
writes a shellscript which will
generate a GMT 3D perspective view
of the data.
mbm_plot: Reads a multibeam data file and
writes a shellscript which will
generate a swath and/or contour
plot of the data.
mbm_rollerror: Reads a multibeam data file,

calculates the noise in the vertical reference
used by the sonar, and generates a file
containing roll corrections which
can be applied to the data.

mbm_stat: Runs mbinfo on a multibeam data
file and extracts beam statistics from
the output of mbinfo.

mbm_vrefcheck: Generates a plot of high pass
filtered apparent crosstrack seafloor slope.

mbm_xbt: Processes a Sparton XBT data file
and outputs a sound velocity profile
file which can be used to process
multibeam data.

PROCESSING SEABEAM 2100 DATA

The following data processing stream is recommended for data obtained with SeaBeam 2100 series multibeam sonars. SeaBeam 2112 sonars are currently in use on the R/V Knorr (operated by the Woods Hole Oceanographic Institution and the R/V Nathaniel B. Palmer (operated by Antarctic Support Associates for the NSF), among other vessels.

Consider a data file "sb2112_example.mb41" containing one hour's worth of SeaBeam 2112 data in the vendor format (format 41). This file contains bathymetry, beam amplitude, and sidescan data. The following commands are typical for processing such data and generating preliminary maps.

Step 1: What's in the data file?

First we run mbinfo to obtain statistics about the contents of the data file:

```
mbinfo -F41 -I sb2112_example.mb41
```

Step 2: Edit the data.

We edit the data to check the quality of the bathymetry and to flag artifacts as necessary by running mbedit. The edited data are output when the done or quit button is clicked.

```
mbedit -F41 -I sb2112_example.mb41 -O sb2112_example_e.mb41
```

Step 3: Generate some first cut plots of the data.

Now, we use the mbm_plot macro to generate the following plots:

bathcont: color fill bathymetry with contours
bathshade: shaded relief color fill bathymetry
bathamp: color fill bathymetry overlaid with amplitude
amp: grayscale amplitude
ss: grayscale sidescan

We use the -S option to apply histogram equalization to sidescan and amplitude data; note that for the bathymetry overlaid with amplitude map we use -S0/1 so that the bathymetry is not histogram equalized but the amplitude data used for shading is histogram equalized. First generate plot shellscripts:

```
mbm_plot -F41 -I sb2112_example_e.mb41 -G1 \
  -C  -N -V -Obathcont
```

```
mbm_plot -F41 -I sb2112_example_e.mb41 -G2 \
        -N -V -Obathshade
mbm_plot -F41 -I sb2112_example_e.mb41 -G3 \
    -S0/1 -N -V -Obathamp
mbm_plot -F41 -I sb2112_example_e.mb41 -G4 \
    -S   -N -V -Oamp
mbm_plot -F41 -I sb2112_example_e.mb41 -G5 \
    -S   -N -V -Oss
```

Then run plot shellscripts to generate plots (postscript files) and put them on the screen:

```
bathcont.cmd
bathshade.cmd
bathamp.cmd
amp.cmd
ss.cmd
```

If we had used the -X option in mbm_plot the plotting shellscripts would have been executed automatically, bringing the plots up on the screen after they were generated.

Step 4: Destripe the sidescan data.

Hull mounted multibeam sonars are subject to a variety of noise sources (e.g. bubbles under the hull) which can cause the sidescan from some pings to be "bad". If parts of some sidescan pings are much higher or lower in amplitude than the adjacent pings, a map made with the sidescan will appear "stripey". The program mbfilter can be used to despike and destripe data by operating a boxcar median filter with low and high ration thresholds (each value is changed only if the original value divided by the local median is less than the low threshold or greater than the high threshold). The -A2 causes the program to operated on sidescan data. We use -S3/3/3/1 so that a median smoothing filter with dimensions of 3 acrosstrack by 3 alongtrack is applied to the data with 1 iteration. The -T0.6/1.67 causes the median filter to work with ratio thresholds of 0.6 and 1.67.

```
mbfilter -F41 -I sb2112_example_e.mb41  \
    -O sb2112_example_ed.mb41 \
    -A2 -S3/3/3/1 -T0.6/1.67 -V
```

We run mbm_plot again to see the results of the destriping:

```
mbm_plot -F41 -I sb2112_example_ed.mb41 -G5 \
    -S   -N -V -Ossdestripe
ssdestripe.cmd
```

Step 5: Obtain amplitude vs grazing angle functions.

The raw sidescan and amplitude images are dominated by the contrast between the high values at the center of the swath and the low values at the swath edges. In order to focus variations in grayscale or color on structure of interest, we seek to "flatten" the images by correcting the sidescan and amplitude for the drop in backscatter amplitude with increasing seafloor grazing angle. In order to make such corrections, we first use mbbackangle to obtain amplitude vs grazing angle functions for both amplitude and sidescan data:

```
mbbackangle -F41 -I sb2112_example_ed.mb41 \
   -A1 -V > avsga_amplitude.dat
mbbackangle -F41 -I sb2112_example_ed.mb41 \
   -A2 -V > avsga_ss.dat
```

Step 6: Apply amplitude vs grazing angle corrections.

Use mbanglecorrect to apply the amplitude vs grazing angle correction to both amplitude and sidescan data in turn (note the output of first run is input of second). In the first case, the -G argument forces the program to ignore seafloor slope in calculating grazing angles, so bathymetric features remain in the amplitude or sidescan data:

```
mbanglecorrect -F41 -I sb2112_example_ed.mb41 \
   -A1/100 -S avsga_amplitude.dat -G -Z3800 \
   -O sb2112_example_eda1.mb41 -V
mbanglecorrect -F41 -I sb2112_example_eda1.mb41 \
   -A2/1000 -S avsga_ss.dat -G -Z3800 \
   -O sb2112_example_eda1a1.mb41 -V
```

In the second case, the program uses seafloor slope in calculating grazing angles, so bathymetric features tend to be removed from the amplitude or sidescan data:

```
mbanglecorrect -F41 -I sb2112_example_ed.mb41 \
   -A1/100 -S avsga_amplitude.dat -Z3800 \
   -O sb2112_example_eda2.mb41 -V
mbanglecorrect -F41 -I sb2112_example_eda2.mb41 \
   -A2/1000 -S avsga_ss.dat -Z3800 \
   -O sb2112_example_eda2a2.mb41 -V
```

Now generate first cut plots of the corrected amplitude and sidescan data:

```
mbm_plot -F41 -I sb2112_example_eda1a1.mb41 -G3 \
   -S0/1 -N -V -Obathampcorr1
mbm_plot -F41 -I sb2112_example_eda1a1.mb41 -G4 \
   -S   -N -V -Oampcorr1
mbm_plot -F41 -I sb2112_example_eda1a1.mb41 -G5 \
   -S   -N -V -Osscorr1
mbm_plot -F41 -I sb2112_example_eda2a2.mb41 -G3 \
   -S0/1 -N -V -Obathampcorr2
mbm_plot -F41 -I sb2112_example_eda2a2.mb41 -G4 \
   -S   -N -V -Oampcorr2
mbm_plot -F41 -I sb2112_example_eda2a2.mb41 -G5 \
   -S   -N -V -Osscorr2

bathampcorr1.cmd
ampcorr1.cmd
sscorr1.cmd
bathampcorr2.cmd
ampcorr2.cmd
sscorr2.cmd
```

Step 7: Apply filtering to fully corrected sidescan.

Now use mbfilter to try to bring out fine scale features and differences in seafloor texture from the multibeam data. In order to avoid biasing the filtered data, we first

use mbcut to remove the nadir region from the sidescan swath. This is necessary because the amplitude vs angle correction does not entirely clean up the specular reflection that dominates the nadir region. For filtering we apply simple boxcar mean hi-pass and lo-pass filters to the sidescan. The sizes of the boxcar filters are specified in terms of across and along-track numbers of pixels. In this case we use:

```
Mean subtraction filter for hipass
  filter acrosstrack dimension: 301
  filter alongtrack dimension:  1
  filter iterations:          1
  filter offset:          10000.000000
Mean filter for smoothing
  filter acrosstrack dimension: 3
  filter alongtrack dimension:  3
  filter iterations:          1
```

Since the alongtrack dimension is 1, the sidescan data in each ping is hipass filtered separately.

```
mbcut -F41 -I sb2112_example_eda2a2.mb41 \
   -O sb2112_example_eda2a2c.mb41 \
   -A2/1/951/1048 -V
mbfilter -F41 -I sb2112_example_eda2a2c.mb41 \
   -O sb2112_example_eda2a2cf.mb41 \
   -A2 -D1/301/1/1 -S1/3/3/1 -V
```

Now generate first cut plots of the corrected and filtered sidescan:

```
mbm_plot -F41 -I sb2112_example_eda2a2cf.mb41 \
   -G5 -S -N -V -Ossfilter

ssfilter.cmd
```

Step 8: Grid the bathymetry and corrected sidescan data.

Now use mbgrid to grid the bathymetry. The greatest depth in the file is 4502 meters (from the mbinfo output). The 120 degree swath is 3.4 times the water depth wide, or 15.3 km wide. This translates to an average acrosstrack spacing of 15300 m / 120 = 127.5 m. If a region of a grid has more than one data point in each grid cell or bin, we say that this region is "oversampled". If some bins in a region have no data points, we say that this region is "undersampled". We choose a grid cell spacing of 150 m, which will cause the grid to be oversampled towards the center of the swath, but undersampled towards the edges of the swath.

The program mbgrid takes a datalist as input, so we first construct that file:

```
echo sb2112_example_eda2a2.mb41 41 > datalist_grid
```

Now run mbgrid using the gaussian weighted mean algorithm (-F1) and longitude and latitude grid cell spacings of 150 m (-E150/150/m) to grid bathymetry (-A1). We use spline interpolation to fill in small gaps in the data no larger than two grid cells (-C2), and we set regions with no data to Nan values for compatibility with GMT programs (-N). We also specify -M so that grids of data density and data standard deviation will be generated:

```
mbgrid -Idatalist_grid -E150/150/m \
    -R114.2208/114.4209/-31.9001/-31.6377 \
    -Osb2112_example_bath -A1 -N -C2 -M -V
```

The program mbgrid creates a shellscript which, when executed, will generate a color fill plot overlaid with contours of the gridded bathymetry. Now run that shellscript:

```
grd_sb2112_example_bath.cmd
```

Shellscripts have also been created to generate plots of the data density and standard deviation grids:

```
grd_sb2112_example_bath_num.cmd
grd_sb2112_example_bath_sd.cmd
```

Now run mbgrid to grid the bathymetry corrected sidescan data (-A4):

```
mbgrid -Idatalist_grid -E150/150/m \
    -R114.2208/114.4209/-31.9001/-31.6377 \
    -Osb2112_example_ss -A4 -N -C2 -M -V
```

Now run the plot shellscripts to view the gridded sidescan data in grayscale and the ancillary data density and standard deviation plots:

```
grd_sb2112_example_ss.cmd
grd_sb2112_example_ss_num.cmd
grd_sb2112_example_ss_sd.cmd
```

Step 9: Generate additional maps of the gridded data.

First, we use mbm_grdplot to generate a color shaded relief view of the bathymetry. We choose to illuminate the bathymetry from the northeast (azimuth of 45 degrees) and to use a shading magnitude of 0.4 (-A0.4/45). Because the data has been gridded as bathymetry (positive down) rather than as topography (positive up), the default plot will have "hot" colors for deep regions and "cold" colors for shallow regions; this is the opposite of the convention we usually use. In order to fix the colors, we have to either rescale the data by multiplying the bathymetry by -1 (accomplished with -MGS-1), or flip the color pallette (accomplished with -D). We use the latter approach. Finally, because this grid is so small, the default shaded relief image is likely to be grainy. To fix this problem, we specify a dots per inch resolution of 72 (-MGQ72); this will take longer and generate a larger plotfile, but the plot will look better. We also use the -L option to specify the title and color scale label for the plot. Here is the command:

```
mbm_grdplot -Igrd_sb2112_example_bath -G2 -A0.4/45 -D -MGQ72 -V
  -L"Shaded Relief Bathymetry":"Depth (meters)" -Osb2112_example_bathshade

sb2112_example_bathshade.cmd
```

Second, we use mbm_grdplot to generate a color fill view of the bathymetry overlaid with the gridded sidescan. The sidescan overlay is specified using the -K option. We want the colors for the bathymetry to be chosen without histogram equalization, but we also want histogram equalization to be applied to the sidescan data used for shading. To do this, we use -S0/1, where the first number (0) specifies no histogram equalization of the color scale and the second number (1) causes histogram equalization of the shading

sidescan data to be implemented. In order to maintain the convention that high sidescan amplitudes are black, we flip both the color pallette (as in the previous example) and the shading scale with -D1/1. We could also flip the shading by specifing a negative shading magnitude (-A-0.4). In this case, we forgo specifying the image resolution, resulting in a grainy plot:

```
mbm_grdplot -Igrd_sb2112_example_bath -G3 -Kgrd_sb2112_example_ss
 -S0/1 -D1/1 -A0.4 -V -L"Bathymetry Overlaid With Sidescan":"Depth
(meters)" -Osb2112_example_bathss

sb2112_example_bathss.cmd
```

Step 10: Generate 3D perspective views of the gridded data.

Now, generate a 3D perspective view of the gridded bathymetry with shading through synthetic illumination (-G2) using the macro mbm_grd3dplot. The grid file is in bathymetry (positive down) rather in topography (positive up), so the bathymetry needs to be rescaled by multiplying by -1 (-MGS-1). We choose an illumination magnitude of 0.4 and an illumination azimuth of 45 degrees (-A0.4/45). We also choose a perspective azimuth of 250 degrees and an elevation of 30 degrees (-E240/30):

```
mbm_grd3dplot -Igrd_sb2112_example_bath -G2 -A0.4/45 -E250/30 -MGS-1 -V
 -Osb2112_example_bath3d

sb2112_example_bath3d.cmd
```

Now, generate a 3D perspective view of the gridded bathymetry shaded using the gridded sidescan data (-Kgrd_sb2112_example_ss). We want the sidescan data to be histogram equalized, so we use -S0/1. We also want the shading to be more prominent than the default shading magnitude of 0.2 would produce, so we use -A0.5:

```
mbm_grd3dplot -Igrd_sb2112_example_bath -Kgrd_sb2112_example_ss
 -G3 -A0.5 -E250/30 -D0/1 -S0/1 -MGS-1 -V -Osb2112_example_bathss3d

sb2112_example_bathss3d.cmd
```

PROCESSING HYDROSWEEP DS DATA

The following data processing stream is recommended for Hydrosweep DS data. Hydrosweep DS sonars are currently in use on the R/V Maurice Ewing (operated by the Lamont-Doherty Earth Observatory and the R/V Thomas Thompson (operated by the University of Washington), among other vessels.

Consider a data file "hs_ew9204_134.mb21" containing one day's worth of Hydrosweep data in the raw "real-time" ascii format (format 21).

Step 1: What's in the data file?

Run mbinfo to obtain statistics about the contents of the data file:

```
mbinfo -F21 -Ihs_ew9204_134.mb21
```

Step 2: Copy the data from the raw ascii format (format 21) to a binary format (format 24) containing the same information. Data i/o for format 24 is 15 times faster than format 21. For example, give the command:

```
mbcopy -F21/24 -Ihs_ew9204_134.mb21 \
   -Ohsih_ew9204_134.mb24
```

Step 3: Obtain a reasonable water velocity profile. Use the command:

```
mblevitus -R-140/-9 -Owvel_ew9204
```

to obtain an initial model. Then use mbvelocitytool to interactively model the affect of changes in the water velocity profile on the bathymetry; the best model may be output for use with mbbath. Velocity profiles from XBT or CTD casts may also be viewed in mbvelocitytool to help define the profile to be used in processing.

Step 4: Construct bathymetry data by full raytracing through the water velocity profile. This step is NECESSARY because the sonar calculates bathymetry assuming a homogeneous water sound velocity. Use the command:

```
mbbath -F24 -Ihsnav_ew9204_134.mb24 \
   -Wwvel_ew9204 \
   -Ohsbat_ew9204_134_b.mb24
```

Step 5: Apply autoediting to flag the really bad bathymetric artifacts as bad. This step is optional - sometimes it makes the interactive editing easier and sometimes it just makes things worse. Use the command:

```
mbclean -F24 -Ihsbat_ew9204_134_b.mb24 \
   -Ohscln_ew9204_134_bc.mb24 \
   -D0.01/0.25 -G0.97/1.03 -C2.5 -V
```

Step 6: Interactively edit the data using mbedit. Start up mbedit and load the data file "hscln_ew9204_134_bc.mb24"; the output file will be "hscln_ew9204_134_bce.mb24".

Step 7a: Merge the data with good, processed navigation. If nav_ew9204_134 is a file containing a day's worth of navigation in the L-DEO navigation format, then give the command:

```
mbmerge -F24 -Ihsih_ew9204_134_bce.mb24 \
   -Nnav_ew9204_134 \
   -Ohsnav_ew9204_134_bcen.mb24
```

Step 7b: If separately processed navigation is not available, edit the existing navigation using the program mbnavedit. This allows the user to remove spikes and other problems from the navigation. Start up mbedit and load the data file "hscln_ew9204_134_bce.mb24"; the output file will be "hscln_ew9204_134_bcen.mb24".

Step 8: Display the data as swath contour plots or swath color fill/shaded relief plots using the macro mbm_plot. See the manual page for examples.

Step 9: Grid and display the data. Supposing that the area of interest is in the bounds -140 < longitude < -139 and -10 < latitude < -9, that the desired grid has a 500 m node spacing, that the gridded data should be in the form of topography (positive upward), and that the grid should be spline interpolated to fill in small gaps (2 grid cells wide), try:

```
cat hsnav_ew9204_134_bcen.mb24 24 > data.list
```

```
mbgrid -R-140/-139/-10/-9 -E500/500/meters \
   -A1 -C2 -Idata.list -Oew9204_bath
```

Here the first command creates a data list file used by mbgrid, and the second command runs mbgrid, generating a file of gridded topography. This command will produce a shellscript grd_ew9204_bath.cmd which will run GMT programs to display the data.

Step 10: Display the gridded data in 2D maps using the macro mbm_grdplot or in 3D perspective views using the macro mbm_grd3dplot. See the appropriate manual pages for examples.

PROCESSING HYDROSWEEP MD DATA

Processing Hydrosweep MD data is similar to processing Hydrosweep DS data. In particular, recalculating bathymetry from the travel times is necessary because the sonar uses a homogeneous water velocity model. The difference is simply in the data formats used. The raw Hydrosweep MD data files (typically named with a ".R" suffix) contain only the travel times; the sonar calculated bathymetry is contained in parallel files (typically named with a ".P" suffix). The ".R" files are supported as format 101. Format 102 data files contain bathymetry in addition to the travel times.

Thus, the first step in processing Hydrosweep MD data is to copy the data from format 101 to format 102. For example, give the command:

```
mbcopy -F101/102 -Iys9409040607.R \
   -Oys9409040607.mb102
```

Bathymetry equivalent to that generated by the sonar will automatically be calculated in the copy process (the data stream includes the mean water velocity used by the sonar). Following this step, process the Hydrosweep MD data in the same fashion as that recommended above for Hydrosweep DS data.

Should one wish to export data from format 102 files back to data files usable by Atlas data processing software, be sure to run mbcopy with the -°-N°N so that comments are stripped from the data. The MB-System implementation of format 101 includes comment records that are not supported by the Atlas software. For example:

```
mbcopy -F102/101 -Iys9409040607.mb102 \
   -Oys9409040607.R -N
```

PROCESSING WHOI DSL AMS-120 DATA

The WHOI DSL group typically provides processed AMS-120 data as a set of parallel files. An example of the filenames used is:

```
DSL120.940630_1100.amp.dat   (sidescan)
DSL120.940630_1100.bat.dat   (bathymetry)
DSL120.940630_1100.nav       (navigation)
```

The sidescan and bathymetry files contain navigation, but it is generally poor. The processed navigation resides in the *.nav file, but it is sampled less frequently than the sonar pings, and is reported in UTM easting and northing meters. The mbm_dslnavfix macro is used to translate the eastings and northings into the geographic coordinates (longitude and latitude) used by MB-System programs. The program mbmerge can be used to merge the translated navigation in with the bathymetry and sidescan.

The parallel files listed above are supported as format 111 by MB-System; a similar sin-

gle-file format is supported as format 112. We recommend converting the data to format 112 before proceeding with processing to simplify data management. However, all relevent MB-System programs will work with format 111 files. In either case, the data is voluminous (~75 MB/hour) and processing will be slow relative to data from multibeam sonars.

We recommend the following initial procedure when processing WHOI DSL AMS-120 data with MB-System:

Step 1: Concatenate all of the processed navigation files into a single file.

```
cat *.nav > cruise.rnav
```

Step 2: Translate the DSL UTM navigation into geographic coordinates. Note that you must know the UTM zone used - here we assume the zone is +27.

```
mbm_dslnavfix -Icruise.rnav \
    -Ocruise.mrg2 -J27 -V
```

Step 3: Copy the bathymetry and sidescan data in each file from format 111 to format 112.

```
mbcopy -F111/112 \
    -IDSL120.940630_1100.bat.dat \
    -ODSL120.940630_1100.mb112
```

Step 4: Merge the translated navigation with each of the format 112 data files. Note that mbm_dslnavfix produces navigation in mbmerge format 2 so that the -M_ option is used.

```
mbmerge -F112 \
    -IDSL120.940630_1100.mb112 \
    -ODSL120.940630_1100_n.mb112 \
    -M2 -Ncruise.mrg2 -V
```

At present, there is no capability for recalculating bathymetry using different water sound velocity models than that used originally. Adding such a capability would require defining a new data format and seems unnecessary for any of the data we have examined.

The large number of bathymetry values (2048) per ping and the high (> 1/second) ping rate make interactive editing of the bathymetry extremely time consuming. It is probably more realistic to use auto-editing (e.g. mbclean) to clean up the worst artifacts. One may also find it necessary to use median filtering rather than weighted mean filtering in the gridding process (mbgrid). Other than the above, the generation of grids and maps can proceed in the same fashion as outlined above for SeaBeam 2112 data.

SEE ALSO

mbio(l)

BUGS

It doesn't do everything we want it to yet, it doesn't work with every kind of swath data ever collected, and sometimes it breaks.

mbinfo(l) MB-System 4.4 mbinfo(l)

NAME

mbinfo - Output some basic statistics of multibeam bathymetry data files.

VERSION

Version 4.4

SYNOPSIS

mbinfo [-Byr/mo/da/hr/mn/sc -C -Eyr/mo/da/hr/mn/sc -Fformat -G -Ifilename -Llonflip -Ppings -Rwest/east/south/north -Sspeed -W -V -H]

DESCRIPTION

mbinfo is an utility for reading a multibeam data file and outputting some basic statistics. The data input may be averaged or windowed in time and space. If pings are averaged (pings > 2) mbinfo estimates the variance for each of the multibeam beams and sidescan pixels by reading a set number of pings (>2) and then finding the variance of the values for each beam. The bathymetry values are detrended before variances are calculated. The results are dumped to stdout.

AUTHORSHIP

David W. Caress (caress@lamont.ldeo.columbia.edu)
Dale N. Chayes (dale@lamont.ldeo.columbia.edu)
Lamont-Doherty Earth Observatory
Palisades, NY 10964

OPTIONS

-Byr/mo/da/hr/mn/sc

Sets the starting time for data allowed in the input data; pings with times before the starting time will be ignored. Default: yr/mo/da/hr/mn/sc = 1962/2/21/10/30/0.

-C Normally, mbinfo ignores comments encountered in the data file. If the -C flag is given, all of the comments will be printed out.

-Eyr/mo/da/hr/mn/sc

Sets the ending time for data allowed in the input data; pings with times after the ending time will be ignored. Default: yr/mo/da/hr/mn/sc = 2062/2/21/10/30/0.

-Fformat

Sets the data format used if the input is read from stdin or from a file. If format < 0, then the input file specified with the -I option will actually contain a list of input multibeam data files. This program uses the MBIO library and will read or write any multibeam format supported by MBIO. A list of the multibeam data formats currently supported by MBIO and their identifier values is given in the MBIO manual page. Default: format = 11.

-G Enables checking for reasonable navigation to be used in calculating the minimum and maximum longitude and latitude values. This mode excludes longitude and latitude values of 0.0 or those values associated with very large apparent speeds between pings (calculated using distance and time differences between adjacent pings). This option is particularly useful when one is trying to obtain reasonable bounds for a plot.

-H This "help" flag cause the program to print out a description of its operation and then exit immediately.

-Ifilename

Sets the input filename. If format > 0 (set with the -f option) then the multibeam data contained in infile is read and processed. If format < 0, then infile is assumed to be an ascii file containing a list of the input multibeam data files to be processed and their formats. The program will read the data in each one of these files. In the infile file, each data file should be followed by a data format identifier, e.g.:

datafile1 11
datafile2 24

This program uses the MBIO library and will read or write any multibeam format supported by MBIO. A list of the multibeam data formats currently supported by MBIO and their identifier values is given in the MBIO manual page. Default: infile = "stdin".

-Llonflip

Sets the range of the longitude values returned. If lonflip=-1 then the longitude values will be in the range from -360 to 0 degrees. If lonflip=0 then the longitude values will be in the range from -180 to 180 degrees. If lonflip=1 then the longitude values will be in the range from 0 to 360 degrees. Default: lonflip = 0.

-Ppings

Turns on variance calculations for the bathymetry, amplitude, and sidescan data (as available in the data stream). If pings = 1, then no variance calculations are made. If pings > 1, then variances are calculated for each beam and pixel using groups of pings values. The bathymetry values are detrended before the variances are calculated; the amplitude and sidescan values are not detrended. The variance calculations can provide crude measures of noise and/or signal as a function of beam and pixel number. Default: pings = 1 (no variance calculations).

-Rwest/east/south/north

Sets the longitude and latitude bounds within which multibeam data will be read. Only the data which lies within these bounds will be read. Default: west=-360, east=360, south=-90, north=90.

-Sspeed

Sets the minimum speed in km/hr (5.5 kts ~ 10 km/hr) allowed in the input data; pings associated with a smaller ship speed will not be copied. Default: speed = 0.

-Ttimegap

Sets the maximum time gap in minutes between adjacent pings allowed before the data is considered to have a gap. Default: timegap = 1.

-V Normally, mbinfo only prints out the statistics obtained by reading all of the data. If the -V flag is given, then mbinfo works in a "verbose" mode and outputs the program version being used and all read error status messages.

-W Normally, mbinfo reports depth values in meters. If the -W flag is given, then mbinfo reports these values in feet.

EXAMPLES

Suppose one wishes to know something about the contents of a Hydrosweep file (format 24) called example_hs.mb24. The following will suffice:

```
mbinfo -F24 -Iexample_hs.mb24
```

The following output is produced:

```
Multibeam Data File:  example_hs.mb24
MBIO Data Format ID:  24
Format name:          MBF_HSLDEOIH
Informal Description: L-DEO in-house binary Hydrosweep
Attributes:           Hydrosweep DS, 59 beams, bathymetry and amplitude,
                      binary, centered, L-DEO.

Data Totals:
Number of Records:          263
Bathymetry Data (59 beams):
  Number of Beams:          15517
  Number of Good Beams:     13661   88.04%
  Number of Zero Beams:       868    5.59%
  Number of Flagged Beams:    988    6.37%
Amplitude Data (59 beams):
  Number of Beams:          15517
  Number of Good Beams:         0    0.00%
  Number of Zero Beams:     15517  100.00%
  Number of Flagged Beams:      0    0.00%
Sidescan Data (0 pixels):
  Number of Pixels:             0
  Number of Good Pixels:        0    0.00%
  Number of Zero Pixels:        0    0.00%
  Number of Flagged Pixels:     0    0.00%

Navigation Totals:
Total Time:           1.2425 hours
Total Track Length:   20.9421 km
Average Speed:        16.8548 km/hr ( 9.1107 knots)

Start of Data:
Time:  08 14 1993 18:00:25.000000  JD226
Lon: -49.3011   Lat:  12.1444   Depth:  4920.0000 meters
Speed: 18.3600 km/hr ( 9.9243 knots)  Heading:  97.2000 degrees

End of Data:
Time:  08 14 1993 19:14:58.000000  JD226
Lon: -49.1111   Lat:  12.1149   Depth:  5021.0000 meters
Speed: 17.2800 km/hr ( 9.3405 knots)  Heading:  97.0000 degrees
```

```
Limits:
Minimum Longitude:  -49.3061   Maximum Longitude:  -49.1064
Minimum Latitude:    12.0750   Maximum Latitude:    12.1806

Minimum Depth:    3726.0000   Maximum Depth:    5190.0000
```

Suppose we wanted to know how noisy the outer beams are relative to the inner beams. We might try:

```
mbinfo -F24 -P5 -Iexample_hs.mb24
```

obtaining:

```
Multibeam Data File: example_hs.mb24
MBIO Data Format ID: 24
Format name:        MBF_HSLDEOIH
Informal Description: L-DEO in-house binary Hydrosweep
Attributes:         Hydrosweep DS, 59 beams, bathymetry and amplitude,
                    binary, centered, L-DEO.

Data Totals:
Number of Records:          263
Bathymetry Data (59 beams):
 Number of Beams:          15517
 Number of Good Beams:     13661    88.04%
 Number of Zero Beams:       868     5.59%
 Number of Flagged Beams:    988     6.37%
Amplitude Data (59 beams):
 Number of Beams:          15517
 Number of Good Beams:         0     0.00%
 Number of Zero Beams:     15517   100.00%
 Number of Flagged Beams:      0     0.00%
Sidescan Data (0 pixels):
 Number of Pixels:             0
 Number of Good Pixels:        0     0.00%
 Number of Zero Pixels:        0     0.00%
 Number of Flagged Pixels:     0     0.00%

Navigation Totals:
Total Time:           1.2425 hours
Total Track Length:   20.9421 km
Average Speed:        16.8548 km/hr ( 9.1107 knots)

Start of Data:
Time:  08 14 1993 18:00:25.000000  JD226
Lon:  -49.3011   Lat:   12.1444   Depth:  4920.0000 meters
Speed:  18.3600 km/hr ( 9.9243 knots)  Heading:  97.2000 degrees

End of Data:
Time:  08 14 1993 19:14:58.000000  JD226
Lon:  -49.1111   Lat:   12.1149   Depth:  5021.0000 meters
Speed:  17.2800 km/hr ( 9.3405 knots)  Heading:  97.0000 degrees

Limits:
Minimum Longitude:  -49.3061   Maximum Longitude:  -49.1064
```

Minimum Latitude: 12.0750 Maximum Latitude: 12.1806
Minimum Depth: 3726.0000 Maximum Depth: 5190.0000

Beam Bathymetry Variances:

Pings Averaged: 5

Beam	N	Mean	Variance	Sigma
0	0	0.00	0.00	0.00
1	0	0.00	0.00	0.00
2	110	4719.59	342.69	18.51
3	105	4779.49	399.15	19.98
4	155	4748.81	280.18	16.74
5	155	4817.12	194.62	13.95
6	150	4826.44	197.76	14.06
7	160	4863.82	155.50	12.47
8	215	4806.08	229.11	15.14
9	235	4807.09	220.23	14.84
10	240	4766.29	158.83	12.60
11	250	4764.34	221.09	14.87
12	245	4765.35	146.24	12.09
13	250	4782.02	167.34	12.94
14	240	4798.38	92.98	9.64
15	245	4775.16	98.27	9.91
16	225	4782.35	136.30	11.67
17	210	4820.37	80.70	8.98
18	215	4821.15	80.97	9.00
19	215	4827.71	76.20	8.73
20	195	4842.65	84.22	9.18
21	190	4843.02	155.87	12.48
22	185	4884.28	73.69	8.58
23	175	4885.21	69.88	8.36
24	175	4871.47	52.01	7.21
25	180	4871.92	34.71	5.89
26	200	4830.80	36.83	6.07
27	205	4835.16	33.47	5.79
28	210	4809.96	43.07	6.56
29	190	4850.77	40.97	6.40
30	240	4768.69	64.23	8.01
31	240	4772.90	74.44	8.63
32	245	4760.11	57.97	7.61
33	255	4734.01	81.72	9.04
34	255	4728.19	82.21	9.07
35	260	4722.94	83.45	9.14
36	260	4721.95	102.02	10.10
37	260	4713.48	83.85	9.16
38	250	4715.40	101.33	10.07
39	255	4722.56	118.20	10.87
40	250	4727.48	109.13	10.45
41	255	4734.96	127.97	11.31
42	255	4724.53	124.06	11.14
43	230	4744.74	122.96	11.09
44	225	4752.16	98.22	9.91
45	230	4692.27	107.96	10.39
46	240	4696.93	95.93	9.79

47	230	4699.80	129.08	11.36
48	225	4696.32	145.20	12.05
49	220	4681.50	140.29	11.84
50	210	4676.16	103.35	10.17
51	180	4627.31	105.22	10.26
52	200	4654.55	207.85	14.42
53	130	4665.82	250.97	15.84
54	185	4704.29	300.80	17.34
55	135	4731.13	218.16	14.77
56	150	4736.29	178.16	13.35
57	115	4691.45	217.31	14.74
58	0	0.00	0.00	0.00

SEE ALSO
mbsystem(l)

BUGS
No bugs, only features...

9.1.4 mbnavedit man page

mbnavedit(l) MB-System 4.4 mbnavedit(l)

NAME
mbnavedit - Interactive navigation editor for multibeam swath data.

VERSION
Version 4.4

SYNOPSIS
mbnavedit [-Byr/mo/da/hr/mn/sc -D -Eyr/mo/da/hr/mn/sc -Fformat -Iinfile -V -H]

DESCRIPTION
mbnavedit is an interactive editor used to identify and fix problems with the navigation of multibeam swath data. Once a file has been read in, mbnavedit displays autoscaled plots of the longitude, latitude, speed, and heading time series. The user may select outliers and interpolate over them or, in the case of heading or speed data, replace the erroneous values with estimates derived from "course-made-good" or "speed-made-good", respectively. Data can only be selected and altered in a single plot at a time (this avoids confusion and prevents mistakes). The user can also inspect plots of roll, pitch, and heaved data to determine if the ship's vertical reference sensors were working properly. The edited data is usually output to a file, but the program can be operated in a "browse" mode where no data is output.

AUTHORSHIP
David W. Caress (caress@lamont.ldeo.columbia.edu)

Dale N. Chayes (dale@lamont.ldeo.columbia.edu)
Lamont-Doherty Earth Observatory
Palisades, NY 10964

OPTIONS

-Byr/mo/da/hr/mn/sc Sets the starting time for data allowed in the input data; pings with times before the starting time will be ignored. Default: yr/mo/da/hr/mn/sc = 1962/2/21/10/30/0.

-D Starts up the program in "browse" mode. If a file is opened in browse mode (either at startup or later), none of the edited data will be output to a file. The default is to output the edited data to a file.

-Eyr/mo/da/hr/mn/sc
Sets the ending time for data allowed in the input data; pings with times after the ending time will be ignored. Default: yr/mo/da/hr/mn/sc = 2062/2/21/10/30/0.

-Fformat
Sets the format at startup for the input and output multibeam data using MBIO integer format identifiers. This value can also be set interactively when specifying the input file. This program uses the MBIO library and will read or write any multibeam format supported by MBIO. A list of the multibeam data formats currently supported by MBIO and their identifier values is given in the MBIO manual page. Default: format = 11.

-H This "help" flag cause the program to print out a description of its operation and then exit immediately.

-Iinfile
Sets the data file from which the input data will be read at startup. This value can also be set interactively. If the input file is named using the MB-System convention of an ".mbXX" suffix (the XX corresponds to the MBIO format id), then the output file name will have an "e.mbXX" suffix. Otherwise, the output file will be infile with ".ed" appended.

-Ioutfile
Sets the output data file, overriding the file naming conventions discussed above in the -I option.

-V Normally, mbnavedit outputs information to the stderr stream regarding the number of records loaded and dumped. If the -V flag is given, then mbnavedit works in a "verbose" mode and outputs the program version being used, all error status messages, and a large amount of other information including all of the beams flagged or zeroed.

INTERACTIVE CONTROLS

File This menu brings up a popup window which allows the user to specify the input multibeam bathymetry data file, its MBIO format id, the output mode, and the output filename. This program uses the MBIO library and will read or write any multibeam format supported by MBIO. A list of the multibeam data formats currently supported by MBIO and their identifier values is given in the MBIO manual page. If the multibeam data file is named using the MB-System suffix convention (format 11 files end with ".mb11", format 41 files end with ".mb41", etc.), then

the program will automatically use the appropriate format id and the default output filename will have an "e.mbXX" suffix; otherwise the format must be set by the user and the default output filename will be infile with ".ed" appended. The popup window also allows the output mode to be set to "browse" so that no data is output. When a valid file is specified and the OK button is clicked, as much data as will fit into the data buffer is read and the navigition times series plots are displayed.

Next Buffer
This button causes the program to write out the data from the current buffer and then read in and display the next buffer. If there is no more data to be read in after the current buffer has been written out, then the input and output files are closed.

Forward
This button causes the set of displayed pings to step nstep pings forward in the current buffer. The right mouse button causes the same action.

Reverse
This button causes the set of displayed pings to step nstep pings backward in the current buffer. The middle mouse button causes the same action.

Done This button causes the program to write out all of the data from the input file and then close the input and output files.

Quit This button causes the program to exit gracefully. If a data file has been read, all of the data will be written to the output file before exiting.

Pick Clicking on this toggle button sets the edit mode to "pick". In this case, clicking the left mouse button will cause the nearest data value to toggle between selected (red) and unselected.

Select Clicking on this toggle button sets the edit mode to "select". In this case, clicking and dragging the left mouse button will cause any data value touched by the cursor to become selected.

Deselect
Clicking on this toggle button sets the edit mode to "deselect". In this case, clicking and dragging the left mouse button will cause any data value touched by the cursor to become unselected.

Select All
Clicking on this toggle button sets the edit mode to "select all". In this case, clicking in any of the editable plots will cause all of the data in that plot to be selected.

Deselect All
Clicking on this toggle button sets the edit mode to "select all". In this case, clicking in any of the editable plots will cause all of the data in that plot to be unselected.

Time Span Shown
This slider sets the number of seconds shown at a time.

Time Step

This slider sets the number of seconds to step when the Forward or Reverse buttons are pushed.

Interpolate Selection

This button causes the selected data to be replaced by linear interpolation of the surrounding unselected data. If the selection extends to the edge of the data, the selected data will be replaced by the first unselected datum on the other side.

Revert Selection

This button causes the selected data to revert to their original values.

Show Entire Buffer

This button causes the plots to expand to show all of the data in the current buffer.

Pick Time Interval

This button allows users to focus the plots on a particular time interval. Once the button is pushed, the left mouse button is used to select the left edge of the time selection. Similarly, the middle mouse button selects the right edge of the time selection. Both ends of the time selection can be adjusted multiple times. Once the time interval of interest is selected to the users satisfaction, pressing the right mouse button will cause the plots to be redrawn with the selected beginning and ending times.

Longitude Plot

This toggle button turns the longitude plot on and off.

Show Original Data (Longitude Plot)

When this toggle button is on, the longitude plot includes a green line representing the original longitude values.

Latitude Plot

This toggle button turns the latitude plot on and off.

Show Original Data (Latitude Plot)

When this toggle button is on, the latitude plot includes a green line representing the original latitude values.

Speed Plot

This toggle button turns the speed plot on and off.

Show Original Data (Speed Plot)

When this toggle button is on, the speed plot includes a green line representing the original speed values.

Show Speed-Made-Good (Speed Plot)

When this toggle button is on, the speed plot includes a blue line representing the speed-made-good values derived from the longitude and latitude time series.

Use Speed-Made-Good (Speed Plot)

When this button is pushed, any selected data in the speed plot will be replaced by the current estimates of speed-made-good.

Heading Plot

This toggle button turns the heading plot on and off.

Show Original Data (Heading Plot)
When this toggle button is on, the speed plot includes a green line representing the original speed values.

Show Course-Made-Good (Heading Plot)
When this toggle button is on, the heading plot includes a blue line representing the course-made-good values derived from the longitude and latitude time series.

Use Course-Made-Good (Heading Plot)
When this button is pushed, any selected data in the heading plot will be replaced by the current estimates of course-made-good.

Roll, Pitch, and Heave Plots
This toggle button turns the roll, pitch, and heave plots on and off. Unlike the other plots, the data in these plots is not editable. These data are shown purely to allow users to determine if the vertical reference sensors were working properly at the time the data was collected (note: many swath data formats do not contain roll, pitch, and heave data).

MOUSE ACTIONS

Left Mouse Button
The left mouse button is used to pick data values. Unselected data values are shown as filled black squares and selected values as empty red squares. The manner in which data are selected or unselected is controlled by the edit mode, as set by the Pick, Select, Deselect, Select All, and Deselect All buttons.

Middle Mouse Button
The middle mouse button causes the set of displayed data to step backward in the current buffer by the amount of time set on the Time Step slider. The control button Reverse causes the same action.

Right Mouse Button
The right mouse button causes the set of displayed data to step forward in the current buffer by the amount of time set on the Time Step slider. The control button Forward causes the same action.

SEE AALSO
mbsystem(l), mbmerge(l), mbedit(l), mbinfo(l)

BUGS
Please let us know.

9.1.5 mbedit man page

mbedit(l) MB-System 4.4 mbedit(l)

NAME

mbedit - Interactive editor used to flag bad beams in multibeam bathymetry data.

VERSION

Version 4.4

SYNOPSIS

mbedit [-Byr/mo/da/hr/mn/sc -D -Eyr/mo/da/hr/mn/sc -Fformat -Iinfile -V -H]

DESCRIPTION

MBEDIT is an interactive editor used to identify and flag artifacts in multibeam bathymetry data. Once a file has been read in, MBEDIT displays the bathymetry profiles from several pings, allowing the user to identify and flag anomalous beams. Flagging is handled internally by setting depth values negative, so that no information is lost. In extreme circumstances, pings may be zeroed (see "KEYBOARD ACTIONS"). The edited is usually output to a file, but the program can be operated in a "browse" mode where no data is output.

AUTHORSHIP

David W. Caress (caress@lamont.ldeo.columbia.edu)
Dale N. Chayes (dale@lamont.ldeo.columbia.edu)
Lamont-Doherty Earth Observatory
Palisades, NY 10964
Rod McCabe (mccabe@endeavor.asa.org)
David Brock (brockda@endeavor.asa.org)
Antarctic Support Associates
61 Inverness Drive East
Englewood, CO

OPTIONS

-Byr/mo/da/hr/mn/sc

Sets the starting time for data allowed in the input data; pings with times before the starting time will be ignored. Default: yr/mo/da/hr/mn/sc = 1962/2/21/10/30/0.

-D Starts up the program in "browse" mode. If a file is opened in browse mode (either at startup or later), none of the edited data will be output to a file. The default is to output the edited data to a file.

-Eyr/mo/da/hr/mn/sc

Sets the ending time for data allowed in the input data; pings with times after the ending time will be ignored. Default: yr/mo/da/hr/mn/sc = 2062/2/21/10/30/0.

-Fformat

Sets the format at startup for the input and output multibeam data using MBIO integer format identifiers. This value can also be set interactively when specifying the input file. This program uses the MBIO library and will read or write any multibeam format supported by MBIO. A list of the multibeam data formats currently supported by MBIO and their identifier values is given in the MBIO manual page. Default: format = 11.

-H This "help" flag cause the program to print out a description of its operation and then exit immediately.

-Iinfile
Sets the data file from which the input data will be read at startup. This value can also be set interactively. If the input file is named using the MB-System convention of an ".mbXX" suffix (the XX corresponds to the MBIO format id), then the output file name will have an "e.mbXX" suffix. Otherwise, the output file will be infile with ".ed" appended.

-V Normally, MBEDIT outputs information to the stderr stream regarding the number of records loaded and dumped. If the -V flag is given, then MBEDIT works in a "verbose" mode and outputs the program version being used, all error status messages, and a large amount of other information including all of the beams flagged or zeroed.

INTERACTIVE CONTROLS

File This button brings up a popup window which allows the user to specify the input multibeam bathymetry data file, its MBIO format id, the output mode, and the output filename. This program uses the MBIO library and will read or write any multibeam format supported by MBIO. A list of the multibeam data formats currently supported by MBIO and their identifier values is given in the MBIO manual page. If the multibeam data file is named using the MB-System suffix convention (format 11 files end with ".mb11", format 41 files end with ".mb41", etc.), then the program will automatically use the appropriate format id and the default output filename will have an "e.mbXX" suffix; otherwise the format must be set by the user and the default output filename will be infile with ".ed" appended. The popup window also allows the output mode to be set to "browse" so that no data is output. When a valid file is specified and the OK button is clicked, as much data as will fit into the data buffer is read and several pings are displayed as stacked bathymetry profiles.

Quit This button causes the program to exit gracefully. If a data file has been read, all of the data will be written to the output file before exiting.

Next Buffer
This button causes the program to write out the data from the current buffer and then read in and display the next buffer. If there is no more data to be read in after the current buffer has been written out, then the input and output files are closed.

Done This button causes the program to write out all of the data from the input file and then close the input and output files.

Forward
This button causes the set of displayed pings to step nstep pings forward in the current buffer. The right mouse button causes the same action.

Reverse
This button causes the set of displayed pings to step nstep pings backward in the current buffer. The middle mouse button causes the same action.

Mode This choice item allows the user to specify the edit mode. If mode is set to Pick, then clicking the left mouse button will cause the nearest beam to toggle between flagged and unflagged. If mode is set to Erase, then the cursor will change to an erasor and any beam with the cursor while the left mouse button is held down will be flagged. If mode is set to Restore, the behavior will be the same as for Erase except that the affected beams will be unflagged instead of flagged. The edit mode can also be set using key macros (see the keyboard action section):

Pick:	'Q', 'q', 'U', 'u'
Erase:	'W', 'w', 'I', 'i'
Restore:	'E', 'e', 'O', 'o'

X Scale
This slider sets the width of the plot in meters; in general this value should be slightly larger than the swath width of the data being edited. If a particular value is desired which cannot be obtained by dragging the slider (e.g., the user wants a plot width of 10 meters but the slider jumps from 1 to 47), the slider can be changed by increments of 1 by clicking with the left button inside the slider area, but not on the slider itself.
.TP Vertical Exageration This slider sets the depth scale in terms of vertical exageration. The depth scale will change as the cross track distance scale is changed to maintain the same vertical exageration. If a particular value is desired which cannot be obtained by dragging the slider, the slider can be changed by increments of 0.01 by clicking with the left button inside the slider area, but not on the slider itself.

Number of pings shown
This slider sets the number of pings shown at a time.

Number of pings to step
This textitem sets the number of pings to step when the Forward or Reverse buttons are pushed.

Go To This button brings up a popup window which allows the user to specify the time of a particular ping to be displayed. Once the year, month, day, hour, minute, and second values are entered, clicking the Apply button causes mbedit to seek the specified target time. If the current data buffer begins after the target time, an error is returned. If the target time is later than the end of the current data buffer, then mbedit will dump and load buffers until the target time is reached or the data file ends. If the end of the file is reached during the search, the file will be closed.

X Axis Tick Interval
This textitem sets the tick interval in m for the cross track distance axis. If a particular value is desired which cannot be obtained by dragging the slider, the slider can be changed by increments of 1 by clicking with the left button inside the slider area, but not on the slider itself.

Y Axis Tick Interval
This textitem sets the tick interval in m for the depth axis. If a particular

value is desired which cannot be obtained by dragging the slider, the slider can be changed by increments of 1 by clicking with the left button inside the slider area, but not on the slider itself.

Data Buffer Size

This slider sets the number of data records which can be held in the data buffer. Any change becomes effective the next time that a data file is read in.

Buffer Retain Size

This slider sets the number of data records which are held over in the buffer each time the old buffer is written out.

MOUSE ACTIONS

Left Mouse Button

The left mouse button is used to pick beams. Good beams are shown as filled squares and bad (flagged) beams as empty squares. The result of picking a particular beam depends on the current edit mode, as set by the Mode button or keyboard macros defined below. The last picked beam (and ping) is remembered for use with some of the keyboard actions described below.

Middle Mouse Button

The middle mouse button causes the set of displayed pings to step nstep pings backward in the current buffer. The control button Reverse causes the same action.

Right Mouse Button

The right mouse button causes the set of displayed pings to step nstep pings forward in the current buffer. The control button Forward causes the same action.

KEYBOARD ACTIONS

'Z', 'z', 'M', or 'm'

Bad Ping: Pressing one of these keys causes all of the beams in the last picked ping to be flagged as bad.

'S', 's', 'K', or 'k'

Good Ping: Pressing one of these keys causes all of the beams in the last picked ping to be unflagged as good.

'A', 'a', 'J', or 'j'

Left: Pressing one of these keys causes all of the beams including and to the left of the last picked beam to be flagged as bad.

'D', 'd', 'L', or 'l'

Right: Pressing one of these keys causes all of the beams including and to the right of the last picked beam to be flagged as bad.

'!' Zero Ping: Pressing this key causes all of the beams in the ping associated with the last picked beam to be zeroed. This should be used only for completely ridiculous values, as the values are not recoverable.

'Q', 'q', 'U', or 'u'

Pick Mode: Pressing one of these keys sets the edit mode to "pick" so that clicking the left mouse button will cause the nearest beam to toggle between flagged

and unflagged. The edit mode can also be set using the Mode button.

'W', 'w', 'I', or 'i'
Erase Mode: Pressing one of these keys sets the edit mode to "erase" so that clicking the left mouse button will cause any beam under the cursor while the left mouse button is held down to be flagged. The edit mode can also be set using the Mode button.

'E', 'e', 'O', or 'o'
Restore Mode: Pressing one of these keys sets the edit mode to "restore" so that clicking the left mouse button will cause any beam under the cursor while the left mouse button is held down to be unflagged. The edit mode can also be set using the Mode button.

SEE ALSO
mbsystem(l), mbclean(l), mbunclean(l), mbinfo(l)

BUGS
Much fewer than before.

Bibliography

Aangeenbrug, R. T. "A Critique of GIS." In: Maguire, D., M. Goodchild, and D. Rhind (eds.) Geographical Information Systems: Principles and applications. John Wiley and Sons, New York, NY. 1 (1991): 101-107.

Aronoff, S. Geographic Information Systems: A management Perspective. WDL Publications. Ottawa, Canada. (1989): 31-45.

August, P., W. Wright, and R. Bendick. "Incorporating GIS methods into the natural resource manager's toolbox: Lessons from the Rhode Island GIS project." Proceedings of the International Geographic Information System Symposium. American Society of Geographers. Arlington, VA. 3 (1987): 473-485.

August, P., J. Michaud, C. Labash, and C. Smith. "GPS for Environmental Applications: Accuracy and Precision of Locational Data." Photogrammetric Engineering and Remote Sensing. 60.1. (1994): 41-45.

Burroughs, R. "Data Structures for Thematic Maps." :13-38.

Byrne, J. S. The Acquisition, Processing, and Display of Digital Side Scan Sonar Data. M.S. Thesis. The University of Rhode Island, Kingston, (1990). 87 p.

Caress, D. W. and D. N. Chayes. Improved Processing of Hydrosweep DS Multibeam Data on the R/V *Maurice Ewing*. Marine Geophysical Researches, 18:631-650, (1996).

Caswell, D. A. "GIS: The 'Big Picture' in Underwater Search Operations." Sea Technology. 33.2 (1992): 40-47.

Clapp, D. L. The Design of a Sidescan Sonar Data Acquisition System. M.S. Thesis. The University of Rhode Island, Kingston, (1994). 114 p.

Clarke, J.E.H. Swath Bathymetry Multibeam Training Course Overview. Coastal Multibeam Hydrographic Surveys Course Book. US/Canada Hydrographic Commission Coastal Multibeam Working Group. 1994. 19 p.

Cyclades Corporation. Cyclom-Y User's Guide, Version 6.0. Fremont, CA. March, 1995. 33 p.

Danforth, W. W. Xsonar/ShowImage: A Complete Processing and Display System for Rapid Sidescan Sonar Processing. U.S. Geological Survey Open-File Report (in progress). 1996. 59 p.

Doytsher, Y. and J. K. Hall. Gridded affine transformation and rubber-sheeting algorithm with FORTRAN program for calibrating scanned hydrographic survey maps. Computers and Geosciences. 1997. (in press).

Ehlers, M., G. Edwards, and Y. Bedard. "Integration of Remote Sensing with Geographic Information Systems: A Necessary Evolution." Photogrammetric Engineering and Remote Sensing. 55.11 (1989): 1619-1627.

Ertep, S. A. "GRASS Supports Natural Resources Management at Fort Benning." GIS World. 9.9 (1996): 52-55.

Evenden, G. I. Cartographic Projection Procedures for the UNIX Environment - A User's Manual. U.S. Geological Survey Open-File Report 90-284. 1990. 64 p.

Flemming, B. Side-Scan Sonar: A Practical Guide. International Hydrographic Review, LIII:65-92. (1976).

Fox, Jim. RIM User's Manual. University Computing Services, University of Washington. 1990. 98 p.

Gann, J. T. YoNav: Your Own Integrated Navigation System for DOS Platforms. U.S. Geological Survey Open-File Report 92-565. 1992. 62 p.

Guptill, S. C. "Evaluating Geographic Information Systems Technology." Photogrammetric Engineering and Remote Sensing. 55.11 (1989): 1583-1587.

Hale, S. S. "The Narragansett Bay Data System." Current. The Narragansett Bay Project, Providence, RI. (Summer, 1990).

Hall, J. K. Digital topography and bathymetry of the area of the Dead Sea Depression. Tectonophysics 266 (1996): 177-185.

Hatcher Jr., G. A. A Geographic Information System as a Data Management Tool for Seafloor Mapping. M. S. Thesis. The University of Rhode Island, Kingston, (1993). 160 p.

Hydrolab Corp. H20 Multiparameter Water Quality Data Transmitter Operating Manual. Austin, TX. 1991. 63 p.

LaBash, C. L., D. H. Walters, P. V. August, and C. G. Smith. "Standardizing a GIS Data Base." Proceedings of the Ninth Annual Environmental Systems Research Institute User Conference.

Leenhardt, O. Side Scanning Sonar - A Theoretical Study. International Hydrographic Review. LI:62-80. (1974).

The Linux Ethernet HOWTO. http://www.ssc.com/linux/howto.html.

The Linux Hardware Compatibility HOWTO. http://www.ssc.com/linux/howto.html.

The Linux Installation HOWTO. http://www.ssc.com/linux/howto.html.

The Linux Serial HOWTO. http://www.ssc.com/linux/howto.html

The Linux Xfree86 HOWTO. http://www.ssc.com/linux/howto.html

Maguire, D. "An Overview and Definition of GIS." In: Maguire, D., M. Goodchild, and D. Rhind (eds,) Geographical Information Systems: Principles and applications. John Wiley and Sons, New York, NY. 1 (1991): 9-20.

Maher, R. V. "Applications of GIS to the Ocean Environment.. The Ocean - An International Workplace." Proceedings Oceans '87. 3 (1987): 1065-1067.

Merz, T. Ghostscript Manual. http://ftp.cs.wisc.edu/ghost/gs5man_e.pdf. 1997.

NCSA Data Transfer Mechanism Programming Manual Version 2.3. National Center for Supercomputing Applications. University of Illinois at Urbana-Champaign. 1992. 61 p.

Ousterhout, John K. Tcl and the Tk Toolkit. Addison-Wesley Publishing Company. Reading, MA. 1994. 439 p.

Ricketts, P., A. McIver, and M. Butler. "Integrated Information Systems, the Key to Coastal Zone Management." Coastal Zone '89. 5 (1989): 4138-4150.

Shapiro, Michael, James Westervelt, Dave Gerdes, Marjorie Larson, and Kenneth R. Brownfield. GRASS4.1 Programmer's Manual. U.S. Army Construction Engineering Research Laboratory. 1993. 338 p.

Snyder, John P. Map Projections- A Working Manual. U.S. Geological Survey Professional Paper 1395. United States Government Printing Office, Washington. 1987.

Soneira, G., W. Woodward, and C. Noe. "Marine Data Platforms - An Interactive Inventory." Coastal Zone '89. 5 (1989): 4085-4091.

Stevens, W. Richard. UNIX Network Programming. Published by Prentice-Hall, Inc. 1990. 748 p.

Tackett Jr., Jack and David Gunter. Special Edition Using Linux, Second Edition. Que Corporation, 1996.

Terstriep, J. A. and D. E. Weber. The NCSA Data Transfer Mechanism Programming Manual. http://xtc.ncsa.uiuc.edu/DTM/Documentation/DTM2.4/dtm.tp.html. 1993.

Turner, A. K. "What's the Difference Among 2-D, 2.5-D, 3-D, and 4-D?" GIS World. 10.3 (1997): 54.

Tyce, R. "Deep Seafloor Mapping Systems - A Review." Marine Technology Society Journal. 2.4 (1986): 4-16.

Tyce, R.C., Dzurenko, S.M., Cohen, P.A., and D.W. Caress, "A PC/Linux Software Toolkit for Coastal Swath Mapping." Proceedings of the Conference on High Frequency Acoustics in Shallow Water, Lerici, Italy, 30 June to 4 July, NATO SACLANT Undersea Research Centre, La Spezia, Italy. pp. 579-586.

Tyce, R., Hatcher, G. A. "Real Time Geographic Information Systems For Survey Data Management." Proceedings, 22nd Joint Meeting of UJNR Sea-Bottom Surveys Panel. 22. (1993).

Urick, R. Principles of Underwater Sound, 2nd Edition. McGraw-Hill, New York. (1983). 417 p.

Walser, E., R. Hughes, and S. Rabalais. "Multiple Interface Data Acquisition Speeds At-Sea Research." Sea Technology. 33.8 (1992): 29-33.

Welch, Brent B. Practical Programming in Tcl and Tk. Published by Prentice-Hall, Inc. 1995.

Welsh, Matt. Linux Installation and Getting Started. The Linux Documentation Project, ftp://sunsite.unc.edu/pub/Linux/docs/LDP. 1994.

Wessel, P. and W. H. F. Smith. New Version of the Generic Mapping Tools Released. EOS Trans. *AGU* 76, 329. 1995.

Wessel, P. and W. H. F. Smith. The Generic Mapping Tools Version 3 Technical Reference and Cookbook. 1995.

Westervelt, J. Introduction to GRASS4.0. Central Washington University, Extended University Programs, Ellensburg, Washington 98926. (1991).

Woodcock, C. E., C. H. Sham, and B. Shaw. "Comments on Selecting a Geographic Information System for Environmental Management." Environmental Management. 14.3. (1990): 307-315.

Yggdrasil Computing Incorporated. The Linux Bible, The GNU Testament - 2nd Edition. San Jose, CA. 1994. 173 p.