PURL II,
A RAPID DEPLOYMENT
SEARCH AND SURVEY
AUTONOMOUS UNDERWATER VEHICLE

by

Peter D. Helland
B.A.Sc., Simon Fraser University, 1995

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in the School

of

Engineering Science

SIMON FRASER UNIVERSITY

October 1997

0-612-24152-1

Approval

Name: Peter D. Helland

Degree: Master of Applied Science

Title of Thesis: PURL II, A RAPID DEPLOYMENT SEARCH AND SURVEY AUTONOMOUS UNDERWATER VEHICLE

Examining Committee: Dr. John Jones, Chairman

Dr. John Bird
Professor, School of Engineering Science, SFU
Senior Supervisor

Dr. Steve Hardy
Professor, School of Engineering Science, SFU
Supervisor

Dr. Jim Cavers
Professor, School of Engineering Science, SFU
Examiner

Date Approved: October 16, 1997

Abstract

The Underwater Research Lab (URL) at Simon Fraser University in Burnaby, Canada, is conducting research in the use of autonomous underwater vehicles (AUVs) for marine science and site survey applications. AUVs are unmanned, untethered, underwater vehicles that fly through the water without external control. Until recently, AUVs have traditionally been large, expensive vehicles that were employed only for research or military applications. The Underwater Research Lab feels there is a need to develop small AUVs capable of performing a variety of scientific and commercial missions in aqueous environments.

This thesis focuses on the development of a small and inexpensive AUV, called PURL II, that can be rapidly deployed in a remote location with little or no logistical support. Moreover, PURL II will also act as a test bed for acoustic imaging research and limnology research. Improving AUV sensors, and developing techniques for collecting scientific data quickly and efficiently increases the usability of AUVs for potential users. The utility of a small AUV is investigated through various lake trials which culminate in the performance of a scientific mission measuring the internal waves in a small lake. PURL II successfully demonstrates many of the capabilities and limitations of a small AUV developed using only off-the-shelf components.

Acknowledgements

Many people helped make this thesis possible and deserve more recognition and thanks than a mention in this Acknowledgements. I thank John Bird for giving me the room to explore and make the mistakes. Harry Bohm for supporting me throughout my thesis, grounding me in the real world, fabricating the mechanical components, and wiring PURL II. Bernard Laval and Paul Krautener for listening to my crazy ideas and providing insightful feedback on the merits and demerits of these ideas. Andreas Huster, for adding structure to my programming, and solving numerous problems much faster and enthusiastically than I ever could hope to. From International Submarine Engineering, Sam Roberts and Mike Boghart for providing me and the URL with excellent support for the control software PROTEUS. Doug Girling and Dennis Michaelson for friendship and support when development was going poorly. Kevin Maier for working on the altimeter system. And finally my family, for putting up with my school schedule and understanding when I was not a contributing member of the household.

Table Of Contents

List of Tables

List of Figures

1. Introduction

The Underwater Research Lab (URL) at Simon Fraser University in Burnaby, Canada, is conducting research and development in four areas: underwater acoustics, autonomous underwater vehicles (AUVs), limnology, and oceanography. AUVs are unmanned, untethered, underwater vehicles that fly through the water without external control. Until recently, AUVs have traditionally been large, expensive vehicles that were employed only for research or military applications. The Underwater Research Lab feels there is a need to develop small AUVs capable of performing a variety of scientific missions in aqueous environments. This thesis focuses on the development of a small AUV that can be rapidly deployed to a remote site for search and survey missions in lakes. Moreover, the AUV will also act as a test bed for acoustic imaging and limnology research.

This work was done in partnership with International Submarine Engineering Research (ISER) of Port Coquitlam, Canada. ISER is investigating the use of AUVs as instrument platforms for marine science missions where scientific data about phenomenon such as out fall plumes and pollution dispersion are collected. ISER is interested in selling the data to interested parties such as BC Hydro or pulp mills. The URL's focus is on very small AUVs that can be easily handled and rapidly deployed by two or three people in remote locations. Such a small platform can be used for acoustic imaging and limnology research, and also allows the URL and ISER to develop unique techniques for collecting scientific data quickly and efficiently.

1.1 General Background

Fresh and salt water covers approximately seventy percent of the Earth's surface and, when compared to the land masses, is relatively unexplored. From Egyptians on the Nile, to the British Empire, to the oil tankers and cargo freighters of today, controlling and conducting commerce on the water's surface has always been a path to wealth and power. Until recently it was assumed that the waters of the world were infinite sinks for our pollution, and infinite sources for our harvesting activities. With the collapse of fish stocks around the world and pollution washing up on foreign shores, it is now apparent that there is nothing infinite about our waters. However, finite as the world's waters are, they remain essentially unexplored and unknown because of the difficulties involved in penetrating their depths.

Depending on what properties are being measured, there are a variety of conventional methods for collecting data and surveying the world's waters. Divers, profilers, moored sensors, towed arrays, manned vehicles, and remotely operated vehicles (ROVs), are all employed in collecting information about the aqueous environment. Unfortunately, most of these data collection methods require support platforms and equipment that are expensive and have inadequate response times for certain natural phenomenon (Chryssostomidis, 1993). For example, ROVs and towed bodies require support platforms that are directly proportional in size and expense to the size of the ROV or towed body, and the depth of the water to be studied. For the missions proposed by the URL and ISER, these conventional data collection methods require support platforms that are not necessarily large, but they must be transported to, or acquired near, the mission site. Furthermore, the support platforms may be required to support hydraulic equipment such as a winch, or provide significant quantities of electrical power for the support equipment and platforms.

1.2 AUVs

Autonomous underwater vehicles (AUVs) are relatively new tools for collecting data and performing underwater missions. The major difference between AUVs and other data collection tools is that AUVs are self-propelled and not physically connected to a human operator. Other data collection platforms such as ROVs, manned submersibles, towed arrays and moored sensors are not both mobile and disconnected from a human operator. AUVs have not gained wide spread acceptance in the various underwater communities because they have not demonstrated real benefits at a price and level of risk that is acceptable to these communities (Krieder, 1997). When an AUV demonstrates its ability to complete a task cost effectively, and with low risk, then and only then will that AUV be considered viable.

Large, sophisticated and expensive AUVs have been employed by the military for missions in the Arctic and other inaccessible regions because the mission requirements were such that AUVs provided a feasible and economic alternative to other technologies. Cable laying, surveying, mine detection, and mine countermeasures have been the high priority items for naval organisations while collecting oceanographic data has been secondary (Ferguson, 1997; Krieder, 1997). To support anti-submarine warfare activities, the Applied Physics Laboratory at the University of Washington built two AUVs, the Self-Propelled Underwater Research Vehicle (SPURV) in 1967, and the Unmanned Arctic Research Vehicle (UARS) in 1972 (Ferguson, 1997). The Advanced Unmanned Search System (AUSS), fielded in 1984 by the United States Navy's Ocean Systems Center, was the first deep water AUV that performed search and identification missions (Ferguson, 1997; Bellingham, 1993). The AUSS had a depth rating of 6000m, a displacement of 1300 kg, and a range of 130 km at 13 km per hour (Bellingham, 1993). In 1995, the AUV Theseus, built by International Submarine Engineering for the Canadian Department of National Defence, laid a fibre optic cable under the Arctic ice cap during a mission that exceeded 350 km in range (Ferguson, 1997). The special requirements of military organisations allow them to develop and deploy large, sophisticated AUVs because they possess the budget, labour, and support equipment required to handle these vehicles in the field.

Inexpensive and small AUVs had never been built for anything other than technology demonstrations until James Bellingham built Odyssey I at MIT Sea Grant (Ferguson, 1997). The goal of the Odyssey AUV program at MIT Sea Grant is the development of a smaller, longer range AUV that can perform a wide variety of ocean research missions (Bellingham, 1992). These research missions include acoustic mapping of an ice canopy, biological surveys, seep (plume) detection, mid-ocean ridge magnetics surveys, rapid response to episodic ocean events, and bottom imaging and scene reconstruction (Chryssostomidis, 1993). Adding to Odyssey's success is the ability to carry a variety of mission dependent payloads. The Odyssey II AUV displaces 120 kilograms, is 2.2 meters long, has a depth rating of 6000 meters, and an endurance of 6-10 hours at 2-3 knots depending on the sensor payload (Bellingham, 1994). A small, robust AUV does not require the support equipment and personnel that a ROV, towed body or manned submersible requires, thus increasing its utility in remote applications where it is difficult to bring support to bear.

In addition to MIT Sea Grant, other research institutions such as Woods Hole Oceanographic Institute (WHOI), and Florida Atlantic University (FAU) are also conducting research into AUVs and related technologies. The Autonomous Benthic Explorer (ABE) performs scientific surveys of the sea floor for an extended period of time without support from the surface. ABE complements existing manned submersible and ROV technology by repeatedly surveying a hydrothermal vent area over a period of six weeks up to one year (Yoerger, 1990). After each survey iteration, ABE moors itself to a hitching post and goes into a low power sleep mode until it is time to perform another survey iteration (Anderson, 1992). Manned submersibles and ROVs cannot perform repeated surveys of a 6000m deep hydrothermal vent because it is too expensive to be on site for an extended duration of time. ABE solves this problem because surface support is only required for a short period of time during the initial deployment and final retrieval phases of a mission. ABE has a displacement of 450 kg, a 6000m depth rating, a maximum speed of 2 knots, a cruise speed of 1 knot, a total survey distance of at least 30 km, and an on site mission time of up to one year (Yoerger, 1990).

WHOI is also developing extremely small AUVs called REMUS vehicles (Remote Environmental Measuring Units). REMUS vehicles are intended to provide researchers with a simple, low cost, rapid response capability which facilitates the collection of water property data (Alt, 1994). The REMUS concept is similar to the SEA SHUTTLE AUV developed in the 1980's by the Applied Physics Laboratory at the University of Washington. SEA SHUTTLE carried conductivity, temperature and depth (CTD) sensors as payload, and performed autonomous missions under the arctic ice cap. REMUS vehicles displace almost 40kg and are larger than SEA SHUTTLE AUVs, but REMUS vehicles are designed to carry not only CTD sensors, but also dissolved oxygen sensors. REMUS vehicles will be operated and controlled from a short base line acoustic tracking system, or pre-programmed to follow a trajectory determined by bottom moored acoustic transponders (Alt, 1994). The small size, low cost, and mission specific nature of the REMUS vehicles makes them suited to data collection tasks where logistics or response time exclude other survey platforms.

The Ocean Voyager II, developed in the Ocean Engineering Department of Florida Atlantic University (FAU), is designed to perform coastal oceanography for the purpose of sampling coastal regions and ground-truthing satellite spectrometry (Smith, 1994). The primary difference between Ocean Voyager II (OVII) and similar AUVs such as ABE and Odyssey is that OVII is designed for shallow-water long-range missions, not deep-water or under-ice missions. Multiple OVII AUVs can work in conjunction with a support ship to survey a coastal sea floor more efficiently and inexpensively than if the support ship employed traditional techniques such as ROVs or towed sleds (Smith, 1994). The OVII has a 250 kg displacement, 2.4m length, 0.6m diameter, 600m maximum depth, and an eight hour endurance at 3 knots (Smith, 1994). Ocean Voyager II is designed to increase the effectiveness of a surface support ship by increasing the surveyed area for a given cost.

1.3 URL and the PURL Program

In recent years, the Underwater Research Lab (URL) at SFU has focused on underwater acoustic imaging, small AUVs, limnology, and oceanography. The URL has determined that a gap exists in the current AUV research, and this gap manifests itself in the absence of a lake only AUV. Because all of the current AUV research is focused on oceanographic platforms, the AUVs currently being researched may not be appropriate for many potential lake missions because they provide more capability than is necessary in a comparatively benign lake environment. Reducing the capability of an AUV can reduce its cost to manufacture and operate. As stated by Krieder (1997) and Alt (1994), AUVs must offer real benefits over the current technology at a price that users are willing to pay, and a level of risk they are willing to accept. The specifications for lake missions are different than those for ocean missions, and as a result, the URL's PURL (Probe for the URL) program is researching potential applications and specifications for AUVs operating in small lakes. Unlike other AUVs such as ABE, Odyssey or the REMUS vehicles (Yoeger, 1990; Bellingham, 1994; Alt, 1994), the AUVs built for the PURL program avoid custom electrical components wherever possible. The goal of the PURL program is to develop small AUVs that employ commercially available off-the-shelf technology, and then test the capabilities of these AUVs in small lakes.

The PURL program has resulted in the development of two AUVs, PURL I and PURL II. PURL I (Figure 1-1) was designed for small area search and survey missions in lakes, but was too slow and never able to carry a sufficient payload. However, PURL I was considered a success because the URL gained experience designing and operating AUVs, and some of PURL I's software and electronics were ported to PURL II (Figure 1-2). Similar to PURL I, PURL II was designed to perform search and survey missions. In addition, PURL II was also designed to perform rapid response missions, and operate as a research test bed for limnology and acoustic imaging. Also, greater effort was expended refining PURL II's deployment and retrieval system to create an AUV than could be easily deployed by a two or three person team operating in a remote environment with no support equipment except that which can be carried by the team. To facilitate limnology research, acoustic imaging research, and search and survey missions, PURL II has a larger payload and greater speed than PURL I.

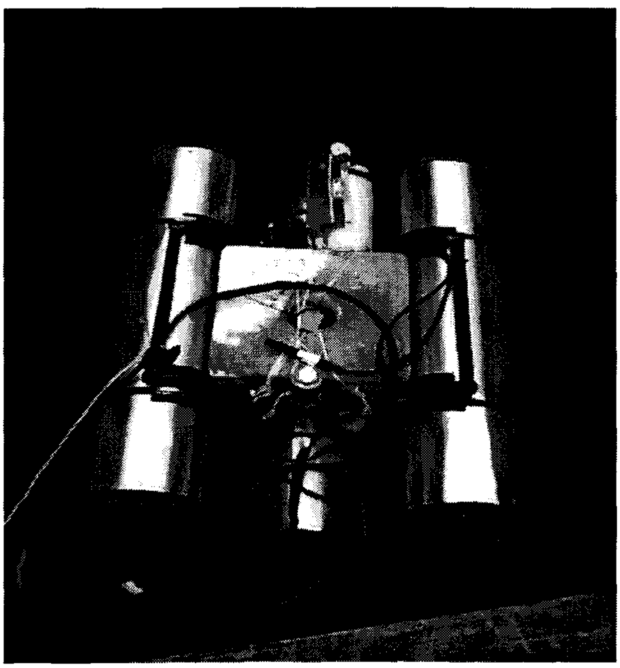


Figure 1-1: PURL I

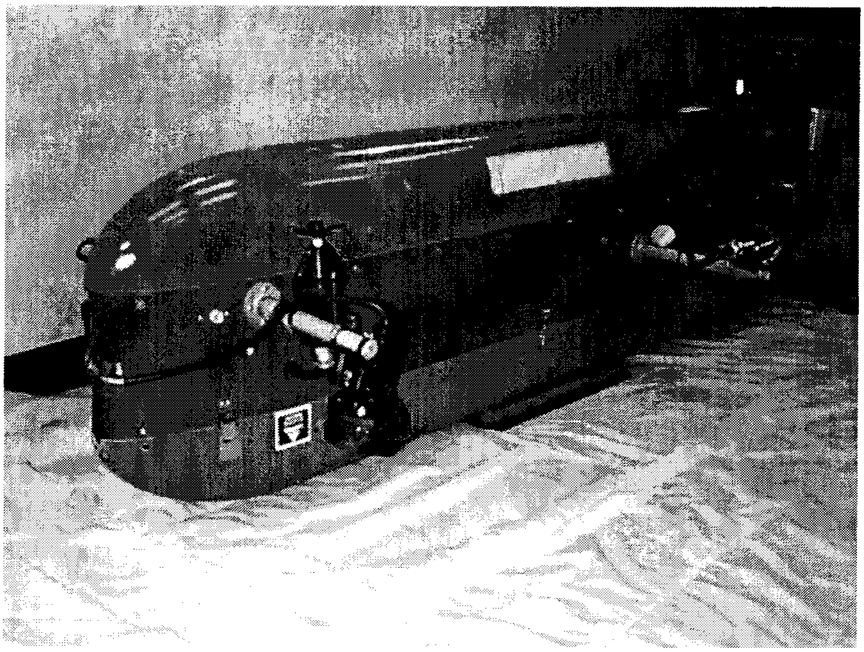


Figure 1-2: PURL II

1.4 Outline Of Thesis

The remainder of this thesis details the design, specifications and field trials of PURL II. First, the many factors influencing the design of PURL II, including the most important factor, being able to meet mission performance specifications, are discussed. Next, the mechanical, electrical, and software designs are described from a functional viewpoint. A detailed description of PURL II's electrical and software implementation is located in the Appendices. The field trial results and performance specifications come next with brief descriptions of the successes and failures of PURL II as a research platform, and search and survey AUV. Finally, conclusions are drawn about the utility of PURL II as an autonomous platform, and the suitability of off-the-shelf technologies for autonomous underwater vehicles.

2. Vehicle Concept and Missions

Autonomous underwater vehicles have been developed in various military and research institutions, and a few companies are even beginning to sell AUVs commercially. PURL II is the URL's autonomous underwater vehicle, and it provides the URL with the ability to perform underwater robotics research, and search and survey missions in lakes. PURL II is not intended to be a commercial product but instead a proof of concept that small AUVs can perform meaningful scientific missions in a variety of operating environments, specifically remote lakes where little support equipment is available. Nevertheless, PURL II could be redeveloped as a commercial product if there were customers to support such an endeavour.

2.1 Vehicle Concept

PURL II is a self-propelled underwater platform capable of delivering a payload to a predetermined area in small lakes. PURL II is designed for two different, yet interconnected, purposes. First, PURL II is designed to be rapidly deployed in a remote lake by two or three people, and once deployed, perform search and survey missions of the lake bottom and water column. Second, PURL II is a URL research test bed for both limnology and underwater acoustic imaging. Depending on the mission or research requirements, PURL II is equipped with different payloads which can be added and subtracted without affecting the standard instrumentation. Employing the standard instrumentation, PURL II is capable of depth keeping, altitude keeping, rudimentary navigation by dead reckoning, and data logging. Combining a research test bed with a rapid deployment search and survey vehicle results in an AUV that carries many different sensor payloads on a variety of lake missions. PURL II is not a commercial product, but instead a proof of concept vehicle showing that a small, rapid deployment AUV can perform search and survey missions in lakes. Doubling as a research platform is an additional requirement for the URL because it facilitates future research and funding for the URL.

As a proof of concept vehicle, PURL II must meet its operational window, but it is not necessary to optimise vehicle parameters such as drag, power consumption, cost, and size. The operational window for PURL II is the ability to perform search and survey missions in lakes while carrying a variety of payloads. This thesis focuses on the components required to perform search and survey missions, but is not an extensive description of the trade-offs made between the various components. The trade-offs that were done involve balancing time, money, components already in our possession (PURL I), donated components, and the window of operation. If they met our operational window, components the URL already possessed were employed whenever possible because they are generally the cheapest and require the least time to integrate. We are a Canadian university research lab that does not have the funding to purchase expensive components or pursue exotic technologies. Achieving the operational window is a demonstration utility under the constraints of finite money, time and labour.

2.2 URL Missions

The window of operation for PURL II is defined by three autonomous missions and a piloted mission. The three autonomous missions are a constant depth mission, a sawtooth mission, and a bottom following mission. When operating autonomously, PURL II is not connected to the surface in any way except during the launch and recovery phases of the mission when communications must be established to start-up and shut down PURL II. During a piloted mission, PURL II is connected to a human pilot via a tether. The payloads will change depending on the type of mission being performed. For example, if PURL II is carrying a conductivity, temperature and depth profiler (CTD), an autonomous limnological research mission could be performed. However, if a video camera is carried, the mission could be a bottom following search and survey mission or a piloted video inspection mission. Table 2-1 lists the payloads that PURL II can carry, and the missions that PURL II can perform with those payloads.

Table 2-1: Payloads For Different Missions And Research

	Autonomous Missions			Piloted Missions	Research	
Payload	Constant Depth	Sawtooth	Bottom Following		Limnology	Imaging
CTD	X	X	X	X	X	
Camera			X	X		X
Side Scan Sonar	X		X	X		X

Note: An 'X' indicates that a payload item is employed for that mission type or research

2.2.1 Autonomous Constant Depth Mission

Constant depth missions are a cornerstone in any AUV's portfolio of missions. Constant depth missions are useful for limnology because they can measure parameters with horizontal variability that are sometimes difficult to measure using traditional techniques. CTD profilers and moored sensor chains operate in a fixed horizontal position, thus requiring multiple chains or profiles to determine horizontal variability. The density of horizontal profiles or moored chains may be too sparse to accurately reconstruct the investigated phenomenon. An AUV travelling at constant depth can provide an excellent platform for side scan sonar imaging of the bottom because an AUV is unaffected by surface conditions such as wind, waves, and ice. Figure 2-1 is a diagram of what a constant depth mission profile may look like.

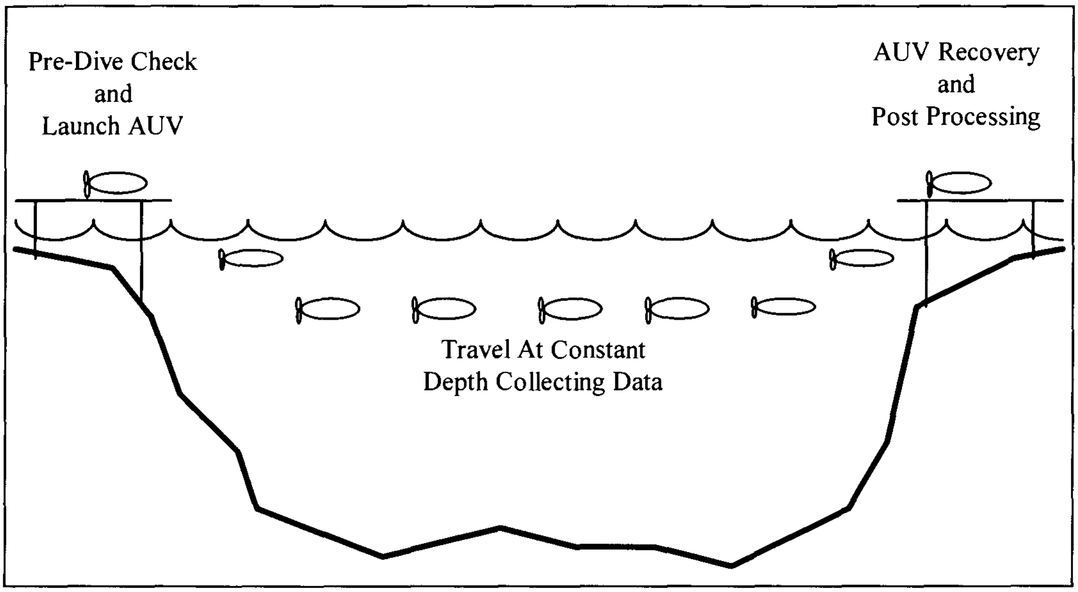


Figure 2-1: Constant Depth Mission Profile

2.2.2 Autonomous Sawtooth Mission

The sawtooth mission is aimed at surveying the water column with a water property profiler such as a CTD. For example, the AUV can repeatedly fly up and down through the water column as the AUV traverses the lake looking for horizontal variability in the temperature structure of the lake. If the lake is warmer in a particular region, this may indicate the presence of a hot fluid out-fall from a source such as a pulp mill or power generation station. Figure 2-2 shows a sawtooth profile and its primary components of launch, execution, recovery, and data processing.

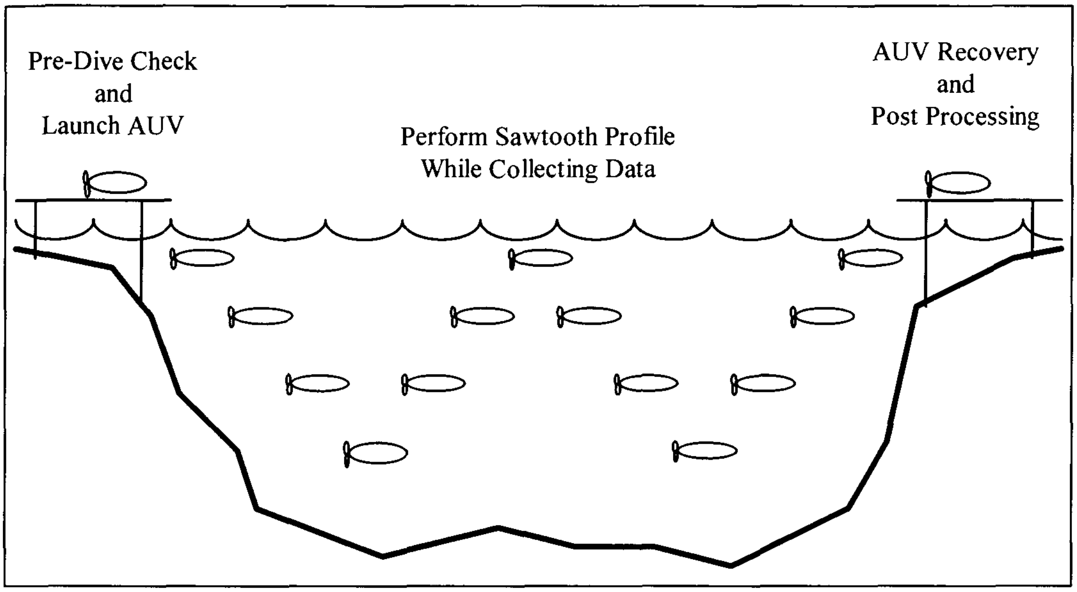


Figure 2-2 : Sawtooth Mission Profile

2.2.3 Autonomous Bottom Following Mission

Bottom following missions are well suited to AUVs because AUVs have better manoeuvrability than other sensor platforms such as towed arrays. The bottom following mission is one of the most dangerous missions for an underwater vehicle because of the risk of collision. Due to the short range of video cameras underwater, the camera must be brought close to the bottom to obtain a useful image. AUVs are one of the few sensor platforms that have both the horizontal and vertical manoeuvrability required to perform a video survey. Platforms such as ROVs are well suited to vertical video inspection, but because of their umbilical, they lack the horizontal manoeuvrability required for long range survey work. Side scan sonar benefits from the constant altitude maintained during a bottom following mission because the swath width will also be constant. Figure 2-3 shows the primary components of a bottom following mission.

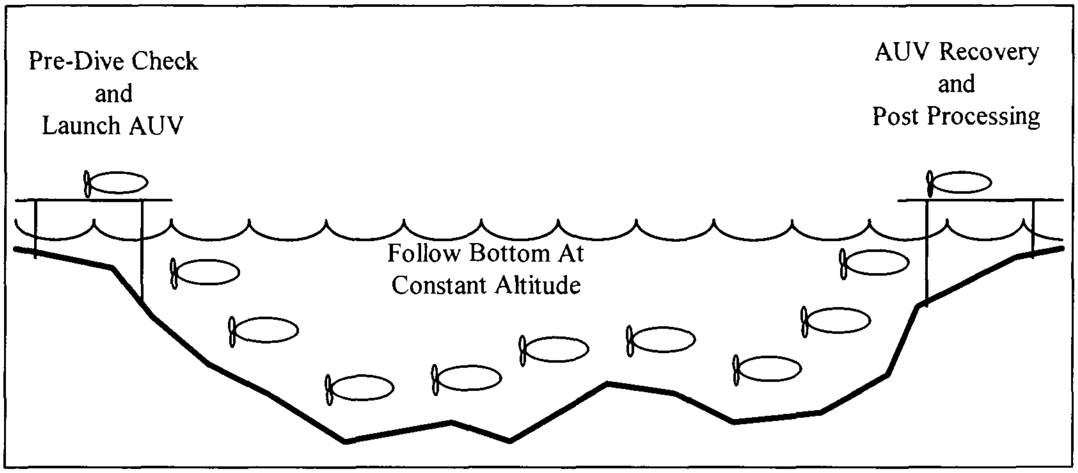


Figure 2-3: Bottom Following Mission Profile

2.2.4 Piloted Mission

When performing a piloted mission the human pilot at the surface controls PURL II via a tether in a manner similar to the way an ROV is controlled via an umbilical. In piloted missions such as video inspection, a pilot's intelligence is particularly useful because only a human is capable of determining what is important during an inspection. The tether also allows the operator to monitor the instrumentation and payload aboard PURL II. During certain research missions or sensor trials, having real time sensor feedback is important. For these missions the data is continuously monitored and evaluated, and the mission modified accordingly. The piloted mission supplements autonomous missions because once an area of interest has been isolated after an autonomous search and survey mission, the tether is connected and a human can investigate the area of interest in detail. Figure 2-4 shows an example of what a piloted mission profile may look like.

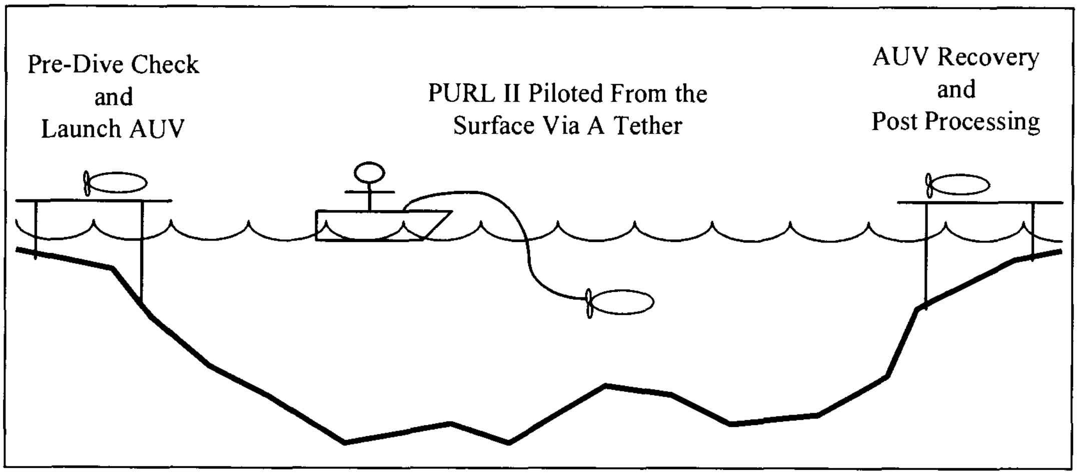


Figure 2-4: A Piloted Mission Profile

3. Design Specifications and Constraints

The design specifications and constraints for PURL II reflect the need to develop a proof-of-concept vehicle that is capable of performing search and survey, and research missions in lakes (Table 3-1). In addition, the specifications also reflect the limited resources that the URL possesses and is willing to allocate to the PURL program.

Table 3-1 : Vehicle Specifications and Design Constraints for PURL II

Desired Completion Date	December 1996
Price	Is there money in the research account?
Maximum Speed	1 m/s
Minimum Speed	Stationary but still maintain depth and heading control
Maximum Depth	70 m
Endurance (min. / max. payload)	2 hours @ 1 m/s / 1 hour @ 1 m/s
Maximum Displacement	70 kg
Maximum Size	2 m long, 0.5 m wide, 0.5 m tall
Pre and Post Autonomous Mission Communication	Telemetry and Video Via Copper Cable
Tethered Mission Communication	Telemetry and Video Via Fibre Optic Cable
Transportation	2 compact cars or one pickup truck
Crew	2 or 3 people
Operating Temperatures	-5 to 40 °C
Navigation	Dead Reckoning
Minimum Instrumentation	Heading, Depth, Altitude, Battery Monitor, Leak Sensor
Payload	CTD Profiler, Video Camera + Lights, Side Scan Sonar, and 2 kg Ballast

3.1 Constraints

Time and money are always two very important constraints in any engineering design, and PURL II is no exception. The desired completion date for PURL II was December 1996, and the maximum cost was not determined by a maximum value but by the URL's cash flow. Components were purchased to minimise cost, and expensive components were purchased only when the URL's research accounts had sufficient funds to cover the expense. Like all ROVs, but few AUVs, PURL II must maintain depth and heading control at zero speed because PURL II must be able to perform inspection tasks. A zero speed requirement precludes the use of actuating planes and rudders for depth and heading control because these control surface require a forward velocity to be effective. An endurance of one to two hours, a maximum depth of 70m, a 70kg displacement, and 2m x 0.5m x 0.5m maximum size were selected so that a two or three person team could operate PURL II in Loon Lake, Maple Ridge (Figure 12-1). With a two hour endurance and 1 m/s maximum velocity, PURL II could perform two out-and-back missions before depleting its energy stores. PURL II and its personnel can be driven to Loon Lake in either two compact cars or a pickup truck, and PURL II can be carried over uneven terrain to the actual launch site.

3.2 Navigation and Basic Instrumentation

PURL II's navigation method has been employed by mariners for many millennia; dead reckoning. The minimum instrumentation required to perform dead reckoning is a heading sensor, a depth sensor, and an altimeter. With these three basic sensors, PURL II can perform the desired missions with very little knowledge about the structure of a lake. Strictly speaking, an altimeter is not required for dead reckoning, but due to lack of knowledge about the lake bottom, an altimeter is required to prevent a bottom collision. Furthermore, an altimeter allows PURL II to navigate by landmarks such as specified bottom depths or bottom events such as pinnacles. The navigation system for PURL II is not sophisticated, but future research in the URL may result in a new acoustic navigation tool which can be tested on PURL II.

Other standard instrumentation aboard PURL II consists of a leak sensor, and a battery monitor. A leak sensor is standard on almost any vehicle because leaks must be caught before they cause serious damage to the electrical components. The battery monitor is employed in order to obtain the maximum endurance from a vehicle before the battery is considered discharged, and to ensure the electronics do not fail because of low input voltages. It is expensive to travel into the field, and it would be an inefficient use of resources if the AUV did not maximise its mission time.

3.3 Payload

Payload delivery is the reason most vehicle exist, be they personal automobiles, jet fighters, or autonomous underwater vehicles. PURL II must provide all the resources (power, space, mountings, buoyancy) required to successfully deploy the payload items listed in Table 3-1. Payload installation and removal from PURL II must also be transparent to the standard instrumentation and control.

Table 3-1: Payload Sensors

Sensor	Specifications
Depth	0 to 100m Accuracy <±0.15% F.S.
Temperature	-3 to 40 °C Accuracy <±0.01°C
Conductivity	0.0 to 7.0 S/m
Side Scan Sonar	Range > 100m, Resolution < 15 cm
Video Camera	Resolution > 400,000 Pixels, Colour or B/W, Low Light

4. Mechanical Design

The mechanical design of PURL II was kept as simple as possible to reduce cost, maintenance, and fabrication time. PURL II is divided into a dry section and a flooded section, and both are enclosed in the fibreglass faring (Figure 4-1). The dry section is housed inside an aluminium pressure vessel that maintains a one atmosphere environment for the batteries and electronics. The flooded section contains the floatation, submersible switch, altimeter, depth sensor, thrusters, aluminium pressure vessel, and payload. If any of the components in the flooded section of the AUV require a one atmosphere environment, they must be in their own pressure canisters.

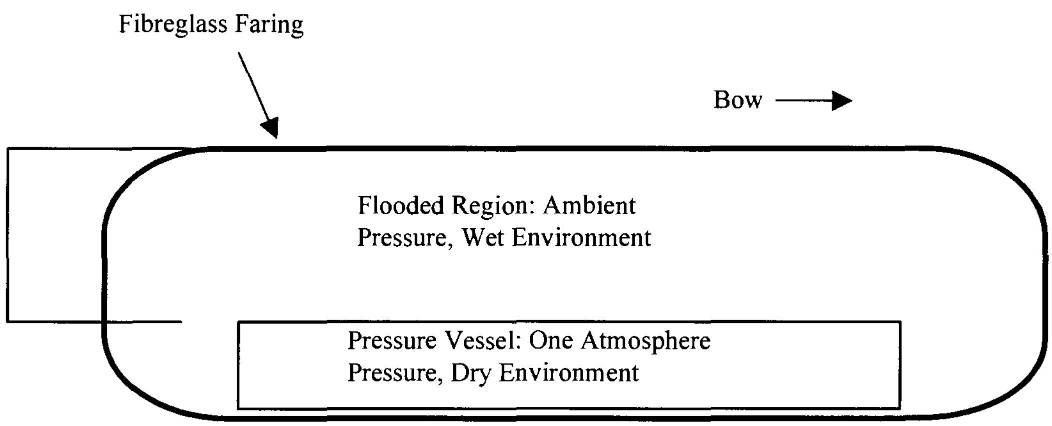


Figure 4-1: Dry and Flooded Sections Of PURL II

4.1 Faring

The fibreglass faring for PURL II was purchased from Simrad Mesotech along with an aluminium pressure vessel. The faring was originally a tow fish body for a side scan sonar, but it was modified in the URL by adding a PVC insert that made the faring eighteen centimetres taller (Figure 4-2). Not having to create our own faring saved both time and money. A new aluminium tail fin was also fabricated in the URL to provide directional stability for the AUV. The faring and tail fin are 1.92m in length and 0.47m in height, the width of the faring is 0.18m, and the width of the tail fin is 0.28m.

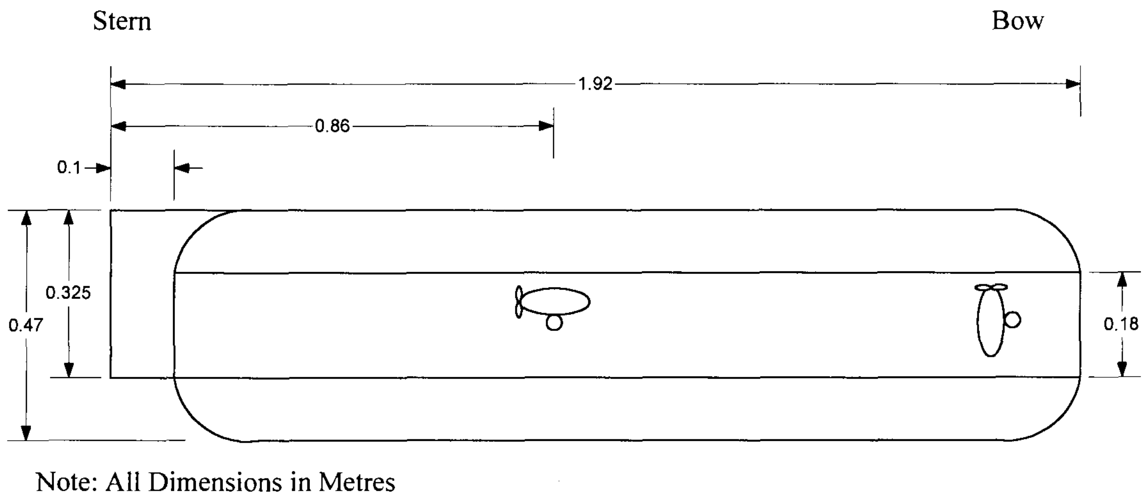


Figure 4-2: Faring and Tail Fin

4.2 Floatation

Rigid floatation is an essential component of any submersible that does not possess a variable ballast system. Floatation provides displacement so that PURL II is slightly positively buoyant. PURL II must maintain positive buoyancy for three reasons. First, when PURL II is resting idle at the beginning or end of a mission, the vertical thrusters are disabled, and positive buoyancy is required to keep the AUV at the surface. Second, in the event of a power or vertical thrusters failure, positive buoyancy is required to return PURL II to the surface for retrieval. Third and finally, the vertical thrusters provide more downward thrust than upward thrust, and positive buoyancy is required to make the ascent and descent rates of PURL II symmetric. If PURL II dives more quickly than it rises, the sawtooth profiles that PURL II creates will not be symmetric. PURL II employs polyurethane foam with a depth rating of 100m and a mass of 290 kg per cubic meter of displacement. The polyurethane foam is located inside the top third of the faring, and helps provide PURL II with a large difference between the location of the centre of gravity (centre of mass) and the centre of buoyancy. This difference is called the b.g. (buoyancy gravity) and is directly proportional to the magnitude of the restoring moment when the vehicle is rotated away from its static equilibrium position (Figure 4-3).

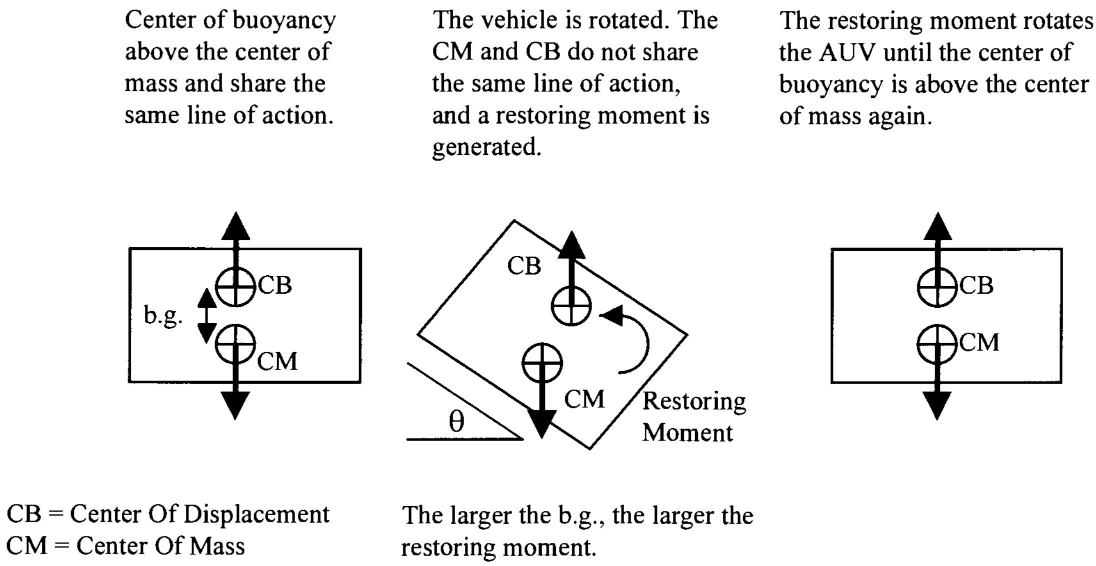


Figure 4-3: The Buoyancy/Gravity And Restoring Moment For A Submerged Body

The equation of the restoring moment for a submerged neutrally buoyant body is:

$$\text{Restoring Moment} = \text{Mass} * g * \text{b.g.} * \sin\theta = \text{Displacement} * \rho * g * \text{b.g.} * \sin\theta \quad (1)$$

where the mass is in kilograms, displacement in cubic meters, b.g. is expressed in meters, ρ is the density of water in kilograms per cubic meter, g is 9.81 N/kg and θ is the pitch or roll angle. The restoring moment will have units of Nm as expected in the S.I. system. As a source of displacement for a small and inexpensive AUV, polyurethane foam is appropriate because it is simple and has no moving parts or seals that can fail.

4.3 Pressure Vessel

PURL II's aluminium pressure vessel provides a dry, one atmosphere environment for the batteries and electronics. When the pressure vessel was originally employed in a Simrad Mesotech tow fish, the depth rating was greater than 300 meters, but we are only employing it to the 70 meter depth rating of PURL II. Figure 4-4 is a picture of the aluminium pressure vessel and Table 4-1 lists its specifications.

Figure 4-4: Aluminium Pressure Vessel

Table 4-1 : Pressure Vessel

Material	Shape	Dimensions (cm)	Mass (kg)	Displacement (kg)	Notes
6061 Aluminium	Ribbed Cylinder with Flat End Caps	15 ID, 16.5 OD, 122 Length	8.3w/o endcaps, 10.8 w endcaps	25.3	Cylinder is showing some pitting due to corrosion.

Note - ID = inner diameter, OD = outer diameter

4.4 Actuators / Propulsion

Propulsion, heading, and depth control are the largest energy consumers in AUVs. PURL II employs four thrusters for propulsion, heading and depth control. The two thrusters mounted horizontally control the heading and propulsion, and the two thrusters mounted vertically control the depth (Figure 4-2 and Figure 4-5).

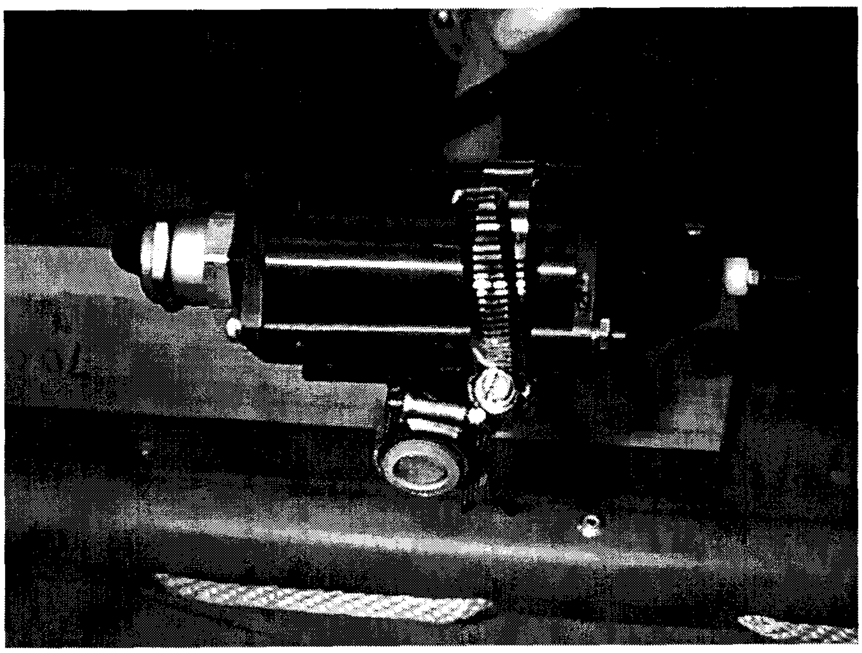


Figure 4-5: Thruster

The thrusters were designed and fabricated by Inuktun Services (Table 4-1). Control planes are not employed to control the heading because PURL II has a zero speed manoeuvring requirement which precludes the sole use of planes. Mounting thrusters on side mounted struts reduces the likelihood of snarling the fibre optic tether in the thrusters. Also, the mechanical complexity of strut mounted thrusters is low in comparison to actuating surfaces such as planes or rudders. The absence of a protective cage renders the thrusters vulnerable to damage during transport and handling. For zero speed control, simplicity, low cost, low weight, and easy maintenance, strut mounted thrusters are consistent with the goals of PURL II.

Table 4-1: Inuktun Thruster Specifications

Motor	Depth Rating	Weight in Air	Weight in Water	Thrust	Power Consumption	Input Voltage	Encoders
Pitman #9412	50m	0.43 kg	0.17 kg	0.7 kg @ 0.6 m/s	50 W	24 VDC	Incremental, 512 Counts Per Revolution

4.5 Card Cage

The card cage is an internal frame inside the pressure vessel where the batteries, wiring and electronics are mounted. A strong, light and versatile card cage is essential for any submersible system where easy access to the electrical systems must be maintained. Mounting components inside pressure vessels is often troublesome because pressure vessels are either spherical or cylindrical, but the components are generally rectangular. The card cage creates a rectangular space and consists of three mounting bars running the entire length of the card cage, and two shorter bars at the fore and aft ends. The open space between the two shorter bars is used for changing the battery during field operations. The mounting bars are connected by octagonal bulkheads which provide rigidity and additional mounting surfaces (Figure 4-6). The bulkheads are 0.25 inch thick octagonal PVC plates, and the mounting bars are 0.125 inches by 1.0 inches by 46.75 inches with two rows of countersunk 4-40 machine screw holes with 0.5 inch spacing. This card cage allowed the URL to pack the components quite tightly, thus reducing the overall size and weight of PURL II.

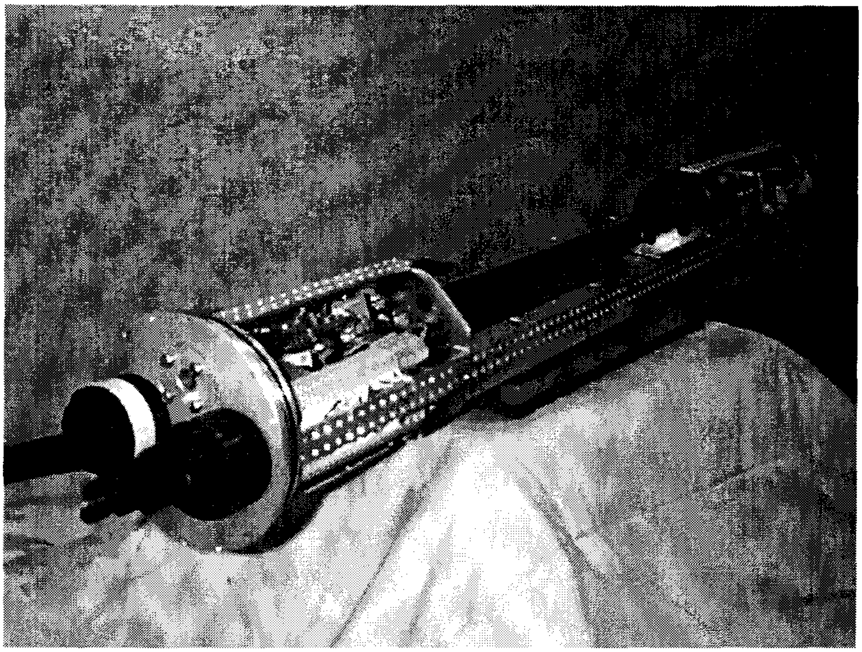


Figure 4-6: Card Cage And Electrical Components

4.6 Cabling And Penetrators

The cabling and bulkhead penetrators for PURL II were kept simple and robust as appropriate for an underwater vehicle deployed with little logistical support. The two SEACON AWQ-6/36 penetrators each provide six pie-shaped connectors with six 18 AWG Teflon insulated conductors (Figure 4-7). To distinguish between the two penetrators and their twelve cables, coloured electrical tape was wrapped around the penetrators and the ends of the cables. The penetrators were wrapped with one band of either blue or yellow electrical tape. The cables connecting into the blue penetrator were labelled with one to six bands of blue tape corresponding to their position in the alphabet. Position ‘A’ was represented by one band, ‘B’ by two bands, and so on up to ‘F’ with six bands of blue tape. The yellow penetrator and cables followed the same scheme as the blue penetrator. The port for the fibre optic cable is occupied by a one inch blanking plate when the fibre optic tether is not connected, and a fibre optic penetrator when the fibre optic tether is connected. The seventy two conductors and the fibre optic penetrator enter the pressure vessel through the aft end of the pressure vessel.

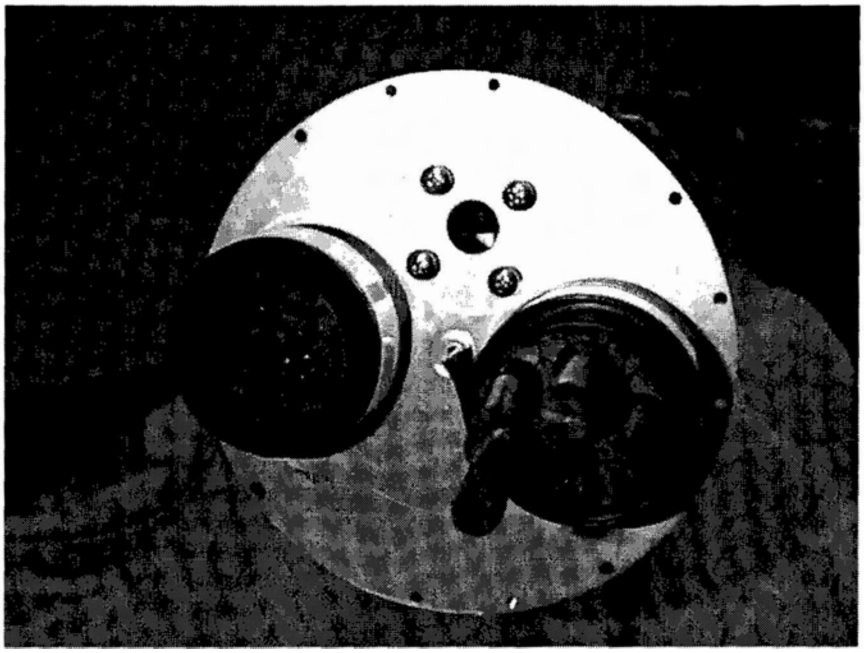


Figure 4-7: SEACON Penetrators And The Fibre Optic Port

4.7 Mass and Displacement

The mass and displacement of any submersible vehicle are extremely important because these parameters in part determine the utility, size, and cost of a submersible vehicle. As the mass of a vehicle increases, the displacement must also increase to maintain neutral (or slightly positive) buoyancy. If the mass increases beyond the displacement achievable in a particular vehicle volume, the physical extents of the vehicle must be increased to gain more displacement. Successful submersible vehicles have a significant payload capacity in the sense that the displacement of these vehicles is large enough that payloads can be added without becoming negatively buoyant. A summary of the mass and displacement of PURL II's components is shown in Table 4-1. Without any payload or batteries, the total mass of PURL II is 49 kg, and with battery pack Beta added the mass increases to 66.2 kg. The displacement of PURL II without any payload is 71.1 kg, which provides PURL II with a payload carrying capacity of 4.9kg of wet weight. Wet weight is the difference between the mass of the payload and its displacement. When the CTD profiler, CTD pump, video camera and lights are added, the mass of PURL II increases to 74.8 kg with a displacement of 76.8 kg leaving 2.0 kg free for additional payload. To obtain slightly positive buoyancy and proper trim, PURL II is ballasted with lead.

Table 4-1: Mass and Displacement

Component	Mass (kg)	Displacement (kg of H_2O)
Battery Pack Alpha (24V 24Ah)	17.0	0
Battery Pack Beta (24V 24Ah)	17.2	0
Battery Pack Gamma (24V 20Ah)	15.8	0
Aluminium Pressure Vessel + End Caps	10.3	25.3
Faring and Floatation	23.1	40.8
Four Thrusters and Cables	4.1	2.5
Card Cage with Electrical and Electronics Components	5.9	0
Altimeter and Cable	3.4	2.0
Depth Sensor, Submersible Switch and Cable	1.8	0.37
Ethernet Cable	0.4	0.14
Video Camera, Lights and Cables	2.1	1.2
SBE-19 CTD Profiler, CTD Pump and Cable	6.5	4.5
Lead Ballast	To Trim	To Trim

5. Electrical Design

The electrical design for PURL II focuses on simplicity. Each of the electrical subsystems is easily isolated from the whole so that in the event of a failure, diagnostics and bench testing are easy to perform. High quality connectors are employed throughout PURL II to ensure reliable electrical connections despite the vibration and jarring that PURL II experiences during transport, handling, and missions. Furthermore, printed circuit boards were created for all the non-commercial circuits for which point-to-point soldering or wire wrapping techniques are often employed.

5.1 Power Distribution

The power distribution system for PURL II is divided into two groups of buses, the main power supply buses, and the CPU power supply buses (Table 5-1). PURL II's battery pack provides 24 VDC and is comprised of sealed lead acid batteries which were chosen for their low cost and ability to survive abuse. Appendix One contains the schematic representation of the power distribution system.

Table 5-1: Power Supply Buses

Buses	Voltages (VDC)
Main Supply	+24, +15, -15, +5, GND
CPU Supply	+12, -12, +5, -5, GND

5.1.1 Battery Pack

Secondary (rechargeable) batteries are the standard energy source for almost all existing AUVs. Secondary batteries generally have a higher initial purchase cost and lower energy density than primary (non-rechargeable) batteries, but because they can be recharged, their cost is amortised over many missions. Power for PURL II is provided by a rechargeable battery pack located inside the main pressure vessel (Figure 5-1). Three battery packs are maintained so that a fully recharged pack is always available.

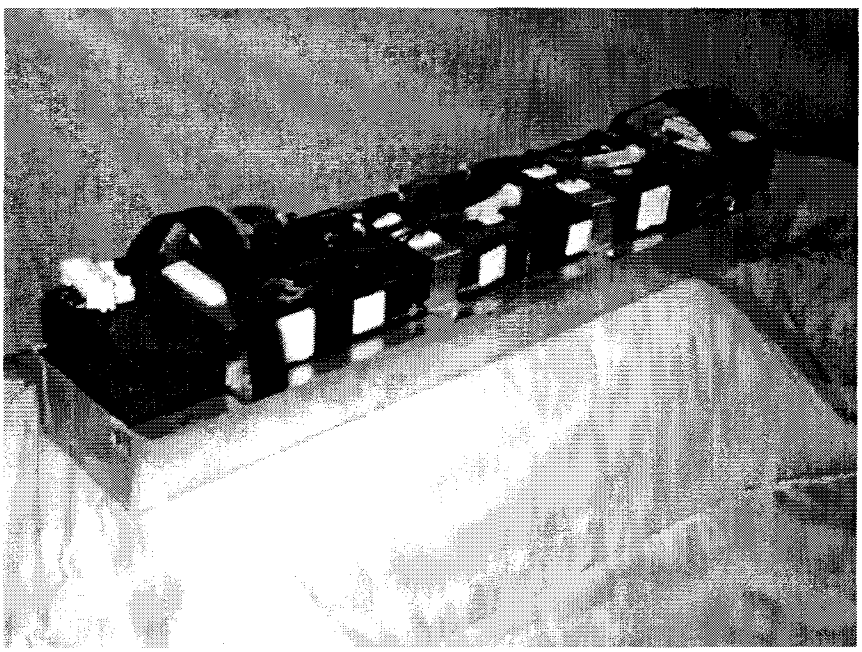


Figure 5-1: Battery Pack

The battery packs are comprised of four sealed lead acid batteries which combine to create a 24VDC pack rated at 24 amp hours based on a 20 hour discharge rate at 25°C. Although sealed lead acid batteries have a poor energy density when compared to other secondary batteries such as Silver Zinc, Nickel Metal Hydride or Lithium Ion, they have other attributes which make them well suited for PURL II. Only sealed lead acid batteries are inexpensive, easily maintained, tolerant to improper recharging practices, tolerant to deep cycle discharging, tolerant to shallow discharging, and possess a long cycle life. Because PURL II may operate in water as cold as 0°C, and discharge times range from one to three hours, the battery pack may be de-rated as much as 40% for a one hour discharge rate, or 25% for a three hour discharge rate. Fortunately, the battery pack shares the pressure vessel with heat generating devices such as electronics and motor controllers, and therefore it is not anticipated that the temperature inside the pressure vessel will ever fall to zero degrees Celsius. If users require more energy for a mission than can be supplied by a sealed lead acid battery pack, they can build their own packs as long as the packs conform to the physical size, output voltage, and connector specifications of the original sealed lead acid battery pack (Table 5-1 and Appendix One).

Table 5-1: Battery Pack Specifications

Battery Pack	Output Voltage	Amp Hours	Watt Hours	Mass (kg)	Dimensions l x w x h	Notes
α	24	24	576	17	61 x 10 x 9.5 cm	Four 12 Ah Cells
β	24	24	576	17.2	61 x 10 x 9.5 cm	Four 12 Ah Cells
χ	24	20	480	15.8	61 x 10 x 9.5 cm	Newest Pack, Four 10 Ah Cells

Note: All specifications are at 25°C with a 20 hour discharge rate.

5.1.2 Power Switch and Relay PCB

PURL II employs a submersible switch for turning power off and on. A submersible switch is much more expensive than a magnetic reed switch or a cable with a shorting plug, but the URL has found that once in the field, switching power off and on, and knowing its present state is sometimes difficult. However, with a submersible switch the power is either off or on, and there is no doubt about its state. The submersible switch controls five relays, the first four control power to the thrusters, and the fifth switches power to the four main buses (+24, +15, -15, +5). The relay outputs are fused at five amps for the thrusters, and three amps for the buses.

5.1.3 Main Power Supply Buses

The main power supply buses are +24 VDC, ±15 VDC, and +5 VDC. The +24 VDC bus is connected directly to the Relay PCB, and the other buses are connected to the Relay PCB through Vicor DC to DC converters. The converters take +24VDC inputs, have a 25 Watt maximum output, and are between 80% to 90% efficient depending on loading and output voltage. All the buses share a common ground which is connected to the negative side of the battery pack.

5.1.4 CPU Power Supply Buses

The PC-104 stack has a Tri-M Systems Inc. power supply card, the HE104-512-16, that outputs ±5 VDC, and ±12 VDC for the CPU power supply buses. This power supply card provides power for the PC-104 stack and the components connected to the PC-104 stack. The input for the Tri-M power supply card is connected to the +24 VDC main power supply bus and shares a common ground with the main supply buses. The specifications for the HE104-512-16 are listed in Table 5-1.

Table 5-1: Specifications For The HE104-512-16 Power Supply Card

HE104-512-16 Specifications	
5V Output	10 Amps including current supplied to 12V, -12V and -5V regulators
12 V Output	2 Amps
-5 V Output	0.4 Amps
-12 V Output	0.5 Amps
Input Range	6 to 40 V
Efficiency	< 95%
Temperature Range	-40°C to 85°C
Output Ripple	20 mV on 5V supply

5.2 CPU and PC-104 Stack

The embedded controller for PURL II is a PC-104 stack which employs an 80486-DX4 100 as the CPU. The PC-104 standard was chosen for the PURL program because it is off the shelf, small, inexpensive, moderately rugged, a low power consumer, and it provides native mode software development. The PC-104 stack has six cards and they provide all the I/O, data storage, and processing required to operate the standard equipment aboard PURL II (Table 5-1). Table 5-2 lists the I/O requirements for PURL II, and the resources currently supplied by the PC-104 stack. When operating autonomously PURL II performs data logging, and time stamps the data so it can be correlated during post processing.

Table 5-1: PC-104 Stack Cards

Manufacturer: Card	Resources
Advanced Digital Logic Inc.: MSM 486 DX6 100	Intel 486 DX4-100 CPU, 16 Mbytes Ram, VGA controller, floppy controller, hard drive controller, 2 RS-232 serial ports, 1 parallel port, keyboard controller, speaker.
Tri-M: HE104-512-16	± 5V output, ±12V output
Sealevel Communications & I/O: C4-104-3521	4 RS-232 Ports
Diamond Systems Corporation: Sapphire-MM	8 14-bit differential analogue inputs, 2 12-bit analogue outputs, 4 TTL inputs, 4 TTL outputs.
Ampro: MiniModule /Ethernet-II	1 10Base-T Ethernet Port or 1 AUI Port
URL Breakout	A breakout card for the PC-104 power supply buses and the Sapphire-MM inputs and outputs.

Table 5-2: PC-104 Stack Resources, Supply and Demand

Signal	Required	Supplied
14-bit Analogue Inputs	3	8
8-bit Analogue Outputs	0	4
RS-232 Serial Ports	4	4 (7 with a PROTEUS upgrade)
Parallel Ports	0	1
Digital Inputs	1-bit	4-bits
Digital Outputs	2-bits	4-bits
UTP Ethernet Ports	1	1
+5 VDC	5.2 Amps*	10 Amps (including current to +12V, -12V and -5V buses)
+12 VDC	0.6 Amps*	2 Amps
-5 VDC	0*	0.4 Amps
-12 VDC	0.06 Amps*	0.5 Amps
VGA Video	1	1
Keyboard + Speaker	1	1
IDE Hard Disk Controller	1	1

*Note: The required currents are steady state values, the peak values are higher. Therefore, when adding new components, the required currents should be multiplied by a safety factor of 1.5 to determine if there is enough current available from the power supply during peak periods.

5.3 Instrumentation

The standard instrumentation aboard PURL II is employed for navigation, status monitoring and alarms. All of the instrumentation interfaces with the PC-104 stack, and is monitored and controlled through the PROTEUS software. The instrumentation suite is the minimum suite required to meet PURL II's operational window.

5.3.1 Navigation

Navigation of PURL II is performed by dead reckoning. With a chronometer (clock), compass, depth sensor, and altimeter PURL II was able to successfully and repeatedly execute a variety of survey missions. The chronometer is the heart of any navigation system and the CPU clock is more than sufficient for all of PURL II's navigational requirements. A heading reference and depth sensor are the two most important sensors in the navigation system. As was demonstrated by PURL I, many useful missions can be performed in areas free of obstacles when only heading, depth and time are combined. The third navigation sensor is an altimeter. The altimeter measures the time of flight of an acoustic pulse to determine the altitude of PURL II above the bottom. The altimeter gives PURL II bottom following, bottom avoidance, and bathymetry generation capabilities. For example, PURL II can fly at a predetermined altitude off the bottom while performing a video or side scan sonar survey, with the heading, depth and time determining where the AUV is located. Table 5-1 lists the sensors employed for navigating PURL II through the water and over the bottom.

Table 5-1: Navigation Sensors For PURL II

Sensor	Range	Accuracy	Resolution	Interface	Power	Notes
KVH C-100 Fluxgate Compass	±180°	±0.5°	0.1°	RS-232, COM 1, 9600 Baud, IRQ 4, Address 0x3F8	0.51 W	Accuracy is reduced by ferrous materials and local magnetic fields.
Pressure Sensor	0 to 70m	±0.20m	0.01m	RS-232, COM 3, 9600 Baud, IRQ 5, Address 0x3E8	1.25 W	Hysteresis of less than 0.05m
Simrad Mesotech Mdl. 819 Altimeter	0.75 to 200m	±0.125m	0.125m	RS-232, COM 4, 4800 Baud, IRQ 7, Address 0x2E8	4.49 W	Distances less than 0.75m are reported as 0.75m

5.3.2 Pitch and Roll

Measuring pitch and roll (attitude) is not required for PURL II to meet its operational window, but attitude can be employed as a diagnostics tool and performance monitor. During the pre-mission check, PURL II is ballasted so that it is positively buoyant with zero pitch and roll. Establishing zero attitude minimises the energy required to maintain constant depth because a tendency to pitch up or down must be compensated by increased energy consumption in the vertical thrusters. Table 5-1 lists the specifications for the pitch and roll sensor employed on PURL II.

Table 5-1: Spectron Systems Technology Inc. Dual Axis Inclinometer

Model	Range	Accuracy	Resolution	Interface	Time Constant	Power	Notes
SSY0091	±20°	±0.1° to 10°, ±0.6° to 20°	0.002°	400mV per degree	150ms	0.19 W	The output is displayed in increments of 0.1°

5.3.3 Monitoring and Alarms

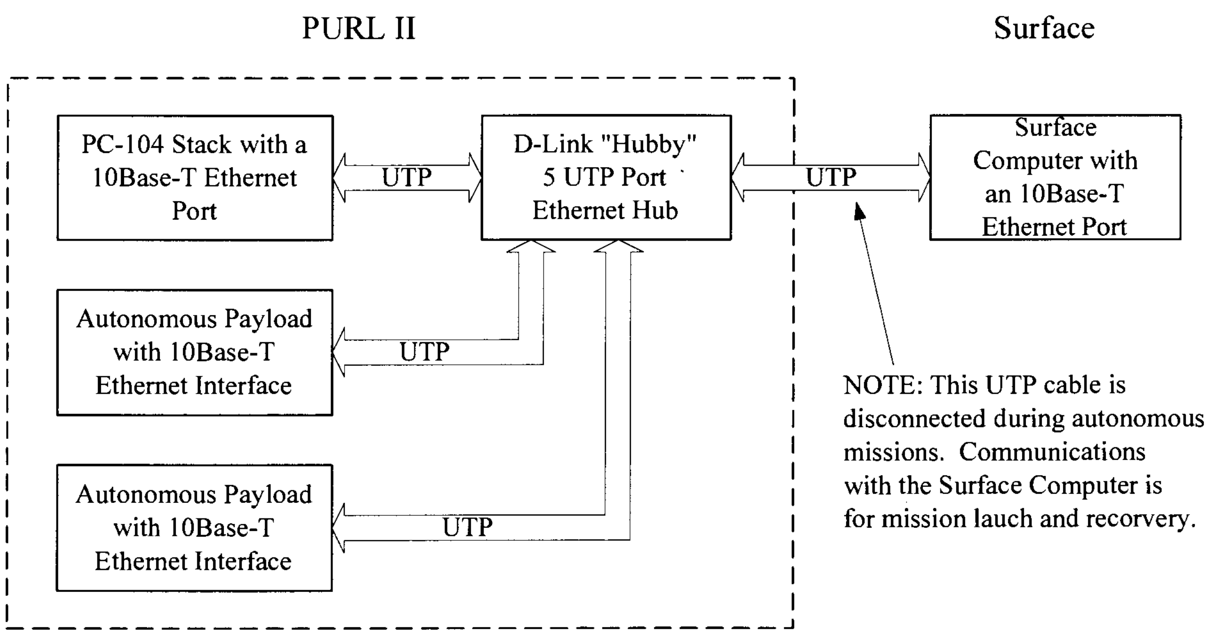
PURL II lacks the system monitoring and redundancy that typify manned vehicles because PURL II is small, and human life is not endangered. The monitored items are pressure vessel leaks, low battery voltage, the presence of telemetry, and communication with the motor controllers. Regardless of the mode of operation, if the leak sensor detects a leak or the battery voltage falls below 20 VDC, PURL II goes into ABORT mode. If the telemetry is lost for more than forty seconds when PURL II is tethered and operating in PILOT mode, PURL II also goes into ABORT mode. Upon entering ABORT mode, the horizontal thrusters are set to full forward, and the vertical thrusters are set to full up to return PURL II to the surface for retrieval.

When the motor controllers are operating during an autonomous mission, a watch dog timer is enabled. The watch dog timer is reset whenever the motor controllers receive messages via their RS-232 serial link. If communications is lost and the watch dog expires, it is assumed a software or hardware failure has occurred, and the horizontal thrusters are set to full forward and the vertical thrusters to full up. The fault management philosophy for PURL II is that if anything goes wrong, return to the surface and stay there.

5.4 Ethernet Network

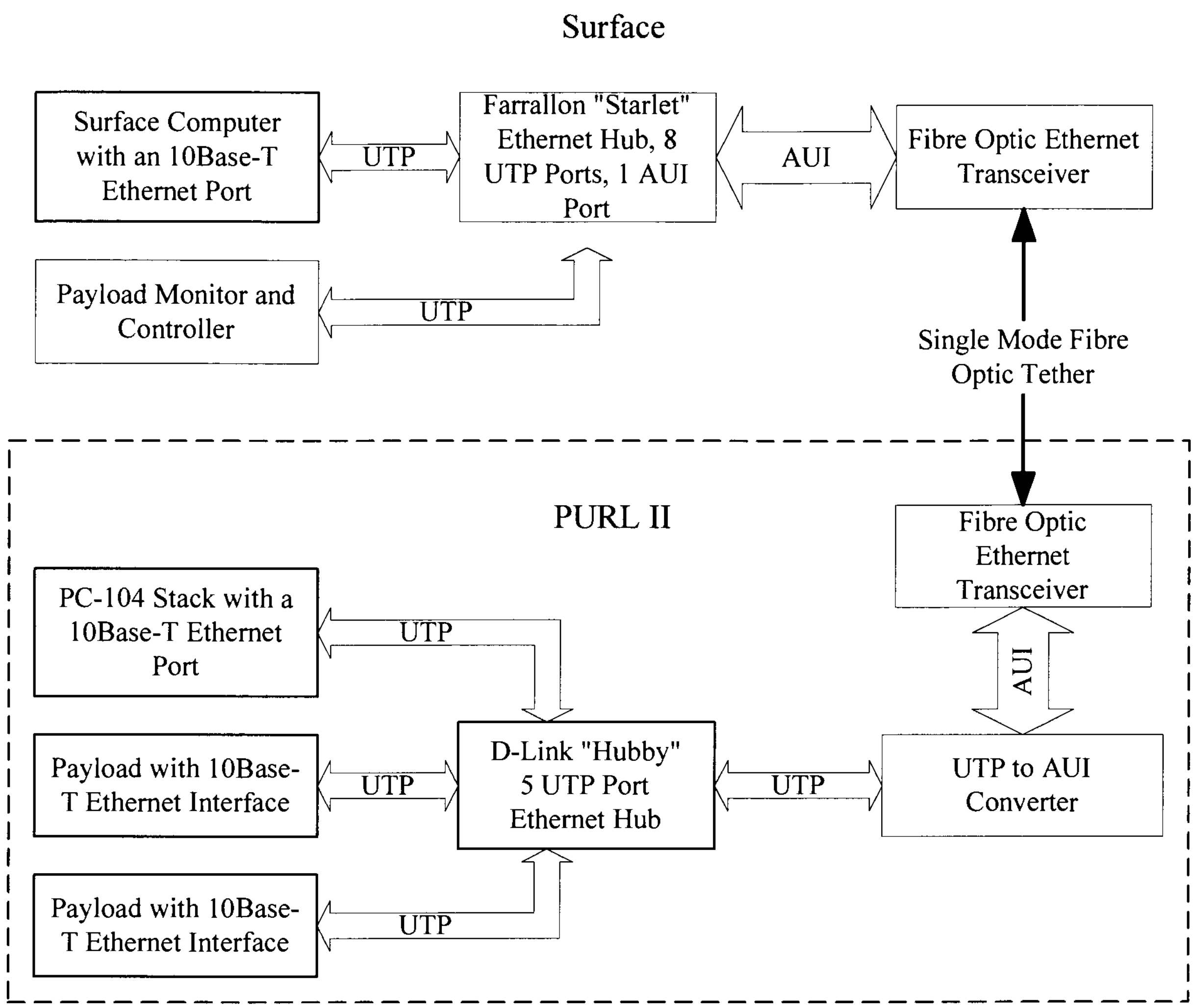
Communications between PURL II, the surface, and Ethernet capable payloads is conducted through an Ethernet network. An Ethernet network was chosen because off the shelf equipment is widely available, it allows multiple nodes to communicate with each other over a single medium, and the addition or subtraction of nodes can be transparent to the other nodes in the network. PURL II does not currently carry any Ethernet capable payloads, but the URL is planning to add a digital side scan sonar which may employ an Ethernet interface.

Depending on whether PURL II is operating autonomously (without the fibre optic cable) or is piloted (with the fibre optic cable), the Ethernet network is configured differently. During autonomous operation, PURL II is not connected to the surface computer except during start-up, pre-mission and post-mission systems checks, and shut down. Because surface monitoring is not occurring during an autonomous mission, a single Ethernet node is sufficient for the pre and post mission communications (Figure 5-2). The autonomous Ethernet configuration has a single five port hub located aboard PURL II, with a single unshielded twisted pair (UTP) cable to the surface computer. For piloted missions where monitoring and control is performed by a human operator, multiple surface nodes may be monitoring the PC-104 stack as well as scientific payloads that communicate via Ethernet (Figure 5-3). To facilitate multiple nodes at the surface, the fibre optic backbone is connected to a hub which can handle up to eight surface nodes. Employing an Ethernet network allows a small AUV such as PURL II to support multiple configurations and payloads with no impact on the internal wiring. If wiring changes were required each time the vehicle changed missions and payloads, PURL II would lose its ability to perform as a rapid deployment vehicle, and also run an increased risk of faulty wiring.



Note: UTP - Unshielded Twisted Pair (10Base-T)

Figure 5-2: Autonomous Ethernet Configuration



Note: UTP - Unshielded Twisted Pair (10Base-T)
AUI - Attachment Unit Interface

Figure 5-3: Piloted Ethernet Configuration

6. Software Design

Similar to many other AUVs, PURL II has a significant software component which handles control, mission execution, telemetry, and sensor interfacing. To simplify software development for both PURL I and PURL II, an AUV control software package called PROTEUS was utilised. PROTEUS was developed by International Submarine Engineering Research, in Port Coquitlam, Canada. Utilising an off-the-shelf software package greatly reduced the development time for PURL II, and provided the URL with a stable platform from which to develop the software components specific to PURL II.

6.1 PROTEUS

PROTEUS is a real-time scheduler developed by International Submarine Engineering Research for controlling remotely operated and autonomous underwater vehicles. PROTEUS employs an object-oriented software architecture implemented in C++. New software modules, called components, added to the overall software system can be based on existing components via class inheritance (ISER, 1991). From the users' standpoint, the most important property of PROTEUS is that the control systems, telemetry, and mission scripting can be modified without working on the underlying C++ source code. An AUV's control system and mission scripts are entirely described by a set of text configuration files which are parsed by the mission executor at run time and executed by the PROTEUS kernel. Because the text files are parsed at run time, configuration changes can be implemented without re-compiling PROTEUS. The ability to reconfigure the system is particularly useful while testing and during field work where parameters must be changed to suit new situations. All of the control loops and signal arbitration are fixed throughout a mission however, and changes to the configuration files must be made off line while PROTEUS is not running.

6.2 Event-Based Software

Data propagation is the fundamental operating mechanism for PROTEUS. Instead of real-time software modules communicating through a typical send-receive-reply mechanism or a global variable blackboard, PROTEUS specifies what actions are performed when a piece of data is updated. If a piece of data (a variable) changes value, an "event" occurs, and this event is propagated to each of the relevant software modules (components). A signal from a peripheral device, a value change in a system variable, a timer tick, or a keystroke are all examples of what can be considered events in PROTEUS. Events connect components together, and these components all have well defined inputs, outputs, parameters, and input to output mappings. Component functions include device interface, operator interface, sensory processing, communications, control, navigation, obstacle avoidance, and fault diagnostics.

Event-based means that the function of each component is entirely event driven. A component interacts with the outside world by connecting its inputs and outputs to other components via events. When the output from one component changes, an event occurs. This event (new variable value) is propagated to the components whose inputs are connected to that event. The procedures in the components that are triggered by the event are called actions. These triggered procedures transform the component inputs into outputs, according to a well defined mapping specific to that component.

In addition to event propagation, PROTEUS also handles real-time scheduling of the components once they have been triggered by an event. Each component has a different priority level, and action procedures with higher priority are executed before those with lower priority. When an event, or multiple events, trigger a set of components, the corresponding action procedures are enqueued on PROTEUS's scheduler waiting list. Actions enqueued into the scheduler's waiting list are executed from the highest priority action down, and on a first-in first-out basis for a given priority level. Actions are always processed to completion before another action is started unless a hardware interrupt pre-empts the current action and places a higher priority action into the scheduler's queue. Action procedures do not wait for resources or semaphores in the middle of their execution, synchronisation is always accomplished through event propagation.

6.3 Configuration Files

Although PROTEUS was written in C++, users program their systems in a configuration language called Control System Probe (CSP). A configuration file is a textual interface for defining and building control systems. Configuration files list instantiations of component types, with specifications for the attributes of each component instantiation, and the input and output events that interconnect components. Configurations draw on a set of library components that together with the scheduler make up PROTEUS.

CSP is a declarative language wherein all events and components are defined one at a time in no order other than they not reference an event or component not yet defined. Within a component there is no rigid ordering of parameters because each attribute has a name. Constructing a real-time system from the PROTEUS library of components involves three steps, choosing the templates for the required components, specifying the parameters for those components, and defining the interconnections between components. The following syntax is employed to define a component in a configuration file:

```
% <component type>
        <attribute name> = <attribute value>
        <attribute name> = <attribute value>
        <attribute name> = <attribute value>
        ...
```

where the component type is one of the PROTEUS library components, attribute name is one of the inputs, outputs or parameters, and attribute value is either an event or a constant. Employing the above syntax, the following is an example of a Boolean AND component:

```
%uncontrolled.int    name = Event1         initial = FALSE
%uncontrolled.int    name = Event2         initial = FALSE
%uncontrolled.int    name = Result_Event   initial = FALSE
%and
        input_1 = Event1
        input_2 = Event2
        output = Result_Event
```

where Event1, Event2, and Result_Event are all integer events that are either FALSE = 0 or TRUE ≠ 0. Appendix Two contains the CSP files for both PURL II itself and the surface interface.

6.4 CSP Design and Configuration

When designing the configuration for PURL II, several options were available for implementing operational mode switches and control signal arbitration. It is beyond the scope of this thesis to describes all the forms of control hierarchies and methodologies that can be configured using PROTEUS. To simplify configuration and debugging, a mode switching and arbitration approach was implemented which resembles a discrete component hardware design. All of the components are always enabled, and they are always taking input events, acting on them, and outputting the results. Arbitration between a few specific inputs and outputs is controlled by multiplexers that use the current mode of operation or other decision criteria to arbitrate signal propagation. The latest release of the PROTEUS executable is entitled "sfu32-h.exe". The file "PURL.BAT" launches the PURL II configuration, and the file "SURFACE.BAT" launches the Surface configuration.

6.4.1 Interface Components

Although PROTEUS's library components provided almost all the functionality required to operate and control PURL II, several interface components had to be written. Writing interface components was expected because the interface components are specific to the sensors and peripherals installed aboard PURL II. Table 6-1 lists the interface components that were added to PROTEUS.

Table 6-1: PROTEUS Interface Components

Component	Interface	Author(s)	Notes
KVH_compass	KVH C100 compass via RS-232	Peter Helland, Doug Girling	Interrogates the compass whenever it is triggered.
Depth_Sensor	Digitec 4-20mA to RS-232 converter via RS-232	Peter Helland, Doug Girling	Interrogates the depth sensor whenever it is triggered
mesotech809_ altimeter	Simrad Mesotech 809 Echo Sounder via RS-232	Kevin Maier, ISER	See "The Integration and Characterisation of a Mesotech 809 Altimeter for the PURL AUV" by Kevin Maier for more information.
sbe19	SBE-19 CTD via RS-232	Peter Helland	Requires up to 1 minute after starting to get initialisation information from the SBE-19.
thruster_interface	Servo~LINK motor controller via RS-232	Andreas Huster, Peter Helland	Adaptive gain control loops dynamically match the thrusters.
Sapphire_Board	PC-104 bus	Peter Helland, Doug Girling	Contains a busy wait that hangs PROTEUS if the Sapphire-MM card is not in the PC-104 stack.

6.4.2 Operator Interface

The operator interface provided by the SURFACE configuration allows a human operator to control PURL II and monitor its status. The operator interface is a graphical user interface where the inputs are controlled by mouse pointer. PROTEUS supports keyboard inputs, but when the operator is in the field, a mouse is the most practical input device. The interface is divided into four main sections; mode control and feedback, setpoint control and feedback, status feedback, and menu bar. Mode control is located on the upper right hand side of the screen, and allows the operator to specify the desired mode of operation, and receive feedback about which mode PURL II is currently in. Setpoint control and feedback is located on the left half of the screen where the operator specifies the desired setpoints while PURL II is in PILOT mode. Feedback from the navigation sensors is also displayed on the left half of the screen. Status feedback, located in the lower right hand side of the screen, monitors the status of the pitch, roll, battery voltage, telemetry, leak sensor, thrusters, CTD, CTD pump, camera lights, and data logging. The operator can also switch the camera lights, CTD pump, and data logging off and on using buttons located beside their status indicators. The menu bar is where the operator can monitor debugging signals, exit the SURFACE configuration, and put PURL II into EXIT mode.

6.4.3 Modes Of Operation

There are five modes of operation for the PURL II configuration; IDLE, PILOT, MISSION, ABORT, and EXIT. Each of these modes have specific attributes that govern the behaviour of PURL II. The SURFACE configuration does not employ modes of operation because there is no mode-dependent change of the behaviour of the SURFACE configuration. As a result, the discussion of modes of operation will focus on the PURL II configuration.

6.4.3.1 IDLE Mode

IDLE mode is the default mode for PURL II when the configuration is initially parsed at start up. IDLE mode is the dormant state for PURL II, and is employed during the launch and recovery phases of a mission. In IDLE mode the thrusters are disabled by forcing their inputs to zero RPM. Disabling the thrusters ensures that it is safe to approach the vehicle.

6.4.3.2 PILOT Mode

PILOT mode is generally entered from IDLE mode, but can be entered from any of the other modes except EXIT. PILOT mode is employed during the pre-mission checkout to ensure all the actuators and sensors are functioning properly, and when an operator is piloting the vehicle via the fibre optic tether. In PILOT mode the operator specifies the desired setpoints for heading, depth, altitude and velocity via the user interface provided by the SURFACE configuration. Specifying setpoints is similar to how a ship's captain specifies the heading and speed to the crew. PILOT mode assumes that there is a telemetry connection between the SURFACE and PURL II because the new setpoints must be transmitted from the SURFACE to PURL II. If the telemetry connection is broken for more than forty seconds, PURL II enters ABORT mode. The thrusters are enabled in PILOT mode, and receive their inputs from the depth, altitude, and heading control loops.

6.4.3.3 MISSION Mode

MISSION mode is employed during autonomous missions where one of six mission scripts is executed. MISSION mode is the same as PILOT mode, except the telemetry connection is not present, and the setpoints are set by the mission scripts not an operator.

Mission scripting is an essential component of any autonomous mission because a mission script is responsible for sequencing the tasks that will be undertaken by the vehicle. Mission scripts are the only CSP files that the operator will be changing on a regular basis. Allowing the operator to modify mission scripts and sequence tasks is potentially dangerous undertaking because it is likely bugs will be introduced. As a result, templates have been created for the depth following, bottom following and sawtooth missions. A detailed description of mission scripts, and mission planning is contained in the PROTEUS documentation provided by International Submarine Engineering (ISER, 1991). Even an abbreviated description of mission planning and writing will remain absent from this thesis because a full and comprehensive understanding of mission scripts is required before writing mission scripts. If there is a bug in an autonomous mission script, the risk of losing PURL II increases dramatically, and therefore script writing must be learned properly or not attempted at all.

6.4.3.4 ABORT Mode

ABORT mode is entered whenever a system failure is detected. Low battery, a leak, loss of telemetry while in PILOT mode, or the ABORT command from the operator all result in PURL II entering ABORT mode. Upon entering ABORT mode, the horizontal thrusters are set to full forward, and the vertical thrusters to full up. These settings attempt to return PURL II to the surface and keep it there until retrieved. As mentioned previously, there is no ambiguity about how PURL II is to behave in an ABORT situation, it should return to the surface and stay there for as long as possible.

In the event that the surface operator determines that the system failure is not critical, the ABORT condition can be overridden by commanding PURL II into another mode. The ABORT condition will still exist, but it will no longer send PURL II into ABORT mode. Overriding the ABORT condition will be necessary during retrieval as a way of stopping the

the PROTEUS system employed in PURL II.
thrusters. Otherwise, overriding ABORT should be discouraged and treated very carefully because it increases the risk of losing or damaging PURL II.

6.4.3.5 EXIT Mode

EXIT Mode is commanded by the surface operator when it is time to exit the PURL II configuration and return to DOS. EXIT Mode executes a script that shuts down the thrusters, lights and CTD pump, thus ensuring that it is safe to shut down PROTEUS. If PURL II is turned off before PROTEUS has exited properly, any logged data that was collected during a mission may be lost, and PURL II's hard drive corrupted. To fix a corrupted hard drive, a disk utility such as DOS's SCANDISK should be run.

6.4.4 Telemetry

Telemetry between the surface and PURL II is controlled by a telemetry component in PROTEUS. The telemetry items are listed in the CSP file "xtelem.csp". Telemetry items are sent when their corresponding events change value, when a periodic timer expires, or both. Telemetry items are sent when their value changes because it is desirable to have updated event values on both the surface and PURL II as soon as possible. Although Ethernet is not deterministic and maximum message transmission times cannot be guaranteed, adverse affects due to collisions and lost packets should be minimal. The Ethernet bandwidth is much larger than the bandwidth consumed by the telemetry information, and telemetry items are on periodic timers that ensure the latest event values are updated even if previous telemetry packets were lost.

6.5 Bugs, Conflicts and Deficiencies

There are no confirmed bugs in the current PURL II configuration of PROTEUS, but there is one bug in the surface configuration. The PROTEUS system also has a variety of deficiencies that can cause problems if they are not handled correctly or are misinterpreted by the programmers. Table 6-1 lists the conflicts and deficiencies that are currently known for

Table 6-1: Bugs and Deficiencies

Problem	Classification	Description
Serial Port Errors	Deficiency, but possibly a bug	PURL II: Occasionally one of the interface components reports a reception error on the serial port. It is likely that the serial port suffers from an overrun error because the serial port interrupts are not serviced quickly enough.
Logged Errors on Start-up	Deficiency	PURL II: PROTEUS logs that the motor controllers fail to respond after being polled during start up. What is actually happening is that several polls get enqueued during start-up and are not sent until after start up is complete. These enqueued polls get sent in such rapid succession that the motor controllers are not given enough time to respond before the next poll is sent, and a polling error is logged.
Mouse Pointer Stops Moving	Bug	SURFACE: The mouse pointer stops accepting inputs from the operator and the surface computer must be re-booted to get the mouse functioning.
DOS Timer	Conflict	PROTEUS seizes the DOS tick and changes it from 65ms to 10ms. This causes problems with Carbon Copy and therefore Carbon Copy must be disabled before PROTEUS is run, and re-enabled after PROTEUS has been exited.
Mission Script Infinite Loop	Deficiency	If a component does not have a specified priority level, it operates at the highest priority level. Mission scripts do not have a priority level. Any loops implemented in a mission script must incorporate timed waits or else they may run freely and consume all the processor time.
Not propagating "interval"	Deficiency	PROTEUS does not propagate the "interval" field of an event through the multiply, add, subtract or divide components.

7. Vehicle Control

Controlling the velocity, heading, depth and altitude of PURL II is performed by a relatively simple yet robust control structure. The dynamics of PURL II change when the payload, ballast, or tether changes. A fixed set of control parameters were chosen which work for all vehicle configurations but are not optimal for any. The heading and velocity is controlled by the horizontal thrusters and the depth and altitude are controlled by the vertical thrusters. The control loops for PURL II are implemented in CSP. Feedback from the heading, depth and altitude sensors are processed to create the appropriate control signals for the horizontal and vertical thrusters. Also, adaptive control loops within the thruster interface utilise the encoder feedback from the thrusters to ensure that all the thrusters are properly matched and have the same response to a specified control signal.

7.1 Propulsion and Heading Control

The propulsion and heading control for PURL II is performed by the two horizontal thrusters mounted on struts off the side of the AUV. Differential thrust is employed by the horizontal thrusters to control heading and propulsion. The thrust from the two horizontal thrusters can be converted from two vectored thrusts to a equivalent net propulsive force and a turning moment. The vector sum of the thrust from the two horizontal thrusters is the net propulsive force (NPF), and the vector difference between the two horizontal thrusters multiplied by their moment arms generates the turning moment (TM) (Figure 7-1).

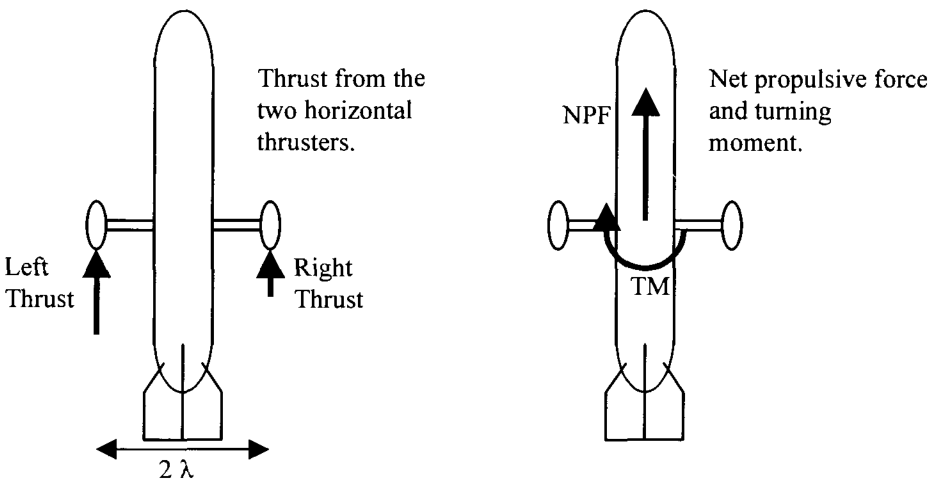


Figure 7-1: Net Propulsive Force (NPF) And Turning Moment (TM)

The equations of the NPF and TM are as follows:

$$\text{Net Propulsive Force} = \text{Left Thrust} + \text{Right Thrust} \quad (2)$$

$$\text{Turning Moment} = \text{Left Thrust} * \lambda - \text{Right Thrust} * \lambda \quad (3)$$

where λ is the length of the moment arms for the horizontal thrusters. Although the 0.65 m/s maximum velocity of PURL II is well below the desired maximum velocity of 1 m/s, the thrusters were deemed adequate for this phase of the PURL program because they can still be employed to demonstrate the utility of a small AUV.

The control loop for heading and propulsion control is anchored by a PID controller (Figure 7-2). This PID controller accepts the heading setpoint from the heading setpoint selection logic which arbitrates between the various heading setpoint sources. The feedback for the PID controller comes from the KVH C100 compass heading, and together with the heading setpoint generates the heading control signal. After the heading control signal is converted to a thruster RPM, it is either added or subtracted from the velocity setpoint RPM to generate the left and right horizontal thrusts respectively. The Thruster Motor Safety Interlock disables the thrusters in IDLE mode, sets them to full forward in ABORT mode, or passes the left and right thrusts through in MISSION and PILOT modes.

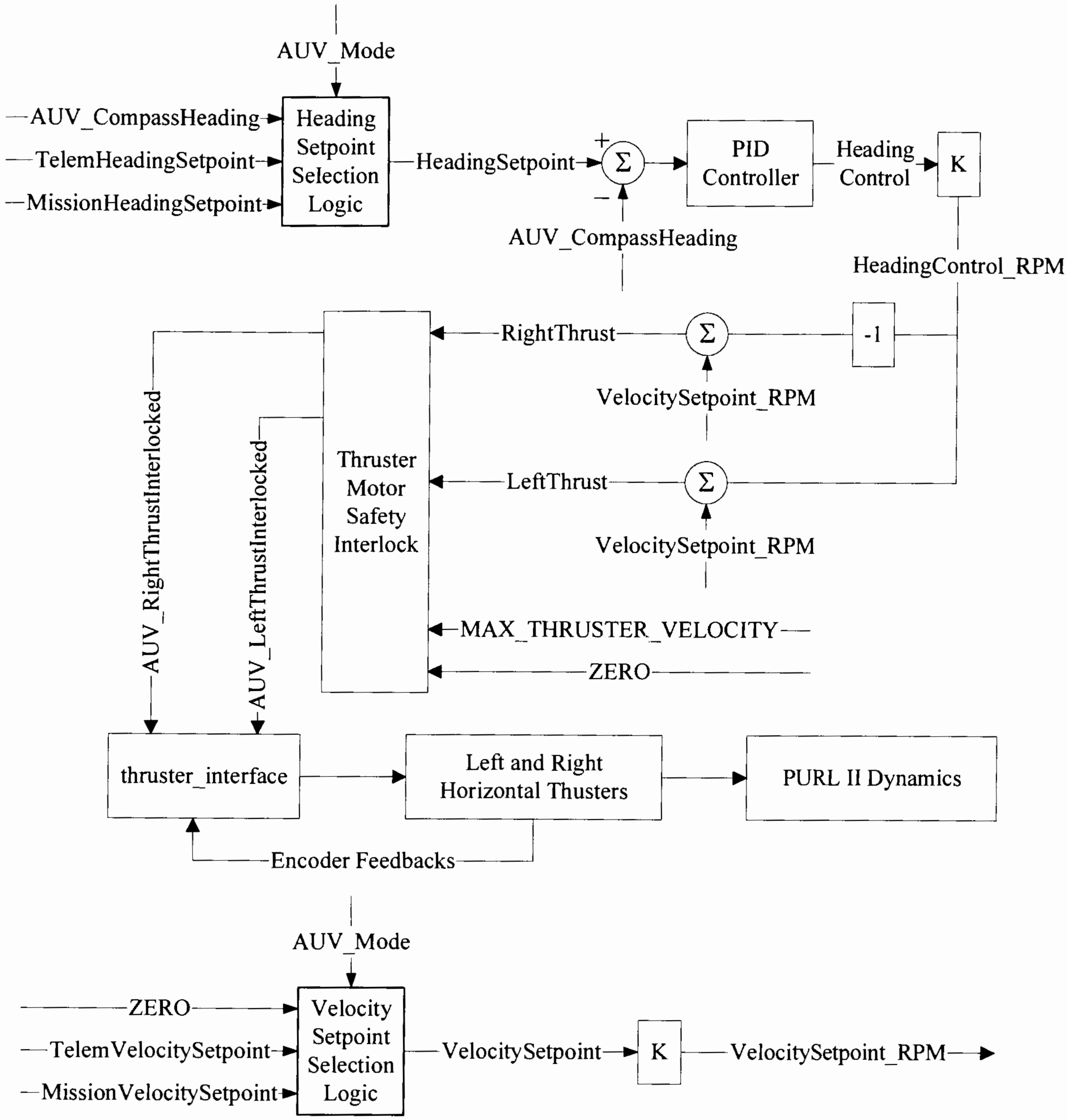
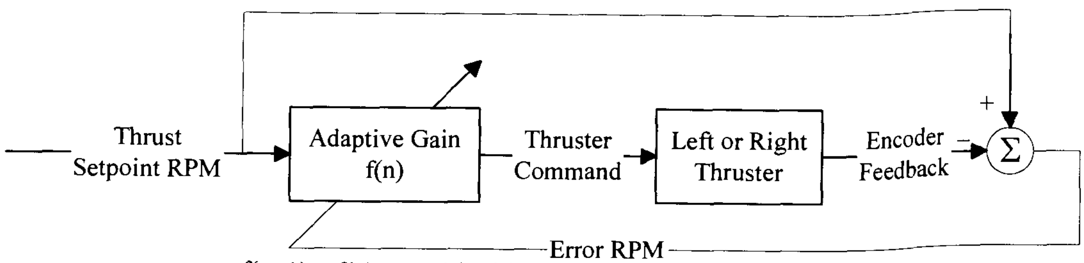


Figure 7-2: Horizontal and Propulsion Control Diagram

Upon entering the thruster interface, the control signals for the right and left thrusters are sent through adaptive control loops to ensure that the thrusters achieve the desired RPM (Figure 7-3). The motor controllers operate in a velocity mode where the control signal is proportional to the error between the desired and actual velocity of the thruster. As a result, there would normally be a steady state error between the desired and actual thruster velocities. However, the thruster interface multiplies the desired RPM by a variable gain (>1) which increases the commanded RPM sent to the motor controllers. The adaptive control loops within the thruster interface employ the encoder feedbacks to adjust the gains for each of the thrusters. The adaptive control loops ensure all the thrusters are matched and turn at the desired RPM when commanded to do so. The PID heading controller cannot fully compensate for mismatched thrusters, but the adaptive control loops within the thruster interface can.



$f(n+1) = f(n) + \alpha\, e(n)\, x(n)$

where $e(n)$ is Error RPM and

$x(n)$ is Thrust Sepoint RPM

α is the step size

Figure 7-3: Thruster Adaptive Control

7.2 Depth Control

Depth control utilises two vertical thrusters mounted on struts at the bow of the AUV. Once again, thrusters are employed for control instead of planes because depth control must be maintained even at zero speed. At zero horizontal speed the vertical thrusters push PURL II up and down through the water column, whereas when PURL II is moving, the vertical thrusters pitch the bow up or down, thus making PURL II behave more like a wing flying itself up and down through the water column (Figure 7-4).

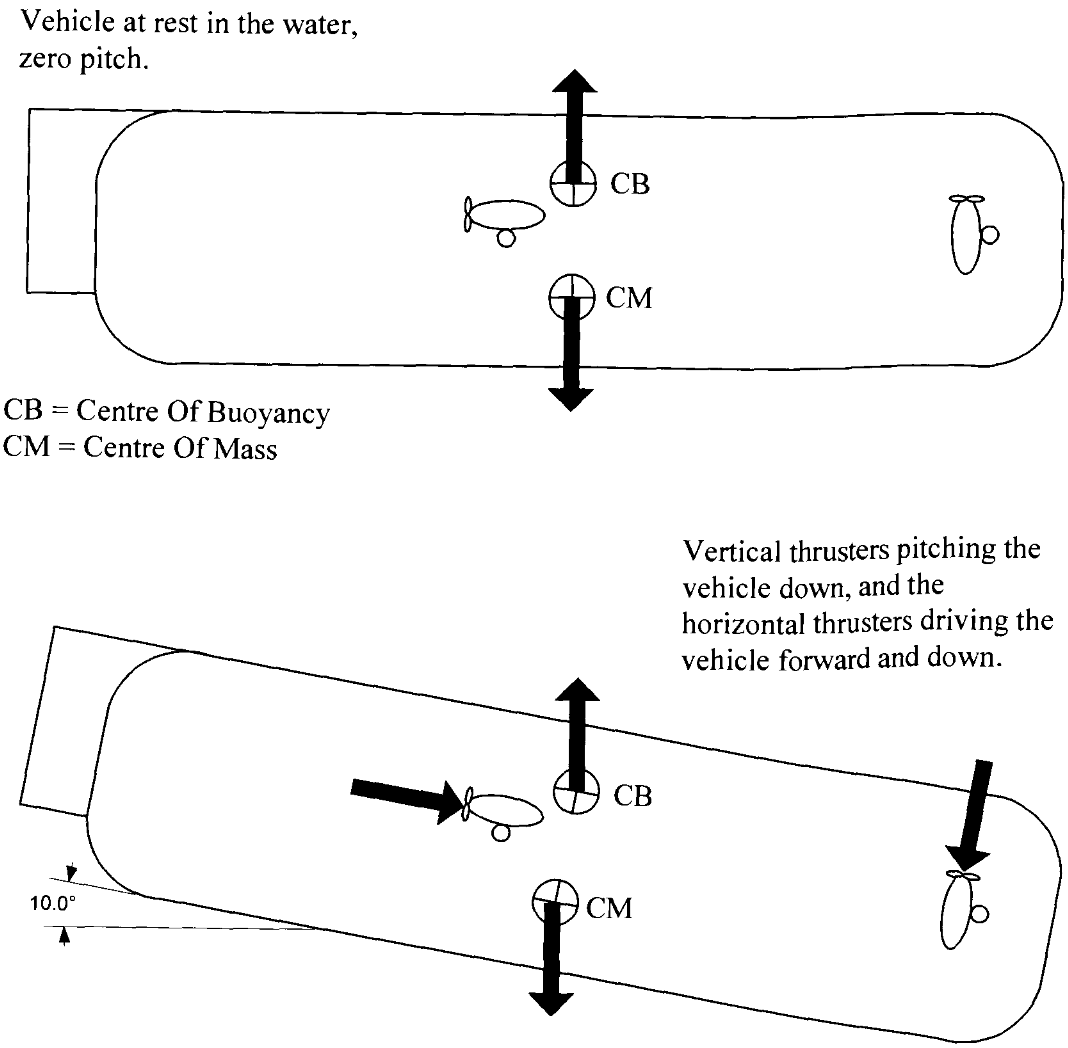


Figure 7-4: Depth Control Employing Vertical Thrusters

Similar to the heading control system, PURL II's operating mode determines which setpoint source will be passed to the vertical PID controllers (Figure 7-5). The vertical control feedbacks are depth and altitude. The depth and altitude control signals enter arbitration logic which determines if PURL II should be controlled by altitude or depth (Table 7-1). Similar again to heading control, the vertical control signal is converted to RPM, passed through the Thruster Safety Interlock, and finally into the thruster interface where adaptive control loops ensure that the vertical thrusters are turning at the desired velocity.

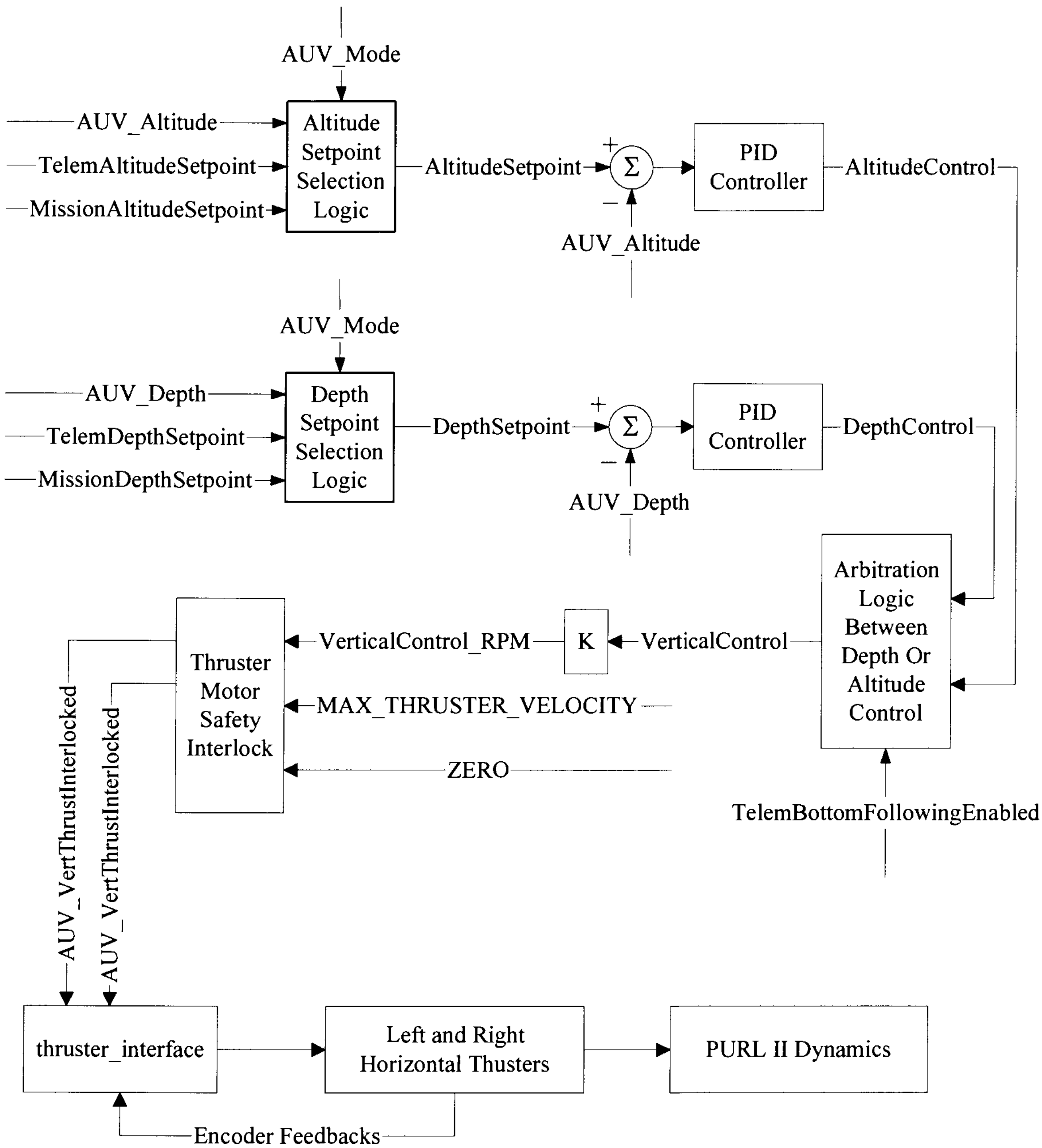


Figure 7-5: Depth and Altitude Control Diagram

Table 7-1: Depth and Altitude Arbitration Logic

	bc = 00	bc = 01	bc = 11	bc = 10
a = 1	Altitude	Altitude	Altitude	Depth
a = 0	Depth	Altitude	Altitude	Depth

where: a = 1 is Bottom Following Enabled
a = 0 is Bottom Following Disabled (i.e. Depth Following)
b = 1 is Deeper than the Depth Setpoint
b = 0 is Shallower than the Depth Setpoint
c = 1 is Less Altitude than the Altitude Setpoint
c = 0 is More Altitude than the Altitude Setpoint

Note: This Karnaugh map is based on the work of both Peter Helland and Maier, 1997.

8. Fibre Optic Link

The fibre optic link is connected to PURL II whenever a survey task requires the intervention or supervision of a human operator. The fibre optic link allows a human operator to view the outputs of the various sensors, especially the video camera, and make decisions about controlling PURL II.

8.1 Design Specifications

The specifications for the fibre optic link are deceptively simple, but were difficult to implement because of physical size and monetary constraints. The specifications for the fibre optic link are listed in Table 8-1.

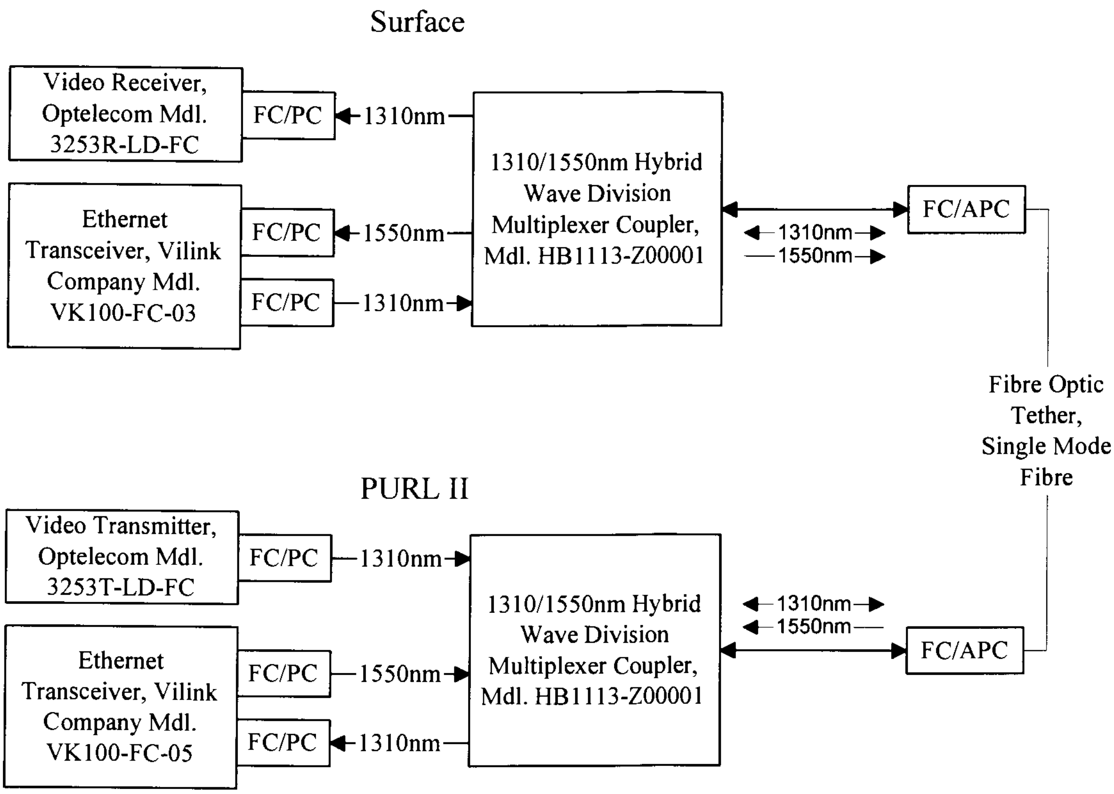
Table 8-1: Fibre Optic Specifications

Parameter	Specification
Fibre Type	Single or Multi-Mode Fibre
Number of Fibres	1
Cable Length	> 150 m
Depth Rating	100 m
Maximum Cable Diameter	< 2.5 mm
Armoured	Yes
Signals From PURL II to the Surface	10Base-T Ethernet, and Colour or Black and White Video
Signals From Surface to PURL II	10Base-T Ethernet

8.2 Fibre Optic Design

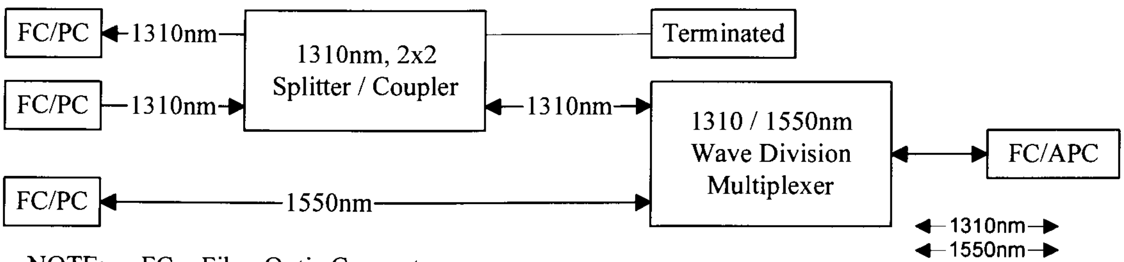
Sending Ethernet bi-directionally and video uni-directionally through a single fibre requires electrical and/or optical multiplexing. When the components for PURL II were sourced there were no electrical mulitplexers for video and Ethernet that could fit within the physical constraints imposed by the pressure vessel. However, Ethernet transceivers, video transmitters and video receivers that could operate with either single mode or multi-mode fibres were located. Multiplexing the Ethernet and video signals had to be performed optically, but a decision had to be made between a multi-mode or a single mode system. Employing off-the-shelf equipment constrained the number of wavelengths available for both multi and single mode fibre systems; multi mode components commonly employ 850nm and 1310nm, and single mode components employ 1310nm and 1550nm. Forced to employ two distinct wavelengths to transmit three signals (Ethernet up, video up, Ethernet down) meant that one of the two wavelengths would have to be sent in both directions, and back reflection would cause interference between the signals that shared a common wavelength.

The possibility of using less expensive multi-mode equipment was eliminated because the back reflection of multi-mode connectors was too high to ensure that interference would not be a problem. Fortunately, the cost of single mode components (optical and electrical) has been falling in recent years, and low back reflection components could be sourced for single mode applications. To ensure that back reflection would not be a problem, link budget and back reflection calculations were performed on the optical multiplexing system shown in Figure 8-1 and Figure 8-2.



NOTE: FC = A Fibre Optic Connector
PC = Physical Contact
APC = Angled Physical Contact

Figure 8-1: Fibre Optic System For Ethernet and Video



NOTE: FC = Fibre Optic Connector
PC = Physical Contact
APC = Angled Physical Contact

Figure 8-2: 1310/1550nm Hybrid Wave Division Multiplexer Coupler

8.3 Link Budget

The link budget for the fibre optic system shown in Figure 8-1 and Figure 8-2 was calculated based on a reasonable worst case estimate of the attenuation of the various components in the system. The maximum total attenuation in the system is 10.6 dB for the 1310nm signals and 3.4 dB for the 1550nm signal. The Ethernet transceiver has a transmission power of -12dBm and a maximum receive sensitivity of -30 dBm, thus leaving an allowable loss of 18 dB in the transmission path. Therefore, the 1310nm Ethernet signal has 7.4dB of signal power margin, and the 1550nm Ethernet signal has 14.4dB. The video transmitter has a transmission power of -14 dBm and a maximum receive sensitivity of -37 dBm thus leaving 12.4 dB of signal power margin.

8.4 Back Reflection

The sources of back reflection in the 1310nm optical system are connectors, coupler/splitters, terminations, and wavelength division multiplexers. If a worst case scenario is assumed, a first order approximation of the back reflection is good enough to determine whether or not the desired multiplexing scheme would be successful. A first order approximation means that only the primary back reflections are included in the back reflection calculation because secondary back reflections would be so small that they are insignificant (<-150dB) when compared to the primary back reflection (<-50dB). A worst case scenario assumes that the all the back reflection sources sum coherently, each back reflection source provides the maximum back reflection specified by the manufacturer (instead of the typical back reflection), and system attenuation is calculated to increase back reflection. These three assumptions yield a pessimistic estimate of the total back reflection in the system, and should provide sufficient safety margin.

The calculated back reflection was -44.2 dB (Appendix Three) and this resulted in a signal to interference ratio of 32.8 dB for the 1310nm Video signal, and a signal to interference ratio of 36.8 dB for the 1310nm Ethernet signal. The manufacturers of the Video and Ethernet receivers would not guarantee that these interference levels were low enough, but the fibre optic equipment was ordered anyway. Upon receipt of the fibre optic equipment, the entire system was tested and confirmed to function properly.

8.5 Cables and Penetrators

The fibre optic cable and penetrator employed on PURL II is a copy of the system employed by ISER for the Theseus AUV which deployed over 200 nautical miles of fibre optic cable under the arctic ice cap in the spring of 1996. The fibre optic cable contains one 9/125μm single mode fibre housed inside a stainless steel tube with a fibre glass and Hytrel jacket. The outer diameter of the fibre optic cable is 0.1 inches. The fibre optic penetrator was constructed from a modified Brantner and Associates Inc. XSA-BCL Type 2 penetrator. This XSA-BCL penetrator has a hollow stainless steel tube that the fibre (the jacket and stainless steel tubing were removed) was fed through and potted into place with epoxy. The potted fibre provides the pressure barrier between the ambient pressure outside PURL II and the one atmosphere environment inside the pressure vessel. When the fibre was potted, the epoxy also spread into the stainless steel tube, thus preventing water from entering the tube. The fibre optic cable, penetrator, and connectors form a single unit. If the fibre optic cable is not in use, the entire cable must be removed from PURL II, and a penetrator blanking plate placed across the hole vacated by the penetrator.

8.6 Carrying Case / Handling

Safely transporting the fibre optic cable to the mission site, and handling the fibre optic cable during a tethered mission are two important tasks of the fibre optic case. We employed a waterproof Pelican case which was modified to provide posts for coiling the fibre optic cable in a figure eight. The Pelican case also houses another smaller Pelican case which holds the fibre optic transmitter and receivers, and the Ethernet Hub (Note: UTP - Unshielded Twisted Pair (10Base-T)

AUI - Attachment Unit Interface

Figure 5-3).

9. Payload

PURL II is designed to carry a variety of payloads that can be interchanged with little or no modification to the vehicle itself. PURL II is currently configured to carry only two items, a water property profiler (CTD), and a video camera. However, power, communications, and reserve buoyancy are available for additional payloads such as a side scan sonar.

9.1 Water Property Sensors

The water property sensors are housed in a conductivity, temperature and depth (CTD) profiler manufactured by Sea-Bird Electronics Inc, the SBE-19 (Table 9-1).

Table 9-1: Sea-Bird SBE-19 CTD

Component	Depth Rating	Weight Air / H_2O	Dimensions	Notes
SBE-19	300m	5.1kg / 1.2 kg	9.9cm ϕ, 73.9cm Len.	Contains conductivity, temperature and depth sensors with up to four additional sensors such as dissolved oxygen, pH, and fluorometer.
SBE-5T Pump	10500	0.7 kg / 0.3 kg	4.5cm ϕ, 22.1cm Len.	100mL per second at 2000 RPM, 10-18VDC power, 0.2 Amps

The SBE-19 is a self-contained CTD profiler that does not require external power sources or data logging. When the SBE-19 is mounted inside PURL II it requires a pump and plumbing to flush the sensors with water. The pump employed on PURL II is the SBE-5T, and it is controlled by switching its power off and on. The SBE-19 can be interfaced with the PC-104 stack via a RS-232 link, and the CTD data logged by PROTEUS. Because PROTEUS can interface with the SBE-19, PURL II can be reconfigured to use the SBE-19 outputs as control signals. For example, if the goal of a particular mission is to survey the thermocline in a lake, PURL II could be configured to perform a sawtooth survey with two temperatures defining the top and bottom of the vertical profiles.

9.2 Video Camera

Underwater vehicles employ cameras for a wide variety of tasks including bio-diversity surveys, bottom mosaics, visual inspection, and vehicle piloting. Low light cameras are essential for autonomous surveys because an AUV must carry energy for the lights. Low light cameras also reduce optical back scatter because as less light is put into the water, less light is scattered back into the camera from particles suspended in the water. An analogy to back scatter is the reduction in visibility experienced when high beams are employed while driving in fog or snow. Extremely low light, black and white SIT and ICCD cameras provide the best light sensitivity of all cameras (10^{-4} to 10^{-6} Lux), whereas low light colour cameras provides light sensitivities as low as 0.1 Lux. Due to cost and size constraints PURL II does not carry an extremely low light camera. Instead, PURL II carries a black and white video camera manufactured by Deep Sea Power and Light called the Micro Sea Cam 1000 (Table 9-1).

Table 9-1: Video Camera and Lights Specifications

Component	Type	Dimensions	Depth (m)	Weight Air / Water (kg)	Power (Watts)	Notes
DSP&L MSC 1000	Black & White	6cm Len. 4.4cm Dia.	1000	0.175 / 0.100	1.4	0.3 Lux Scene Illumination
DSP&L Light	Halogen	11.6cm Len. 7.9cm Dia.	1000	0.3 / Neutral	50 or 100	Inrush current up to 10 times nominal value.

10. Transportation, Launch and Recovery

Transporting, launching, and recovering AUVs are the three tasks that often make them unwieldy for many potential users because AUVs are generally too large to be handled inexpensively. PURL II is small and light enough that two or three people can transport it to the launch site, launch it, run a mission, and recover it with little difficulty or expense. To facilitate easy handling, a search and rescue stretcher was purchased and modified to act as a carriage for PURL II. Search and rescue equipment is designed for handling loads (injured people) that are approximately the same weight and size as PURL II. To aid movement over long distances and rough terrain, a large wheel is attached to one end of the stretcher, and the unit is rolled over the ground. The stretcher has convenient hand holds and is designed for carrying, dragging, rolling, lifting and other tasks associated with moving, launching and retrieving PURL II. PURL II is small enough to be transported in a compact car with a hatch back and the passenger seat removed, a pick up truck, or a mini-van. Employing readily available personal vehicles reduces the cost of transporting PURL II because special vehicles do not need to be rented or purchased. The support equipment also travels in the same vehicle as PURL II, and is contained in toolboxes and Pelican cases. The final result is a mobile AUV that can be transported from the lab to a mission site quickly and easily.

11. Vehicle Performance

For the purpose of mission planning and analysis, the performance parameters of PURL II can be represented empirically by a few simplified equations. Quantifying the performance of AUVs reduces the costs and resources required to obtain underwater data because missions can be planned to make the best use of the available AUV resource. Also, if an AUV is designed for a specific set of missions, a minimum AUV can be developed that minimises the resources consumed while operating the AUV. The volume of water surveyed during a mission is determined not only by the types of sensors carried and their ranges, but also by the path that the AUV is able to complete in a specified time. In general, the faster an AUV travels, the more water surveyed. Trade-offs between the speed, size, depth and cost of AUVs constrain the maximum performance available from current AUV technology.

11.1 Specific Energy

The base components for PURL II include everything but the payload (i.e. the faring, floatation, pressure vessel, thrusters, navigation sensors, batteries, and electronics). These base components can be represented in relation to the vehicle's mission by four parameters: volume, mass, velocity and power consumption. The power consumption calculated without including the propulsion and payload is commonly referred to as the hotel load. Hotel load represents the overhead energy required to operate all the vehicle's systems before any useful survey work is done. As in a company, reducing the overhead (hotel load) is an important part of increasing efficiency. If all of an AUV's energy is consumed by the hotel load, there will be no energy for propulsion and payload sensors. An equation for specific hotel energy for a given mission with duration t is shown in (4):

$$e_H = \frac{P_H\, t}{s\, w} = \frac{P_H}{v\, w} \tag{4}$$

where e_H is the specific hotel energy, P_H is the hotel load, s is the survey distance, w is the total weight of the vehicle, and v is the survey speed (Bird, 1997). The specific energy represents the energy required per unit distance travelled per unit weight of the vehicle. By definition, e_H is directly proportional to the hotel load, and inversely proportional to the velocity and weight of the AUV. The faster the AUV travels, the less time the hotel load has to consume the AUV's energy reserves, and the heavier the AUV the more energy that can be stored in the form of batteries for a given hotel load.

Similar to the hotel load, the payload also consumes power, and the specific payload energy is shown in (5):

$$e_P = \frac{P_P\, t}{s\, w} = \frac{P_P}{v\, w} \tag{5}$$

where e_P is the specific payload energy, P_P is the payload power consumption, s is the survey distance, w is again the total weight of the vehicle, and v is the survey speed. It is not obvious that the specific energies for both the payload and hotel use the total weight of the vehicle until you remember that the total weight of the vehicle also includes the weight of whatever payload is carried. Part of a payload could be an extra battery pack, similar to drop tanks that fighter aircraft carry to increase their range. The mission defines the payload, but the weight of the vehicle used to calculate the specific energies must include everything including the payload.

The sum of the specific hotel and payload energies represents the energy required to complete a mission if no energy is expended moving the vehicle through the water. However, energy is required to thrust an AUV through the water, and the specific energy of propulsion is shown in (6):

$$e_T = \frac{D_g}{\alpha w} + \frac{P_v}{v w} = \frac{C_D A_w \rho_w v^2}{2 \alpha w} + \frac{P_v}{v w} \tag{6}$$

where e_T is the specific energy of propulsion (thrust), D_g is the drag force, C_D is the drag coefficient, A_w is the wetted area of the vehicle, α is the efficiency converting electrical energy into thrust, P_v is the battery power required for the vertical thrusters (typically 50W), and ρ_w is the density of the water. The efficiency converting electrical energy into thrust is assumed to be constant, and at 0.6 m/s α was determined to be 0.0825. 8.25% is an extremely low efficiency rating, but one of the reasons for this low efficiency rating is that the thrusters turn a small, low pitch propeller at high speeds. To increase the propulsion efficiency, the thruster should turn a larger propeller at lower speeds.

The coefficient of drag is determined by friction drag, pressure drag, and Reynolds number as follows:

$$C_D = C_f\left(1 + \frac{1.5}{r^{1.5}} + \frac{7}{r^3}\right) \tag{7}$$

where C_f is the friction drag, r is the length to diameter ratio, and the terms in the brackets represent the pressure drag on a slender body (Bird, 1997). Since PURL II does not have a cylindrical cross section, the length to diameter ratio for PURL II was approximated by an equivalent diameter (0.33m) which yields the same frontal area as PURL II. The equivalent length to diameter ratio for PURL II is r = 5.82. Friction drag depends on the Reynolds number (r_e) as shown in (8) and (9).

$$C_f = \frac{C_1}{r_e^{C_2}} \tag{8}$$

$$r_e = \frac{v\,L}{\gamma} \tag{9}$$

Where C_1 and C_2 are determined by the flow around the vehicle, L is the length, v is the velocity, and γ is the kinematic viscosity of water ($\gamma = 1.1\text{x}10^{-6}$ at 18°C). An estimate of the coefficient of drag could be determined by approximating PURL II's shape with known shapes such as cylinders and rectangular boxes, but the shape of PURL II is more complex because it has appendages. Determining the values of C_1 and C_2 relies on the assumption that the flow around PURL II is turbulent. At the speeds PURL II generally operates, assuming turbulent flow is reasonable because PURL II is not well fared and it has non-fared appendages. For bodies with turbulent flow $C_2 = 0.2$ (Bird, 1997), and C_1 must be determined from measured data. When the two forms for the specific energy of propulsion (e_T) are equated at v = 0.6m/s, $A_w = 2.6\ m^2$, $\rho_w = 1000\ kg/m^3$ and $D_g = 1.4\ kg * 9.81\ N/kg$, C_1 equals 0.41.

The total specific energy for a vehicle (e_V) is the sum of e_H, e_P and e_T as shown in (10). It must be reiterated that the calculation of specific energy is an approximation to the actual performance of PURL II and the development of a detailed model is outside the scope of this thesis.

$$e_V = \frac{P_H + P_P + P_V}{v\,w} + \frac{C_D\,A_W\,\rho_W\,v^2}{2\,\alpha\,w} \tag{10}$$

The graphs for specific energies have different minima depending on the payload and ballasting. For this crude empirical analysis it is assumed that the payload ranges from zero watts (a self contained CTD) to 125 watts (CTD, pump, video camera and 100 watt lights), and that the vehicle is ballasted so that 50 watts is required on average to keep the vehicle at the desired depth and altitude (Figure 11-1 and Figure 11-2). The wetted area of PURL II is approximately 2.6 m^2, the mass is 70 kg. The minimum for a zero power payload occurs at a velocity of approximately 0.45m/s and the minimum for a 125 watts payload occurs at approximately 0.65m/s. The area between the two curves in Figure 11-3 shows the range of specific energies for payload power consumption between zero and 125 watts. If the payload power consumption increases above 125 watts, the minimum energy velocity will increase beyond the maximum velocity of PURL II (0.65m/s).

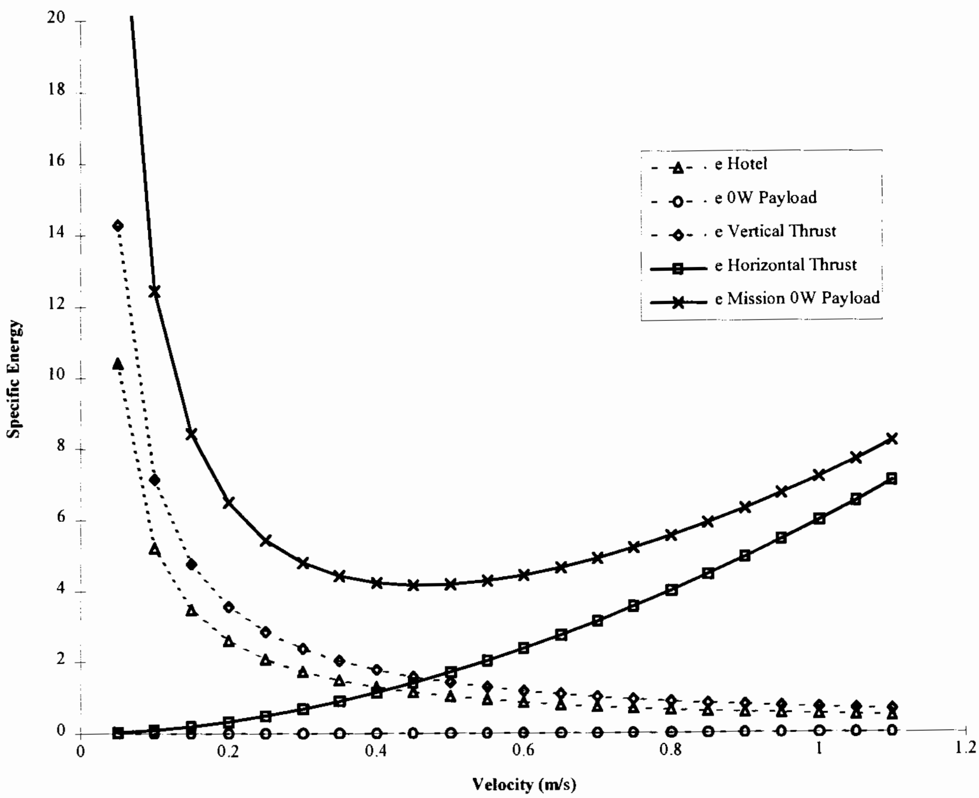


Figure 11-1: Velocity Vs. Specific Energy (0 Watt Payload)

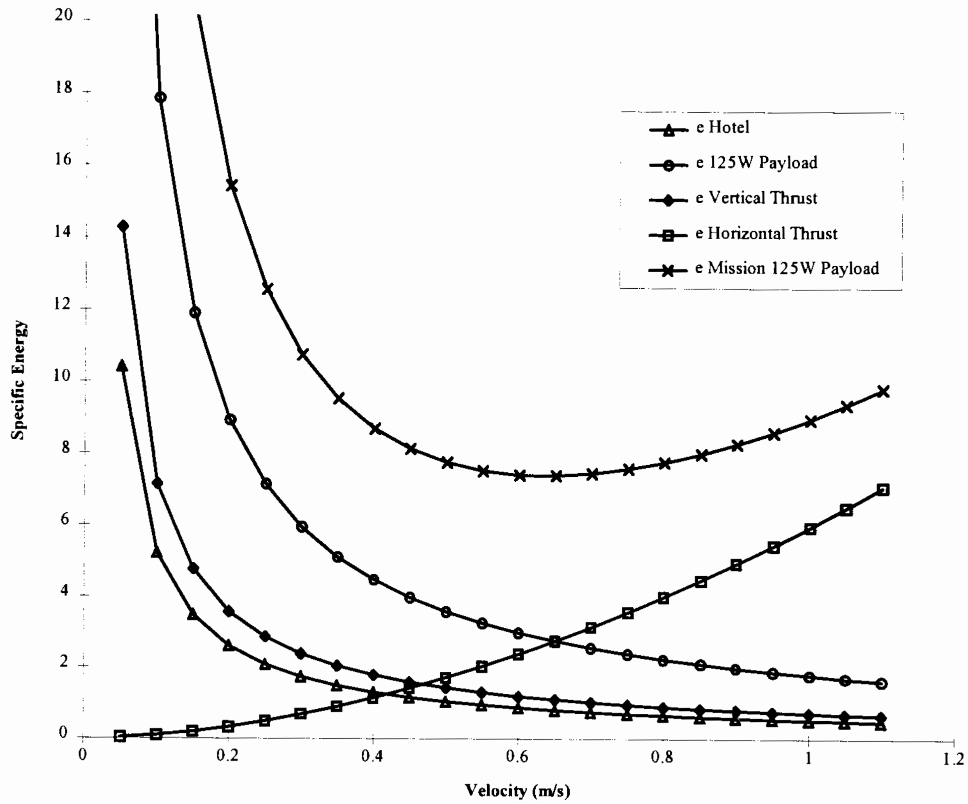


Figure 11-2: Velocity Vs. Specific Energy (125 Watt Payload)

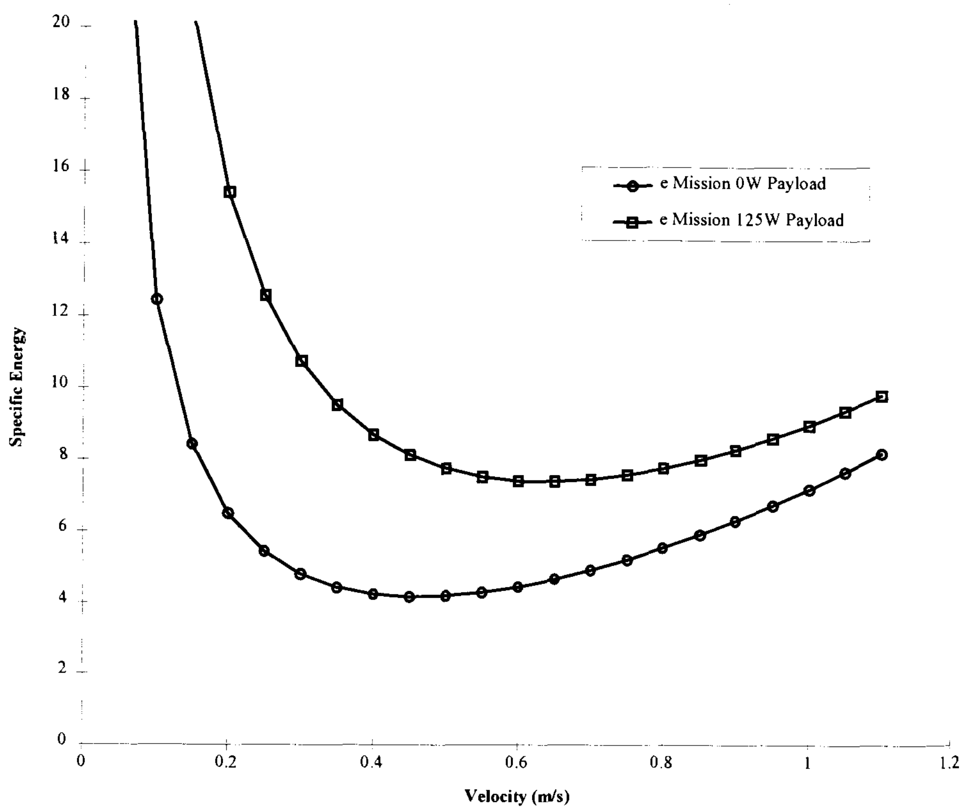


Figure 11-3: Velocity Vs. Specific Energies

Rarely will PURL II be able to operate at the minimum specific energy point because the desired mission velocity will be determined by the physical phenomenon being measured or surveyed. Full throttle is usually the desired velocity setpoint because the phenomenon being measured must be sampled as quickly as possible. The scientist operating PURL II does not care what is the most efficient operating point, he or she wants to collect their data in as timely a manner as possible.

11.2 Vehicle Specifications

Table 3-1 lists the desired specifications for PURL II. Table 11-1 shows a comparison between the desired and actual specifications for PURL II.

Table 11-1: Design Goals and Actual Implementation

Design Parameter	Goal	PURL II
Completion Date	December 1996	June 1997
Max. : Min. Speed	1 m/s : Stationary	0.65 m/s : Stationary
Maximum Depth	70m	Tested to 50m
Endurance (min. : max. payload)	2 hr. @ 1m/s : 1 hr. @ 1m/s	3 hr. @ 0.65m/s : 1.5 hr. @ 0.65m/s
Maximum Displacement	< 70 kg	< 70 kg (payload dependent)
Maximum Size	2m x 0.5 m x 0.5m	1.92m x 0.47m x 0.7m
Communication Link	Ethernet (via Fibre and 10Base-T)	Ethernet (Fibre and 10Base-T)
Transportation	2 compact cars or a pickup truck	2 compact cars or a pickup truck
Crew	2 or 3 people	2, but preferably 3 people
Operating Temperature	-5 to 40 °C	Field Tested in -2 to 25°C
Navigation	Dead Reckoning	Dead Reckoning with compass, depth sensor, and an altimeter
Minimum Instrumentation	Heading, Depth, Altitude, Battery Monitor, and Leak Sensor	Heading, Depth, Altitude, Battery Monitor, and Leak Sensor
Payload	CTD, Camera + Lights, Side Scan Sonar, and 2kg ballast.	CTD, Camera + Lights and 4 kg ballast.

PURL II does not meet all of the desired specifications. PURL II's maximum velocity of 0.65m/s falls well short of the desired maximum velocity of 1m/s. The URL has plans to upgrade the thrusters aboard PURL II which will hopefully bring the maximum velocity closer to the desired velocity of 1m/s. Also, the faring and its appendages could be streamlined to reduce drag and increase velocity. A side scan sonar payload has also not been added, but the URL has recently purchased an Imagenex side scan sonar and hopes to add it to PURL II in the coming year. Other than these two shortcomings, PURL II meets the desired operational window and is therefore considered a success.

12. Trials and Missions

In order to prove PURL II as a useful AUV in small lakes, major components of its development were lake trials and lake missions. When a design is placed under the constraints of finite money, finite labour, finite time, and employing only off the shelf components, the effectiveness of such an AUV is thrown into doubt. All of the subsystems, software, and sensors were tested in the swimming pool at Simon Fraser University, or at Loon Lake in the University of British Columbia's research forest in Maple Ridge (Figure

12-1). The small arm in the Southeast corner of Loon Lake is where much of the testing was conducted because of its shallow maximum depth (<15m), and its smaller size. If PURL II is lost in the arm of Loon Lake, divers could search the arm and locate the vehicle within a few working days.

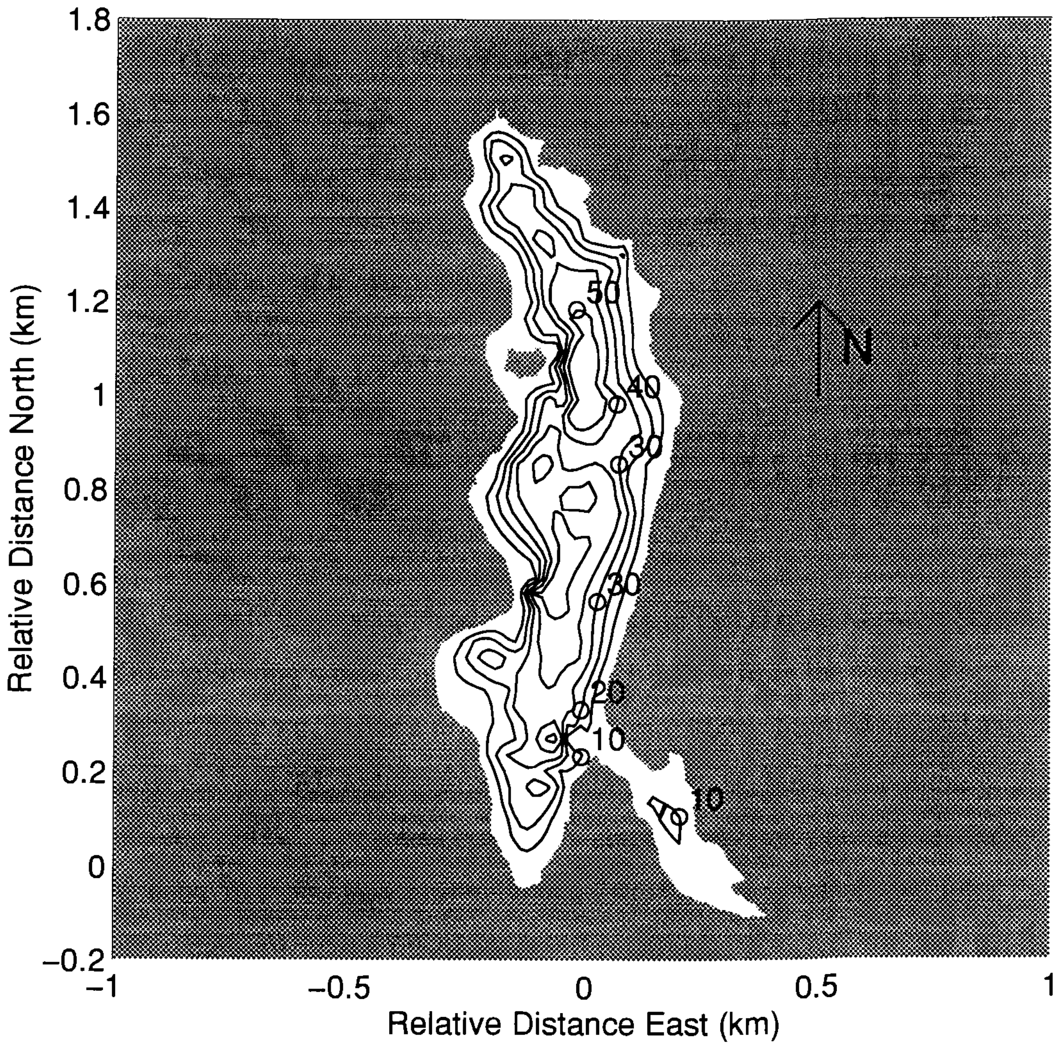


Figure 12-1: Loon Lake, University of British Columbia Research Forest

12.1 Swimming Pool and Lake Trials

PURL II was usually tested in the SFU swimming pool and then moved to Loon Lake for field verification. However, towards the end of PURL II's development cycle, the pool tests were sometimes skipped. Moving from bench testing straight to Loon Lake is an indication of the confidence we had in PURL II's ability to perform missions successfully.

The first PURL II missions were actually run aboard PURL I. PURL I performed as a test platform for software and sensors until PURL II was operational in September 1996. Employing PURL I as a test platform allowed us to develop software in parallel with the electrical, electronic, and mechanical components of PURL II. Moreover, lessons learned from operating PURL I were applied to PURL II, especially in the areas of mission planning, and vehicle handling and transportation.

12.1.1 October 19, 1995

Trial: Constant Compass Heading

Vehicle: PURL I

Location: SFU Pool

Tests: KVH Fluxgate Compass, Heading Control, and PROTEUS Mission Scripting

While following a constant heading, PURL I flew from the shallow to the deep end of the SFU swimming pool. After a timed wait expired, PURL I turned around and headed back to the shallow end. PURL I maintained a constant altitude above the bottom of the pool by dragging a chain. PURL I behaved erratically and turned circles during several of the runs. This erratic behaviour was later attributed to a PROTEUS bug which corrupted the serial port receive buffer, thus corrupting the heading information.

12.1.2 February 20, 1996

Trial: 50m Deep Dive

Vehicle: PURL I

Location: Loon Lake Main Body

Tests: Thruster 50m Depth Rating, Depth Sensor, and Depth Control.

PURL I dove from the surface to 50m where the vehicle apparently embedded itself into the bottom and was unable to return to the surface under its own power. PURL I was retrieved by pulling up the rope it followed to the bottom. We are unsure why PURL I could not return to the surface, but it appears that the vehicle could not over come the lost buoyancy when the floatation compressed at depth. The thrusters were unharmed by their descent to 50m, and the depth sensor functioned properly.

Trial: Sawtooth Profiling
Vehicle: PURL I
Location: Loon Lake Arm
Tests: PROTEUS Mission Scripting (Looping), Depth Sensor, Depth Control System, CTD Profiler, First Completely Autonomous Mission

PURL I went out and back on fixed headings in the Arm of Loon Lake. While following the fixed headings, the vehicle saw-toothed up and down through the water column between 0.25m and 4m depth. The sensors and control loops for depth and heading appeared to work well, and the error detection added to the sensor interfaces appeared to catch the corrupted serial port messages. The CTD profiler was mounted on PURL I for this mission and it was able to collect data. This is the first completely autonomous mission for PURL I.

12.1.3 March 12, 1996

Trial: Out and Back at Fixed Depths
Vehicle: PURL I
Location: Loon Lake Main Body
Tests: Depth Control, CTD Profiler and CTD Pump

PURL I went out from the dock in the main body of Loon Lake at 5m depth for 35 minutes, and then turned around and headed back to the dock at 10m depth. PURL I dragged a surface float to ensure that we would not lose the vehicle if it dived unpredictably or had a system failure. The CTD pump did not work because its power cable had an intermittent connection.

When the same mission was run a second time, PURL I dived to almost 9m before returning to the desired out-bound depth of 5m. After examining the data logging and error logging files, it was found that the serial port data was corrupted for an extended period of time. PURL I did not receive accurate depth data for most of its dive to 9m. The remainder of the mission was uneventful. After contacting ISER, an updated release of PROTEUS was sent to the URL which contained several bug fixes.

12.1.4 September 4, 1996

Trial: PURL II Shakedown

Location: SFU Swimming Pool

Tests: Heading and Depth Control, Ethernet Link, Vehicle Velocity and Manoeuvrability.

We had a hard time establishing a reliable Ethernet connection with PURL II because we encountered general protection faults in memory, and the Ethernet cable may have had a problem with water intrusion. Once launched successfully, PURL II ran several out and back missions without any problems. The out and back runs were conducted with either a 0.25m depth setpoint for the entire mission, or a 0.25m depth setpoint in the shallow end of the pool, and a 1.0m depth setpoint in the deep end. After the trials, we cleaned and lubricated all the underwater connectors.

12.1.5 September 17, 1996

Trial: PURL II Shakedown and Altimeter Test

Location: SFU Swimming Pool

Tests: Altimeter, Vertical Control System

Tested the altimeter in bottom following mode and depth following mode. The altimeter and vertical control systems worked well (for additional information see Maier, 1997). Communications with PURL II was reliable throughout the twelve runs that this trial entailed.

12.1.6 September 19, 1996

Trial: Altimeter Test and Autonomous Missions For The NSERC Demonstration

Location: Loon Lake Arm

Tests: Depth Following Mission, Bottom Following Mission, Sawtooth Mission

Operating autonomously in the arm of Loon Lake, PURL II successfully completed the constant depth following and bottom following missions. These trials were the first autonomous missions for PURL II in a lake. Moreover, this was the first bottom following mission in a lake. Unfortunately, the sawtooth mission was unsuccessful because PURL II veered off course and wedged itself under a submerged log. PURL II eventually extracted itself and was lost for approximately 30 minutes before it surfaced for retrieval. The mission script loop controlling the sawtoothing entered a state where it seized all the available processing time and pre-empted all of the other control tasks. This is a serious deficiency in the PROTEUS mission scripting language. To prevent a seizure from occurring, timed waits were added to the sawtooth loop.

12.1.7 September 23, 1996

Trial: Autonomous Missions For The NSERC Demonstration

Location: Loon Lake Small Arm

Tests: Depth Following, Bottom Following, and Sawtooth Missions.

The Depth Following, Bottom Following and Sawtooth Missions were all performed properly by PURL II.

Trial: Long Distance Mission

Location: Loon Lake Main Body

Test: Endurance

Following a constant heading and constant depth, PURL II travelled out from the dock in the main body of Loon Lake. After 1500 seconds, PURL II turned around and headed back to the dock. PURL II travelled approximately 1.8 kilometres round trip, at a depth of 2 meters and a velocity of 0.6 m/s.

12.1.8 September 27, 1996

Trial: NSERC Demonstration Of Depth Following

Location: Loon Lake Arm

Goal: Demonstrate Depth Following, Bottom Following, and Sawtooth Profiles To NSERC

PURL II successfully performed Depth Following , Bottom Following and Sawtooth Profiling missions for NSERC.

Trial: Altimeter Analysis

Location: Loon Lake Arm

Tests: Altimeter Parameters

PURL II ran sixteen missions across the short axis of the Loon Lake arm. Different altimeter parameters and settings were tested at different altitudes. For more information on these trials please see Maier, 1997.

12.1.9 May 28, 1997

Trial: Fibre Optic Cable and Video System Test

Location: Loon Lake Arm

Tests: Altitude For Employing The Video Camera

PURL II ran a variety of trials across the short axis of the Loon Lake Arm while towing the fibre optic tether and employing a video camera. The goal of these trials was to determine what altitudes worked well for the Micro Sea Cam video camera. It appears that an altitude of between one meter and two meters provides the best results. At higher altitudes the lights carried by PURL II do not provide enough illumination to achieve enough contrast for general viewing and object identification. The fibre optic cable and the surface viewing equipment worked well. The fibre optic cable snagged on the bottom during one of the runs, but we were able to free the vehicle without entering the water. PURL II can also be stopped and dragged backwards by the fibre optic cable which allows us to operate in deep water. If a failure occurs, PURL II can be retrieved by manually pulling it to the surface.

Trial Fibre Optic Cable and Video System Test

Location: Loon Lake Main Body

Tests: Deep Tests of the Fibre Optic Cable and Lights

PURL II ran along the bottom of Loon Lake at depths ranging from 30 meters to 15 meters. PURL II was able to pull the fibre optic cable through the water although its speed was reduced. The lights appeared to work well, there were no "hot spots" and they provided enough illumination to travel one meter to two meters above the bottom of the lake. It was difficult to determine how much fibre optic cable was in the water, therefore pay out markings should be added to the cable. PURL II and its fibre optic system was deployed and operated from the URL canoe.

12.2 Missions

During trials, PURL II performed the three basic mission types that can be combined to create more complex survey and scientific data collection missions. The trials also showed that PURL II can be deployed in a remote location with little logistical support. One can drive close to Loon Lake, but for the final 100 to 200 meters to the launch site, PURL II must be wheeled over rough terrain. To demonstrate that PURL II can actually survey and collect scientific data, we examined the internal waves caused by wind action across Loon Lake..

Internal waves are found in stratified waters and have traditionally been investigated using an array of self-recording sensors such as thermistor chains. While the temporal resolution of thermistor chains is generally excellent, the spatial resolution is often poor because the chains are spaced far apart (Laval, 1997b). An AUV can provide good spatial resolution because it moves horizontally through the water.

Internal waves are studied because density stratification is a barrier to vertical mixing and transport within the water column (Laval, 1997b). Surface waters tend to be oxygen rich but nutrient poor, and deeper water tend to be oxygen poor and nutrient rich. Exchanging oxygen and nutrients between the surface and deeper water is essential for maintaining a healthy lake ecology (Laval, 1997b). When a pollutant is added to a body of water, proper dispersal is inhibited by stratification, or when water is withdrawn from a reservoir, stratification determines what type of water is withdrawn. In both cases, knowing the effects of stratification is important for maintaining healthy bodies of water.

On November 26 (JED 330), November 27 (JED 331) and December 2 (JED 336), 1996, PURL II performed CTD surveys along the long axis of the main body of Loon Lake. In order to measure the position of the thermocline along the length of the lake, repeated sawtooth profiles were performed. There is no meteorological data for Loon Lake other than qualitative data collected on the survey dates. On JED 330 and JED 331 it was overcast with a light rain and no wind. On JED 336 there was a strong southerly wind and it was snowing heavily. In the days between JED 331 and JED 336 there was a wind and snow storm throughout the Greater Vancouver Region including Loon Lake. The wind storm provided us with the opportunity to collect data that compares the internal waves in Loon Lake during a period of calm weather with data collected during a wind event.

12.2.1 Mission Tracks

PURL II completed five runs during the three mission days. Figure 12-2 and Figure 12-3 show two of the five mission tracks that were followed (Laval, 1997a). Because PURL II is not yet equipped with deep water retrieval equipment, a float and string was attached to prevent loss in the event of a system failure. Figure 12-4 and Figure 12-5 show the sawtooth profiles and bathymetry data generated during the outbound legs of Mission 4 and Mission 5. The sawtooth profiles were conducted between ten and twenty meters depth unless the bottom prevented PURL II from reaching the bottom of the profile. Table 12-1 shows the mission specifics for the five missions with the outbound leg containing an "a" suffix and the return leg a "b" suffix (Laval, 1997a).

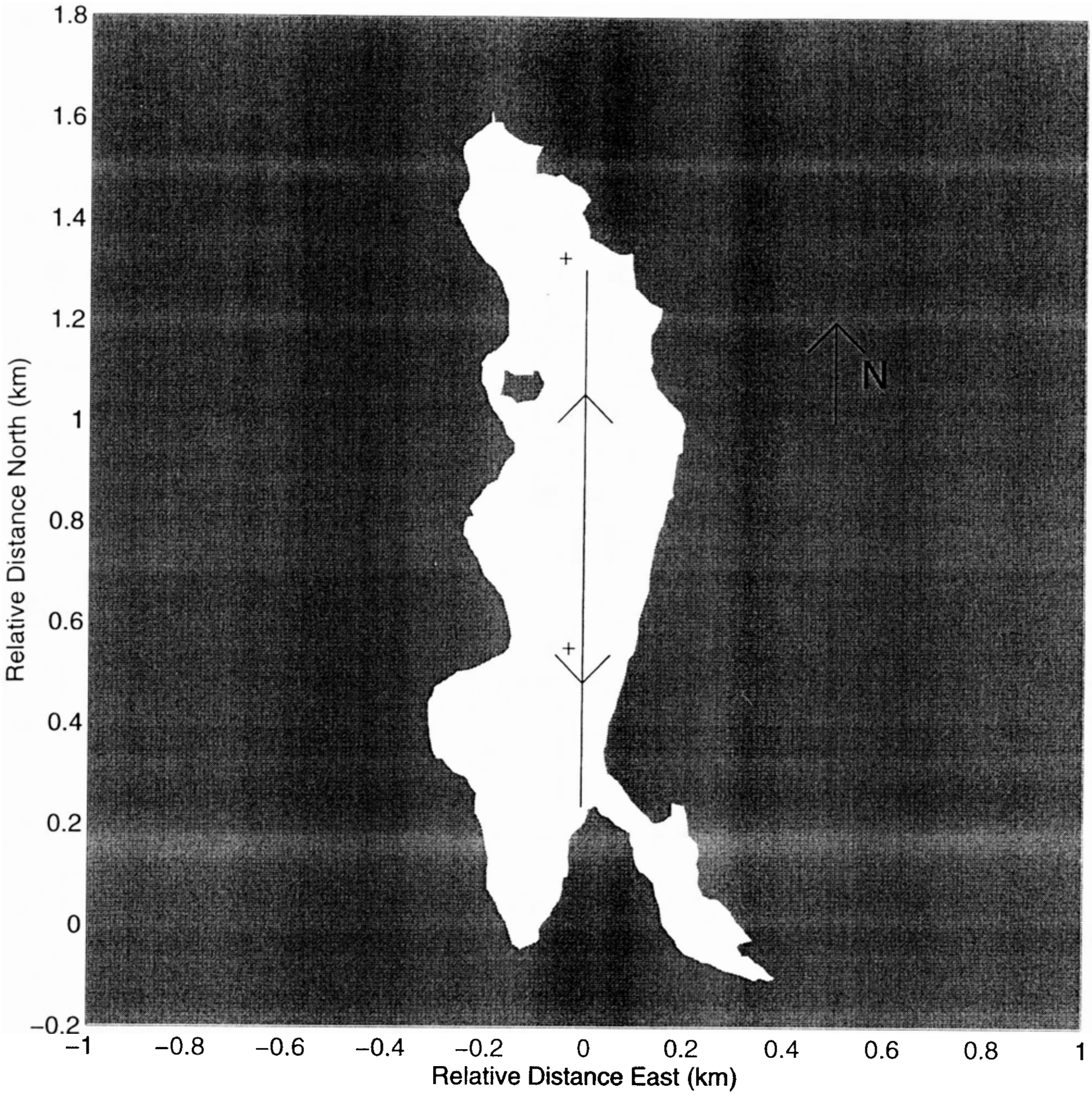


Figure 12-2: Mission 4, JED 330, Mission Path

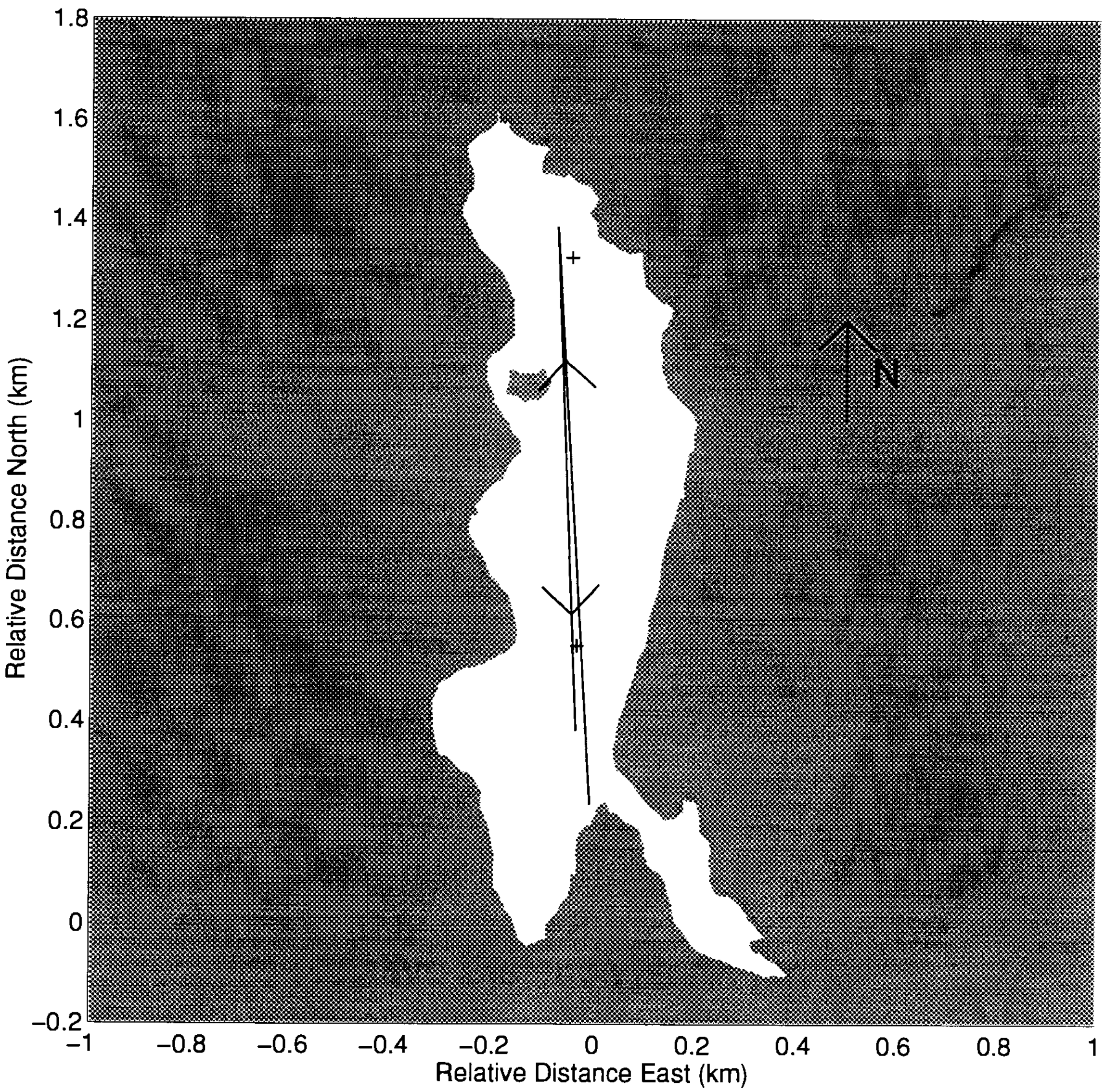


Figure 12-3: Mission 5, JED 336, Mission Path

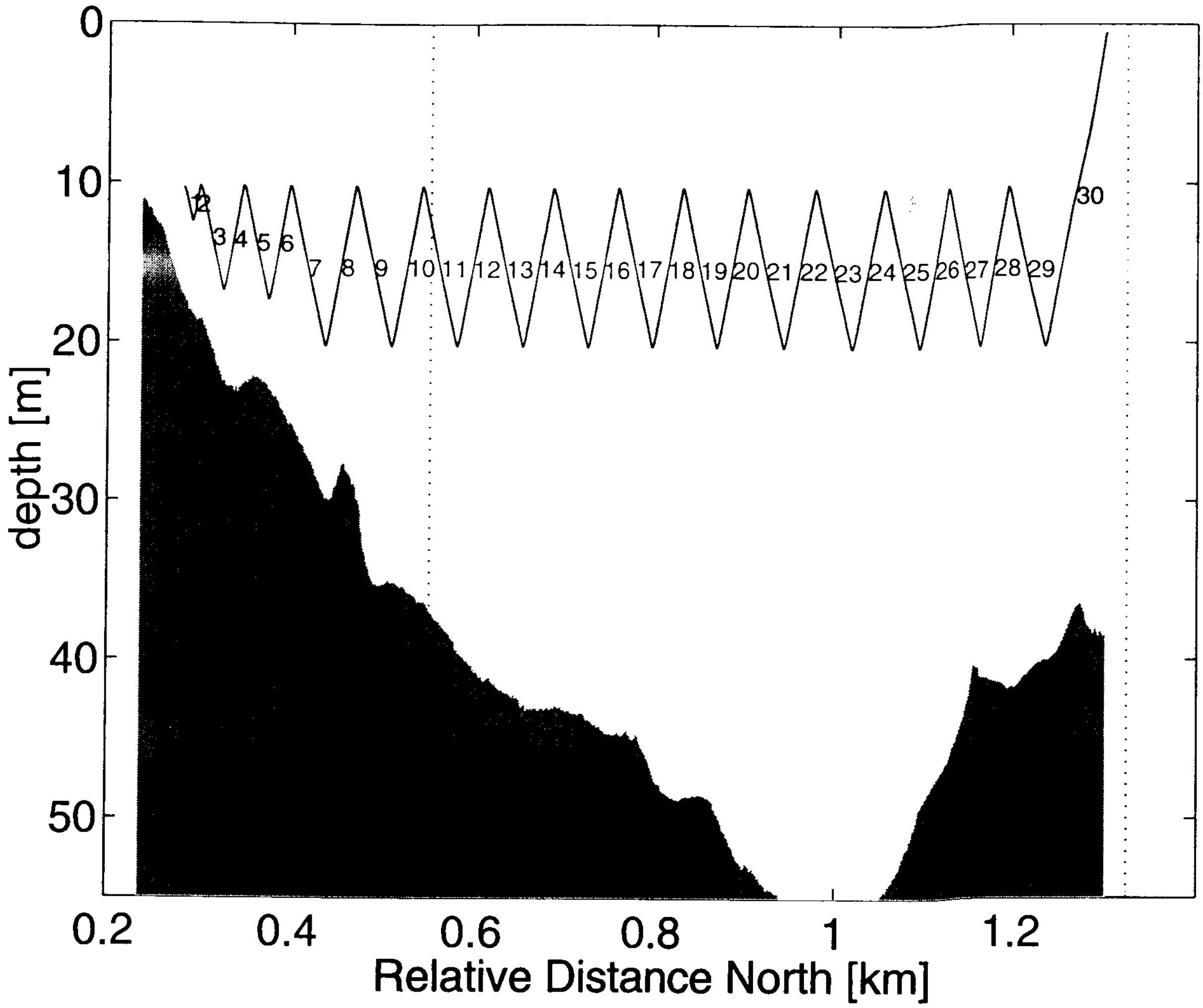


Figure 12-4: Mission 4, JED 331, Outbound Sawtooth Profiles

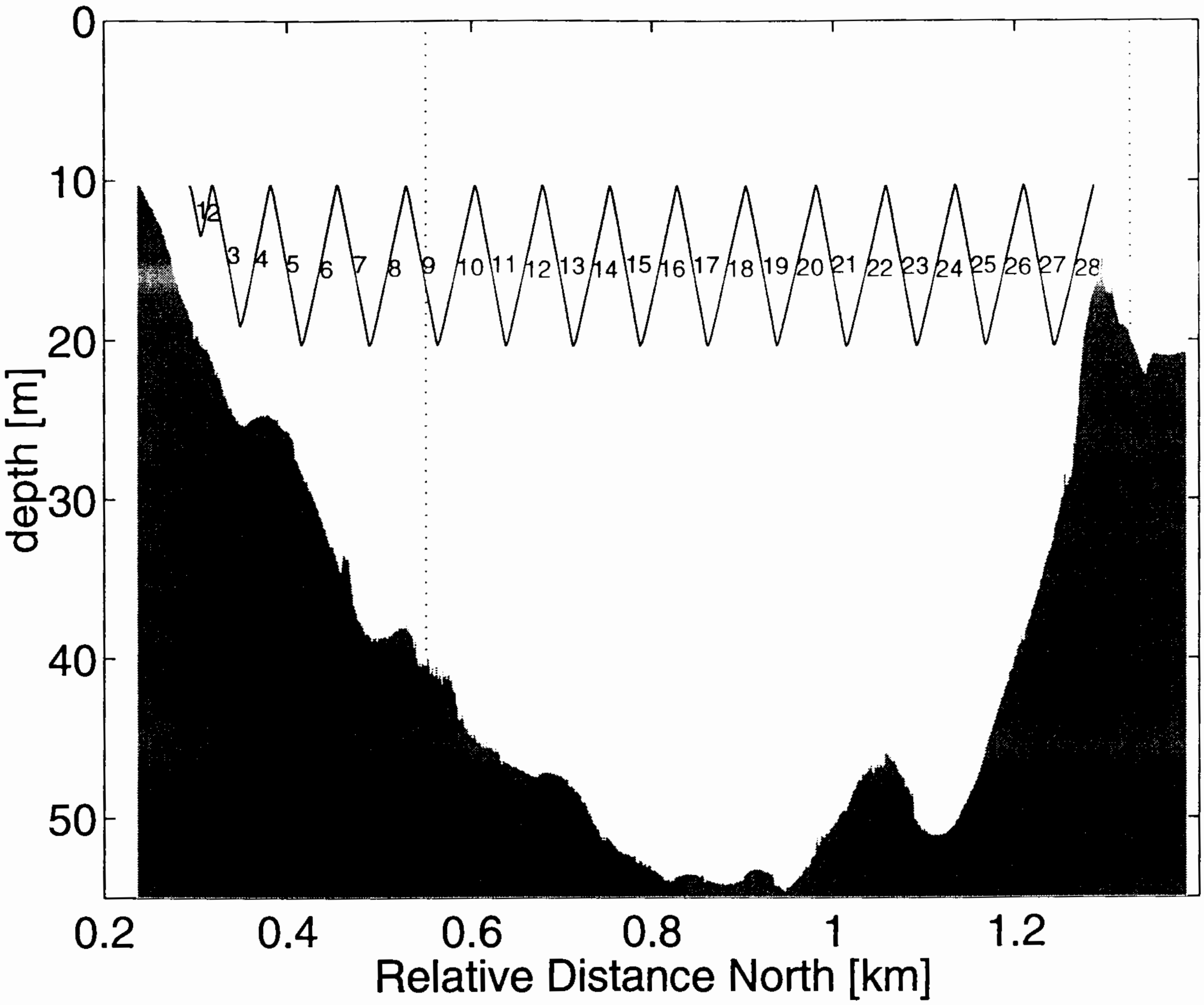


Figure 12-5: Mission 5, JED 336, Outbound Sawtooth Profiles

Table 12-1: Mission Specifics

Mission	Day	Start	End	Length (m)	Duration (min)	# of Profiles
1 a	330	9:58	10:28	640	30	18
1 b	330	10:30	11:00	640	30	18
2 a	330	11:53	12:35	860	40	26
2 b	330	12:35	13:15	860	40	20
3 a	331	9:08	10:02	1050	50	28
3 b	331	10:02	10:41	850	40	18
4 a	331	13:57	14:50	1060	50	30
4 b	331	14:50	15:36	1060	50	26
5 a	336	10:23	11:15	1150	50	28
5 b	336	11:16	12:09	1000	50	26

12.2.2 Mission Profiles

An analysis of the sawtooth profile data collected during the five mission series can be found in "PURL II / Loon Lake Fall 1996 Raw Data Report" by Bernard Laval. Figure 12-6 and Figure 12-7 show the profiles for the outbound legs of Mission 4 and Mission 5. After comparing these two profiles, it is possible to see the increased variability in thermocline depth caused by the wind event.

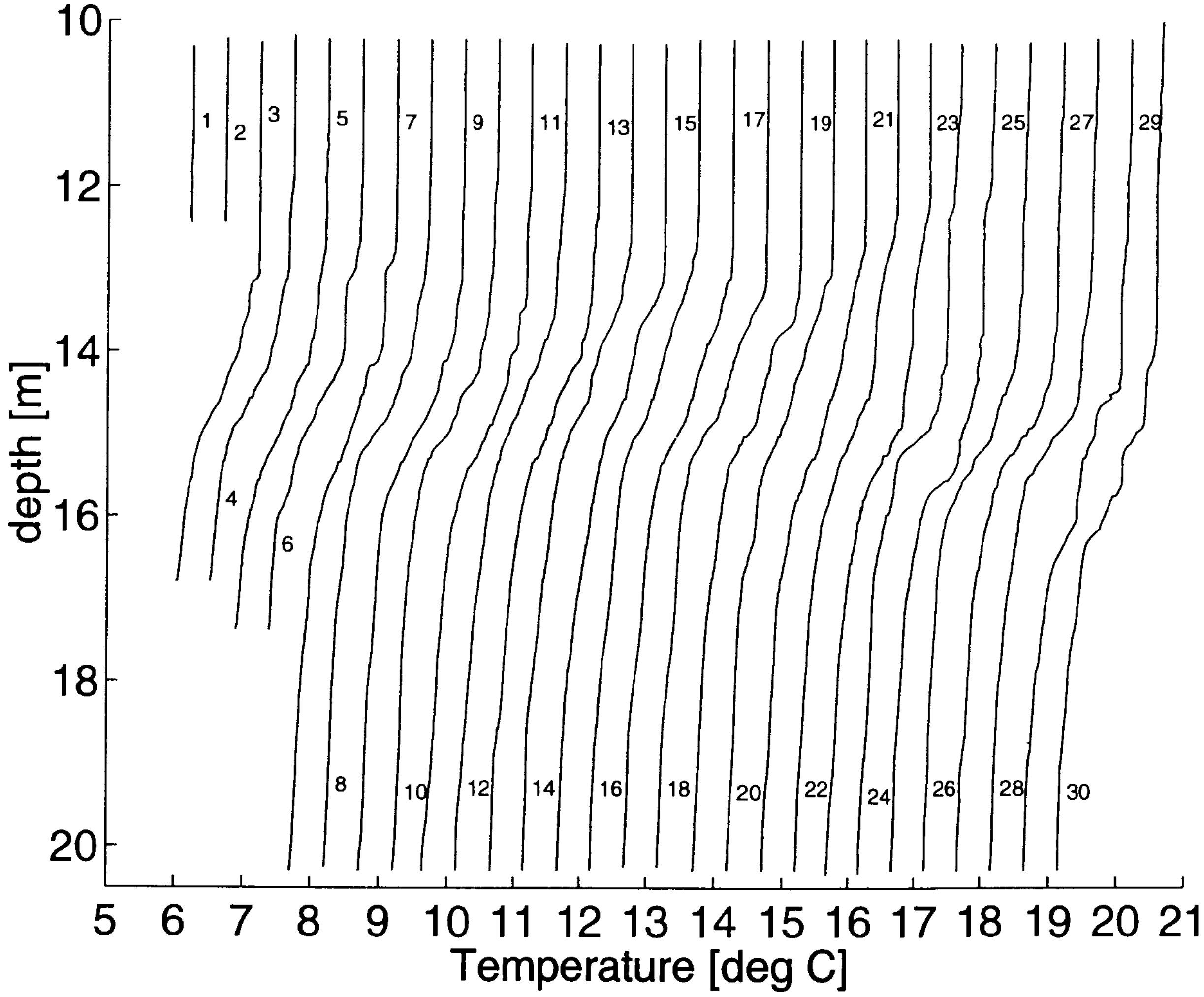


Figure 12-6: JED 331, Mission 4 a, Profiles Spaced 0.5°C Apart

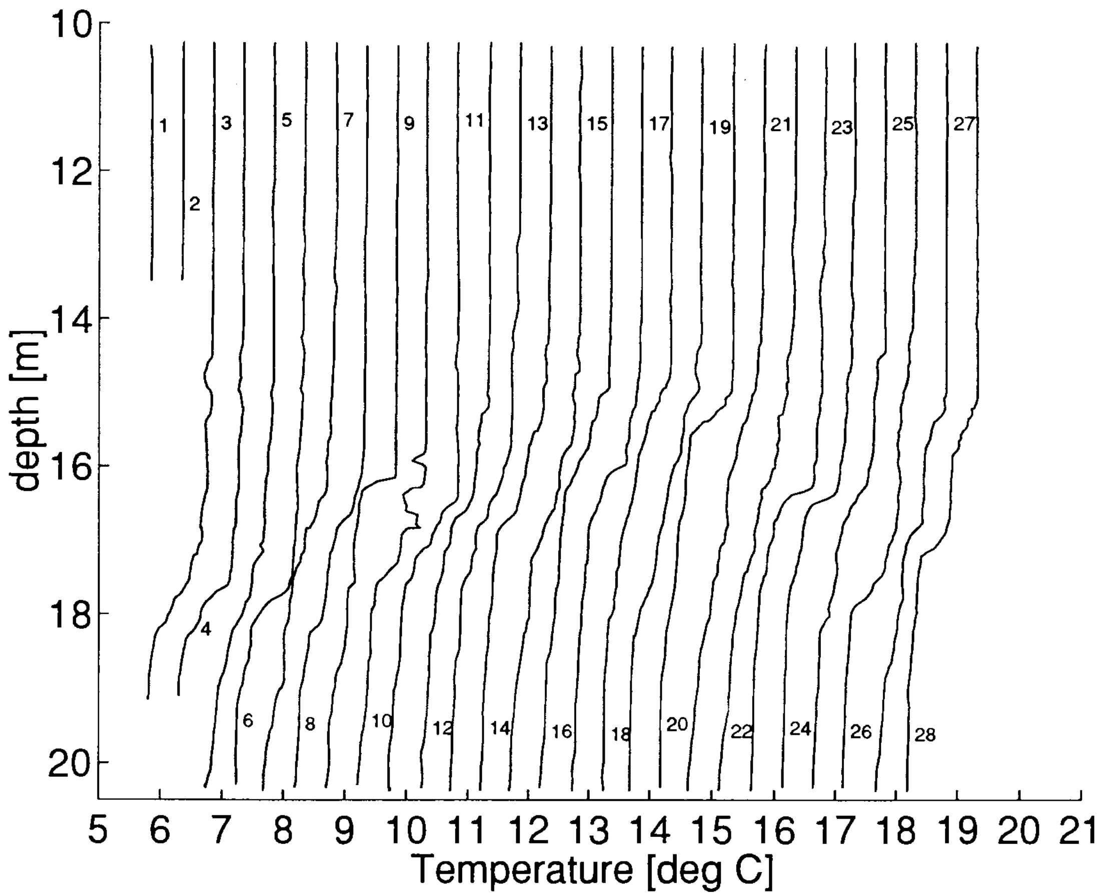


Figure 12-7: JED 336, Mission 5 a, Profiles Spaced 0.5°C Apart

After the missions were complete, Bernard Laval made several recommendations about improving the usability of the data collected by PURL II. It is important to stress that the desires and needs of scientists should be fulfilled by future AUV designs. Scientists will employ AUVs if, and only if, tasks can be performed at a level of risk and cost acceptable to the scientists. The following is an abbreviated list of the recommendations made by Mr. Laval (1997a).

1) Better positioning can be facilitated by breaking missions into two parts where the return leg starts at a fixed location such as a buoy, not where the outbound leg ends.
2) In order to perform limnological studies, a meteorological station recording parameters such as wind speed and temperature is essential.
3) Increase the resolution of the data logging time stamp to 0.1 seconds. The current time stamp resolution is one second.
4) Record the times when PURL II passes known landmarks as a method of updating its position.

13. Future Enhancements

The list of PURL II's future enhancements could be extensive if the scope is not constrained by time, money, and the URL's research plans. The following is a list of enhancements that may be pursued in future phases of the PURL II project.

Mechanical Enhancements

- New mounts for both the horizontal and vertical thrusters. They should be fared to reduce vehicle drag.
- Reducing vehicle drag by filling the holes and depressions in the faring
- Add removable foam sections in the bow and stern of the faring to increase the payload capacity.

Electrical Enhancements

- Add more powerful horizontal thrusters with larger propellers and lower screw speed. The horizontal motor controllers may also require changes to handle the increased current draw of the new thrusters.

Software Enhancements

- Upgrade the Ethernet utilities available to the operator so that PROTEUS can be launched without employing Carbon Copy LAN. TCP\IP command calls may be the solution. Carbon Copy LAN should always be maintained for debugging and verification purposes.
- As ISER upgrades PROTEUS and moves from DOS to QNX, the URL should also move PURL II to QNX. QNX would eliminate restrictions such as a maximum of five serial ports. QNX can operate as a real time OS, and DOS cannot.
- Increase the data logging resolution from one second to 0.1 seconds.

General Enhancements

- Add deep water (>10m) retrieval equipment so PURL II can be relocated and retrieved if lost in deeper waters.
- Finish building up the other fibre optic penetrators and cables that are currently in the URL.
- Add distance markings to the fibre optic cable so that the pay out length is known.
- Develop a payload pressure vessel. The payload pressure vessel should be supplied with power and an Ethernet connection to the main pressure vessel.
- Efforts should be made to reduce the time it takes to change the battery pack. Although not prohibitive, reducing down time increases mission time.

14. Conclusions

Successfully completing the PURL II missions proved that AUVs can be employed to collect data in lakes and other remote locations. PURL II demonstrated its ability to operate as a rapid deployment search and survey AUV by performing a survey of the internal waves within Loon Lake. PURL II also operated in a variety of weather conditions ranging from calm, warm and sunny days, through to windy, cold and snowy ones. PURL II has many deficiencies such as the absence of an accurate positioning system, but dead reckoning is adequate for many missions and inspection tasks.

The utility of AUVs is inversely proportional to their cost and fear of losing them. If one cannot afford to lose the AUV, it will never be permitted to perform an autonomous mission. PURL II operated autonomously in the arm of Loon Lake where SCUBA divers could retrieve a lost vehicle, but PURL II was always tethered in the main body where retrieval is not possible at deeper depths. Once deep water location and retrieval equipment is developed for PURL II, it will operate autonomously in the main body of Loon Lake. Reducing the cost of a AUV increases its utility because it will be permitted to perform a greater variety of missions.

Reductions in the cost, size, and power consumption of electronic components will lead to higher levels of physical and electrical integration, thus reducing the size weight, and cost of AUVs. Smaller, cheaper and faster are the desired attributes of AUVs because as size decreases, operating costs decrease, and as speed increases, the quantity of data collected in a specified period increases. When AUVs demonstrate they can perform tasks at a cost and level of risk acceptable to users, AUVs will start gaining acceptance in the various underwater communities.

Off-the-shelf software such as PROTEUS dramatically reduced the time and cost of developing PURL II. Interface components specific to PURL II were the only software components that were added to PROTEUS. Employing off-the-shelf software allows developers to amortise costs over time and among different users. Without PROTEUS it would have taken substantially more time and money to develop PURL II.

PURL II does not contain many redundant systems because human life is not endangered by a failure, and the money, weight and space consumed by redundant systems reduce the AUV's utility. If a failure is detected, PURL II attempts to return to the surface for retrieval. If PURL II cannot return to the surface, it must be located and retrieved by external means such as SCUBA divers or a grapple. Losing the AUV is an unwanted but acceptable option because the value of the tasks performed are greater than the risk adjusted cost of losing the AUV and replacing it.

Employing the fibre optic system is warranted only for sophisticated tasks that require a human operator to apply his or her intelligence to the mission. Whenever possible, autonomous missions should be employed because they consume fewer resources; the AUV travels faster and surveys a larger volume of water, the pre-mission and post-mission times are reduced, and the operator(s) can perform other tasks while the AUV is surveying autonomously. For the foreseeable future, there will always be missions, especially visual inspection and identification tasks, that require a human operator.

Clients should be defining the performance specifications and mission parameters from which future AUVs are designed. Bounding what an AUV must accomplish reduces costs because the design can be minimised. Also, the client will allow the AUV to operate autonomously because its risks and costs are outweighed by the value of the tasks it performs.

PURL II is a first step along the path of developing small, inexpensive AUVs. Performing autonomous and tethered missions in Loon Lake with little logistical support demonstrated PURL II's ability to be a rapid deployment survey vehicle. Future enhancements such as deep water location and retrieval equipment will increase PURL II's utility and expand the scope of missions it performs. PURL II is a continuing project in the Underwater Research Lab at Simon Fraser University that will build on the utility demonstrated in this thesis.

References

Alt, Christopher von, Ben Allen, Thomas Austin and Roger Stokey. July 1994. "Remote environmental measuring units," in *Proceedings of the 1994 Symposium on AUV Technology*. Cambridge, Massachusetts. 13-19.

Anderson, Jamie M. June 1992. "Model development for control of the autonomous benthic explorer," in *Proc. of the International Offshore and Polar Engineering Conference*, San Francisco, CA: Vol. 2, 468-472.

Bellingham, J. G., C. A. Goudey, T. R. Consi and C. Chryssostomidis. June 1992. "A small, long-range autonomous vehicle for deep ocean exploration," in *Proc. of the International Offshore and Polar Engineering Conference*, San Francisco, CA: Vol. 2, 461-467.

Bellingham, J. G., C. A. Goudey and Chryssostomos Chryssostomidis. April 1993. "Economic ocean survey capability with AUVs. Systems overview: intelligent control, navigation, communications, energy storage, propulsion, subsystem power use," *Sea Technology* 12-17.

Bellingham, J. G., C. A. Goudey, T. R. Consi, J. W. Bales, D. K. Atwood, J. J. Leonard and C. Chryssostomidis. July 1994. "A second generation survey AUV," in *Proceedings of the 1994 Symposium on AUV Technology*. Cambridge, Massachusetts. 148-155.

Bird, John S., April 1997. "Size Bounds and Survey Limits of Autonomous Underwater Vehicles and Marine Mammals," Manuscript in preparation.

Chryssostomidis, Chryssostomos, Henrik Schmidt and James Bellingham. June 1993. "Autonomous underwater vehicles," *Research Department Of Ocean Engineering At MIT, 100th Anniversary Issue*, Cambridge, Massachusetts: 16-22.

Ferguson, James. 1997. "UUV Evolution: Exploration to Commercialization," *Aviation Week & Space Technology - Association for Unmanned Vehicles Systems International, International Guide To Unmanned Vehciles / 1997-98.*

International Submarine Engineering Research Ltd., 1991. "AUV Mission Planner And Executor,"

Kreider, John R.February 1997. "UUVs for Underwater Work - Innovation or High Tech Toy?," *Sea Technology*. 25-32.

Laval, Bernard, May 1997a. "PURL II / Loon Lake Fall 1996 Raw Data Report,"

Laval, Bernard, John S. Bird, and Peter D. Helland, 1997b. "Observations Of The Spatial Structure Of Internal Waves In A Small Mid-Latitude Lake," presented at *Oceans 97*. Halifax, Nova Scotia. October 6-9, 1997.

Smith, Samuel M. and Stanley E. Dunn. July 1994. "The Ocean Voyager II: An AUV designed for coastal oceanography," in *Proceedings of the 1994 Symposium on AUV Technology*. Cambridge, Massachusetts. 139-147.

Yeorger, Dana R., Albert M. Bradley and Barrie B. Walden. Nov 1990. "The autonomous benthic explorer (ABE): A deep ocean AUV for scientific seafloor survey," presented at Seminar On Autonomous Underwater Vehicles, Tokyo, Japan

Appendix One

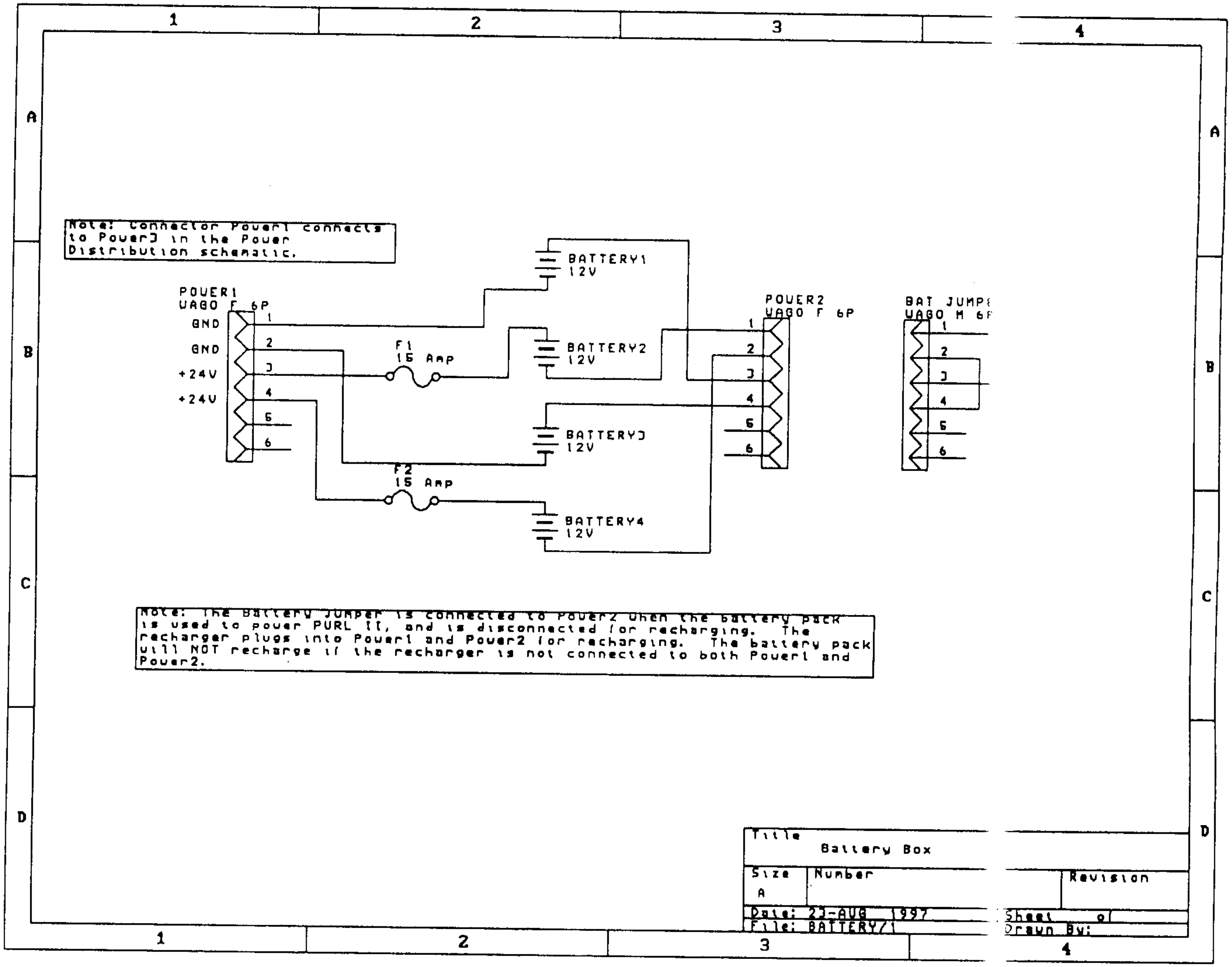
The following is a list of the schematics that make up the wiring and printed circuit boards inside PURL II.

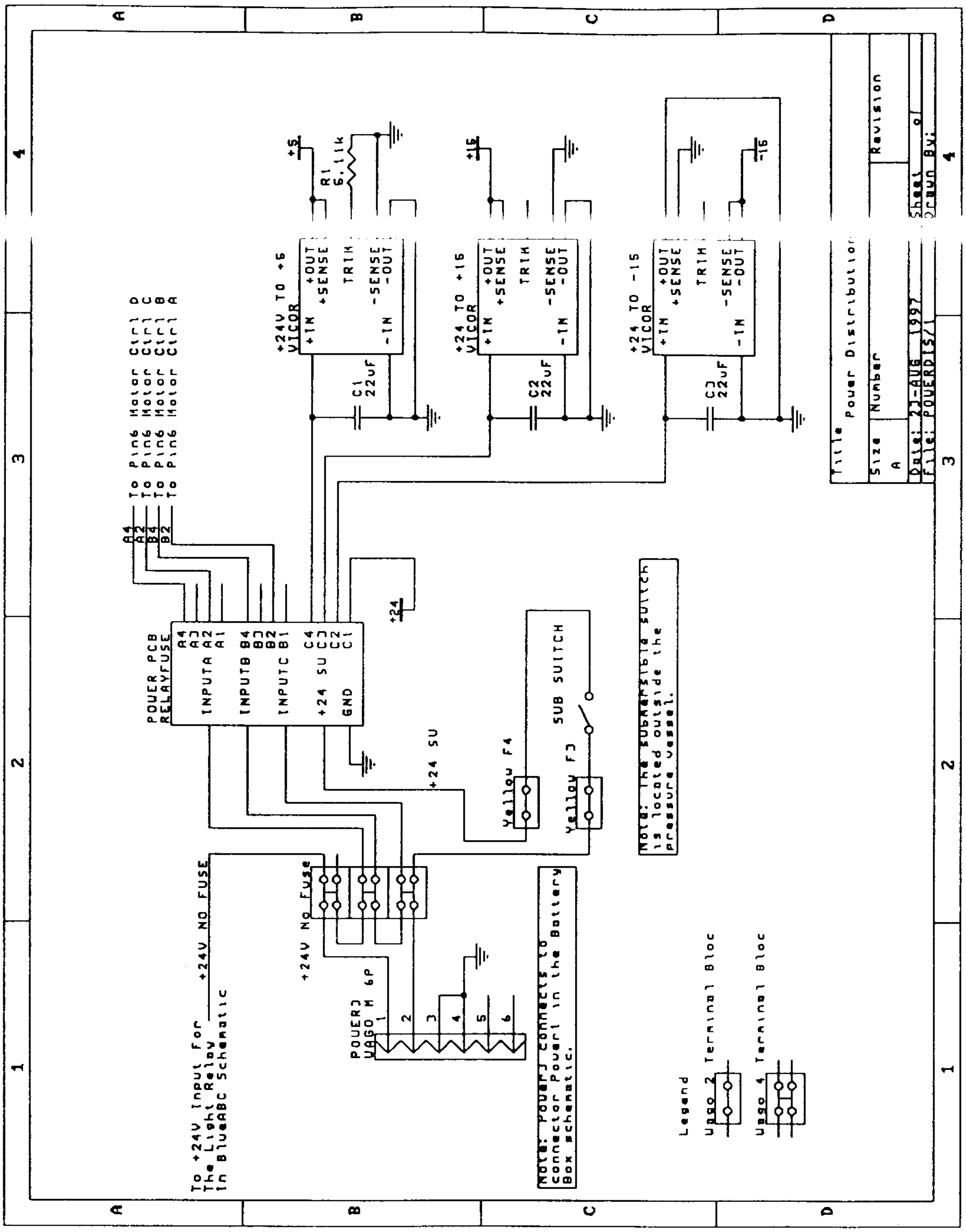
Wiring Schematics

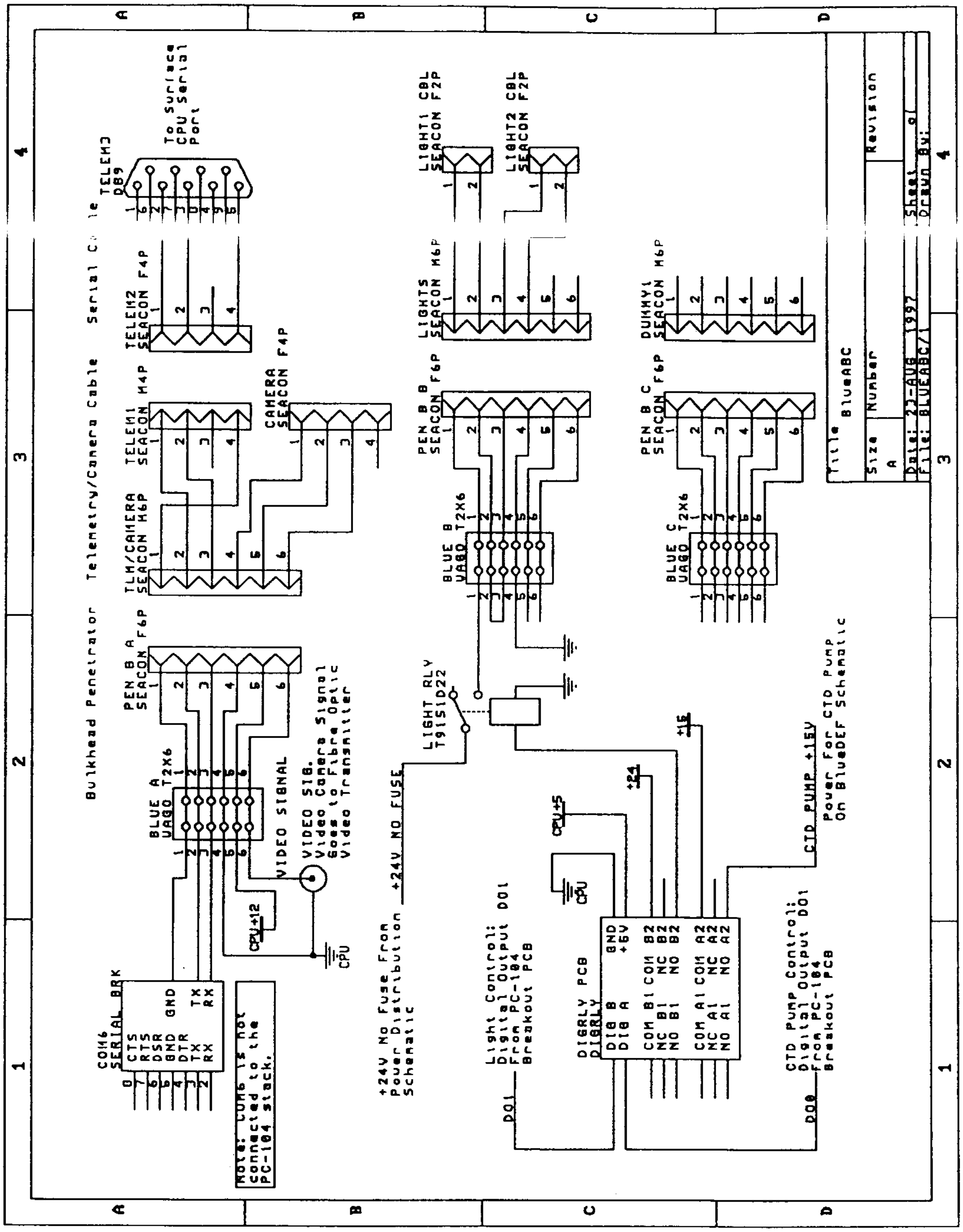
Schematic Name	Description
Battery Box	Wiring and pin out of the battery packs
Power Distribution	Power distribution for the main power supply buses
BlueABC	Wiring and pin outs for the Blue A,B,C penetrators
BlueDEF	Wiring and pin outs for the Blue D,E,F penetrators
YellABCD	Wiring and pin outs for the Yellow A,B,C,D penetrators
YellEF	Wiring and pin outs for the Yellow E,F penetrators
Fibre Optic/Ethernet	Fibre Optic and Ethernet Connections and Cabling
Miscellaneous	Compass, Tilt Sensor and Leak Sensor Wiring

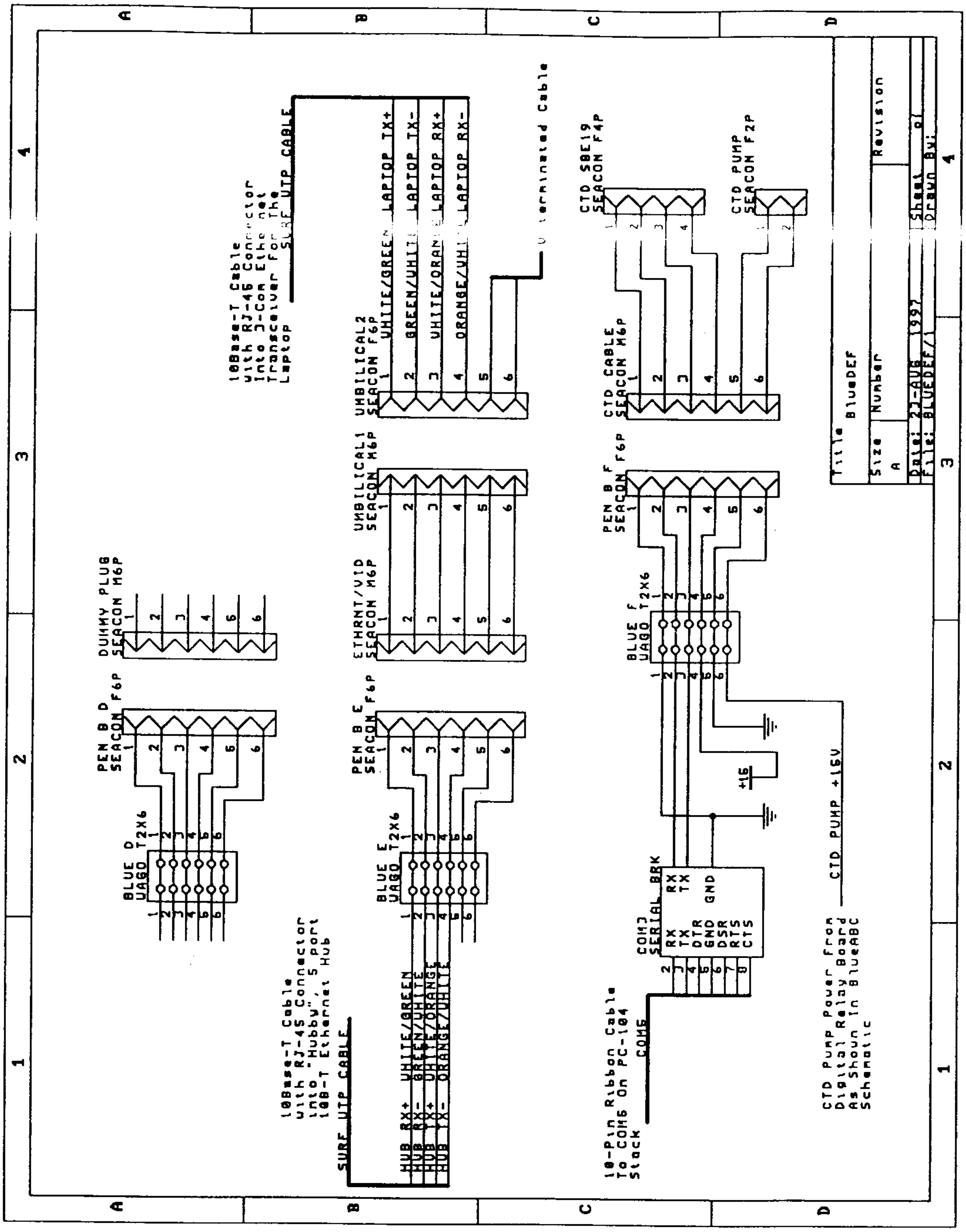
Printed Circuit Boards

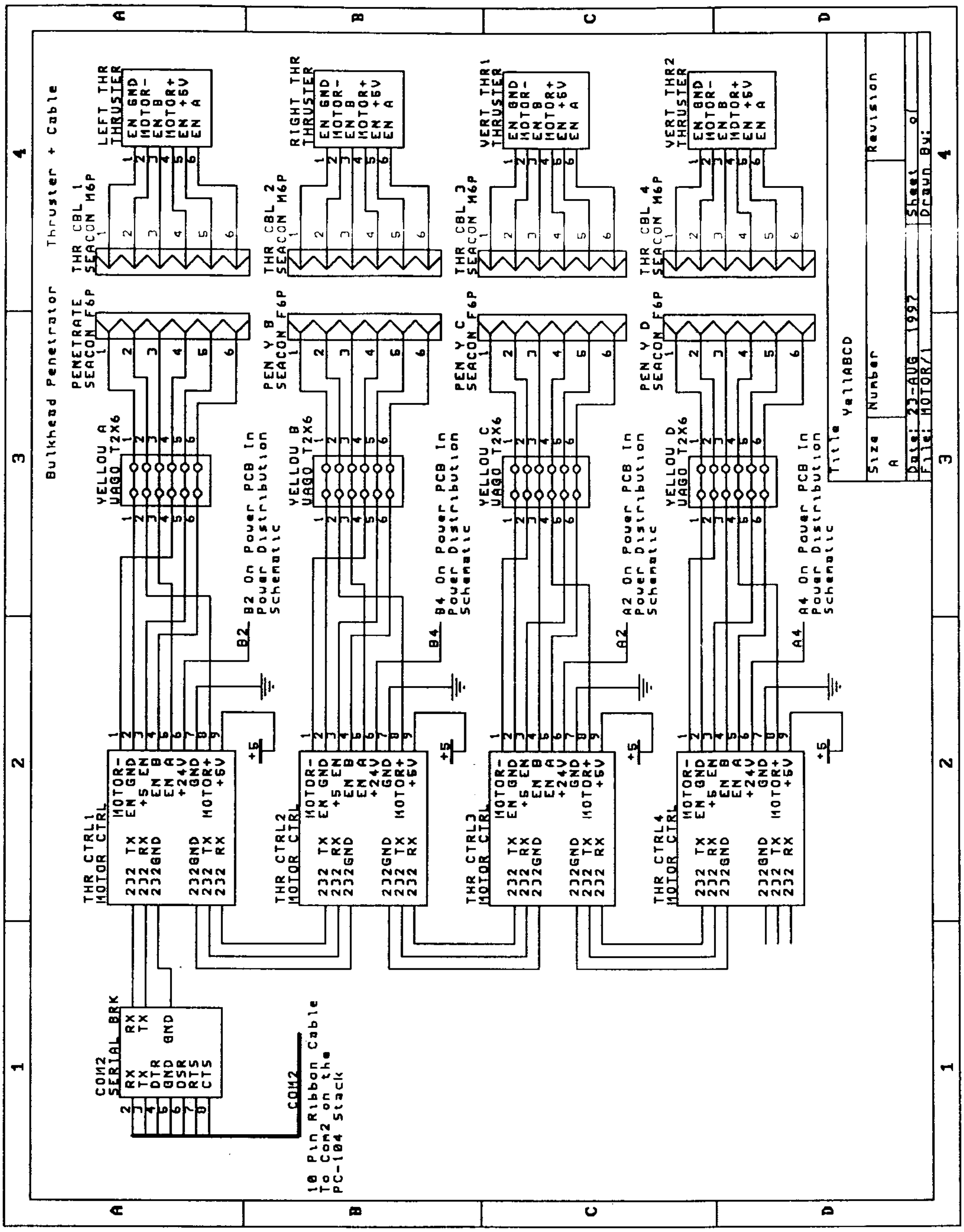
Schematic Name	Description
PC-104 Breakout	PC-104 Breakout Board on top of the PC-104 stack
Relay and Fuse	Power control relays and fuses
Serial Breakout	10-pin ribbon cable to 3-wire RS-232 converter
Digital Relay	Two relays controlled by TTL outputs
Leak Sensor	Leak Detector board


Note: Connector Power1 connects to Power3 in the Power Distribution schematic.
POWER1
WAGO F 6P
GND
GND
+24V
+24V
F1
15 Amp
F2
15 Amp
BATTERY1
12V
BATTERY2
12V
BATTERY3
12V
BATTERY4
12V
POWER2
WAGO F 6P
BAT JUMP
WAGO M 6
Note: The Battery Jumper is connected to Power2 when the battery pack is used to power PURL II, and is disconnected for recharging. The recharger plugs into Power1 and Power2 for recharging. The battery pack will NOT recharge if the recharger is not connected to both Power1 and Power2.
Title
Battery Box
Size
A
Number
Revision
Date: 23-AUG 1997
File: BATTERY/1
Sheet of
Drawn By:

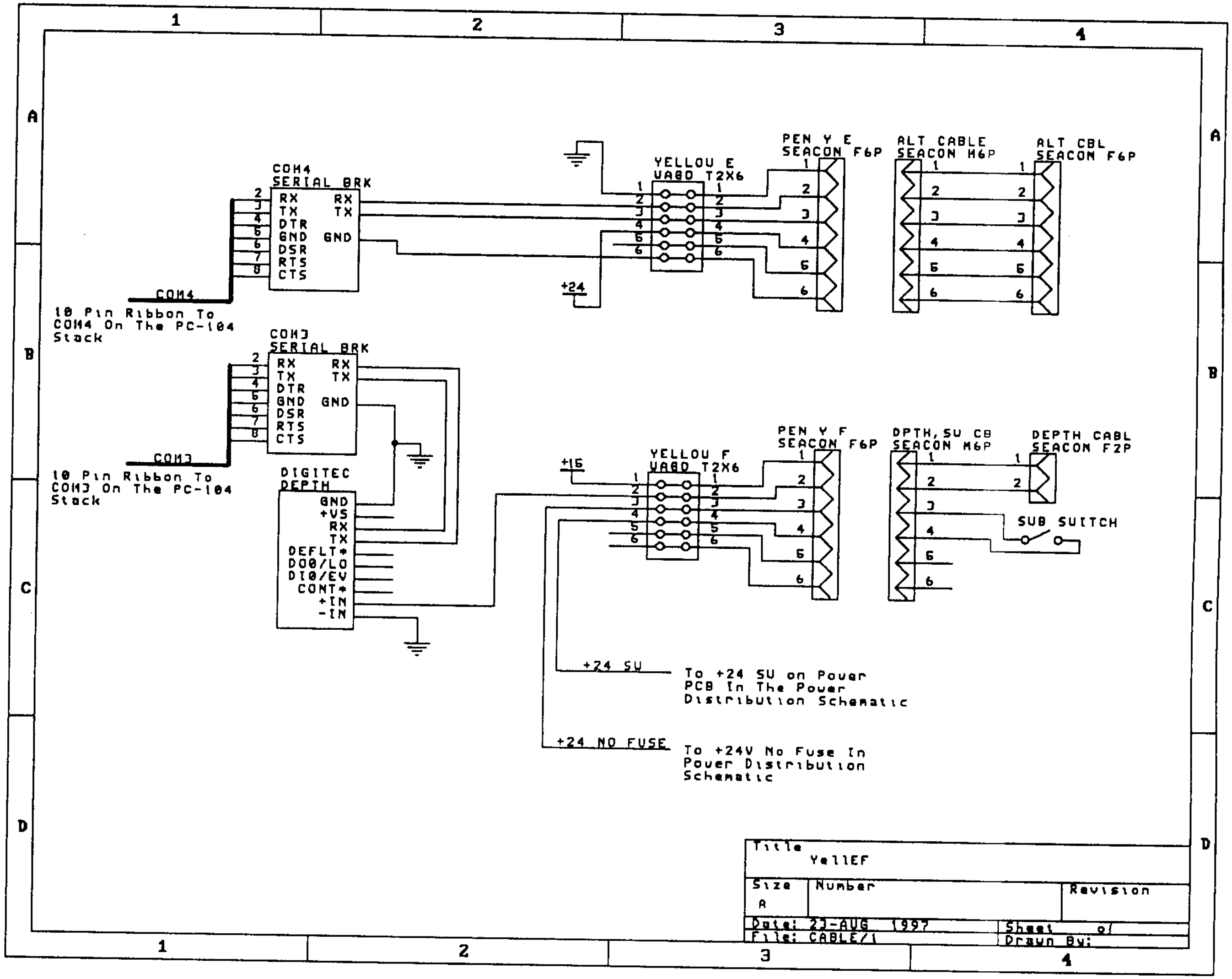

POWER PCB
RELAYFUSE
INPUTA A4 A3 A2 A1
INPUTB B4 B3 B2 B1
INPUTC B1
+24 SW C4 C3 C2 C1
GND
To Pin6 Motor Ctrl D
To Pin6 Motor Ctrl C
To Pin6 Motor Ctrl B
To Pin6 Motor Ctrl A
+24V TO +5
VICOR
+IN +OUT +SENSE TRIM -SENSE -OUT -IN
R1
6.11k
C1
22uF
+24 TO +15
VICOR
C2
22uF
+24 TO -15
VICOR
C3
22uF
+5
+15
-15
+24
+24 SW
SUB SWITCH
Yellow F4
Yellow F3
Note: The submersible switch is located outside the pressure vessel.
To +24V Input For
The Light Relay
In BlueABC Schematic
+24V NO FUSE
+24V No Fuse
POWER3
WAGO M 6P
1 2 3 4 5 6
Note: Power3 connects to connector Power1 in the Battery Box schematic.
Legend
Wago 2 Terminal Bloc
Wago 4 Terminal Bloc
Title
Power Distribution
Size
A
Number
Revision
Date: 23-AUG-1997
File: POWERDIS/1
Sheet of
Drawn By:

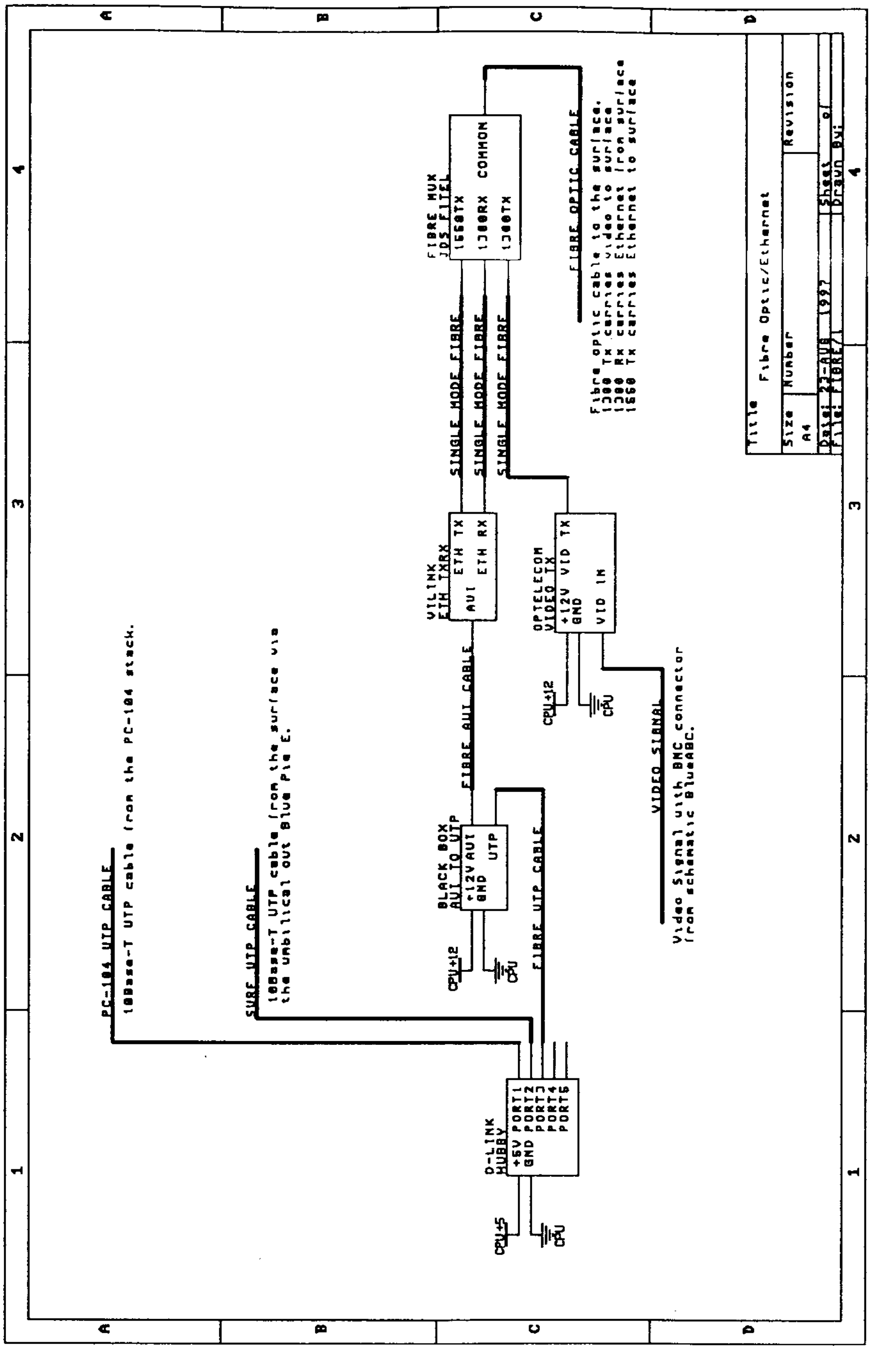

Bulkhead Penetrator
Telemetry/Camera Cable
Serial Cable
TELEM3
DB9
To Surface
CPU Serial
Port
TELEM2
SEACON F4P
TELEM1
SEACON M4P
TLM/CAMERA
SEACON M6P
CAMERA
SEACON F4P
PEN B A
SEACON F6P
BLUE A
VAGO T2X6
COM6
SERIAL BRK
CTS
RTS
DSR
GND
DTR
TX
RX
Note: COM6 is not connected to the PC-104 stack.
CPU+12
CPU
VIDEO SIGNAL
VIDEO SIG.
Video Camera Signal
Goes to Fibre Optic
Video Transmitter
+24V NO FUSE
+24V No Fuse From
Power Distribution
Schematic
LIGHT1 CBL
SEACON F2P
LIGHT2 CBL
SEACON F2P
LIGHTS
SEACON M6P
PEN B B
SEACON F6P
BLUE B
VAGO T2X6
LIGHT RLY
T91S1D22
Light Control:
Digital Output DO1
From PC-104
Breakout PCB
DO1
CPU+5
+24
+15
DIGRLY PCB
DIGRLY
DIG B
DIG A
GND
+5V
COM B1
NC B1
NO B1
COM B2
NC B2
NO B2
COM A1
NC A1
NO A1
COM A2
NC A2
NO A2
DUMMY1
SEACON M6P
PEN B C
SEACON F6P
BLUE C
VAGO T2X6
CTD PUMP +15V
Power For CTD Pump
On BlueDEF Schematic
CTD Pump Control:
Digital Output DO0
From PC-104
Breakout PCB
DO0
Title
BlueABC
Size
A
Number
Revision
Date: 23-AUG 1997
Sheet of
File: BLUEABC/1
Drawn By:

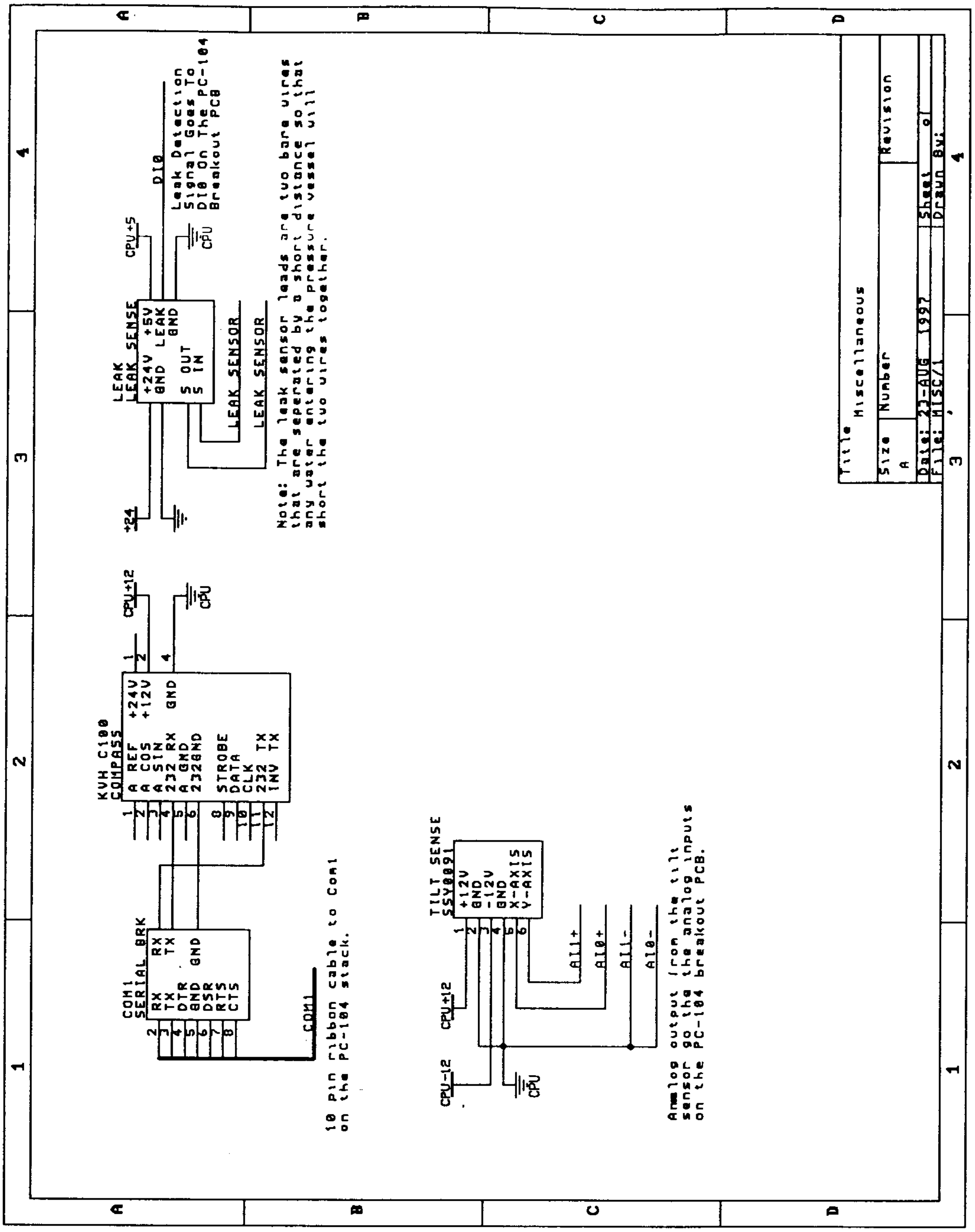

10Base-T Cable with RJ-46 Connector Into 3-Com Ethernet Transceiver For The Laptop
SURF UTP CABLE
UMBILICAL2 SEACON F6P
WHITE/GREEN LAPTOP TX+
GREEN/WHITE LAPTOP TX-
WHITE/ORANGE LAPTOP RX+
ORANGE/WHITE LAPTOP RX-
Unterminated Cable
CTD SBE19 SEACON F4P
CTD PUMP SEACON F2P
CTD CABLE SEACON M6P
PEN B D SEACON F6P
DUMMY PLUG SEACON M6P
BLUE D WAGO T2X6
PEN B E SEACON F6P
ETHRNT/VID SEACON M6P
UMBILICAL1 SEACON M6P
BLUE E WAGO T2X6
PEN B F SEACON F6P
BLUE F WAGO T2X6
10Base-T Cable with RJ-45 Connector Into "Hubby", 5 port 10B-T Ethernet Hub
SURF UTP CABLE
HUB RX+ WHITE/GREEN
HUB RX- GREEN/WHITE
HUB TX+ WHITE/ORANGE
HUB TX- ORANGE/WHITE
10-Pin Ribbon Cable To COM6 On PC-104 Stack
COM6
COM3 SERIAL BRK
RX TX DTR GND DSR RTS CTS
RX TX GND
+16
CTD PUMP +16V
CTD Pump Power From Digital Relay Board As Shown In BlueABC Schematic
Title BlueDEF
Size A
Number
Revision
Date: 23-AUG-1997
Sheet of
File: BLUEDEF/1
Drawn By:

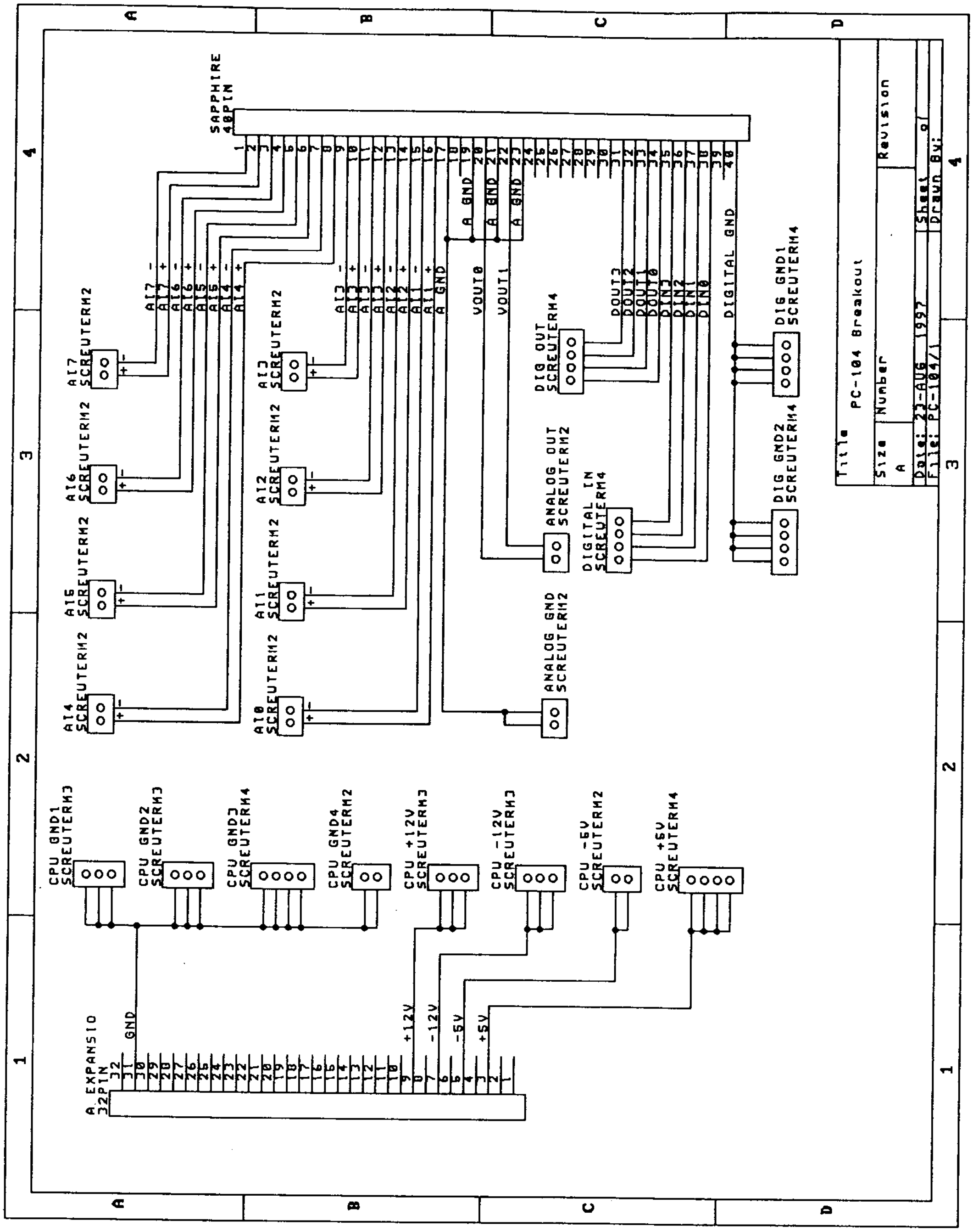


Bulkhead Penetrator
Thruster + Cable
LEFT THR THRUSTER
RIGHT THR THRUSTER
VERT THR1 THRUSTER
VERT THR2 THRUSTER
THR CBL 1 SEACON M6P
THR CBL 2 SEACON M6P
THR CBL 3 SEACON M6P
THR CBL 4 SEACON M6P
PENETRATE SEACON F6P
PEN Y B SEACON F6P
PEN Y C SEACON F6P
PEN Y D SEACON F6P
VELLOU A UAGO T2X6
VELLOU B UAGO T2X6
VELLOU C UAGO T2X6
VELLOU D UAGO T2X6
B2 On Power PCB In Power Distribution Schematic
B4 On Power PCB In Power Distribution Schematic
A2 On Power PCB In Power Distribution Schematic
A4 On Power PCB In Power Distribution Schematic
THR CTRL1 MOTOR CTRL
THR CTRL2 MOTOR CTRL
THR CTRL3 MOTOR CTRL
THR CTRL4 MOTOR CTRL
COM2 SERIAL BRK
COM2
10 Pin Ribbon Cable To Com2 on the PC-104 Stack
Title VallABCD
Size A
Number
Revision
Date: 23-AUG-1997
File: MOTOR/1
Sheet
Drawn By:

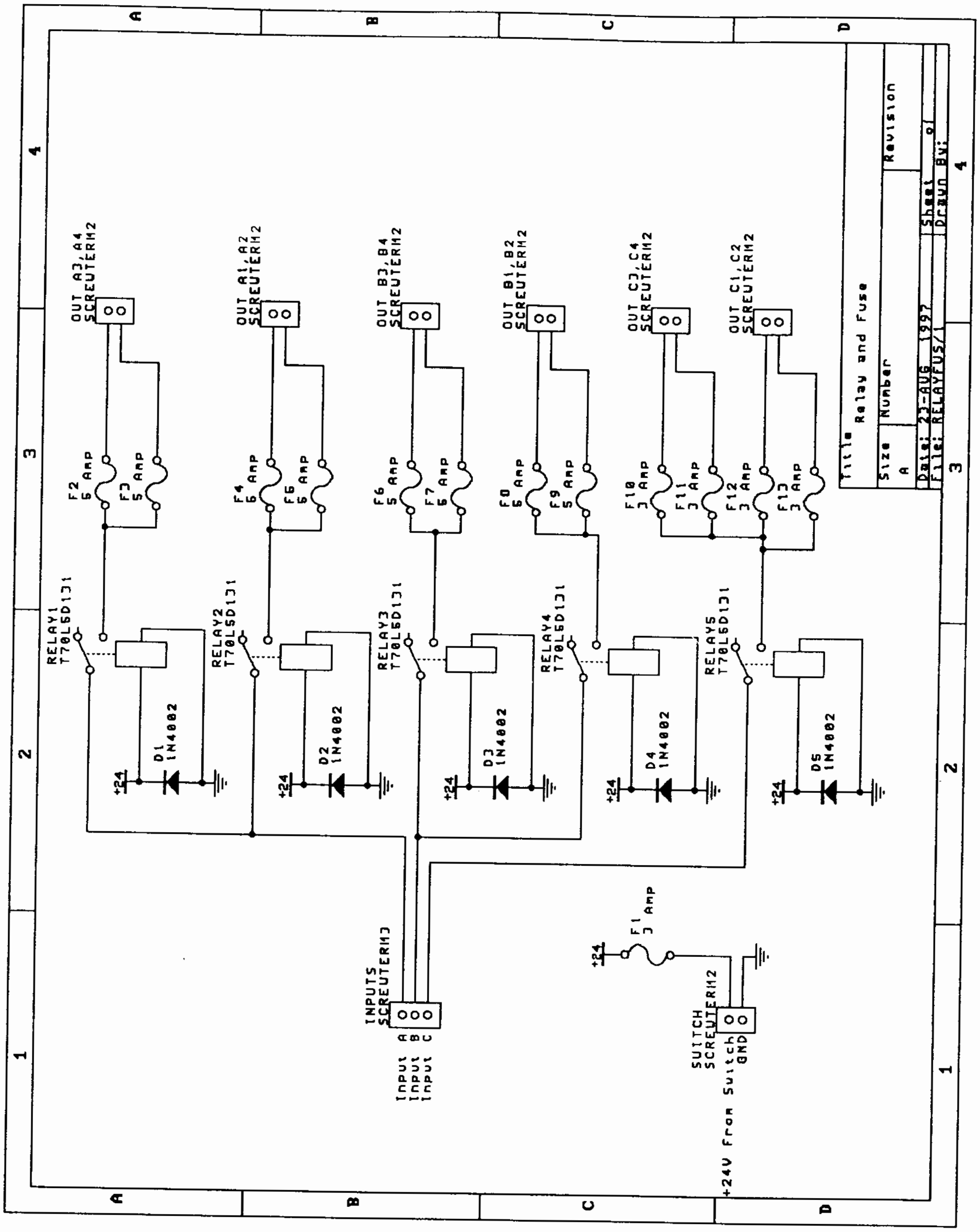

COM4
SERIAL BRK
RX
TX
DTR
GND
DSR
RTS
CTS
GND
COM4
10 Pin Ribbon To COM4 On The PC-104 Stack
COM3
SERIAL BRK
COM3
10 Pin Ribbon To COM3 On The PC-104 Stack
DIGITEC DEPTH
GND
+VS
DEFLT+
DO0/LO
DI0/EV
CONT+
+IN
-IN
+24
YELLOW E
VA80 T2X6
PEN Y E
SEACON F6P
ALT CABLE
SEACON M6P
ALT CBL
SEACON F6P
+15
YELLOW F
VA80 T2X6
PEN Y F
SEACON F6P
DPTH,SW CB
SEACON M6P
DEPTH CABL
SEACON F2P
SUB SWITCH
+24 SW
To +24 SW on Power PCB In The Power Distribution Schematic
+24 NO FUSE
To +24V No Fuse In Power Distribution Schematic
Title
YellEF
Size
A
Number
Revision
Date: 23-AUG 1997
File: CABLE/1
Sheet of
Drawn By:

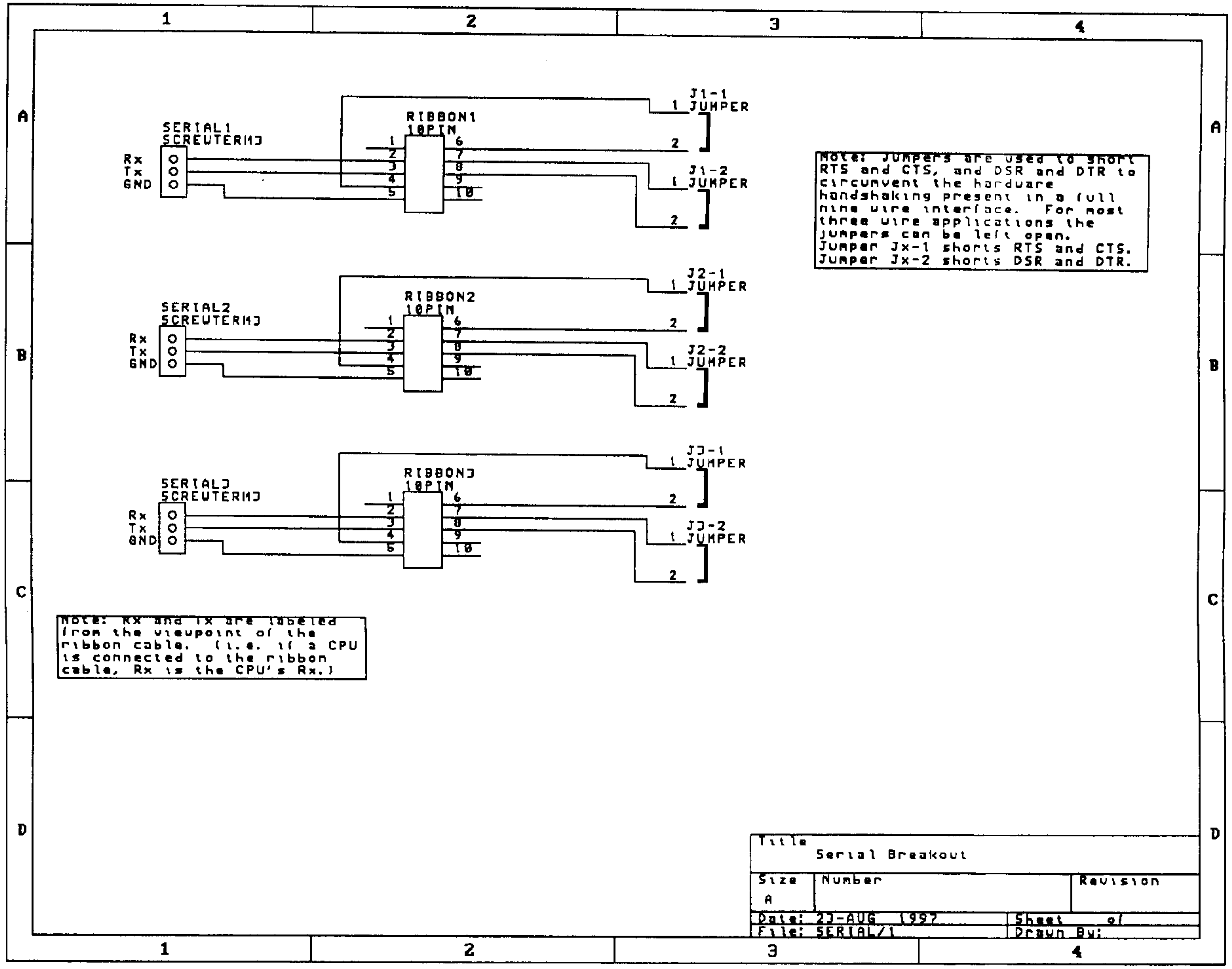

PC-104 UTP CABLE
10Base-T UTP cable from the PC-104 stack.
SURF UTP CABLE
10Base-T UTP cable from the surface via the umbilical out Blue Pie E.
D-LINK
HUBBY
+5V PORT1
GND PORT2
PORT3
PORT4
PORT6
CPU+5
CPU
BLACK BOX
AUI TO UTP
+12V AUI
GND
UTP
CPU+12
CPU
FIBRE UTP CABLE
FIBRE AUI CABLE
VILINK
ETH TXRX
AUI
ETH TX
ETH RX
OPTELECOM
VIDEO TX
+12V VID TX
GND
VID IN
CPU+12
CPU
VIDEO SIGNAL
Video Signal with BNC connector
from schematic BlueABC.
SINGLE MODE FIBRE
SINGLE MODE FIBRE
SINGLE MODE FIBRE
FIBRE MUX
JDS FITEL
1660TX
1300RX COMMON
1300TX
FIBRE OPTIC CABLE
Fibre optic cable to the surface.
1300 Tx carries video to surface
1300 Rx carries Ethernet from surface
1660 Tx carries Ethernet to surface
Title
Fibre Optic/Ethernet
Size
A4
Number
Revision
Date: 22-AUG-1997
Sheet of
File: FIBRE/1
Drawn By:

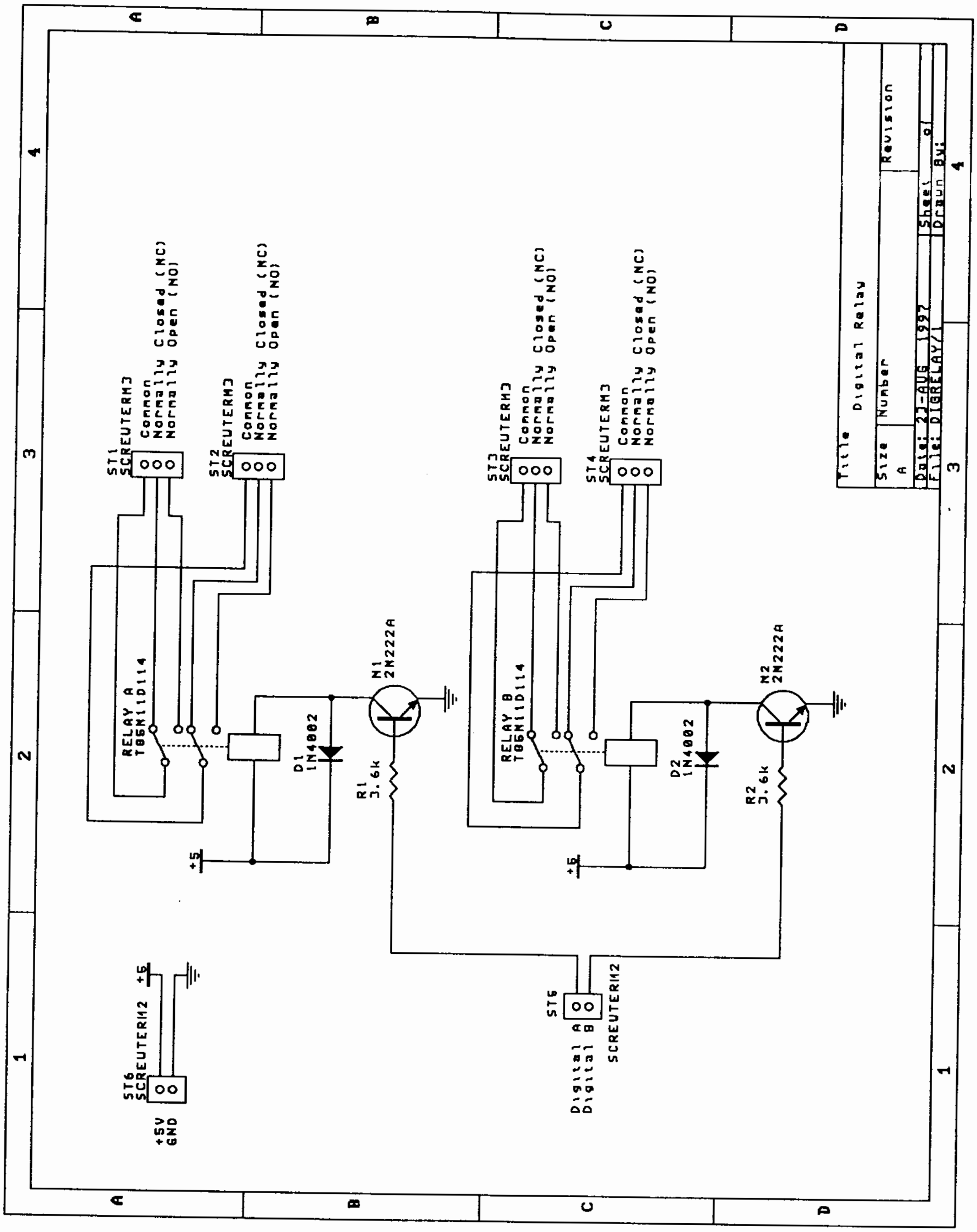

COM1
SERIAL BRK
RX
TX
DTR
GND
DSR
RTS
CTS
GND
COM1
10 pin ribbon cable to Com1 on the PC-104 stack.
KVH C100
COMPASS
A REF
A COS
A SIN
232 RX
A GND
232GND
STROBE
DATA
CLK
232 TX
INV TX
+24V
+12V
GND
CPU+12
CPU
LEAK
LEAK SENSE
+24V
GND
S OUT
S IN
+5V
LEAK
GND
LEAK SENSOR
LEAK SENSOR
+24
CPU+5
CPU
DI0
Leak Detection Signal Goes To DI0 On The PC-104 Breakout PCB
Note: The leak sensor leads are two bare wires that are seperated by a short distance so that any water entering the pressure vessel will short the two wires together.
TILT SENSE
SSY0091
+12V
GND
-12V
GND
X-AXIS
Y-AXIS
CPU+12
CPU-12
CPU
AI1+
AI0+
AI1-
AI0-
Analog output from the tilt sensor go the analog inputs on the PC-104 breakout PCB.
Title
Miscellaneous
Size
A
Number
Revision
Date: 23-AUG-1997
Sheet of
File: MISC/1
Drawn By:

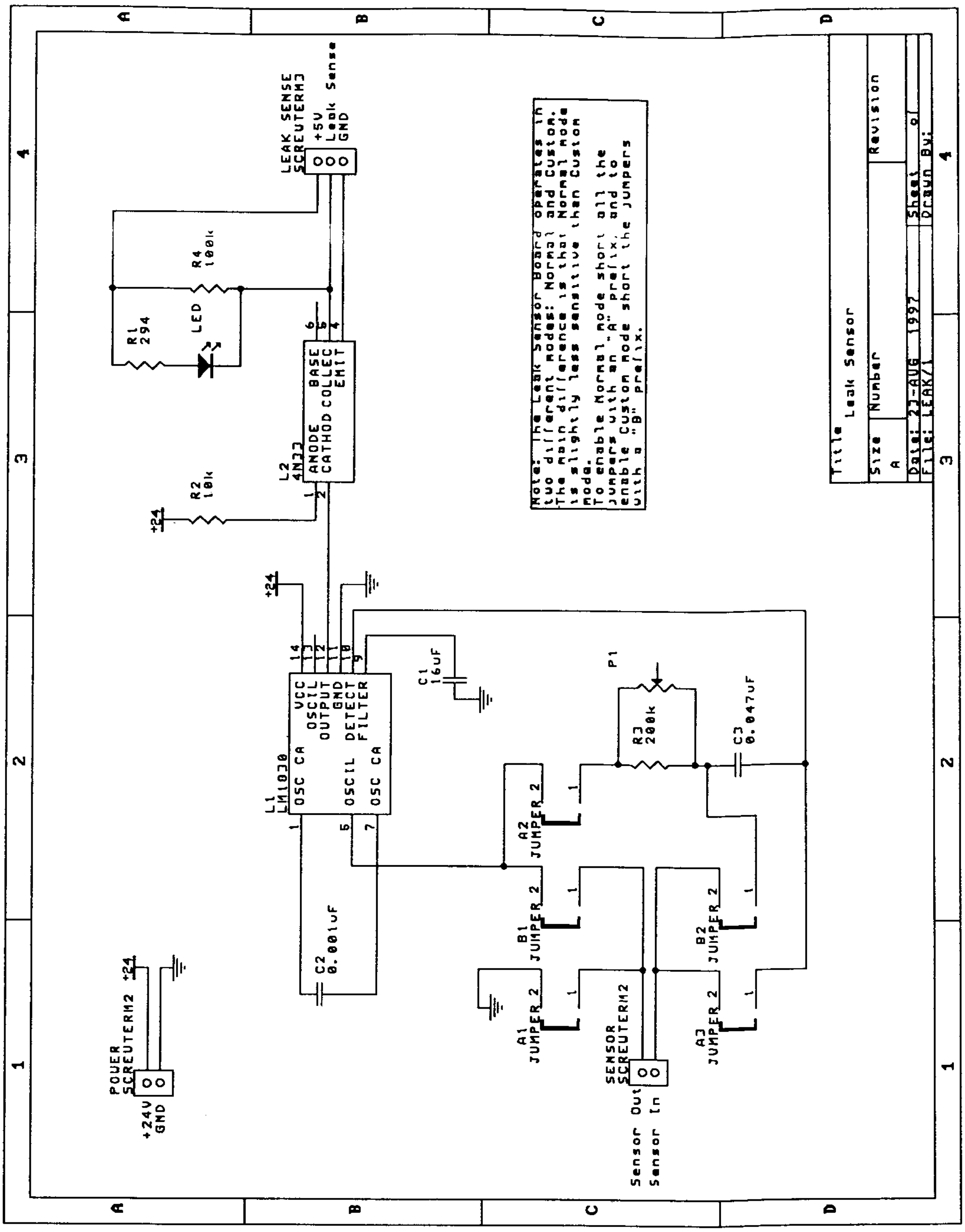


SAPPHIRE
40PIN
A EXPANSIO
32PIN
AI7 SCREUTERM2
AI6 SCREUTERM2
AI5 SCREUTERM2
AI4 SCREUTERM2
AI3 SCREUTERM2
AI2 SCREUTERM2
AI1 SCREUTERM2
AI0 SCREUTERM2
A GND
VOUT0
VOUT1
DOUT3
DOUT2
DOUT1
DOUT0
DIN3
DIN2
DIN1
DIN0
DIGITAL GND
DIG OUT SCREUTERM4
ANALOG OUT SCREUTERM2
DIGITAL IN SCREUTERM4
ANALOG GND SCREUTERM2
DIG GND1 SCREUTERM4
DIG GND2 SCREUTERM4
CPU GND1 SCREUTERM3
CPU GND2 SCREUTERM3
CPU GND3 SCREUTERM4
CPU GND4 SCREUTERM2
CPU +12V SCREUTERM3
CPU -12V SCREUTERM3
CPU -5V SCREUTERM2
CPU +5V SCREUTERM4
GND
+12V
-12V
-5V
+5V
Title
PC-104 Breakout
Size
A
Number
Revision
Date: 23-AUG-1997
Sheet of
File: PC-104/1
Drawn By:


Title
Relay and Fuse
Size
A
Number
Revision
Date: 23-AUG-1997
Sheet of
File: RELAYFUS/1
Drawn By:
OUT A3,A4
SCREWTERM2
OUT A1,A2
SCREWTERM2
OUT B3,B4
SCREWTERM2
OUT B1,B2
SCREWTERM2
OUT C3,C4
SCREWTERM2
OUT C1,C2
SCREWTERM2
F2
5 Amp
F3
5 Amp
F4
5 Amp
F5
5 Amp
F6
5 Amp
F7
5 Amp
F8
5 Amp
F9
5 Amp
F10
3 Amp
F11
3 Amp
F12
3 Amp
F13
3 Amp
RELAY1
T70L5D131
RELAY2
T70L5D131
RELAY3
T70L5D131
RELAY4
T70L5D131
RELAY5
T70L5D131
D1
1N4002
D2
1N4002
D3
1N4002
D4
1N4002
D5
1N4002
+24
INPUTS
SCREWTERM3
Input A
Input B
Input C
F1
3 Amp
SWITCH
SCREWTERM2
+24V From Switch
GND


SERIAL1
SCREWTERM3
Rx
Tx
GND
RIBBON1
10PIN
J1-1
JUMPER
J1-2
JUMPER
SERIAL2
SCREWTERM3
RIBBON2
10PIN
J2-1
JUMPER
J2-2
JUMPER
SERIAL3
SCREWTERM3
RIBBON3
10PIN
J3-1
JUMPER
J3-2
JUMPER
Note: Jumpers are used to short RTS and CTS, and DSR and DTR to circumvent the hardware handshaking present in a full nine wire interface. For most three wire applications the jumpers can be left open. Jumper Jx-1 shorts RTS and CTS. Jumper Jx-2 shorts DSR and DTR.
Note: Rx and Tx are labeled from the viewpoint of the ribbon cable. (i.e. if a CPU is connected to the ribbon cable, Rx is the CPU's Rx.)
Title
Serial Breakout
Size
A
Number
Revision
Date: 23-AUG 1997
Sheet of
File: SERIAL/1
Drawn By:


ST1
SCREUTERM3
Common
Normally Closed (NC)
Normally Open (NO)
ST2
SCREUTERM3
Common
Normally Closed (NC)
Normally Open (NO)
RELAY A
T86N11D114
D1
1N4002
R1
3.6k
N1
2N2222A
+5
ST3
SCREUTERM3
Common
Normally Closed (NC)
Normally Open (NO)
ST4
SCREUTERM3
Common
Normally Closed (NC)
Normally Open (NO)
RELAY B
T86N11D114
D2
1N4002
R2
3.6k
N2
2N2222A
+5
ST6
SCREUTERM2
+5V
GND
+5
ST6
Digital A
Digital B
SCREUTERM2
Title
Digital Relay
Size
A
Number
Revision
Date: 23-AUG-1997
Sheet of
File: DIGRELAY/1
Drawn By:



LEAK SENSE SCREWTERM3
+5V
Leak Sense
GND
R4 100k
R1 294
LED
BASE
COLLEC
EMIT
ANODE
CATHOD
L2 4N33
R2 10k
+24
Note: The Leak Sensor Board operates in two different modes: Normal and Custom. The main difference is that Normal mode is slightly less sensitive than Custom mode. To enable Normal mode short all the jumpers with an "A" prefix, and to enable Custom mode short the jumpers with a "B" prefix.
Title Leak Sensor
Size A
Number
Revision
Date: 23-AUG-1997
Sheet of
File: LEAK/1
Drawn By:
L1 LM1830
VCC
OSCIL
OUTPUT
GND
DETECT
FILTER
OSC CA
C1 16uF
P1
R3 200k
C3 0.047uF
A2 JUMPER 2
B1 JUMPER 2
B2 JUMPER 2
A1 JUMPER 2
A3 JUMPER 2
C2 0.001uF
POWER SCREWTERM2
+24V
GND
SENSOR SCREWTERM2
Sensor Out
Sensor In

Appendix Two

The CSP configuration files for PURL II are divided into three groups; common files, surface files, and PURL II files. Generally, common file names are pre-fixed with an "x", surface files are prefixed with an "s", and PURL II files are pre-fixed with an "p".

Common Configuration Files:
xconst.csp
xtimer.csp
xtelem.csp
display.csp
xstyle.csp

Surface Configuration Files:
sdef.csp
sheader.csp
sports.csp
main_win.csp
setpoint.csp
stat_win.csp
dbg_win.csp
auv_mode.csp

PURL II Configuration Files:
pdef.csp
pheader.csp
pports.csp
control.csp
sapphire.csp
exitscpt.csp
purl_win.csp
payload.csp
mission1.csp
mission2.csp
mission3.csp
mission4.csp
mission5.csp
mission6.csp
log.csp

xconst.csp

```
/*
$Log: xconst.csp $
# Revision 1.8  1997/02/08  17:26:39  COUSTEAU
# Peter H: Removed ABOVE_WATER_DEPTH and TEN_METERS
#
# Revision 1.7  1996/11/24  17:07:22  COUSTEAU
# Peter H: Added MAX_NEG_THRUSTER_VELOCITY
#
# Revision 1.6  1996/08/15  08:33:45  PURL
# Peter H: ABORT condition improvements.
#
# Revision 1.5  1996/08/09  11:46:55  PURL
# AH: changed the interlocking logic to using MotorMode
#
# Revision 1.4  1996/08/08  17:20:25  NEMO
# changed user interface
#
# Revision 1.3  1996/08/06  17:34:47  COUSTEAU
# AH: added EXIT mode
#
# Revision 1.2  1996/08/06  14:29:21  COUSTEAU
# added two constants: ABOVE_WATER_DEPTH and
MAX_THRUSTER_VELOCITY
#
# Revision 1.1  1996/08/01  12:15:21  dosuser
# Initial revision
#
*/
//    *********************SFUCONST.CSP******************

// Const & Define
%define                  name=FALSE               value=0
%define                  name=TRUE                value=1
%const.int               name=FALSE               value=FALSE
%const.int               name=TRUE                value=TRUE
%const.int               name=ZERO                value=0
%const.int               name=ONE                 value=1
%const.int               name=TWO                 value=2
%const.int               name=TEN                 value=10
%float.param             name=NEGATIVE_ONE        value=-1.0
%const.int               name=MAX_TELEM_COUNTER   value=20

%define               name=MAX_THRUSTER_VELOCITY  value=5000
%define               name=MAX_NEG_THRUSTER_VELOCITY  value=-
5000
%const.int            name=MAX_THRUSTER_VELOCITY
value=MAX_THRUSTER_VELOCITY
%const.int            name=MAX_NEG_THRUSTER_VELOCITY
value=MAX_NEG_THRUSTER_VELOCITY

// AUV Mode Defines and Constants
%define                  name=IDLE                value=10
%define                  name=PILOT               value=20
%define                  name=ABORT               value=30
%define                  name=EXIT                value=35
%define                  name=MISSION1            value=41
%define                  name=MISSION2            value=42
%define                  name=MISSION3            value=43
%define                  name=MISSION4            value=44
%define                  name=MISSION5            value=45
%define                  name=MISSION6            value=46
%const.int               name=IDLE                value=IDLE
%const.int               name=PILOT               value=PILOT
%const.int               name=ABORT               value=ABORT
%const.int               name=EXIT                value=EXIT
%const.int               name=MISSION1
value=MISSION1
%const.int               name=MISSION2
value=MISSION2
%const.int               name=MISSION3
value=MISSION3
%const.int               name=MISSION4
value=MISSION4
%const.int               name=MISSION5
value=MISSION5
%const.int               name=MISSION6
value=MISSION6
%define                  name=DEBUG               value=40
%const.int               name=DEBUG               value=DEBUG

// level
%level                   name = TOP               priority = 0
// Highest Priority
%level                   name = HIGH              priority = 1
%level                   name = MEDIUM            priority = 2
%level                   name = LOW               priority = 3
%level                   name = BOTTOM            priority = 4
// Lowest Priority

// layer
%layer                   name = TOP               rank = 5 //
Lowest Priority
%layer                   name = HIGH              rank = 4
%layer                   name = MEDIUM            rank = 3
%layer                   name = LOW               rank = 2
%layer                   name = BOTTOM            rank = 1 //
Highest Priority
```

xtimer.csp

```
/*
$Log: xtimer.csp $
# Revision 1.1  1996/08/01  12:24:55  NEMO
# Initial revision
#
*/

//
******************TIMER.CSP*****************************

%sync   name = TimeTick
%sync   name = SYS_TwoSecondTrigger
%sync   name = SYS_OneSecondTrigger
%sync   name = SYS_TwoHzTrigger
%sync   name = SYS_FourHzTrigger       // 4 Hz timed trigger
%sync   name = SYS_FiveHzTrigger
%sync   name = SYS_TenHzTrigger

%tick
            output   = TimeTick
            interval = 10

%timer
            name     = SYS_Timer
            level    = TOP
            input    = TimeTick

%timer.output
            timer    = SYS_Timer
            interval = 500    // 2 Hz
            event    = SYS_TwoHzTrigger

%timer.output
            timer    = SYS_Timer
            interval = 250    // 4 Hz
            event    = SYS_FourHzTrigger
%timer.output
            timer    = SYS_Timer
            interval = 100    // 10 Hz
            event    = SYS_TenHzTrigger

%timer.output
            timer    = SYS_Timer
            interval = 200    // 5 Hz
            event    = SYS_FiveHzTrigger

%timer.output
            timer    = SYS_Timer
            interval = 2000   // 2 Second Timer
            event    = SYS_TwoSecondTrigger
%timer.output
            timer    = SYS_Timer
            interval = 1000   // 1 Second Timer
            event    = SYS_OneSecondTrigger
```

xtelem.csp

```
/*
$Id: xtelem.csp 1.19 1996/11/22 14:54:15 COUSTEAU Exp $
$Log: xtelem.csp $
# Revision 1.19  1996/11/22  14:54:15  COUSTEAU
# Peter H: Changed AUV_Standard_Conductivity to
AUV_StandardConductivity
#
# Revision 1.18  1996/11/21  11:45:06  COUSTEAU
# Peter H: Added Conductivity, Temperature and Pressure to
the telemetry list.
#
# Revision 1.17  1996/09/27  16:54:36  PURL
# Peter H: Added the lights and pump enables and feedbacks
#
# Revision 1.16  1996/09/08  17:05:52  COUSTEAU
# Kevin M:added telem for bottom following mode
#
# Revision 1.15  1996/09/08  15:29:17  PURL
# Kevin M:changed DepthThrustInterlocked to
VertThrustInterlocked
# to allow for integrated altimeter and depth control
#
# Revision 1.14  1996/08/24  10:41:43  PURL
# Peter H: Removed AUV_FifoMeter
#
# Revision 1.13  1996/08/15  09:13:42  PURL
# added SurfaceTelemCheckStatus and AUV_TelemCheckStatus
#
# Revision 1.12  1996/08/14  17:31:44  PURL
# added AUV_NoTelemetry to a gtelem group
#
# Revision 1.11  1996/08/14  17:03:44  PURL
# made AUV_NoTelemetry uncontrolled
#
# Revision 1.10  1996/08/14  17:00:19  NEMO
# added the no telemetry items
#
# Revision 1.9  1996/08/09  08:13:35  PURL
# Peter H: Updated the variable types (i.e. int, float) of
several of the
# telemetry items.  Added the low battery flag and changed
the name of
# the leak sensor flag to AUV_Leaking.
#
# Revision 1.8  1996/08/08  15:05:37  PURL
# Peter H: Added AUV_AltSignalStrength and AUV_LowBattery
#
# Revision 1.7  1996/08/08  12:31:24  PURL
# Peter H: Changed several of the setpoint and feedbacks to
floating point values.
#
# Revision 1.6  1996/08/07  10:20:45  COUSTEAU
# removed references to AUV_ExitProteus
#
# Revision 1.5  1996/08/06  17:34:47  COUSTEAU
# AH: added EXIT mode
#
# Revision 1.4  1996/08/02  14:49:31  PURL
# changed names of thruster input events
#
# Revision 1.3  1996/08/02  13:16:50  COUSTEAU
# Peter H: Added the feedbacks for the thrusters
# to file
#
# Revision 1.2  1980/01/04  00:48:52  dosuser
#
*/
// This file lists xtelem groups and is common between purl
and surface.

///////////////////////////////////////////////////////////////
/
// definition of telemetry signals

// Surface-to-AUV
%uncontrolled.float     name=TelemHeadingSetpoint      initial
= 0
%uncontrolled.int       name=TelemVelocitySetpoint     initial
= 0
%uncontrolled.float     name=TelemDepthSetpoint        initial
= 0
%uncontrolled.float     name=TelemAltitudeSetpoint     initial
= 0
%uncontrolled.int       name=TelemBottomFollowingEnabled
initial = FALSE
%uncontrolled.int       name=TelemEnableLogging        initial
= FALSE
%uncontrolled.int       name=TelemModeSelect           initial
= IDLE

%int                    name=SurfaceTelemCounter
%uncontrolled.int       name=SurfaceTelemCheckStatus   initial
= FALSE

%uncontrolled.int       name=TelemEnable_CTD_Pump      initial
= FALSE
%uncontrolled.int       name=TelemEnableLights         initial
= FALSE

// AUV-to-Surface
%uncontrolled.float     name = AUV_CompassHeading      initial
= 0.0
%uncontrolled.float     name = AUV_Depth               initial
= 0.0
%uncontrolled.float     name = AUV_Altitude            initial
= 0.0
%uncontrolled.float     name = AUV_AltSignalStrength   initial
= 0.0
%uncontrolled.int       name = AUV_FollowingBottom     initial
= FALSE
%uncontrolled.int       name = MissionStep             initial
= 0
%uncontrolled.int       name = LocalStep               initial
= 0
%uncontrolled.int       name = AUV_Mode                initial
= 0
%uncontrolled.int       name = AUV_IsLogging           initial
= FALSE
%uncontrolled.float     name = AUV_Pitch               initial
= 0.0
```

```
%uncontrolled.float     name = AUV_Roll                 initial
= 0.0

%uncontrolled.int       name = AUV_LeftThrustInterlocked
initial = 0
%uncontrolled.int       name = AUV_RightThrustInterlocked
initial = 0
%uncontrolled.int       name = AUV_VertThrustInterlocked
initial = 0
%uncontrolled.int       name = AUV_Left_RPM_Fb          initial
= 0
%uncontrolled.int       name = AUV_Right_RPM_Fb         initial
= 0
%uncontrolled.int       name = AUV_VertLeft_RPM_Fb      initial
= 0
%uncontrolled.int       name = AUV_VertRight_RPM_Fb     initial
= 0
%uncontrolled.int       name = AUV_Left_PWM_Fb          initial
= 0
%uncontrolled.int       name = AUV_Right_PWM_Fb         initial
= 0
%uncontrolled.int       name = AUV_VertLeft_PWM_Fb      initial
= 0
%uncontrolled.int       name = AUV_VertRight_PWM_Fb     initial
= 0

%int                    name=LocalTxStatus
%int                    name=LocalRxStatus
%int                    name=LocalTimeoutsOccurred

%long                   name=AUV_IdleMeter
%int                    name=AUV_TxStatus
%int                    name=AUV_RxStatus
%int                    name=AUV_TimeoutsOccurred
%int                    name=AUV_Freemem
%uncontrolled.float     name=AUV_BatteryVoltage         initial
= 30.0  // A High Battery Voltage
%uncontrolled.int       name=AUV_LowBattery             initial
= FALSE
%uncontrolled.int       name=AUV_Leaking                initial
= FALSE
%uncontrolled.int       name=AUV_NoTelemetry            initial
= FALSE
%int                    name=AUV_TelemCounter
%uncontrolled.int       name=AUV_TelemCheckStatus       initial
= FALSE

// Payload
%uncontrolled.int       name=AUV_CTD_PumpOn             initial
= FALSE
%uncontrolled.float     name=AUV_Temperature
initial=0.0
%uncontrolled.float     name=AUV_Conductivity
initial=0.0
%uncontrolled.float     name=AUV_Pressure
initial=0.0
%uncontrolled.int       name=AUV_StandardConductivity
initial=0
%uncontrolled.int       name=AUV_LightsOn               initial
= FALSE


// Debug variables
%uncontrolled.float     name = AUV_DebugFloat1          initial
= 0.0
%uncontrolled.float     name = AUV_DebugFloat2          initial
= 0.0
%uncontrolled.float     name = AUV_DebugFloat3          initial
= 0.0
%uncontrolled.float     name = AUV_DebugFloat4          initial
= 0.0
%uncontrolled.float     name = AUV_DebugFloat5          initial
= 0.0

%uncontrolled.float     name = SurfaceDebugFloat1       initial
= 0.0
%uncontrolled.float     name = SurfaceDebugFloat2       initial
= 0.0
%uncontrolled.float     name = SurfaceDebugFloat3       initial
= 0.0
%uncontrolled.float     name = SurfaceDebugFloat4       initial
= 0.0
%uncontrolled.float     name = SurfaceDebugFloat5       initial
= 0.0

%uncontrolled.int       name = AUV_DebugInt1            initial
= 0
%uncontrolled.int       name = AUV_DebugInt2            initial
= 0
%uncontrolled.int       name = AUV_DebugInt3            initial
= 0
%uncontrolled.int       name = AUV_DebugInt4            initial
= 0
%uncontrolled.int       name = AUV_DebugInt5            initial
= 0

%uncontrolled.int       name = SurfaceDebugInt1         initial
= 0
%uncontrolled.int       name = SurfaceDebugInt2         initial
= 0
%uncontrolled.int       name = SurfaceDebugInt3         initial
= 0
%uncontrolled.int       name = SurfaceDebugInt4         initial
= 0
%uncontrolled.int       name = SurfaceDebugInt5         initial
= 0


/////////////////////////////////////////////////////////////
/
// telemetry setup
%gtelem
                        name             = gt
                        max_frame_length= 128
                        delimiter        = 3
                        timer            = SYS_TwoSecondTrigger
                        escape           = 27
                        trigger          = SYS_TenHzTrigger
                        level            = HIGH
                        port             = TelemPort
                        tx_status_output= LocalTxStatus
                        rx_status_output= LocalRxStatus
                        timeouts_occurred_output =
LocalTimeoutsOccurred
/////////////////////////////////////////////////////////////
/
// source = surface


// position setpoints

%gtelem_float_data
                        telem=gt group=1
                        event=TelemHeadingSetpoint
                        trigger=TelemHeadingSetpoint
                        source=SURFACE
%gtelem_float_data
                        telem=gt group=101
                        event=TelemHeadingSetpoint
                        trigger=SYS_TwoSecondTrigger
                        source=SURFACE

%gtelem_int_data
                        telem=gt group=2
                        event=TelemVelocitySetpoint
                        trigger=TelemVelocitySetpoint
                        source=SURFACE
%gtelem_int_data
                        telem=gt group=101
                        event=TelemVelocitySetpoint
                        source=SURFACE


%gtelem_float_data
                        telem=gt group=3
                        event=TelemDepthSetpoint
                        trigger=TelemDepthSetpoint
                        source=SURFACE
%gtelem_float_data
                        telem=gt group=101
                        event=TelemDepthSetpoint
                        source=SURFACE
%gtelem_float_data
                        telem=gt group=6
                        event=TelemAltitudeSetpoint
                        trigger=TelemAltitudeSetpoint
                        source=SURFACE
%gtelem_float_data
                        telem=gt group=101
                        event=TelemAltitudeSetpoint
                        source=SURFACE

%gtelem_int_data
                        telem=gt group=8
                        event=TelemBottomFollowingEnabled
                        trigger=TelemBottomFollowingEnabled
                        source=SURFACE

// Mode Commands

%gtelem_int_data
                        telem=gt group=4
                        event=TelemModeSelect
```

```
                    trigger=TelemModeSelect
                    source=SURFACE
%gtelem_int_data
                    telem=gt group=5
                    event=TelemEnableLogging
                    trigger=TelemEnableLogging
                    source=SURFACE
%gtelem_int_data
                    telem=gt group=7
                    event=SurfaceTelemCheckStatus
                    trigger=SurfaceTelemCheckStatus
                    source=SURFACE
%gtelem_int_data
                    telem=gt group=9
                    event=TelemEnable_CTD_Pump
                    trigger=TelemEnable_CTD_Pump
                    source=SURFACE
%gtelem_int_data
                    telem=gt group=10
                    event=TelemEnableLights
                    trigger=TelemEnableLights
                    source=SURFACE
// Debug Variables
%gtelem_float_data
                                        telem=gt group=105
          event=SurfaceDebugFloat1
                                        source=SURFACE
          trigger=SYS_TwoSecondTrigger
%gtelem_float_data
                                        telem=gt group=105
          event=SurfaceDebugFloat2
                                        source=SURFACE
%gtelem_float_data
                                        telem=gt group=105
          event=SurfaceDebugFloat3
                                        source=SURFACE
%gtelem_float_data
                                        telem=gt group=105
          event=SurfaceDebugFloat4
                                        source=SURFACE
%gtelem_float_data
                                        telem=gt group=105
          event=SurfaceDebugFloat5
                                        source=SURFACE
%gtelem_int_data
                                        telem=gt group=105
          event=SurfaceDebugInt1
                                        source=SURFACE
%gtelem_int_data
                                        telem=gt group=105
          event=SurfaceDebugInt2
                                        source=SURFACE
%gtelem_int_data
                                        telem=gt group=105
          event=SurfaceDebugInt3
                                        source=SURFACE
%gtelem_int_data
                                        telem=gt group=105
          event=SurfaceDebugInt4
                                        source=SURFACE
%gtelem_int_data
                                        telem=gt group=105
          event=SurfaceDebugInt5
                                        source=SURFACE
%gtelem_int_data
                                        telem=gt group=105
          event=SurfaceTelemCounter
                                        source=SURFACE
//////////////////////////////////////////////////////////////
/
// source = AUV

// Group 111: Position sensors

%gtelem_float_data
                    telem=gt group=20
                    event=AUV_CompassHeading
                    source=AUV
                    trigger=AUV_CompassHeading
%gtelem_float_data
                    telem=gt group=111
                    event=AUV_CompassHeading
                    source=AUV
                    trigger=SYS_TwoSecondTrigger
%gtelem_float_data
                    telem=gt group=21
                    event=AUV_Depth
                    source=AUV
                    trigger=AUV_Depth
%gtelem_float_data
                    telem=gt group=111
                    event=AUV_Depth
                    source=AUV
%gtelem_float_data
                    telem=gt group=27
                    event=AUV_Altitude
                    source=AUV
                    trigger=AUV_Altitude
%gtelem_float_data
                    telem=gt group=111
                    event=AUV_Altitude
                    source=AUV
%gtelem_float_data
                    telem=gt group=28
                    event=AUV_AltSignalStrength
                    source=AUV
                    trigger=AUV_AltSignalStrength
%gtelem_float_data
                    telem=gt group=111
                    event=AUV_AltSignalStrength
                    source=AUV
%gtelem_float_data
                    telem=gt group=31
                    event=AUV_Pitch
                    source=AUV
                    trigger=AUV_Pitch
%gtelem_float_data
                    telem=gt group=111
                    event=AUV_Pitch
                    source=AUV
%gtelem_float_data
                    telem=gt group=32
                    event=AUV_Roll
                    source=AUV
                    trigger=AUV_Roll
%gtelem_float_data
                    telem=gt group=111
                    event=AUV_Roll
                    source=AUV
// mission
%gtelem_int_data
                    telem=gt group=23
                    event=LocalStep
                    source=AUV
                    trigger=LocalStep
%gtelem_int_data
                    telem=gt group=112
                    event=LocalStep
                    source=AUV
                    trigger=SYS_TwoSecondTrigger
%gtelem_int_data
                    telem=gt group=24
                    event=MissionStep
                    source=AUV
                    trigger=MissionStep
%gtelem_int_data
                    telem=gt group=112
                    event=MissionStep
                    source=AUV
// Group 110 -- AUV status and mode feedback
// AUV Status Parameters
%gtelem_int_data
                    telem=gt group=110
                    event=AUV_IdleMeter
                    source=AUV
                    trigger=SYS_TwoSecondTrigger
```

```
%gtelem_int_data
        telem=gt group=110
        event=AUV_TxStatus
        source=AUV

%gtelem_int_data
        telem=gt group=110
        event=AUV_RxStatus
        source=AUV

%gtelem_int_data
        telem=gt group=110
        event=AUV_TimeoutsOccurred
        source=AUV

%gtelem_int_data
        telem=gt group=110
        event=AUV_Freemem
        source=AUV

%gtelem_float_data
        telem=gt group=110
        event=AUV_BatteryVoltage
        source=AUV

%gtelem_int_data
        telem=gt group=29
        event=AUV_LowBattery
        source=AUV
        trigger=AUV_LowBattery
%gtelem_int_data
        telem=gt group=110
        event=AUV_LowBattery
        source=AUV

%gtelem_int_data
        telem=gt group=30
        event=AUV_Leaking
        source=AUV
        trigger=AUV_Leaking
%gtelem_int_data
        telem=gt group=110
        event=AUV_Leaking
        source=AUV

%gtelem_int_data
        telem=gt group=110
        event=AUV_NoTelemetry
        source=AUV

%gtelem_int_data
        telem=gt group=35
        event=AUV_IsLogging
        source=AUV
        trigger=AUV_IsLogging
%gtelem_int_data
        telem=gt group=110
        event=AUV_IsLogging
        source=AUV

%gtelem_int_data
        telem=gt group=36
        event=AUV_Mode
        source=AUV
        trigger=AUV_Mode
%gtelem_int_data
        telem=gt group=110
        event=AUV_Mode
        source=AUV

%gtelem_int_data
        telem=gt group=22
        event=AUV_FollowingBottom
        source=AUV
        trigger=AUV_FollowingBottom
%gtelem_int_data
        telem=gt group=110
        event=AUV_FollowingBottom
        source=AUV

%gtelem_int_data
        telem=gt group=37
        event=AUV_TelemCheckStatus
        source=AUV
        trigger=AUV_TelemCheckStatus
%gtelem_int_data
        telem=gt group=110
        event=AUV_TelemCheckStatus
        source=AUV

%gtelem_int_data
        telem=gt group=110
        event=AUV_TelemCounter
        source=AUV

// Payload
%gtelem_int_data
        telem=gt group=33
        event=AUV_CTD_PumpOn
        source=AUV
        trigger=AUV_CTD_PumpOn
%gtelem_int_data
        telem=gt group=114
        event=AUV_CTD_PumpOn
        source=AUV
        trigger=SYS_TwoSecondTrigger

%gtelem_float_data
        telem=gt group=38
        event=AUV_Conductivity
        source=AUV
        trigger=AUV_Conductivity
%gtelem_float_data
        telem=gt group=114
        event=AUV_Conductivity
        source=AUV

%gtelem_float_data
        telem=gt group=39
        event=AUV_Temperature
        source=AUV
        trigger=AUV_Temperature
%gtelem_float_data
        telem=gt group=114
        event=AUV_Temperature
        source=AUV

%gtelem_float_data
        telem=gt group=40
        event=AUV_Pressure
        source=AUV
        trigger=AUV_Pressure
%gtelem_float_data
        telem=gt group=114
        event=AUV_Pressure
        source=AUV

%gtelem_int_data
        telem=gt group=41
        event=AUV_StandardConductivity
        source=AUV
        trigger=AUV_StandardConductivity
%gtelem_int_data
        telem=gt group=114
        event=AUV_StandardConductivity
        source=AUV

%gtelem_int_data
        telem=gt group=34
        event=AUV_LightsOn
        source=AUV
        trigger=AUV_LightsOn
%gtelem_int_data
        telem=gt group=114
        event=AUV_LightsOn
        source=AUV

// Thruster Feedback
%gtelem_int_data
        telem=gt group=113
        event=AUV_LeftThrustInterlocked
        source=AUV
        trigger=SYS_TwoSecondTrigger

%gtelem_int_data
        telem=gt group=113
        event=AUV_RightThrustInterlocked
        source=AUV

%gtelem_int_data
        telem=gt group=113
        event=AUV_VertThrustInterlocked
        source=AUV

%gtelem_int_data
        telem=gt group=113
        event=AUV_Left_RPM_Fb
        source=AUV

%gtelem_int_data
        telem=gt group=113
```

```
                        event=AUV_Right_RPM_Fb
                        source=AUV

%gtelem_int_data
                        telem=gt group=113
                        event=AUV_VertLeft_RPM_Fb
                        source=AUV

%gtelem_int_data
                        telem=gt group=113
                        event=AUV_VertRight_RPM_Fb
                        source=AUV

%gtelem_int_data
                        telem=gt group=113
                        event=AUV_Left_PWM_Fb
                        source=AUV

%gtelem_int_data
                        telem=gt group=113
                        event=AUV_Right_PWM_Fb
                        source=AUV

%gtelem_int_data
                        telem=gt group=113
                        event=AUV_VertLeft_PWM_Fb
                        source=AUV

%gtelem_int_data
                        telem=gt group=113
                        event=AUV_VertRight_PWM_Fb
                        source=AUV

// Debug Variables

%gtelem_float_data
                        telem=gt group=115
                        event=AUV_DebugFloat1
                        source=AUV
                        trigger=SYS_TwoSecondTrigger
%gtelem_float_data
                        telem=gt group=115
                        event=AUV_DebugFloat2
                        source=AUV
%gtelem_float_data
                        telem=gt group=115
                        event=AUV_DebugFloat3
                        source=AUV
%gtelem_float_data
                        telem=gt group=115
                        event=AUV_DebugFloat4
                        source=AUV
%gtelem_float_data
                        telem=gt group=115
                        event=AUV_DebugFloat5
                        source=AUV

%gtelem_int_data
                        telem=gt group=115
                        event=AUV_DebugInt1
                        source=AUV
%gtelem_int_data
                        telem=gt group=115
                        event=AUV_DebugInt2
                        source=AUV
%gtelem_int_data
                        telem=gt group=115
                        event=AUV_DebugInt3
                        source=AUV
%gtelem_int_data
                        telem=gt group=115
                        event=AUV_DebugInt4
                        source=AUV
%gtelem_int_data
                        telem=gt group=115
                        event=AUV_DebugInt5
                        source=AUV
```

display.csp

```
//  VGA display
%gfx.display
        name      =screen
        enable    =TRUE
        level     =LOW
        refresh   =SYS_TenHzTrigger
//      key_output=KeyPressed

//
// GRAPHICS OBJECTS' X-Y CO-ORDINATES
//
%define name=DEEP_DEPTH                    value=2
%define name=NO_DEPTH                      value=0
%define name=MAN_BUTTON_WIDTH              value=20
%define name=MAN_BUTTON_HEIGHT             value=15
%define name=MAN_MED_BUTTON_WIDTH          value=29
%define name=MAN_MED_BUTTON_HEIGHT         value=20
%define name=MAN_BIG_BUTTON_WIDTH          value=62
%define name=MAN_BIG_BUTTON_HEIGHT         value=25
%define name=MAN_TEXT_WIDTH                value=50

%define name=COL_0  value=0
%define name=COL_1  value=30
%define name=COL_2  value=60
%define name=COL_3  value=90
%define name=COL_4  value=120
%define name=COL_5  value=150
%define name=COL_6  value=180
%define name=COL_7  value=210
%define name=COL_8  value=240
%define name=COL_9  value=270
%define name=COL_10 value=300
%define name=COL_11 value=330
%define name=COL_12 value=360
%define name=COL_13 value=390
%define name=COL_14 value=420
%define name=COL_15 value=450
%define name=COL_16 value=480
%define name=COL_17 value=510
%define name=COL_18 value=540
%define name=COL_19 value=570
%define name=COL_20 value=600

%define name=ROW_HEIGHT  value=15
%define name=ROW_0  value=5
%define name=ROW_1  value=20
%define name=ROW_2  value=35
%define name=ROW_3  value=50
%define name=ROW_4  value=65
%define name=ROW_5  value=80
%define name=ROW_6  value=95
%define name=ROW_7  value=110
%define name=ROW_8  value=125
%define name=ROW_9  value=140
%define name=ROW_10 value=155
%define name=ROW_11 value=170
%define name=ROW_12 value=185
%define name=ROW_13 value=200
%define name=ROW_14 value=215
%define name=ROW_15 value=230
%define name=ROW_16 value=245
%define name=ROW_17 value=260
%define name=ROW_18 value=275
%define name=ROW_19 value=290
%define name=ROW_20 value=305
%define name=ROW_21 value=320
%define name=ROW_22 value=335
%define name=ROW_23 value=350
%define name=ROW_24 value=365
%define name=ROW_25 value=380
%define name=ROW_26 value=395
%define name=ROW_27 value=410
%define name=ROW_28 value=425
%define name=ROW_29 value=440
```

xstyle.csp

```
/*
$Log: xstyle.csp $
# Revision 1.1  1996/08/01  12:14:18  dosuser
# Initial revision
#
*/
//    *******************STYLE.CSP**************************

//  color, & style

@screen.color   display=screen

%screen.color   name=black       r=0     g=0     b=0
%screen.color   name=blue        r=0     g=0     b=130
%screen.color   name=green       r=40    g=150   b=40
%screen.color   name=cyan        r=0     g=170   b=170
// original %screen.color   name=red        r=255   g=0
b=0
%screen.color   name=red         r=230   g=10     b=10
!screen.color   name=magenta     r=170   g=0     b=170
%screen.color   name=brown       r=170   g=85    b=0
%screen.color   name=lightgray   r=170   g=170   b=170
%screen.color   name=darkgray    r=100   g=100   b=100
!screen.color   name=tan         r=162   g=134   b=90
%screen.color   name=lightgreen  r=20    g=200   b=20
%screen.color   name=lightcyan   r=85    g=255   b=255
%screen.color   name=lightred    r=255   g=10    b=0
%screen.color   name=lightblue   r=40    g=40    b=200
%screen.color   name=yellow      r=255   g=255   b=0
%screen.color   name=white       r=255   g=255   b=255

%style
                        name=green_button_on_style
                        foreground=black
                        background=green
                        fillpattern=solid
%style
                        name=red_button_on_style
                        foreground=black
                        background=red
                        fillpattern=solid
%style
                        name=yellow_button_on_style
                        foreground=black
                        background=yellow
                        fillpattern=solid
%style
                        name=button_off_style
                        foreground=black
                        background=lightgray
                        fillpattern=solid
%style
                        name=switch_off_button_style
                        foreground=black
                        background=lightgray
                        fillpattern=solid
                        bwforeground=0
                        bwbackground=1
%style
                        name=switch_on_button_style
                        foreground=black
                        background= cyan
                        fillpattern=solid
                        bwforeground=0
                        bwbackground=1

%style
        name=indicator_bad_style
        background=red
        foreground=black
        fillpattern=solid

%style
        name=fb_compass_style
        foreground=black
        background=black
        fillpattern=interleave

%style
        name=fb_cavity_style
        foreground=blue
        background=lightgray
        fillpattern=solid

%style
        name=neg_fb_cavity_style
        foreground=black
        background=black
        fillpattern=interleave

%style
        name=neg_fb_cavity_style_2
        foreground=red
        background=red
        fillpattern=solid

%style
        name=fb_needle_style
        foreground=green
        background=green
        fillpattern=solid

%style
        name=neg_fb_needle_style
        foreground=blue
        background=lightgray
        fillpattern=solid

%style
                        name=plot_style
                        foreground=yellow
                        background=blue
                        fillpattern=solid

%style
                        name=hyde_plot_style
                        foreground=darkgray
                        background=lightcyan
                        fillpattern=solid

%style
                        name=ruler_style
                        foreground=lightcyan
                        background=darkgray
                        fillpattern=solid

%style
                        name=label_style
                        foreground=black
                        background=lightgray
                        fillpattern=solid

%style
                        name=window_style
                        foreground=black
                        background=lightgray
                        fillpattern=solid

!style
                        name=sp_number_style
                        foreground=yellow
                        background=darkgray
                        fillpattern=interleave   /*cga=solid
vga=interleave*/
                        bwforeground=1
                        bwbackground=0

!style
                        name=fb_number_style
                        foreground=lightcyan
                        background=darkgray
                        fillpattern=interleave   /*cga=solid
vga=interleave*/
                        bwforeground=1
                        bwbackground=0

%style
                        name=general_label_style
                        foreground=blue
                        background=white
                        fillpattern=solid
                        bwforeground=0
                        bwbackground=1


%style.setup
                        style=window_style
                        object_type=window
                        object_state=any

%style.setup
                        style=window_style
                        object_type=title
                        object_state=active

%style.setup
                        style=window_style
                        object_type=title
                        object_state=any
```

```
%style.setup
                        style=window_style
                        object_type=menu_item
                        object_state=any

%style.setup
                        style=plot_style
                        object_type=menu_item
                        object_state=current

%style.setup
                        style=plot_style
                        object_type=scroll_bar
                        object_state=any

%style.setup
                        style=plot_style
                        object_type=menu_item
                        object_state=any
//                      object_state=view

// object_type = "window_object"
//             "window"
//             "icon"
//             "scroll_bar"
//             "border"
//             "title"
//             "prompt"
//             "button"
//             "menu_item"
//             "outline"
//             "white_shadow"
//             "light_shadow"
//             "dark_shadow"
//             "black_shadow"
//
// object_state= "any"
//             "current"
//             "view"
//             "hot_key"
//             "active"
//             "selected"
//             "non_selectable"


// fillpattern= "solid"
//             "line"
//             "ltslash"
//             "slash"
//             "bkslash"
//             "ltbkslash"
//             "hatch"
//             "xhatch"
//             "interleave"
//             "wide_dot"
//             "close_dot"
```

sdef.csp

```
/*
$Log: sdef.csp $
# Revision 1.5  1996/08/16  08:38:13  NEMO
# added telemetry checking stuff
#
# Revision 1.4  1996/08/15  08:47:07  NEMO
# ?
#
# Revision 1.3  1996/08/06  17:34:47  COUSTEAU
# AH: added EXIT mode
#
# Revision 1.2  1996/08/01  14:19:31  NEMO
# changed ports.csp to sports.csp
#
# Revision 1.1  1996/08/01  12:07:03  dosuser
# Initial revision
#
*/
// ******************* SURFDEF.CSP
******************************

%include        file="XCONST.CSP"
%include        file="SHEADER.CSP"
%include        file="XTIMER.CSP"
%include        file="SPORTS.CSP"
%include        file="XTELEM.CSP"

%include        file="DISPLAY.CSP"
%include        file="XSTYLE.CSP"
%include        file="MAIN_WIN.CSP"
%include        file="SETPOINT.CSP"
%include        file="STAT_WIN.CSP"
%include        file="DBG_WIN.CSP"
%include        file="AUV_MODE.CSP"

//  diagnostic functions
%exit
        input=Exit
        message="Exiting PROTEUS normally on command from the
user."

%freemem
        trigger=SYS_TwoSecondTrigger
        output=SurfaceFreemem

%idle.meter
        trigger=SYS_TwoSecondTrigger
        output=SurfaceIdleMeter

!fifo.meter
        trigger=SYS_TwoSecondTrigger
        output=SurfaceFifoMeter
        global_full_recover=1

//  Main Menu & Submenu

%root.menu
        display=screen
        name=root
        enable=TRUE

%submenu
        menu=root
        name=system
        title="~System "

%submenu.int.output
        submenu=system
        title="~Exit"
        output=Exit
        value=TRUE

%submenu.int.output
        submenu=system
        title="~Show Status"
        output=SwitchStatusWindow
        value=1

%submenu.int.output
        submenu=system
        title="~Hide Status"
        output=SwitchStatusWindow
        value=0

%submenu.int.output
        submenu=system
        title="~Show Debug"
        output=SwitchDebugWindow
        value=1

%submenu.int.output
        submenu=system
        title="~Hide Debug"
        output=SwitchDebugWindow
        value=0

%submenu
        menu=root
        name=shutdown
        title="  AUV Shutdown  "

%submenu.int.output
        submenu=shutdown
        title="AUV Exit Mode"
        output=TelemModeSelect
        value=EXIT

!zero
        level=MEDIUM
        trigger=ButtonTimer
        input=KeyPressed
        output=KeyPressed

%exception_log
        file="ERROR.TXT"
        level=LOW
        buffer_size=100

%copy
        input=LocalTxStatus
        output=SurfaceTxStatus
        enable=TRUE

%copy
        input=LocalRxStatus
        output=SurfaceRxStatus
        enable=TRUE

%copy
        input=LocalTimeoutsOccurred
        output=SurfaceTimeoutsOccurred
        enable=TRUE

// telemetry timer

%cycle
    trigger     = SYS_TwoSecondTrigger
    level       = LOW
    init        = 0
    min         = 0
    max         = 1000
    step        = 1
    reset       = AUV_TelemCounter
    output      = SurfaceTelemCounter
```

sheader.csp

```
/*
$Log: sheader.csp $
# Revision 1.6  1996/09/10  10:03:58  COUSTEAU
# Peter H: Added initial values to the Status and Debug
windows to eliminate
# error messages that were appearing in the error log file
#
# Revision 1.5  1996/08/15  08:33:11  NEMO
# added telem checking stuff
#
# Revision 1.4  1996/08/08  17:20:25  NEMO
# changed user interface
#
# Revision 1.3  1996/08/06  17:34:47  COUSTEAU
# AH: added EXIT mode
#
# Revision 1.2  1996/08/02  14:53:37  COUSTEAU
# Peter H: Removed all the DEBUG parameters
#
# Revision 1.1  1996/08/01  12:10:49  dosuser
# Initial revision
#
*/

//    **********************HEADER.CSP*******************

//  Const & Define
%define                  name=SURFACE            value=1
%define                  name=AUV                value=0

//  sync, int, long
%sync                    name=OneSecond
%sync                    name=Exit

%sync                    name=DepthSetpointTrigger
%sync                    name=HeadingSetpointTrigger
%sync                    name=VelocitySetpointTrigger
%sync                    name=TelemModeSelectTrigger
%sync                    name=EnableLoggingTrigger

%int                     name=SurfaceFreemem
%long                    name=SurfaceIdleMeter
%int                     name=SurfaceFifoMeter
%int                     name=TelDiag
%int                     name=SurfaceTxStatus
%int                     name=SurfaceRxStatus
%int                     name=SurfaceTimeoutsOccurred
%uncontrolled.int        name=SwitchStatusWindow
initial=FALSE
%uncontrolled.int        name=SwitchDebugWindow
initial=FALSE
%uncontrolled.int        name=Dummy
%uncontrolled.int        name=KeyPressed
initial=0

%uncontrolled.int        name = HeadingSpJog           initial
= 0
%uncontrolled.int        name = VelocitySpJog          initial
= 0
%uncontrolled.int        name = DepthSpJog             initial
= 0
%uncontrolled.int        name = AltitudeSpJog          initial
= 0

%uncontrolled.int        name = VelocitySp             initial
= 0
%uncontrolled.int        name = ZeroVelocitySp         initial
= 0
%uncontrolled.int        name = LargeVelocitySp
initial = 0
%uncontrolled.int        name = DepthSp                initial
= 0
%uncontrolled.int        name = ZeroDepthSp            initial
= 0
%uncontrolled.int        name = AltitudeSp             initial
= 0
%uncontrolled.int        name = ResetAltitudeSp        initial
= 10

%uncontrolled.int        name = EnableHeadingJog       initial
= 0
%uncontrolled.int        name = EnableVelocityJog      initial
= 0
%uncontrolled.int        name = EnableDepthJog         initial
= 0
%uncontrolled.int        name = EnableAltitudeJog      initial
= 0

// AUV Mode Switching Parameters
%int                     name = IdleModeSelected
%int                     name = PilotModeSelected
%int                     name = AbortModeSelected
%int                     name = Mission1ModeSelected
%int                     name = Mission2ModeSelected
%int                     name = Mission3ModeSelected
%int                     name = Mission4ModeSelected
%int                     name = Mission5ModeSelected
%int                     name = Mission6ModeSelected

%int                     name = IdleModeRequest
%int                     name = PilotModeRequest
%int                     name = AbortModeRequest
%int                     name = Mission1ModeRequest
%int                     name = Mission2ModeRequest
%int                     name = Mission3ModeRequest
%int                     name = Mission4ModeRequest
%int                     name = Mission5ModeRequest
%int                     name = Mission6ModeRequest

// The mode the AUV thinks it is in (Yes PURL does think, but
not too well)
%int                     name = InIdleMode
%int                     name = InPilotMode
%int                     name = InAbortMode
%int                     name = InExitMode
%uncontrolled.int        name = InMissionMode
```

sports.csp

```
/////////////////////////////////////////////////////////////
// Telemetry port

!udp.port
        level = HIGH
        name = TelemPort
        local_socket = 4100
        remote_socket = 4000
        remote_host = "purl"
        signal_size = 4096
        max_packet_size = 256

!com.serial.port
        name            = TelemPort
        baud_rate       = 38400
        parity          = 0
        delimiter       = 3
        port_number     = 1
        buffer_size=256 // Needed for 32-bit PROTEUS.

!serial.port.diag
        output          = TelDiag
        port            = TelemPort
        trigger         = DiagTimer
        overrun_errors  = TRUE
        accumulate      = TRUE
```

main_win.csp

```
/*
$Log: main_win.csp $
# Revision 1.9  1996/11/22  14:52:23  COUSTEAU
# Peter H: Changed AUV_Standard_Conductivity to
AUV_StandardConductivity
#
# Revision 1.8  1996/11/22  12:40:06  COUSTEAU
# Peter H: Added CTD display
#
# Revision 1.7  1996/10/01  20:31:12  COUSTEAU
# Peter H: Added the buttons for the Camera Lights and CTD
Pump
#
# Revision 1.6  1996/09/18  16:27:46  COUSTEAU
# Peter H: Added the altimeter and changed the variables
names accordingly
#
# Revision 1.5  1996/08/15  08:33:11  NEMO
# added telem checking stuff
#
# Revision 1.4  1996/08/08  17:20:25  NEMO
# changed user interface
#
# Revision 1.3  1996/08/03  11:34:47  COUSTEAU
# Peter H: Not sure what I changed
#
# Revision 1.2  1996/08/02  14:52:53  COUSTEAU
# Peter H: Added the display components for the thruster
feedbacks
#
# Revision 1.1  1996/08/01  12:09:50  dosuser
# Initial revision
#
*/
//************************ MAIN_WIN.CSP
********************************
// MAIN_WIN.CSP is the main window wherein all the buttons,
and feebacks for
// controlling the AUV are located.

%graphics.window
                enable=TRUE
                name=AUV_Control
                title="AUV_Control"
                display=screen
                top=1 left=0
                width=-1 height=-1
                //width=80 height=32
                border=TRUE

@AUV_Control.window.label

style=label_style fill=0

@AUV_Control.window.number

style=label_style
                                                fill=TRUE

// **************** Heading Dial **********************
// Dial showing the feedback for the compass heading

%clipping.linear.param
                name            = MAN_HeadingDialParam
                input1          = 0
                input2          = +360
                output1         = +90
                output2         = -270

%AUV_Control.window.dial
                input           = AUV_CompassHeading
                x               = COL_6
                y               = ROW_4
                radius          = 40
                start_angle     = 0
                sweep_angle     = 360
                mapping         = MAN_HeadingDialParam
                major           = 5
                tic_size        = 25
                minor           = 3
                fill            = TRUE
                needle_style    = fb_compass_style
                cavity_style    = fb_cavity_style

%window.label
                string          = "N"
                x               = 180
                y               = 19
                window          = AUV_Control
                style           = label_style
                center          = TRUE
                fill            = TRUE

%window.label
                string          = "E"
                x               = 228
                y               = 65
                window          = AUV_Control
                style           = label_style
                center          = TRUE
                fill            = TRUE

%window.label
                string          = "S"
                x               = 180
                y               = 113
                window          = AUV_Control
                style           = label_style
                center          = TRUE
                fill            = TRUE

%window.label
                string          = "W"
                x               = 133
                y               = 65
                window          = AUV_Control
                style           = label_style
                center          = TRUE
                fill            = TRUE


// **************** Abort Conditions
**********************
// Indicates abort conditions

%window.button
                left            =COL_18
                top             =ROW_10
                on_value        =TRUE
                input           =AUV_LowBattery
                label           ="Low Bat"
                window          =AUV_Control
                width           =MAN_BIG_BUTTON_WIDTH
                height          =MAN_BIG_BUTTON_HEIGHT
                depth           =NO_DEPTH
                border          =TRUE
                off_style       =button_off_style
                on_style        =red_button_on_style

%window.button
                left            =COL_18
                top             =ROW_12
                on_value        =TRUE
                input           =AUV_Leaking
                label           ="Leak"
                window          =AUV_Control
                width           =MAN_BIG_BUTTON_WIDTH
                height          =MAN_BIG_BUTTON_HEIGHT
                depth           =NO_DEPTH
                border          =TRUE
                off_style       =button_off_style
                on_style        =red_button_on_style
%window.button
                left            =COL_18
                top             =ROW_14
                on_value        =TRUE
                input           =AUV_NoTelemetry
                label           ="NoTelem"
                window          =AUV_Control
                width           =MAN_BIG_BUTTON_WIDTH
                height          =MAN_BIG_BUTTON_HEIGHT
                depth           =NO_DEPTH
                border          =TRUE
                off_style       =button_off_style
                on_style        =red_button_on_style

%AUV_Control.window.number
        x=COL_9 y=ROW_11
        input=AUV_TelemCounter
        string="    AUV:"
        width=4

%AUV_Control.window.number
        x=COL_9 y=ROW_12
        input=SurfaceTelemCounter
        string="Surface:"
        width=4

%window.toggle.button
```

```
                input           = SurfaceTelemCheckStatus
                output          = SurfaceTelemCheckStatus
                on_value        = TRUE
                label           = "ChkTelem"
                window          = AUV_Control
                width           = MAN_BIG_BUTTON_WIDTH
                height          = MAN_BIG_BUTTON_HEIGHT
                depth           = DEEP_DEPTH
                border          = FALSE
                off_style       = switch_off_button_style
                on_style        = yellow_button_on_style
                left            = COL_13
                top             = ROW_11
%window.button
                left            =COL_13
                top             =ROW_13
                on_value        =TRUE
                input           =AUV_TelemCheckStatus
                label           ="ChkTelem"
                window          =AUV_Control
                width           =MAN_BIG_BUTTON_WIDTH
                height          =MAN_BIG_BUTTON_HEIGHT
                depth           =NO_DEPTH
                border          =TRUE
                off_style       =button_off_style
                on_style        =yellow_button_on_style

// ********************* Vehicle Depth Feedback Display
*******************
// Display the depth of the vehicle graphically

%clipping.linear.param          // mapping neg propellor rpm
fb to bar graphs
                name            = Depth_Scale_Param
                input1          = 75
                input2          = 0
                output1         = 100
                output2         = 0

%window.vertical.scale
                input           = AUV_Depth
                mapping         = Depth_Scale_Param
                window          = AUV_Control
                width           = 15
                height          = 100
                border_width    = 1
                fill            = TRUE
                mercury_style   = neg_fb_needle_style
                cavity_style    = neg_fb_cavity_style
                surface_style   = neg_fb_needle_style
                left            = 165
                top             = 140

%window.label
                string          = "0"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 183
                y               = 137
%window.label
                string          = "15"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 183
                y               = 157
%window.label
                string          = "30"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 183
                y               = 177
%window.label
                string          = "45"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 183
                y               = 197
%window.label
                string          = "60"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 183
                y               = 217
%window.label
                string          = "75"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 183
                y               = 237

%window.label
                string          = "D"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_7
                y               = 160

%window.label
                string          = "e"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_7
                y               = 170

%window.label
                string          = "p"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_7
                y               = 180

%window.label
                string          = "t"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_7
                y               = 190

%window.label
                string          = "h"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_7
                y               = 200

%window.label
                string          = "(m)"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 203
                y               = 210


// ***************** Vehicle Altitude
************************
// Display the height of PURL off of the bottom.  The range
of the
// is 0 to 200 metres.

%clipping.linear.param          //mapping altimeter output to
bar graph
                name            = Coarse_Alt_Scale_Param
                input1          = 0
                input2          = 200
                output1         = 100
                output2         = 0

%clipping.linear.param          //mapping altimeter output to
bar graph
                name            = Fine_Alt_Scale_Param
                input1          = 0
                input2          = 10
                output1         = 100
                output2         = 0

%window.vertical.scale
                input           = AUV_Altitude
                mapping         = Coarse_Alt_Scale_Param
```

```
                window          = AUV_Control
                width           = 15
                height          = 100
                border_width    = 1
                fill            = TRUE
                mercury_style   = fb_needle_style
                cavity_style    = fb_cavity_style
                surface_style   = fb_needle_style
                left            = 150
                top             = 260

%window.vertical.scale
                input           = AUV_Altitude
                mapping         = Fine_Alt_Scale_Param
                window          = AUV_Control
                width           = 15
                height          = 100
                border_width    = 1
                fill            = TRUE
                mercury_style   = fb_needle_style
                cavity_style    = fb_cavity_style
                surface_style   = fb_needle_style
                left            = 195
                top             = 260

%window.label
                string          = "200"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 170
                y               = 262

%window.label
                string          = "100"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 170
                y               = 310

%window.label
                string          = "0"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 167
                y               = 355

%window.label
                string          = "10"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 215
                y               = 262

%window.label
                string          = "5"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 215
                y               = 310

%window.label
                string          = "0"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 215
                y               = 355

// ****************** Pitch
*******************************************
%clipping.linear.param          // mapping pos propellor rpm
fb to bar graphs
                name            = Positive_Pitch_Scale_Param
                input1          = 0
                input2          = 20
                output1         = 50
                output2         = 0

%clipping.linear.param          // mapping neg propellor rpm
fb to bar graphs
                name            = Negative_Pitch_Scale_Param
                input1          = -20
                input2          = 0
                output1         = 50
                output2         = 0

%window.vertical.scale
                input           = AUV_Pitch
                mapping         = Positive_Pitch_Scale_Param
                window          = AUV_Control
                width           = 15
                height          = 50
                border_width    = 1
                fill            = TRUE
                mercury_style   = fb_needle_style
                cavity_style    = fb_cavity_style
                surface_style   = fb_needle_style
                left            = 315
                top             = 240

%window.vertical.scale
                input           = AUV_Pitch
                mapping         = Negative_Pitch_Scale_Param
                window          = AUV_Control
                width           = 15
                height          = 50
                border_width    = 1
                fill            = TRUE
                mercury_style   = neg_fb_needle_style
                cavity_style    = neg_fb_cavity_style
                surface_style   = neg_fb_needle_style
                left            = 315
                top             = 290

%window.label
                string          = "+20"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 333
                y               = 237

%window.label
                string          = "Pitch"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 333
                y               = 255

%AUV_Control.window.number
                x               = 333
                y               = 270
                input           = AUV_Pitch
                width           = 5
                precision       = 1

%window.label
                string          = "0"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 333
                y               = 288

%window.label
                string          = "-20"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 333
                y               = 335

// ************************ Roll
*******************************************
%clipping.linear.param          // mapping positive roll to
bar graphs
                name            = Positive_Roll_Scale_Param
                input1          = 0
                input2          = 20
                output1         = 0
                output2         = 50
```

```
%clipping.linear.param          // mapping negative roll to
bar graphs
                name            = Negative_Roll_Scale_Param
                input1          = -20
                input2          = 0
                output1         = 0
                output2         = 50

%window.horizontal.scale
                input           = AUV_Roll
                mapping         = Positive_Roll_Scale_Param
                window          = AUV_Control
                width           = 50
                height          = 15
                border_width    = 1
                fill            = TRUE
                mercury_style   = neg_fb_cavity_style_2
                cavity_style    = neg_fb_needle_style
                surface_style   = neg_fb_cavity_style_2
                left            = 323
                top             = 345

%window.horizontal.scale
                input           = AUV_Roll
                mapping         = Negative_Roll_Scale_Param
                window          = AUV_Control
                width           = 50
                height          = 15
                border_width    = 1
                fill            = TRUE
                mercury_style   = fb_cavity_style
                cavity_style    = fb_needle_style
                surface_style   = fb_needle_style
                left            = 273
                top             = 345

%window.label
                string          = "+20"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 363
                y               = 365

%AUV_Control.window.number
                x               = 290
                y               = 375
                string          = "Roll:"
                input           = AUV_Roll
                width           = 5
                precision       = 1

%window.label
                string          = "0"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 320
                y               = 365

%window.label
                string          = "-20"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = 260
                y               = 365

// ****************** Battery Monitor
******************************
// Display the current voltage of the battery inside PURL.
The range
// for the battery voltage is +28 Volts to 15 Volts.

%clipping.linear.param
                name            = Battery_Scale_Param
                input1          = 15
                input2          = 28
                output1         = 80
                output2         = 0

%window.vertical.scale
                input           = AUV_BatteryVoltage
                mapping         = Battery_Scale_Param
                window          = AUV_Control
                width           = 15
                height          = 80
                border_width    = 1
                fill            = TRUE
                mercury_style   = fb_needle_style
                cavity_style    = fb_cavity_style
                surface_style   = fb_needle_style
                left            = 490
                top             = 145

%window.label
                string          = "28V"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_17
                y               = 140

%window.label
                string          = "24V"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_17
                y               = 165

%window.label
                string          = "15V"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_17
                y               = 220

%AUV_Control.window.number
                x               = COL_16
                y               = ROW_15
                input           = AUV_BatteryVoltage
                width           = 3
                precision       = 1

%window.label
                string          = "B"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_16
                y               = ROW_9

%window.label
                string          = "a"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_16
                y               = 150
%window.label
                string          = "t"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_16
                y               = 160
%window.label
                string          = "t"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_16
                y               = 170
%window.label
                string          = "e"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_16
                y               = 180
%window.label
                string          = "r"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_16
```

```
                y               = 190
%window.label
                string          = "y"
                window          = AUV_Control
                style           = label_style
                center          = FALSE
                fill            = TRUE
                x               = COL_16
                y               = 200

// ***************** Lights Control ***********************
// The lights toggle button turns the camera lights off and
on

%window.toggle.button
                input           = TelemEnableLights
                output          = TelemEnableLights
                on_value        = TRUE
                label           = "Lights"
                window          = AUV_Control
                width           = MAN_BIG_BUTTON_WIDTH
                height          = MAN_BIG_BUTTON_HEIGHT
                depth           = DEEP_DEPTH
                border          = FALSE
                off_style       = switch_off_button_style
                on_style        = green_button_on_style
                left            = COL_16
                top             = ROW_17

// CTD Pump Feedback button

%window.button
                left            =COL_18
                top             =ROW_17
                on_value        =TRUE
                input           =AUV_LightsOn
                label           ="Lights"
                window          =AUV_Control
                width           =MAN_BIG_BUTTON_WIDTH
                height          =MAN_BIG_BUTTON_HEIGHT
                depth           =NO_DEPTH
                border          =TRUE
                off_style       =button_off_style
                on_style        =green_button_on_style



// ***************** CTD Pump Control
***********************
// The CTD pump toggle button turns the CTD pump off and on

%window.toggle.button
                input           = TelemEnable_CTD_Pump
                output          = TelemEnable_CTD_Pump
                on_value        = TRUE
                label           = "CTDPump"
                window          = AUV_Control
                width           = MAN_BIG_BUTTON_WIDTH
                height          = MAN_BIG_BUTTON_HEIGHT
                depth           = DEEP_DEPTH
                border          = FALSE
                off_style       = switch_off_button_style
                on_style        = green_button_on_style
                left            = COL_16
                top             = ROW_19

// CTD Pump Feedback button

%window.button
                left            =COL_18
                top             =ROW_19
                on_value        =TRUE
                input           =AUV_CTD_PumpOn
                label           ="Pump On"
                window          =AUV_Control
                width           =MAN_BIG_BUTTON_WIDTH
                height          =MAN_BIG_BUTTON_HEIGHT
                depth           =NO_DEPTH
                border          =TRUE
                off_style       =button_off_style
                on_style        =green_button_on_style



// ***************** Data Logging ***********************
// The logging toggle button turns the general data logging
off and
// on onboard PURL

%window.toggle.button
                input           = TelemEnableLogging
                output          = TelemEnableLogging
                on_value        = TRUE
                label           = "Logging"
                window          = AUV_Control
                width           = MAN_BIG_BUTTON_WIDTH
                height          = MAN_BIG_BUTTON_HEIGHT
                depth           = DEEP_DEPTH
                border          = FALSE
                off_style       = switch_off_button_style
                on_style        = green_button_on_style
                left            = COL_16
                top             = ROW_21

// Data Logging Feedback button

%window.button
                left            =COL_18
                top             =ROW_21
                on_value        =TRUE
                input           =AUV_IsLogging
                label           ="Log FB"
                window          =AUV_Control
                width           =MAN_BIG_BUTTON_WIDTH
                height          =MAN_BIG_BUTTON_HEIGHT
                depth           =NO_DEPTH
                border          =TRUE
                off_style       =button_off_style
                on_style        =green_button_on_style

// ****************** SBE-19 CTD Values Display
***********************

%AUV_Control.window.number
    x=COL_13 y=ROW_17
    string="C:"
    input=AUV_Conductivity
    width=9
    precision=4

%AUV_Control.window.number
    x=COL_13 y=ROW_18
    string="S:"
    input=AUV_StandardConductivity
    width=9
    precision=4

%AUV_Control.window.number
    x=COL_13 y=ROW_19
    string="T:"
    input=AUV_Temperature
    width=9
    precision=4

%AUV_Control.window.number
    x=COL_13 y=ROW_20
    string="D:"
    input=AUV_Pressure
    width=9
    precision=4

// ***************** Motor Feedback Display
******************************

%AUV_Control.window.label
        x=COL_15 y=ROW_23
        string="Thruster RPM"

//command RPM
%AUV_Control.window.number
        x=COL_14 y=ROW_24
        input=AUV_LeftThrustInterlocked
        string=" L:"
        width=6

%AUV_Control.window.number
        x=COL_14 y=ROW_25
        input=AUV_RightThrustInterlocked
        string=" R:"
        width=6

%AUV_Control.window.number
        x=COL_14 y=ROW_26
        input=AUV_VertThrustInterlocked
        string="VL:"
        width=6

%AUV_Control.window.number
        x=COL_14 y=ROW_27
        input=AUV_VertThrustInterlocked
        string="VR:"
```

```
        width=6

//actual RPM
%AUV_Control.window.number
        x=COL_17 y=ROW_24
        input=AUV_Left_RPM_Fb
        width=6

%AUV_Control.window.number
        x=COL_17 y=ROW_25
        input=AUV_Right_RPM_Fb
        width=6

%AUV_Control.window.number
        x=COL_17 y=ROW_26
        input=AUV_VertLeft_RPM_Fb
        width=6

%AUV_Control.window.number
        x=COL_17 y=ROW_27
        input=AUV_VertRight_RPM_Fb
        width=6

//actual PWM
%AUV_Control.window.number
        x=COL_19 y=ROW_24
        input=AUV_Left_PWM_Fb
        width=4

%AUV_Control.window.number
        x=COL_19 y=ROW_25
        input=AUV_Right_PWM_Fb
        width=4

%AUV_Control.window.number
        x=COL_19 y=ROW_26
        input=AUV_VertLeft_PWM_Fb
        width=4

%AUV_Control.window.number
        x=COL_19 y=ROW_27
        input=AUV_VertRight_PWM_Fb
        width=4
```

setpoint.csp

```
/*
$Log: setpoint.csp $
# Revision 1.5  1996/09/10  10:05:03  COUSTEAU
# Peter H: Added the bottom tracking buttons
#
# Revision 1.4  1996/08/15  08:33:11  NEMO
# added telem checking stuff
#
# Revision 1.3  1996/08/08  17:20:25  NEMO
# changed user interface
#
# Revision 1.2  1996/08/06  15:39:19  COUSTEAU
# added altimeter signal strength to user interface
#
# Revision 1.1  1996/08/01  12:09:12  dosuser
# Initial revision
#
*/
//********************** SETPOINT.CSP
*************************************
// The buttons required to specify the setpoints for the AUV
heading, depth,
// and velocity.  This configuration file also displays the
feedbacks for
// the various setpoints.
// All the buttons and feedbacks in this configuration file a
displayed
// in the main graphics window "AUV_Control".

// PILOT MODE LABEL
%AUV_Control.window.label
        x=COL_0 y=ROW_1
        string="Pilot Mode Controls"

// HEADING SETPOINT BUTTONS AND FEEDBACK
%AUV_Control.window.label
        x=COL_1 y=ROW_2
        string="Heading"

%AUV_Control.window.number
        x=COL_1 y=ROW_3
        input=TelemHeadingSetpoint
        width=6

%AUV_Control.window.number
        x=COL_1 y=ROW_4
        input=AUV_CompassHeading
        width=6
        precision=1


// HEADING SETPOINT JOG BUTTONS
%window.momentary.button
        input           = HeadingSpJog
        output          = HeadingSpJog
        on_value        = -2
        label           = "\x11"            // left arrow
        window          = AUV_Control
        width           = MAN_MED_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        left            = COL_0
        top             = ROW_3

%window.momentary.button
        input           = HeadingSpJog
        output          = HeadingSpJog
        on_value        = 2
        label           = "\x10"            // right
arrow
        window          = AUV_Control
        width           = MAN_MED_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        left            = COL_3
        top             = ROW_3


// HEADING SETPOINT OVERRIDE
%circular.range.param
        name    =HeadingRange
        min     =0
        max     =360

%and
        inputs          =TRUE
        inputs          =HeadingSpJog
        output          =EnableHeadingJog

%jog.override
        name            = HeadingSpOvr
        trigger         = SYS_FourHzTrigger
        input           = HeadingSpJog
        rate            = 40
        level           = MEDIUM
        range           = HeadingRange
        output          = TelemHeadingSetpoint
        initialize      = TelemHeadingSetpoint
        enable          = EnableHeadingJog


// VELOCITY SETPOINT BUTTONS AND FEEDBACK
%AUV_Control.window.label
        x=COL_1 y=ROW_6
        string=" Velocity (RPM)"

%AUV_Control.window.number
        x=COL_1 y=ROW_7
        input=TelemVelocitySetpoint
        width=6

// VELOCITY SETPOINT JOG BUTTONS
%window.momentary.button
        input           = VelocitySpJog
        output          = VelocitySpJog
        on_value        = -1
        label           = "\x1f"            // down arrow
        window          = AUV_Control
        width           = MAN_MED_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        left            = COL_0
        top             = ROW_8

%window.momentary.button
        input           = VelocitySpJog
        output          = VelocitySpJog
        on_value        = 1
        label           = "\x1e"            // up arrow
        window          = AUV_Control
        width           = MAN_MED_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        left            = COL_0
        top             = ROW_6

%window.momentary.button
        input           = ZeroVelocitySp
        output          = ZeroVelocitySp
        label           = "0"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        window          = AUV_Control
        width           = MAN_MED_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_3
        top             = ROW_8

%const.int  name=LARGE_VELOCITY  value=3000
%window.momentary.button
        input           = LargeVelocitySp
        output          = LargeVelocitySp
        label           = "3000"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_1
        top             = ROW_8
```

```
// VELOCITY SETPOINT OVERRIDE
%range.param
        name     = VelocityRange
        min      = -5000
        max      =  5000

%and
        inputs          =TRUE
        inputs          =VelocitySpJog
        output          =EnableVelocityJog

%jog.override
        name             = VelocitySpOvr
        trigger          = SYS_FiveHzTrigger
        input            = VelocitySpJog
        rate             = 500
        level            = MEDIUM
        range            = VelocityRange
        output           = TelemVelocitySetpoint
        initialize       = TelemVelocitySetpoint
        enable           = EnableVelocityJog

%copy
        enable           = ZeroVelocitySp
        input            = ZERO
        output           = TelemVelocitySetpoint
%copy
        enable           = LargeVelocitySp
        input            = LARGE_VELOCITY
        output           = TelemVelocitySetpoint

// DEPTH SETPOINT BUTTONS
%AUV_Control.window.label
        x=COL_2 y=ROW_11
        string=" Depth (m)"

%AUV_Control.window.number
        x=COL_2 y=ROW_12
        input=TelemDepthSetpoint
        width=6
        precision=2

%AUV_Control.window.number
        x=COL_2 y=ROW_13
        input=AUV_Depth
        width=6
        precision=2


// DEPTH SETPOINT JOG BUTTONS
// Depth Setpoint: Fine
%window.momentary.button
        input            = DepthSpJog
        output           = DepthSpJog
        on_value         = 1
        label            = "\x1f"               // down arrow
        window           = AUV_Control
        width            = MAN_MED_BUTTON_WIDTH
        height           = MAN_MED_BUTTON_HEIGHT
        depth            = DEEP_DEPTH
        border           = FALSE
        off_style        = switch_off_button_style
        on_style         = switch_on_button_style
        left             = COL_1
        top              = ROW_13

%window.momentary.button
        input            = DepthSpJog
        output           = DepthSpJog
        on_value         = -1
        label            = "\x1e"               // up arrow
        window           = AUV_Control
        width            = MAN_MED_BUTTON_WIDTH
        height           = MAN_MED_BUTTON_HEIGHT
        depth            = DEEP_DEPTH
        border           = FALSE
        off_style        = switch_off_button_style
        on_style         = switch_on_button_style
        left             = COL_1
        top              = ROW_11

// Depth Setpoint: Coarse
%window.momentary.button
        input            = DepthSpJog
        output           = DepthSpJog
        on_value         = 20
        label            = "\x1f\x1f"               // down
arrow
        window           = AUV_Control
        width            = MAN_MED_BUTTON_WIDTH
        height           = MAN_MED_BUTTON_HEIGHT
        depth            = DEEP_DEPTH
        border           = FALSE
        off_style        = switch_off_button_style
        on_style         = switch_on_button_style
        left             = COL_0
        top              = ROW_13

%window.momentary.button
        input            = DepthSpJog
        output           = DepthSpJog
        on_value         = -20
        label            = "\x1e\x1e"                    // up
arrow
        window           = AUV_Control
        width            = MAN_MED_BUTTON_WIDTH
        height           = MAN_MED_BUTTON_HEIGHT
        depth            = DEEP_DEPTH
        border           = FALSE
        off_style        = switch_off_button_style
        on_style         = switch_on_button_style
        left             = COL_0
        top              = ROW_11

%window.momentary.button
        input            = ZeroDepthSp
        output           = ZeroDepthSp
        label            = "0"
        on_value         = 1
        off_style        = switch_off_button_style
        on_style         = switch_on_button_style
        window           = AUV_Control
        width            = MAN_MED_BUTTON_WIDTH
        height           = MAN_MED_BUTTON_HEIGHT
        depth            = DEEP_DEPTH
        border           = FALSE
        left             = COL_4
        top              = ROW_13

// DEPTH SETPOINT OVERRIDE
%range.param    // in DummyDepthSetpoint Units
        name     =DepthRange
        min      = -1
        max      = 70

%and
        inputs          =TRUE
        inputs          =DepthSpJog
        output          =EnableDepthJog

%jog.override
        name             = DepthSpOvr
        trigger          = SYS_FourHzTrigger
        input            = DepthSpJog
        rate             = 1
        level            = MEDIUM
        range            = DepthRange
        output           = TelemDepthSetpoint
        initialize       = TelemDepthSetpoint
        enable           = EnableDepthJog

%copy
        enable           = ZeroDepthSp
        input            = ZERO
        output           = TelemDepthSetpoint

// Altitude Setpoint Buttons

%window.label
        string           = "Altitude (m)"
        window           = AUV_Control
        style            = label_style
        center           = FALSE
        fill             = TRUE
        x                = COL_2
        y                = ROW_17

%AUV_Control.window.number
            x                = COL_2
            y                = ROW_18
            input            = TelemAltitudeSetpoint
            width            = 6
            precision        = 2

%AUV_Control.window.number
            x                = COL_2
            y                = ROW_19
            input            = AUV_Altitude
            width            = 6
            precision        = 2

%AUV_Control.window.number
```

```
        x               = COL_0
        y               = ROW_23
        string          = "AltSigStrength"
        input           = AUV_AltSignalStrength
        width           = 6
        precision       = 0

// Altitude Setpoint: Coarse
%window.momentary.button
        input           = AltitudeSpJog
        output          = AltitudeSpJog
        on_value        = 20
        label           = "\x1e\x1e"              // up
arrows
        window          = AUV_Control
        width           = MAN_MED_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        left            = COL_0
        top             = ROW_17

%window.momentary.button
        input           = AltitudeSpJog
        output          = AltitudeSpJog
        on_value        = -20
        label           = "\x1f\x1f"              // down
arrows
        window          = AUV_Control
        width           = MAN_MED_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        left            = COL_0
        top             = ROW_19

// Altitude Setpoint: Fine
%window.momentary.button
        input           = AltitudeSpJog
        output          = AltitudeSpJog
        on_value        = 1
        label           = "\x1e"          // up arrow
        window          = AUV_Control
        width           = MAN_MED_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        left            = COL_1
        top             = ROW_17

%window.momentary.button
        input           = AltitudeSpJog
        output          = AltitudeSpJog
        on_value        = -1
        label           = "\x1f"          // down arrow
        window          = AUV_Control
        width           = MAN_MED_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        left            = COL_1
        top             = ROW_19

%window.momentary.button
        input           = ResetAltitudeSp
        output          = ResetAltitudeSp
        label           = "10m"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = switch_on_button_style
        window          = AUV_Control
        width           = MAN_MED_BUTTON_WIDTH
        height          = MAN_MED_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_1
        top             = ROW_21

// ALTITUDE SETPOINT OVERRIDE

%range.param
        name    = AltitudeRange
        min     = 0
        max     = 200

%and
        inputs          =TRUE
        inputs          =AltitudeSpJog
        output          =EnableAltitudeJog

%jog.override
        name            = AltitudeSpOvr
        trigger         = SYS_FourHzTrigger
        input           = AltitudeSpJog
        rate            = 1
        level           = MEDIUM
        range           = AltitudeRange
        output          = TelemAltitudeSetpoint
        initialize      = TelemAltitudeSetpoint
        enable          = EnableAltitudeJog

%copy
        enable          = ResetAltitudeSp
        input           = TEN
        output          = TelemAltitudeSetpoint

// toggle AUV into bottom following mode

%window.toggle.button
        input           = TelemBottomFollowingEnabled
        output          = TelemBottomFollowingEnabled
        label           = "Bot Trk"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = green_button_on_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_BIG_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_0
        top             = ROW_25

%window.button
        input           =AUV_FollowingBottom
        label           ="Tracking"
        on_value        =TRUE
        on_style        =green_button_on_style
        off_style       =switch_off_button_style      //This
is really an off style too.
        window          =AUV_Control
        width           =MAN_BIG_BUTTON_WIDTH
        height          =MAN_BIG_BUTTON_HEIGHT
        depth           =NO_DEPTH
        border          =TRUE
        left            =COL_2
        top             =ROW_25
```

stat_win.csp

```
/*
$Log: stat_win.csp $
# Revision 1.2  1996/08/24  10:39:21  PURL
# Peter H: Removed AUV_FifoMeter
#
# Revision 1.1  1996/08/01  12:08:12  dosuser
# Initial revision
#
*/
// *********************** STAT_WIN.CSP
********************************

%data.window
                        display=screen
                        top=3 left=3
                        width=32 height=20
                        name=status
                        title="Status Window"
                        enable=SwitchStatusWindow
                        name_width=20
                        scrollbar=1
                        border=TRUE

%window.int.data        window=status input=AUV_IdleMeter
%window.int.data        window=status input=AUV_TxStatus
%window.int.data        window=status input=AUV_RxStatus
%window.int.data        window=status
input=AUV_TimeoutsOccurred
%window.float.data      window=status width=6 precision=0.01
input=AUV_Freemem
%window.int.data        window=status input=AUV_Mode

%window.int.data        window=status input=SurfaceIdleMeter
%window.int.data        window=status input=SurfaceTxStatus
%window.int.data        window=status input=SurfaceRxStatus
%window.int.data        window=status
input=SurfaceTimeoutsOccurred
%window.int.data        window=status input=SurfaceFifoMeter
%window.float.data      window=status width=6 precision=0.01
input=SurfaceFreemem
%window.int.data        window=status input=TelDiag
```

dbg_win.csp

```
/*
$Log: dbg_win.csp $
# Revision 1.1  1996/08/16  08:44:25  NEMO
# Initial revision
#
*/
// *********************** STAT_WIN.CSP
********************************

%data.window
                        display=screen
                        top=3 left=10
                        width=32 height=20
                        name=debug
                        title="Debug Window"
                        enable=SwitchDebugWindow
                        name_width=20
                        scrollbar=1
                        border=TRUE

%window.int.data        window=debug input=AUV_DebugInt1
%window.int.data        window=debug input=AUV_DebugInt2
%window.int.data        window=debug input=AUV_DebugInt3
%window.int.data        window=debug input=AUV_DebugInt4
%window.int.data        window=debug input=AUV_DebugInt5
%window.float.data      window=debug width=6 precision=2
input=AUV_DebugFloat1
%window.float.data      window=debug width=6 precision=2
input=AUV_DebugFloat2
%window.float.data      window=debug width=6 precision=2
input=AUV_DebugFloat3
%window.float.data      window=debug width=6 precision=2
input=AUV_DebugFloat4
%window.float.data      window=debug width=6 precision=2
input=AUV_DebugFloat5

%window.int.data        window=debug input=SurfaceDebugInt1
%window.int.data        window=debug input=SurfaceDebugInt2
%window.int.data        window=debug input=SurfaceDebugInt3
%window.int.data        window=debug input=SurfaceDebugInt4
%window.int.data        window=debug input=SurfaceDebugInt5
%window.float.data      window=debug width=6 precision=2
input=SurfaceDebugFloat1
%window.float.data      window=debug width=6 precision=2
input=SurfaceDebugFloat2
%window.float.data      window=debug width=6 precision=2
input=SurfaceDebugFloat3
%window.float.data      window=debug width=6 precision=2
input=SurfaceDebugFloat4
%window.float.data      window=debug width=6 precision=2
input=SurfaceDebugFloat5
```

auv_mode.csp

```
/*
$Id: auv_mode.csp 1.4 1996/10/01 20:30:41 COUSTEAU Exp $
$Log: auv_mode.csp $
# Revision 1.4  1996/10/01  20:30:41  COUSTEAU
# Peter H: Moved around some of the buttons
#
# Revision 1.3  1996/08/08  17:20:25  NEMO
# changed user interface
#
# Revision 1.2  1996/08/06  17:34:47  COUSTEAU
# AH: added EXIT mode
#
# Revision 1.1  1996/08/01  11:47:21  dosuser
# Initial revision
#
*/
//*********************** AUV_MODE.CSP
************************************
// This configuration file contains the components used for
selecting the
// mode of AUV operation and displaying the feedback
indicating which mode
// the AUV thinks its in.

// How the mode selection works
//
// Both the surface computer and the AUV start up in IDLE
mode.
// The surface computer sends the desired state to the AUV
whenever the
// desired state changes or a periodic timer expires.  The
periodic timer
// ensures that if the initial state change telemetry was
lost due to a bad
// link, the desired state will reach the AUV in subsequent
telemetry packets.
// The AUV sends its current state to the surface
periodically or on a state
// change.
// The AUV Mode feeback displayed on the surface computer
shows the most
// recent mode information that was successfully transmitted
from the AUV
// to the surface.  Upon startup there is no mode feedback
and the surface
// must wait to receive a periodic update before a value will
be displayed
// on the screen.



// ******** AUV MODE SELECT ********
// The following components are the buttons and glue logic
needed to select
// which mode the AUV should be operating in.

%AUV_Control.window.label
        string          = "AUV Mode Selection"
        x               = COL_10
        y               = ROW_1

%selection
        name            = AUV_mode_select
        input           = TelemModeSelect

%AUV_mode_select.selection.output
        choice          = IDLE
        event           = IdleModeSelected

%AUV_mode_select.selection.output
        choice          = PILOT
        event           = PilotModeSelected

%AUV_mode_select.selection.output
        choice          = ABORT
        event           = AbortModeSelected

%AUV_mode_select.selection.output
        choice          = MISSION1
        event           = Mission1ModeSelected

%AUV_mode_select.selection.output
        choice          = MISSION2
        event           = Mission2ModeSelected

%AUV_mode_select.selection.output
        choice          = MISSION3
        event           = Mission3ModeSelected

%AUV_mode_select.selection.output
        choice          = MISSION4
        event           = Mission4ModeSelected

%AUV_mode_select.selection.output
        choice          = MISSION5
        event           = Mission5ModeSelected

%AUV_mode_select.selection.output
        choice          = MISSION6
        event           = Mission6ModeSelected

// ************* Toggle Buttons For Mode Selection
**********************

%window.toggle.button
        input           = PilotModeSelected
        output          = PilotModeRequest
        label           = "Pilot"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = yellow_button_on_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_BIG_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_11
        top             = ROW_3

%copy
        enable          = PilotModeRequest
        input           = PILOT
        output          = TelemModeSelect

%window.toggle.button
        input           = IdleModeSelected
        output          = IdleModeRequest
        label           = "Idle"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = red_button_on_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_BIG_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_9
        top             = ROW_3

%copy
        enable          = IdleModeRequest
        input           = IDLE
        output          = TelemModeSelect

%window.toggle.button
        input           = AbortModeSelected
        output          = AbortModeRequest
        label           = "Abort"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = red_button_on_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_BIG_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_13
        top             = ROW_3

%copy
        enable          = AbortModeRequest
        input           = ABORT
        output          = TelemModeSelect


%window.toggle.button
        input           = Mission1ModeSelected
        output          = Mission1ModeRequest
        label           = "M 1"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = green_button_on_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_BIG_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_9
```

```
        top             = ROW_5

%copy
        enable          = Mission1ModeRequest
        input           = MISSION1
        output          = TelemModeSelect

%window.toggle.button
        input           = Mission2ModeSelected
        output          = Mission2ModeRequest
        label           = "M 2"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = green_button_on_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_BIG_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_11
        top             = ROW_5

%copy
        enable          = Mission2ModeRequest
        input           = MISSION2
        output          = TelemModeSelect

%window.toggle.button
        input           = Mission3ModeSelected
        output          = Mission3ModeRequest
        label           = "M 3"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = green_button_on_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_BIG_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_13
        top             = ROW_5

%copy
        enable          = Mission3ModeRequest
        input           = MISSION3
        output          = TelemModeSelect

%window.toggle.button
        input           = Mission4ModeSelected
        output          = Mission4ModeRequest
        label           = "M 4"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = green_button_on_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_BIG_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_9
        top             = ROW_7

%copy
        enable          = Mission4ModeRequest
        input           = MISSION4
        output          = TelemModeSelect

%window.toggle.button
        input           = Mission5ModeSelected
        output          = Mission5ModeRequest
        label           = "M 5"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = green_button_on_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_BIG_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_11
        top             = ROW_7

%copy
        enable          = Mission5ModeRequest
        input           = MISSION5
        output          = TelemModeSelect

%window.toggle.button
        input           = Mission6ModeSelected
        output          = Mission6ModeRequest
        label           = "M 6"
        on_value        = 1
        off_style       = switch_off_button_style
        on_style        = green_button_on_style
        window          = AUV_Control
        width           = MAN_BIG_BUTTON_WIDTH
        height          = MAN_BIG_BUTTON_HEIGHT
        depth           = DEEP_DEPTH
        border          = FALSE
        left            = COL_13
        top             = ROW_7

%copy
        enable          = Mission6ModeRequest
        input           = MISSION6
        output          = TelemModeSelect

// Display the current mission step and the local script step
%AUV_Control.window.number
        x=COL_9 y=ROW_9
        string="MStep:"
        input=MissionStep
        width=2

%AUV_Control.window.number
        x=COL_12 y=ROW_9
        string="LStep:"
        input=LocalStep
        width=2


// Mode Feedback Indicating What Mode The AUV Thinks its in
// The initial value of AUVMode is DEBUG so none of the
buttons are
// illuminated, the surface computer must received a
successful telemetry
// message contain the AUV mode before it will show the AUV's
current mode

%selection
        name            = WhatAUVMode
        input           = AUV_Mode

%WhatAUVMode.selection.output
        choice          = PILOT
        event           = InPilotMode

%WhatAUVMode.selection.output
        choice          = IDLE
        event           = InIdleMode

%WhatAUVMode.selection.output
        choice          = EXIT
        event           = InExitMode

%WhatAUVMode.selection.output
        choice          = ABORT
        event           = InAbortMode

%WhatAUVMode.selection.output
        choice          = MISSION1
        event           = InMissionMode

%WhatAUVMode.selection.output
        choice          = MISSION2
        event           = InMissionMode

%WhatAUVMode.selection.output
        choice          = MISSION3
        event           = InMissionMode

%WhatAUVMode.selection.output
        choice          = MISSION4
        event           = InMissionMode

%WhatAUVMode.selection.output
        choice          = MISSION5
        event           = InMissionMode

%WhatAUVMode.selection.output
        choice          = MISSION6
        event           = InMissionMode

// AUV Mode Feedback buttons

%AUV_Control.window.label
        string          = "  Mode Feedback"
        x               = COL_16
        y               = ROW_1

%window.button
        left            =COL_16
```

```
        top             =ROW_3
        on_value        =1
        input           =InIdleMode
        label           ="IDLE"
        window          =AUV_Control
        width           =MAN_BIG_BUTTON_WIDTH
        height          =MAN_BIG_BUTTON_HEIGHT
        depth           =NO_DEPTH
        border          =TRUE
        off_style       =button_off_style
        on_style        =red_button_on_style

%window.button
        left            =COL_18
        top             =ROW_3
        on_value        =1
        input           =InPilotMode
        label           ="Pilot"
        window          =AUV_Control
        width           =MAN_BIG_BUTTON_WIDTH
        height          =MAN_BIG_BUTTON_HEIGHT
        depth           =NO_DEPTH
        border          =TRUE
        off_style       =button_off_style
        on_style        =yellow_button_on_style

%window.button
        left            =COL_18
        top             =ROW_5
        on_value        =1
        input           =InAbortMode
        label           ="Abort"
        window          =AUV_Control
        width           =MAN_BIG_BUTTON_WIDTH
        height          =MAN_BIG_BUTTON_HEIGHT
        depth           =NO_DEPTH
        border          =TRUE
        off_style       =button_off_style
        on_style        =red_button_on_style

%window.button
        left            =COL_18
        top             =ROW_7
        on_value        =1
        input           =InExitMode
        label           ="Exit"
        window          =AUV_Control
        width           =MAN_BIG_BUTTON_WIDTH
        height          =MAN_BIG_BUTTON_HEIGHT
        depth           =NO_DEPTH
        border          =TRUE
        off_style       =button_off_style
        on_style        =red_button_on_style

%window.button
        left            =COL_16
        top             =ROW_5
        on_value        =1
        input           =InMissionMode
        label           ="Mission"
        window          =AUV_Control
        width           =MAN_BIG_BUTTON_WIDTH
        height          =MAN_BIG_BUTTON_HEIGHT
        depth           =NO_DEPTH
        border          =TRUE
        off_style       =button_off_style
        on_style        =green_button_on_style
```

pdef.csp

```
/*
$Log: pdef.csp $
# Revision 1.7  1996/11/22  12:36:23  PURL
# Peter H: Added PAYLOAD.CSP to the list of configuration
files
#
# Revision 1.6  1996/08/24  10:40:08  PURL
# Peter H: Added altimeter items for control
#
# Revision 1.5  1996/08/15  08:29:54  PURL
# Peter H: Additional ABORT condition variables
#
# Revision 1.4  1996/08/14  16:31:32  PURL
# debugging
#
# Revision 1.3  1996/08/07  10:20:45  COUSTEAU
# removed reference to AUV_ExitProteus
#
# Revision 1.2  1996/08/06  17:34:47  COUSTEAU
# AH: added EXIT mode
#
# Revision 1.1  1980/01/04  03:17:38  PURL
# Initial revision
#
*/
//    *******************PURLDEF.CSP***********************

%include        file="XCONST.CSP"
%include        file="PHEADER.CSP"
%include        file="XTIMER.CSP"
%include        file="PPORTS.CSP"
%include        file="XTELEM.CSP"
%include        file="CONTROL.CSP"
%include        file="SAPPHIRE.CSP"
%include        file="EXITSCPT.CSP"
%include        file="DISPLAY.CSP"
%include        file="XSTYLE.CSP"
%include        file="PURL_WIN.CSP"
%include        file="PAYLOAD.CSP"

%include        file="MISSION1.CSP"
%include        file="MISSION2.CSP"
%include        file="MISSION3.CSP"
%include        file="MISSION4.CSP"
%include        file="MISSION5.CSP"
%include        file="MISSION6.CSP"

%include        file="LOG.CSP"

%idle.meter     trigger=SYS_TwoSecondTrigger
                    output=AUV_IdleMeter

%freemem        trigger=SYS_TwoSecondTrigger
                    output=AUV_Freemem

%exit
                    input=ExitProteus
                    message="Exiting Proteus Normally"

%exception_log  file="ERROR.TXT"
                    level=TOP
                    buffer_size=256
%copy
          input=LocalTxStatus
          output=AUV_TxStatus
          enable=TRUE

%copy
          input=LocalRxStatus
          output=AUV_RxStatus
          enable=TRUE

%copy
          input=LocalTimeoutsOccurred
          output=AUV_TimeoutsOccurred
          enable=TRUE

//Telemetry Alive Timer components

//increment the timer
%cycle
          trigger  = SYS_TwoSecondTrigger
          level    = LOW
          init     = 0
          min      = 0
          max      = 10000
          step     = 1
          reset    = SurfaceTelemCounter
          output   = AUV_TelemCounter

%greater
          enable = AUV_TelemCheckStatus
          input1 = AUV_TelemCounter
          input2 = MAX_TELEM_COUNTER
          output = LostDownTelemetry
%greater
          enable = AUV_TelemCheckStatus
          input1 = SurfaceTelemCounter
          input2 = MAX_TELEM_COUNTER
          output = LostUpTelemetry

%or
          enable = TRUE
          inputs = LostDownTelemetry
          inputs = LostUpTelemetry
          output = AUV_NoTelemetry

%copy
          enable = TRUE
          input  = SurfaceTelemCheckStatus
          output = AUV_TelemCheckStatus

%copy
          enable = PilotModeSelected
          input = TRUE
          output = AUV_TelemCheckStatus

%copy
          enable = MissionModeSelected
          input = FALSE
          output = AUV_TelemCheckStatus


// Stop parsing.
!STOP
```

pheader.csp

```
/*
$Log: pheader.csp $
# Revision 1.14  1997/02/08  18:13:37  PURL
# Peter H: Reseting the WatchDogValue
#
# Revision 1.13  1996/09/18  20:37:43  PURL
# Peter H and Kevin M: Added items for integrating the
altimeter into the
# control system
#
# Revision 1.12  1996/08/30  20:39:53  PURL
# Peter H: Removed ThrusterInterlock
#
# Revision 1.11  1996/08/24  10:40:59  PURL
# Peter H: Added altimeter items for control
#
# Revision 1.10  1996/08/15  08:31:55  PURL
# Peter H: ABORT condition additions
#
# Revision 1.9  1996/08/14  16:27:26  PURL
# changed some of the gains, other debugging changes
#
# Revision 1.8  1996/08/09  11:46:55  PURL
# AH: changed the interlocking logic to using MotorMode
#
# Revision 1.7  1996/08/09  09:04:51  COUSTEAU
# added abort when leaking/low battery occurs
#
# Revision 1.6  1996/08/09  08:05:18  PURL
# Peter H: Added the conversion from centimeters to meters.
# Added signals and events for latching the battery and leak
sensor alarms.
#
# Revision 1.5  1996/08/07  09:34:06  PURL
# Peter H: Added the constant for converting from Sapphire A
To D conversions
# to appropriate units for use in the rest of the PURL
system.
# Added the uncontrolled ints for the pitch, roll, and
battery voltage conveons.
#
# Revision 1.4  1996/08/06  17:34:47  COUSTEAU
# AH: added EXIT mode
#
# Revision 1.3  1996/08/06  14:30:18  COUSTEAU
# removed unused signals
#
# Revision 1.2  1996/08/02  14:48:56  PURL
# removed thruster input events
#
# Revision 1.1  1980/01/04  03:13:55  PURL
# Initial revision
#
*/
//   ***********************HEADER.CSP*******************

// *************** General Defines and Constants
************************
%define        name=SURFACE                   value=0
%const.int     name=SURFACE                   value=SURFACE
%define        name=AUV                       value=1
%const.int     name=AUV                       value=AUV


%const.float   name=LOW_BATTERY_VOLTAGE       value=20.0

%const.int     name=DEPTH_TO_RPM          value=1
%const.int     name=HEADING_TO_RPM        value=1
%const.int     name=VELOCITY_TO_RPM       value=1
%const.float   name=CENTIMETERS_TO_METERS          value =
0.01
%const.float   name=SAPPHIRE_TO_BATTERY_VOLTAGE    value =
0.00395
%const.float   name=SAPPHIRE_TO_ANGLE              value =
0.0030368695

%define        name=MOTORS_OFF            value=1
%define        name=MOTORS_CONTROLLED     value=2
%define        name=MOTORS_ABORT          value=3
%const.int     name=MOTORS_OFF
value=MOTORS_OFF
%const.int     name=MOTORS_CONTROLLED
value=MOTORS_CONTROLLED
%const.int     name=MOTORS_ABORT
value=MOTORS_ABORT

// Behaviours
%behavior   name     = ControlBehaviour
                layer    = LOW
                level    = HIGH
                enable   = TRUE

// ********** Syncs are generally timing signals or triggers
************
%sync               name=ExitProteus

// ****************** AUV status parameters
****************************
%int                name=TelDiag
%int                name=CompassDiag
%int                name=DepthDiag

// ******* Sensor Outputs and Events Involved In Thruster
Control *******
%uncontrolled.int       name=HeadingControl
%uncontrolled.int       name=DepthControl
%uncontrolled.int       name=AltitudeControl
%uncontrolled.int       name=VerticalControl

%uncontrolled.int       name=VerticalControl_RPM
%uncontrolled.int       name=HeadingControl_RPM
%uncontrolled.int       name=NegHeadingControl_RPM
%uncontrolled.int       name=VelocityControl_RPM

%uncontrolled.int       name=LeftThrust
%uncontrolled.int       name=RightThrust
%uncontrolled.int       name=RightThrust_RH_Prop
%int                    name=MotorsAbort
%int                    name=MotorsControlled
%int                    name=MotorsOff
%uncontrolled.int       name=MotorMode
initial=MOTORS_OFF

%uncontrolled.float     name=DepthIn_cm
initial=0.0
%uncontrolled.float     name=AltitudeIn_cm
initial=0.0
%uncontrolled.int       name=BelowAltitudeSetpoint
initial=FALSE
%uncontrolled.int       name=AboveAltitudeSetpoint
initial=FALSE
%uncontrolled.int       name=BelowDepthSetpoint
initial=FALSE
%uncontrolled.int       name=AboveDepthSetpoint
initial=FALSE
%uncontrolled.int       name=LogicTermOne
initial=FALSE
%uncontrolled.int       name=AltitudeControlSelected
initial=FALSE
%uncontrolled.int       name=DepthControlSelected
initial=FALSE

%uncontrolled.int       name=WatchDogValue
initial=0
%int                    name=GotoAbortMode
%int                    name=AbortCondition


// Setpoints
%uncontrolled.float     name=HeadingSetpoint
initial=0.0
%uncontrolled.float     name=VelocitySetpoint
initial=0.0
%uncontrolled.float     name=DepthSetpoint
initial=0.0
%uncontrolled.float     name=AltitudeSetpoint
initial=0.0
%uncontrolled.float     name=AbortHeadingSetpoint
initial=0.0
%uncontrolled.float     name=AbortVelocitySetpoint
initial=0.0
%uncontrolled.float     name=AbortDepthSetpoint
initial=0.0
%uncontrolled.float     name=AbortAltitudeSetpoint
initial=0.0
%uncontrolled.float     name=MissionHeadingSetpoint
initial=0.0
%uncontrolled.float     name=MissionVelocitySetpoint
initial=0.0
%uncontrolled.float     name=MissionDepthSetpoint
initial=0.0
%uncontrolled.float     name=MissionAltitudeSetpoint
initial=0.0

// *************** Sapphire Board Inputs and Outputs
*******************
%uncontrolled.int       name = DumbEvent
initial = 0
```

```
%uncontrolled.int       name = PitchAToD
initial = 0
%uncontrolled.int       name = RollAToD
initial = 0
%uncontrolled.int       name = BatteryAToD
initial = 8000 //A High Battery Voltage
%uncontrolled.int       name = EnableLowBattery
initial = FALSE
%uncontrolled.int       name = LeakSensorDigIn
initial = TRUE
%uncontrolled.int       name = EnableLeakAlarm
initial = FALSE

// *************** AUV Mode Switching Parameters
**********************
%int                    name = IdleModeSelected
%int                    name = PilotModeSelected
%int                    name = AbortModeSelected
%int                    name = NotAbortModeSelected
%int                    name = ExitModeSelected
%int                    name = NotExitModeSelected
%int                    name = NotMissionModeSelected
%int                    name = MissionModeSelected

%uncontrolled.int       name=GoMission1
initial=FALSE
%uncontrolled.int       name=GoMission2
initial=FALSE
%uncontrolled.int       name=GoMission3
initial=FALSE
%uncontrolled.int       name=GoMission4
initial=FALSE
%uncontrolled.int       name=GoMission5
initial=FALSE
%uncontrolled.int       name=GoMission6
initial=FALSE

%uncontrolled.int       name=LostDownTelemetry
initial=FALSE
%uncontrolled.int       name=LostUpTelemetry
initial=FALSE

// ********************** Logging Variables
****************************
%sync                   name = LogTrigger
%sync                   name = M1LogTrigger
%sync                   name = M2LogTrigger
%sync                   name = M3LogTrigger
%sync                   name = M4LogTrigger
%sync                   name = M5LogTrigger
%sync                   name = M6LogTrigger

%uncontrolled.int       name = LogSize
%uncontrolled.int       name = M1LogEnable      initial =
FALSE
%uncontrolled.int       name = M1LogSize
%uncontrolled.int       name = M2LogEnable      initial =
FALSE
%uncontrolled.int       name = M2LogSize
%uncontrolled.int       name = M3LogEnable      initial =
FALSE
%uncontrolled.int       name = M3LogSize
%uncontrolled.int       name = M4LogEnable      initial =
FALSE
%uncontrolled.int       name = M4LogSize
%uncontrolled.int       name = M5LogEnable      initial =
FALSE
%uncontrolled.int       name = M5LogSize
%uncontrolled.int       name = M6LogEnable      initial =
FALSE
%uncontrolled.int       name = M6LogSize
```

pports.csp

```
// Ports on Purl
//////////////////////////////////////////////////////////////
//
// Telemetry Ports (UDP or Serial)

%udp.port
	level = HIGH
	name = TelemPort
	local_socket = 4000
	remote_socket = 4100
	remote_host = "cousteau"
//	  remote_host = "nemo"
	signal_size = 4096
	max_packet_size = 256

!com.serial.port
	name=TelemPort
	baud_rate=38400
//	  data_bits	= 8
//	  stop_bits	= 1
//	  parity=0
	delimiter=3
	port_number=1
	buffer_size=512

!serial.port.diag
	output=TelDiag
	port=TelemPort
	trigger=SYS_TwoSecondTrigger
	overrun_errors=TRUE
	accumulate=TRUE

//////////////////////////////////////////////////////////////
// KVH C100 Compass Interface
// COM1
%com.serial.port
	name=CompassPort
	baud_rate=9600
//	  data_bits	= 8
//	  stop_bits	= 1
//	  parity=0
	delimiter=13
	port_number=1
	buffer_size=512 //256  // Needed for 32-bit PROTEUS

!serial.port.diag
	output=CompassDiag
	port=CompassPort
	trigger=SYS_TwoSecondTrigger
	overrun_errors=TRUE
	accumulate=TRUE

//////////////////////////////////////////////////////////////
// Motor Controllers
// COM2
%com.serial.port
	name=ThrusterPort
	baud_rate=9600
//	  data_bits	= 8
//	  stop_bits	= 1
//	  parity=0
	delimiter=0
	port_number=2
	buffer_size=512 //256  // Needed for 32-bit PROTEUS


//////////////////////////////////////////////////////////////
//////
// Data Instruments and Digitec 4-20mA to RS232 Interface for
depth sensor
// COM3
%com.serial.port
	name=DepthPort
	baud_rate=9600
//	baud_rate=300
//	  data_bits	= 8
//	  stop_bits	= 1
//	  parity=0
	delimiter=13
	irq_level=5
	port_base=0x3e8
	buffer_size=512 //256  // Needed for 32-bit PROTEUS

!serial.port.diag
	output=DepthDiag
	port=DepthPort
	trigger=SYS_TwoSecondTrigger
	overrun_errors=TRUE
	accumulate=TRUE

//////////////////////////////////////////////////////////////
// SBE-19 CTD Interface
// COM3
!com.serial.port
	name		= CTD_Port
	baud_rate	= 600
	data_bits	= 7
	stop_bits	= 1
	parity		= 2
	delimiter	= 10
	irq_level	= 5
	port_base	= 0x3e8
	buffer_size	= 256

!serial.port.diag
	output=CTD_Diag
	port=CTD_Port
	trigger=SYS_TwoSecondTrigger
	overrun_errors=TRUE
	accumulate=TRUE

//////////////////////////////////////////////////////////////
// Altimeter
// COM4
%com.serial.port
	name=AltimeterPort
	baud_rate=4800
//	  data_bits	= 8
//	  stop_bits	= 1
//	  parity=0
	delimiter=0
	irq_level=7
	port_base=0x2e8
	buffer_size=512 //256
```

control.csp

```
/*
$Log: control.csp $
# Revision 1.19  1997/05/10  16:20:39  PURL
# Peter H: checked in to restablish the software on the new
hard drive
#
# Revision 1.18  1996/11/24  17:06:26  COUSTEAU
# Peter H: Changed MAX_THRUSTER_VELOCITY to
MAX_NEG_THRUSTER_VELOCITY
#
# Revision 1.17  1996/11/22  12:38:16  PURL
# Peter H: Changed depth control to use the CTD pressure
sensor
#
# Revision 1.16  1996/09/18  20:37:43  PURL
# Peter H and Kevin M: Updated PID gains and integrated
altimeter into the
# control system
#
# Revision 1.15  1996/09/07  13:38:36  PURL
# Peter H: Changed the PID gains for the depth and heading
#
# Revision 1.14  1996/08/24  12:07:15  PURL
# Peter H: Moved the vertical thruster to address 3
#
# Revision 1.13  1996/08/19  14:56:09  COUSTEAU
# Peter H: Added the thruster interface component and a
commented out triggeredcopy.
#
# Revision 1.12  1996/08/15  08:12:37  PURL
# Peter H: Added an extra ABORT condition.
#
# Revision 1.11  1996/08/14  17:34:19  PURL
# debugging and such
#
# Revision 1.10  1996/08/09  17:57:15  PURL
# Peter H: Good Question
#
# Revision 1.9  1996/08/09  11:46:55  PURL
# AH: changed the interlocking logic to using MotorMode
#
# Revision 1.8  1996/08/09  09:04:51  COUSTEAU
# added abort when leaking/low battery occurs
#
# Revision 1.7  1996/08/09  08:08:56  PURL
# Peter H: Uncommented the altimeter component and got it
working properly.
#
# Revision 1.6  1996/08/06  17:34:47  COUSTEAU
# AH: added EXIT mode
#
# Revision 1.5  1996/08/06  17:14:27  COUSTEAU
# AH: added altimeter
#
# Revision 1.4  1996/08/06  14:30:46  COUSTEAU
# rearranged some logic and rearranged the order of
components to improve
# the flow of the file.
#
# Revision 1.3  1996/08/02  19:28:41  PURL
# Peter H: Nothing accomplished the compass serial port still
does not work.
#
# Revision 1.2  1996/08/02  14:47:34  PURL
# added new thruster component description and modified names
# of the thruster inputs and outputs
#
# Revision 1.1  1980/01/04  03:14:27  PURL
# Initial revision
#
*/
//*********************** CONTROL.CSP
************************************

// This configuration file interfaces with the motors, KVH
Compass,
// Depth Sensors, and Velocity stuff, AND is responsible for
generating
// the commands that will be sent to the thruster motors to
control
// the AUV

/////////////////////////////////////////////////////////////
// heading sensor

%KVH_compass
                                level=HIGH
                                port=CompassPort
                                out_heading=AUV_CompassHeading
                                trigger=SYS_FourHzTrigger
                                delimeter=13
                                escape=27

/////////////////////////////////////////////////////////////
%Depth_Sensor
                                level=HIGH
                                port=DepthPort
                                out_depth=DepthIn_cm     //
Depth In Centimeters
                                trigger=SYS_FourHzTrigger
                                delimeter=13
                                escape=27

// Convert the depth from centimeters to meters
%multiply
        enable = TRUE
        inputs = DepthIn_cm
        inputs = CENTIMETERS_TO_METERS
        output = AUV_Depth

/////////////////////////////////////////////////////////////
// -------- Simrad Mesotech 809 Serial Interface -----------

%const.int      name = ALTIMETER_SIGNAL_RANGE     value =
200
%const.int      name = ALTIMETER_THRESHOLD        value = 30

%mesotech809_altimeter
        level           = HIGH
        port            = AltimeterPort
        mes809working   = TRUE
        altitude        = AltitudeIn_cm        //
Altitude In Centimeters
        signal_strength = AUV_AltSignalStrength    // 0
to 255
        signal_range    = ALTIMETER_SIGNAL_RANGE   // 0
to 200
        threshold       = ALTIMETER_THRESHOLD      // 0
to 80

// Convert the altitude to meters.
%multiply
        enable = TRUE
        inputs = AltitudeIn_cm
        inputs = CENTIMETERS_TO_METERS
        output = AUV_Altitude

// ******************* Setpoint Selection Logic
****************************
// TelemModeSelect contains the all the mode information.
This section
// breaks out the information into separate modes


// AUV_Mode is echoed to the surface
%copy
        enable          = TRUE
        input           = TelemModeSelect
        output          = AUV_Mode

!triggered.copy
        layer           = MEDIUM
        enable          = TRUE
        trigger         = GotoAbortMode
        input           = ABORT
        output          = TelemModeSelect
%copy
        enable          = GotoAbortMode
        input           = ABORT
        output          = TelemModeSelect

!multiply
        enable          = GotoAbortMode
        inputs          = GotoAbortMode
        inputs          = ABORT
        output          = TelemModeSelect
!calc.copy
        enable          = TRUE
        level           = LOW
        trigger         = GotoAbortMode
        input           = ABORT
        output          = TelemModeSelect


//  %equal used to be %int_equal
%equal
        enable          = TRUE
        input1          = TelemModeSelect
```

```
        input2          = IDLE
        output          = IdleModeSelected
%equal
        enable          = TRUE
        input1          = TelemModeSelect
        input2          = PILOT
        output          = PilotModeSelected
%equal
        enable          = TRUE
        input1          = TelemModeSelect
        input2          = ABORT
        output          = AbortModeSelected
%not
        enable          = TRUE
        input           = AbortModeSelected
        output          = NotAbortModeSelected
%equal
        enable          = TRUE
        input1          = TelemModeSelect
        input2          = EXIT
        output          = ExitModeSelected
%not
        enable          = TRUE
        input           = ExitModeSelected
        output          = NotExitModeSelected

%or
        enable          = TRUE
        inputs          = IdleModeSelected
        inputs          = PilotModeSelected
        inputs          = AbortModeSelected
        inputs          = ExitModeSelected
        output          = NotMissionModeSelected

%not
        enable          = TRUE
        input           = NotMissionModeSelected
        output          = MissionModeSelected

//////////////////////////////////////////////////////////////
// abort logic
%or
        enable          = TRUE
        inputs          = AUV_Leaking
        inputs          = AUV_LowBattery
        inputs          = AUV_NoTelemetry
        output          = AbortCondition

%and
        enable          = TRUE
        inputs          = AbortCondition
        inputs          = NotExitModeSelected
        output          = GotoAbortMode

//////////////////////////////////////////////////////////////
// mission logic
%equal
        input1          = TelemModeSelect
        input2          = MISSION1
        output          = GoMission1
%equal
        input1          = TelemModeSelect
        input2          = MISSION2
        output          = GoMission2
%equal
        input1          = TelemModeSelect
        input2          = MISSION3
        output          = GoMission3
%equal
        input1          = TelemModeSelect
        input2          = MISSION4
        output          = GoMission4
%equal
        input1          = TelemModeSelect
        input2          = MISSION5
        output          = GoMission5
%equal
        input1          = TelemModeSelect
        input2          = MISSION6
        output          = GoMission6

//////////////////////////////////////////////////////////////
// Heading Setpoint multiplexer.
%copy
        enable          = IdleModeSelected
        input           = AUV_CompassHeading
        output          = HeadingSetpoint
%copy
        enable          = PilotModeSelected
        input           = TelemHeadingSetpoint
        output          = HeadingSetpoint
!copy  // not using controller values for abort
        enable          = AbortModeSelected
        input           = AbortHeadingSetpoint
        output          = HeadingSetpoint
%copy
        enable          = ExitModeSelected
        input           = AUV_CompassHeading
        output          = HeadingSetpoint
%copy
        enable          = MissionModeSelected
        input           = MissionHeadingSetpoint
        output          = HeadingSetpoint

//////////////////////////////////////////////////////////////
// Velocity Setpoint multiplexer.
%copy
        enable          = IdleModeSelected
        input           = ZERO
        output          = VelocitySetpoint
%copy
        enable          = PilotModeSelected
        input           = TelemVelocitySetpoint
        output          = VelocitySetpoint
!copy
        enable          = AbortModeSelected
        input           = AbortVelocitySetpoint
        output          = VelocitySetpoint
%copy
        enable          = ExitModeSelected
        input           = ZERO
        output          = VelocitySetpoint
%copy
        enable          = MissionModeSelected
        input           = MissionVelocitySetpoint
        output          = VelocitySetpoint

//////////////////////////////////////////////////////////////
// Depth Setpoint multiplexer.
%copy
        enable          = IdleModeSelected
        input           = AUV_Depth
        output          = DepthSetpoint
%copy
        enable          = PilotModeSelected
        input           = TelemDepthSetpoint
        output          = DepthSetpoint
!copy
        enable          = AbortModeSelected
        input           = AbortDepthSetpoint
        output          = DepthSetpoint
%copy
        enable          = ExitModeSelected
        input           = AUV_Depth
        output          = DepthSetpoint
%copy
        enable          = MissionModeSelected
        input           = MissionDepthSetpoint
        output          = DepthSetpoint

//////////////////////////////////////////////////////////////
// Altitude Setpoint multiplexer.
%copy
        enable          = IdleModeSelected
        input           = AUV_Altitude
        output          = AltitudeSetpoint
%copy
        enable          = PilotModeSelected
        input           = TelemAltitudeSetpoint
        output          = AltitudeSetpoint
!copy
        enable          = AbortModeSelected
        input           = AbortAltitudeSetpoint
        output          = AltitudeSetpoint
%copy
        enable          = ExitModeSelected
        input           = AUV_Altitude
        output          = AltitudeSetpoint
%copy
        enable          = MissionModeSelected
        input           = MissionAltitudeSetpoint
        output          = AltitudeSetpoint

//////////////////////////////////////////////////////////////
///
// Heading PID Controller
%circular.range.param
                        name=heading_error_range
                        min=-179.99999
                        max=180.0


%pid.param
```

```
                              name=pid_heading_gains
                              proportional=300
                              derivative=-300
                              integral=2
                              max_integral=800

%pid.control
                              error_range=heading_error_range
                              gains=pid_heading_gains
                              setpoint=HeadingSetpoint
                              feedback=AUV_CompassHeading
                              output=HeadingControl
                              behavior=ControlBehaviour

/////////////////////////////////////////////////////////////
/
// Depth PID Controller
%range.param
                              name=depth_error_range
                              min=-10000.0 //-100.0
                              max=10000.0  //100.0

%pid.param
                              name=pid_depth_gains
                              proportional=-50 //-5000 //-50
                              derivative=60    //6000 //60
                              integral=-2      //-200 //-2
                              max_integral=600

// Temporary fix because the depth component outputs in
centimeters not meters
%uncontrolled.float     name=DepthSetpoint_CM   initial = 0.0
%const.float            name=M_TO_CM            value = 100.0

%multiply
          enable = TRUE
          inputs = DepthSetpoint
          inputs = M_TO_CM
          output = DepthSetpoint_CM


%pid.control
                              error_range=depth_error_range
                              gains=pid_depth_gains
                              //setpoint=DepthSetpoint
                              setpoint=DepthSetpoint_CM
                              feedback=DepthIn_cm //AUV_Depth
                              output=DepthControl
                              behavior=ControlBehaviour

/////////////////////////////////////////////////////////////
/
// Altimeter PID Controller
%range.param
                              name=altimeter_error_range
                              min=-20000.0  // -200.0
                              max=20000.0  // 200.0

%pid.param
                              name=pid_altimeter_gains
                              proportional=50    // negative
values because
                              derivative=-60     // altimeter
base of reference
                              integral=2         // opposite
to depth sensor
                              max_integral=600


// Temporary fix because the altimeter component outputs in
centimeters not meters
// in order to keep the time stamp (which doesn't propagate
through multiply)
// the setpoint must be converted rather than the altimeter
feedback.
// The time stamp is required in order to use the time-based
integral
// and derivative components of the PID controller.

%uncontrolled.float     name=AltitudeSetpoint_CM   initial =
0.0

%multiply
          enable = TRUE
          inputs = AltitudeSetpoint
          inputs = M_TO_CM
          output = AltitudeSetpoint_CM

%pid.control

          error_range=altimeter_error_range
                              gains=pid_altimeter_gains
                              setpoint=AltitudeSetpoint_CM
                              feedback= AltitudeIn_cm   //
AUV_Altitude
                              output=AltitudeControl
                              behavior=ControlBehaviour

/////////////////////////////////////////////////////////////
//
//  Arbitration logic between Depth sensor and Altimeter
control
//  of vertical thrusters.  Uses the values in meters - no
time stamp
//  required.  Output = Vertical control which is in cm.
//  Takes BottomFollowingEnabled as an input

%copy
         enable         = TRUE
         input          = TelemBottomFollowingEnabled
         output         = AUV_FollowingBottom

%less_equal
         enable         = TRUE
         input1         = AUV_Altitude
         input2         = AltitudeSetpoint
         output         = BelowAltitudeSetpoint

%greater_equal
         enable         = TRUE
         input1         = AUV_Depth
         input2         = DepthSetpoint
         output         = BelowDepthSetpoint

%not
         enable         = TRUE
         input          = BelowAltitudeSetpoint
         output         = AboveAltitudeSetpoint

%not
         enable         = TRUE
         input          = BelowDepthSetpoint
         output         = AboveDepthSetpoint

%and
         enable         = TRUE
         inputs         = AboveDepthSetpoint
         inputs         = AboveAltitudeSetpoint
         inputs         = AUV_FollowingBottom
         output         = LogicTermOne

%or
         enable         = TRUE
         inputs         = LogicTermOne
         inputs         = BelowAltitudeSetpoint
         output         = AltitudeControlSelected

%not
         enable         = TRUE
         input          = AltitudeControlSelected
         output         = DepthControlSelected

%copy
         enable         = AltitudeControlSelected
         input          = AltitudeControl
         output         = VerticalControl

%copy
         enable         = DepthControlSelected
         input          = DepthControl
         output         = VerticalControl

/////////////////////////////////////////////////////////////
//
// Conversion from controller units to RPM units understood
by motors

%multiply
         inputs         = VerticalControl
         inputs         = DEPTH_TO_RPM
         output         = VerticalControl_RPM
         enable         = TRUE

%multiply
         inputs         = HeadingControl
//       inputs         = ZERO
         inputs         = HEADING_TO_RPM
         output         = HeadingControl_RPM
         enable         = TRUE

%multiply
         inputs         = VelocitySetpoint
```

```
        inputs          = VELOCITY_TO_RPM
        output          = VelocityControl_RPM
        enable          = TRUE


/////////////////////////////////////////////////////////////
//
// Combining heading and forward velocity control
// Motor Command Generation for the left and right thrusters

%multiply
        inputs          = HeadingControl_RPM
        inputs          = NEGATIVE_ONE
        output          = NegHeadingControl_RPM
        enable          = TRUE


%add
        inputs          = NegHeadingControl_RPM
        inputs          = VelocityControl_RPM
        output          = RightThrust
        enable          = TRUE


%multiply
        inputs          = RightThrust
        inputs          = NEGATIVE_ONE
        output          = RightThrust_RH_Prop
        enable          = TRUE


%add
        inputs          = HeadingControl_RPM
        inputs          = VelocityControl_RPM
        output          = LeftThrust
        enable          = TRUE


/////////////////////////////////////////////////////////////
///////
// ***** Thruster Motor Safety/Interlock *****
// Hardwires the motors to 0 when in IDLE mode, or when the
mission scripts
// set the mission interlock.  Prevents divers complaining
about missing
// limbs and all that rot.
// WARNING!!!   WARNING!!!   WARNING!!!   WARNING!!!
WARNING!!!
// WARNING: It is essential that at the beginning on each
MISSION script or
// ABORT script the ThrusterInterlock is set so that it is in
a known state.
// ThrusterInterlock could enter an undetermined state if the
AUV goes
// from one mission to another mission or ABORT mode without
moving through
// the IDLE or PILOT modes.



%copy
        enable          = IdleModeSelected
        input           = MOTORS_OFF
        output          = MotorMode
%copy
        enable          = PilotModeSelected
        input           = MOTORS_CONTROLLED
        output          = MotorMode
%copy
        enable          = AbortModeSelected
        input           = MOTORS_ABORT
        output          = MotorMode
%copy
        enable          = ExitModeSelected
        input           = MOTORS_OFF
        output          = MotorMode
%copy
        enable          = MissionModeSelected
        input           = MOTORS_OFF
        output          = MotorMode
//missions set their own thruster lock status after initial
setting


%selection
        name            = MotorModeSelection
        input           = MotorMode

%MotorModeSelection.selection.output
        choice          = MOTORS_CONTROLLED
        event           = MotorsControlled

%MotorModeSelection.selection.output
        choice          = MOTORS_OFF
        event           = MotorsOff

%MotorModeSelection.selection.output
        choice          = MOTORS_ABORT
        event           = MotorsAbort

/////////////////////////////////////////////////////////////
/
//right thruster
%copy
        enable          = MotorsOff
        input           = ZERO
        output          = AUV_RightThrustInterlocked
%copy
        enable          = MotorsControlled
        input           = RightThrust
        output          = AUV_RightThrustInterlocked
%copy
        enable          = MotorsAbort
        input           = MAX_THRUSTER_VELOCITY
        output          = AUV_RightThrustInterlocked

//left thruster
%copy
        enable          = MotorsOff
        input           = ZERO
        output          = AUV_LeftThrustInterlocked
%copy
        enable          = MotorsControlled
        input           = LeftThrust
        output          = AUV_LeftThrustInterlocked
%copy
        enable          = MotorsAbort
        input           = MAX_THRUSTER_VELOCITY
        output          = AUV_LeftThrustInterlocked

//vertical thruster
%copy
        enable          = MotorsOff
        input           = ZERO
        output          = AUV_VertThrustInterlocked
%copy
        enable          = MotorsControlled
        input           = VerticalControl_RPM
        output          = AUV_VertThrustInterlocked
%copy
        enable          = MotorsAbort
        input           = MAX_THRUSTER_VELOCITY
        output          = AUV_VertThrustInterlocked

/////////////////////////////////////////////////////////////
//////
// Thruster Motors Interface

%thruster_interface
        port                = ThrusterPort
        level               = HIGH
        max_motor_setpoint  = MAX_THRUSTER_VELOCITY
        poll_trigger        = SYS_OneSecondTrigger
        mode_trigger        = SYS_TwoSecondTrigger
        timer_value         = 80              //101
        adaptive_gain       = 10              //value =
xx/1000000
        inner_loop_gain     = 20
        watch_dog_value     = WatchDogValue //FALSE
        warning_level       = 2     // 4=Babbling 3=Debug
2=Normal 1=Sparse 0=None

        command_left_RPM                =
AUV_LeftThrustInterlocked
        actual_left_RPM                 = AUV_Left_RPM_Fb
        actual_left_PWM                 = AUV_Left_PWM_Fb
        left_adaptive_weight            = AUV_DebugFloat1
        l_motor_address                 = 1

        command_right_RPM               =
AUV_RightThrustInterlocked
        actual_right_RPM                = AUV_Right_RPM_Fb
        actual_right_PWM                = AUV_Right_PWM_Fb
        right_adaptive_weight           = AUV_DebugFloat2
        r_motor_address                 = 2

        command_vertical_left_RPM       =
AUV_VertThrustInterlocked
        actual_vertical_left_RPM        =
AUV_VertLeft_RPM_Fb
        actual_vertical_left_PWM        =
AUV_VertLeft_PWM_Fb
```

```
          vertical_left_adaptive_weight  = AUV_DebugFloat3
          vl_motor_address               = 3

          command_vertical_right_RPM     =
AUV_VertThrustInterlocked
          actual_vertical_right_RPM      =
AUV_VertRight_RPM_Fb
          actual_vertical_right_PWM      =
AUV_VertRight_PWM_Fb
          vertical_right_adaptive_weight = AUV_DebugFloat4
          vr_motor_address               = 4


!thruster.control
          port=ThrusterPort
          level=HIGH
          max_motor_setpoint=MAX_THRUSTER_VELOCITY
          trigger=SYS_FourHzTrigger
          TimerValue = 80                //101
          AdaptiveGain = 10              //value =
xx/1000000
          InnerLoopGain = 30
          watch_dog_value = WatchDogValue  //FALSE
//WatchDogValue

          command_left_RPM=AUV_LeftThrustInterlocked
          actual_left_RPM=AUV_Left_RPM_Fb
          actual_left_PWM=AUV_Left_PWM_Fb
          l_motor_address = 1

          command_right_RPM=AUV_RightThrustInterlocked
          actual_right_RPM=AUV_Right_RPM_Fb
          actual_right_PWM=AUV_Right_PWM_Fb
          r_motor_address = 2

          command_vertical_left_RPM=AUV_VertThrustInterlocked
          actual_vertical_left_RPM=AUV_VertLeft_RPM_Fb
          actual_vertical_left_PWM=AUV_VertLeft_PWM_Fb
          vl_motor_address = 3

          command_vertical_right_RPM =
AUV_VertThrustInterlocked
          actual_vertical_right_RPM=AUV_VertRight_RPM_Fb
          actual_vertical_right_PWM=AUV_VertRight_PWM_Fb
          vr_motor_address = 4
```

sapphire.csp

```
/*
$Log: sapphire.csp $
# Revision 1.4  1996/10/01  13:28:07  PURL
# Peter H: Added the digital output controls for the CTD pump
and camera lights
#
# Revision 1.3  1996/08/09  08:11:30  PURL
# Peter H: Added the leak sensor and low battery flags.
#
# Revision 1.2  1996/08/07  09:31:29  PURL
# Peter H: Added the Pitch and Roll A To D conversions and
converted the input to an angle +-20.
# Added the Battery Monitoring A To D conversion and
converted the input tage.
#
# Revision 1.1  1980/01/04  03:17:10  PURL
# Initial revision
#
*/
//*********************** SAPPHIRE.CSP
************************************

// This configuration file interfaces with the Sapphire
Analog I/O, Digital
// I/O board.
//
// The sensors currently attached to the Sapphire board are:
// 1) Tilt Sensor
// 2) Battery Monitor
// 3) Leak Sensor
//
// The actuators/outputs currently attached to the Sapphire
board are:
// 1) CTD Pump
// 2) Camera Lights
//

%copy
          enable = TRUE
          input = TelemEnable_CTD_Pump
          output = AUV_CTD_PumpOn

%copy
          enable = TRUE
          input = TelemEnableLights
          output = AUV_LightsOn

%Sapphire_Board
          level = HIGH
          base_address = 0x300
          analog_in_0 = PitchAToD
          analog_in_1 = RollAToD
          analog_in_2 = BatteryAToD
          analog_in_3 = DumbEvent
          analog_in_4 = DumbEvent
          analog_in_5 = DumbEvent
          analog_in_6 = DumbEvent
          analog_in_7 = DumbEvent
          analog_get_0 = SYS_FourHzTrigger
          analog_get_1 = SYS_FourHzTrigger
          analog_get_2 = SYS_OneSecondTrigger
          analog_get_3 = FALSE
          analog_get_4 = FALSE
          analog_get_5 = FALSE
          analog_get_6 = FALSE
          analog_get_7 = FALSE
          analog_out_0 = ZERO
          analog_out_1 = ZERO
          digital_in_0 = LeakSensorDigIn
          digital_in_1 = DumbEvent
          digital_in_2 = DumbEvent
          digital_in_3 = DumbEvent
          digital_get_0 = SYS_TwoHzTrigger
          digital_get_1 = FALSE
          digital_get_2 = FALSE
          digital_get_3 = FALSE
          digital_out_0 = AUV_CTD_PumpOn
          digital_out_1 = AUV_LightsOn
          digital_out_2 = ZERO
          digital_out_3 = ZERO

// Scale the Analog To Digital Inputs to meaningful units.
// The Pitch and Roll and converted to degrees.
// The Battery Voltage is converted to Volts.

%multiply2
          input1 = PitchAToD
          input2 = SAPPHIRE_TO_ANGLE
          output = AUV_Pitch
          behavior = ControlBehaviour

%multiply2
          input1 = RollAToD
          input2 = SAPPHIRE_TO_ANGLE
          output = AUV_Roll
          behavior = ControlBehaviour

%multiply2
          input1 = BatteryAToD
          input2 = SAPPHIRE_TO_BATTERY_VOLTAGE
          output = AUV_BatteryVoltage
          behavior = ControlBehaviour

// Convert the battery voltage input a boolean flag that
indicates a low
// battery voltage.
%less
          enable = TRUE
          input1 = AUV_BatteryVoltage
          input2 = LOW_BATTERY_VOLTAGE
          output = EnableLowBattery

%copy
          enable = EnableLowBattery
          input = TRUE
          output = AUV_LowBattery

// Convert the LeakSensor Digital Input to a high signal
indicating a leak.
// The %copy acts as a latch where once the leak sensor finds
a leak,
// AUV_Leaking remains TRUE even if the leak sensor no longer
signals a leak.
%not
          enable = TRUE
          input = LeakSensorDigIn
          output = EnableLeakAlarm

%copy
          enable = EnableLeakAlarm
          input = TRUE
          output = AUV_Leaking
```

exitscpt.csp

```
// Exit Script
/*
$Log: exitscpt.csp $
# Revision 1.3  1996/10/01  20:29:05  PURL
# Peter H: Turned off the Camera Lights and CTD Pump
#
# Revision 1.2  1996/08/08  14:14:44  PURL
# Peter H: Changed AUV_ExitProteus to ExitProteus and added
RCS comments.
#
*/
//
// This script traps on a ShutDown = TRUE enable and calls
the Exit sync
// signal when done.
//
//////////////////////////////////////////////////////////////
////////////////

//////////////////////////////////////////////////////////////
//////////////
%global.script
          enable     = ExitModeSelected
          layer      = HIGH
          repeat     = FALSE
          name       = ExitScript

//////////////////////////////////////////////////////////////
////////////////
// STEP 1: Turn OFF anything that should be off before the
vehicle exits
//              Proteus.

%int.set
          script = ExitScript
          output = MissionStep
          value = 1
          step=1 thread=1

%int.set
          script = ExitScript
          output = AUV_LightsOn
          value = FALSE
          step=1 thread=1

%int.set
          script = ExitScript
          output = AUV_CTD_PumpOn
          value = FALSE
          step=1 thread=1

%timed.wait
          script=ExitScript
          trigger=TimeTick
          interval=1000
          step=1 thread=1

//////////////////////////////////////////////////////////////
////////////////
// STEP 2:  Wait for motor velocities to drop to zero

%int.set
          script = ExitScript
          output = MissionStep
          value = 2
          step=2 thread=1

%int.confirm
          script = ExitScript
          input = AUV_Left_PWM_Fb
          value = 0
          step=2 thread=1

%int.confirm
          script = ExitScript
          input = AUV_Right_PWM_Fb
          value = 0
          step=2 thread=1

%int.confirm
          script = ExitScript
          input = AUV_VertLeft_PWM_Fb
          value = 0
          step=2 thread=1

%int.confirm
          script = ExitScript
          input = AUV_VertRight_PWM_Fb
          value = 0
          step=2 thread=1

// Time out in case the motors don't respond
%timed.wait
          script=ExitScript
          trigger=TimeTick
          interval=2000
          step=2 thread=2

//////////////////////////////////////////////////////////////
////////////////
// STEP 3: Clear the motor watch dog timer

%int.set
          script = ExitScript
          output = MissionStep
          value = 3
          step=3 thread=1

%int.set
          script = ExitScript
          output = WatchDogValue
          value = FALSE
          step=3 thread=1

//////////////////////////////////////////////////////////////
////////////////
// STEP 4: Next Step

%int.set
          script = ExitScript
          output = MissionStep
          value = 4
          step=4 thread=1

//////////////////////////////////////////////////////////////
////////////////
// Final Two steps:
// DO NOT CHANGE THESE STEPS UNLESS YOU ARE SURE YOU KNOW
WHAT YOU ARE DOING

%int.set
          script = ExitScript
          output = MissionStep
          value = 10
          step=10 thread=1

// Allow all the signals set in previous steps to move
through the system
%timed.wait
          script=ExitScript
          trigger=TimeTick
          interval=1000
          step=10 thread=1


%int.set
          script = ExitScript
          output = ExitProteus
          value = TRUE
          step=11 thread=1

%timed.wait
          script=ExitScript
          trigger=TimeTick
          interval=1000
          step=11 thread=1
```

purl_win.csp

```
/*
$Log: purl_win.csp $
# Revision 1.6  1996/09/18  20:37:43  PURL
# Peter H and Kevin M: Debugging the altimeter
#
# Revision 1.5  1996/08/15  08:32:45  PURL
# Peter H: ABORT condition testing and debuggin
#
# Revision 1.4  1996/08/09  08:33:57  PURL
# Peter H: Added debugging for leak sensor and low battery
flags.
#
# Revision 1.3  1996/08/07  10:20:45  COUSTEAU
# Exit menu selection now sets mode to EXIT
#
# Revision 1.2  1996/08/07  09:36:22  PURL
# Peter H: Added several display statements for displaying
the Sapphire inputs
#
# Revision 1.1  1980/01/04  03:21:36  PURL
# Initial revision
#
*/
//************************* PURL_WIN.CSP
**********************************
// PURL_WIN.CSP is the main window

%graphics.window
          enable=TRUE
          name=AUV_Control
          title="AUV_Control:Purl"
          display=screen
          top=1 left=0
          width=-1 height=-1
          //width=80 height=32
//        width=80 height=24
          border=TRUE

@AUV_Control.window.label

          style=label_style fill=0

@AUV_Control.window.number

          style=label_style

          fill=TRUE

%root.menu
                    display=screen
                    name=root
                    enable=TRUE

%submenu
                    menu=root
                    name=system
                    title="~System"

%submenu.int.output
                    submenu=system
                    title="~Exit"
                    output=TelemModeSelect
                    value=EXIT

%AUV_Control.window.number
                    x = COL_1
                    y = ROW_2
                    input = TelemModeSelect
                    string="TelemModeSelect:"
                    width=10

%AUV_Control.window.number
                    x = COL_1
                    y = ROW_3
                    input = AUV_Mode
                    string="AUV_Mode:"
                    width=10

%AUV_Control.window.number
                    x = COL_1
                    y = ROW_4
                    input = AUV_Depth
                    string="AUV_Depth:"
                    width=10
                    precision=2

%AUV_Control.window.number
                    x = COL_1
                    y = ROW_5
                    input = AUV_Altitude
                    string="AUV_Altitude:"
                    width=10
                    precision=2

%AUV_Control.window.number
                    x = COL_1
                    y = ROW_6
                    input = AUV_BatteryVoltage
                    string="AUV_BatteryVoltage:"
                    width=10
                    precision=2

%AUV_Control.window.number
                    x = COL_1
                    y = ROW_7
                    input = AUV_CompassHeading
                    string="AUV_CompassHeading:"
                    width=10
                    precision=2

%AUV_Control.window.number
                    x = COL_1
                    y = ROW_8
                    input = AUV_TelemCounter
                    string="AUV_TelemCounter:"
                    width=10

%AUV_Control.window.number
                    x = COL_1
                    y = ROW_9
                    input = AUV_Pitch
                    string="AUV_Pitch:"
                    width=10
                    precision=2

%AUV_Control.window.number
                    x = COL_1
                    y = ROW_10
                    input = AUV_Roll
                    string="AUV_Roll:"
                    width=10
                    precision=2

%AUV_Control.window.number
                    x = COL_1
                    y = ROW_11
                    input = AUV_Leaking
                    string="AUV_Leaking:"
                    width=10
```

payload.csp

```
/*
$Log: payload.csp $
# Revision 1.3  1997/02/08  18:11:42  PURL
# Peter H: Converted the configuration back to one that
utilises 5 serial ports
#
# Revision 1.2  1996/11/22  16:47:36  COUSTEAU
# Peter H: Changed AUV_Standard_Conductivity to
AUV_StandardConductivity
#
# Revision 1.1  1996/11/22  12:37:14  PURL
# Initial revision
#
*/
//    ******************** PAYLOAD.CSP
************************

!sbe19
        level = HIGH
        port = CTD_Port
        delimeter = 10
        escape = 27
        out_temperature = AUV_Temperature
        out_conductivity = AUV_Conductivity
        out_pressure = AUV_Pressure
        standard_conductivity = AUV_StandardConductivity
```

mission1.csp

```
/*
$Log: mission1.csp $
# Revision 1.7  1997/05/10  16:22:02  PURL
# Peter H: checked in to re-establish software on the new
hard drive
#
# Revision 1.6  1996/11/22  16:48:50  COUSTEAU
# Peter H: Added the CTD variables and control
#
# Revision 1.5  1996/09/20  16:09:58  PURL
# Peter H: Mission1 Bathemetry survey of Loon Lake
#
# Revision 1.4  1996/09/18  20:37:43  PURL
# Peter H and Kevin M: Loon Lake Mission for Sept 19,1996
#
# Revision 1.3  1996/08/30  20:38:49  PURL
# Peter H: Pool trials
#
# Revision 1.2  1996/08/24  11:11:08  COUSTEAU
# Peter H: Added new logging items, changed the enable and
log trigger
#
# Revision 1.1  1980/01/04  03:19:08  PURL
# Initial revision
#
*/
// Mission 1 : Loon Lake Trial Purl II
//
// The following is the mission script for MISSION1
//
//
//////////////////////////////////////////////////////////////
/////////////////

%define              name=OUT_HEADING1        value=345.0
//130.0
%define              name=BACK_HEADING1       value=155.0
//320.0
%define              name=TURN_HEADING1       value=75.0
//220.0
%const.float         name=OUT_HEADING1
value=OUT_HEADING1
%const.float         name=BACK_HEADING1
value=BACK_HEADING1
%const.float         name=TURN_HEADING1
value=TURN_HEADING1
%define              name=OUT_DEPTH1          value=10.0
%define              name=BACK_DEPTH1         value=10.0
%define              name=SURFACE1            value=0.5
%const.float         name=OUT_DEPTH1
value=OUT_DEPTH1
%const.float         name=BACK_DEPTH1
value=BACK_DEPTH1
%const.float         name=SURFACE1
value=SURFACE1
%define              name=LOW_ALT1            value=5.0
%define              name=HIGH_ALT1           value=5.0
%const.float         name=LOW_ALT1
value=LOW_ALT1
%const.float         name=HIGH_ALT1
value=HIGH_ALT1

//////////////////////////////////////////////////////////////
//////////////
%global.script
          enable=GoMission1
          layer=HIGH
          repeat=FALSE
          name=Mission1

//////////////////////////////////////////////////////////////
/////////////////
// STEP 1
// Wait for the serial link tether to be disconnected and the
vehicle
// placed into the water.  This whole procedure should not
take more
// than 60 seconds

%int.set
          script = Mission1
          output = MissionStep
          value = 1
          step=1 thread=1
%int.set
          script = Mission1
          output = M1LogEnable
          value = TRUE
          step=1 thread=1
%int.set
          script = Mission1
          output = AUV_CTD_PumpOn
          value = TRUE
          step=1 thread=1
%int.set
          script = Mission1
          output = WatchDogValue
          value = 250
          step=1 thread=1

%timed.wait
          script=Mission1
          trigger=TimeTick
          interval=60000
          step=1 thread=1

//////////////////////////////////////////////////////////////
////////////////
// STEP 2
// Initialize the heading setpoint to the desired heading
// Initialize the depth setpoint to the desired depth
// Initialize the altitude setpoint to the desired altitude
and set either
// bottom following mode or depth following mode
// Initialize the velocity setpoints to zero

%int.set
          script = Mission1
          output = MissionStep
          value = 2
          step=2 thread=1
%int.set
          script = Mission1
          output = MotorMode
          value = MOTORS_CONTROLLED
          step=2 thread=1
%float.set
          script = Mission1
          output = MissionHeadingSetpoint
          value = OUT_HEADING1
          step=2 thread=1
%int.set
          script = Mission1
          output = MissionVelocitySetpoint
          value = 0
          step=2 thread=1
%float.set
          script = Mission1
          output = MissionDepthSetpoint
          value = OUT_DEPTH1
          step=2 thread=1
%float.set
          script = Mission1
          output = MissionAltitudeSetpoint
          value = LOW_ALT1
          step=2 thread=1
%int.set
          script = Mission1
          output = AUV_FollowingBottom
          value = FALSE
          step=2 thread=1

%float.confirm
          script = Mission1
          input = AUV_CompassHeading
          value = OUT_HEADING1
          range = 3.0
          step=2 thread=1

!float.confirm
          script = Mission1
          input = AUV_Depth
          value = OUT_DEPTH1
          range = 0.2
          step=2 thread=1

!float.confirm
          script = Mission1
          input = AUV_Altitude
          value = LOW_ALT1
          range = 0.5
          step=2 thread=1

//////////////////////////////////////////////////////////////
////////////////
// STEP 3
// Go out for a predetermined time  (1800 sec)

%int.set
```

```
          script = Mission1
          output = MissionStep
          value = 3
          step=3 thread=1

%int.set
          script = Mission1
          output = MissionVelocitySetpoint
          value = 4000
          step=3 thread=1

%timed.wait
          script=Mission1
          trigger=TimeTick
          interval=1800000
          step=3 thread=1

/////////////////////////////////////////////////////////////
/////////////////
// STEP 4
// Start the turn to BACK_HEADING1 by going through
TURN_HEADING1
// Change the depth setpoint to BACK_DEPTH1
// Change the altitude setpoint
//

%int.set
          script = Mission1
          output = MissionStep
          value = 4
          step=4 thread=1

%float.set
          script = Mission1
          output = MissionHeadingSetpoint
          value = TURN_HEADING1
          step=4 thread=1

%float.set
          script = Mission1
          output = MissionDepthSetpoint
          value = BACK_DEPTH1
          step=4 thread=1

%float.set
          script = Mission1
          output = MissionAltitudeSetpoint
          value = LOW_ALT1
          step=4 thread=1

%int.set
          script = Mission1
          output = MissionVelocitySetpoint
//        value = 0
          value = -4000
          step=4 thread=1

%float.confirm
          script = Mission1
          input = AUV_CompassHeading
          value = TURN_HEADING1
          range = 30.0
          step=4 thread=1

!float.confirm
          script = Mission1
          input = AUV_Depth
          value = BACK_DEPTH1
          range = 0.2
          step=4 thread=1

!float.confirm
          script = Mission1
          input = AUV_Altitude
          value = LOW_ALT1
          range = 0.5
          step=4 thread=1

/////////////////////////////////////////////////////////////
/////////////////
// STEP 5
// Change the heading setpoint to BACK_HEADING1
// Wait until the AUV reaches the desired heading, then move
to the next step

%int.set
          script = Mission1
          output = MissionStep
          value = 5
          step=5 thread=1

%float.set
          script = Mission1
          output = MissionHeadingSetpoint
          value = BACK_HEADING1
          step=5 thread=1

%float.confirm
          script = Mission1
          input = AUV_CompassHeading
          value = BACK_HEADING1
          range = 3.0
          step=5 thread=1

!float.confirm
          script = Mission1
          input = AUV_Depth
          value = BACK_DEPTH1
          range = 0.2
          step=5 thread=1

!float.confirm
          script = Mission1
          input = AUV_Altitude
          value = LOW_ALT1
          range = 0.5
          step=5 thread=1

/////////////////////////////////////////////////////////////
/////////////////
// STEP 6
// Go back for a predetermined time  (1800 sec)

%int.set
          script = Mission1
          output = MissionStep
          value = 6
          step=6 thread=1

%int.set
          script = Mission1
          output = MissionVelocitySetpoint
          value = 4000
          step=6 thread=1

%timed.wait
          script=Mission1
          trigger=TimeTick
          interval=1800000
          step=6 thread=1

/////////////////////////////////////////////////////////////
////////////////
// STEP 7
// Stop the vehicle and return to the surface

%int.set
          script = Mission1
          output = MissionStep
          value = 7
          step=7 thread=1

%int.set
          script = Mission1
          output = MissionVelocitySetpoint
          value = 0
          step=7 thread=1

%float.set
          script = Mission1
          output = MissionDepthSetpoint
          value = SURFACE1
          step = 7 thread=1

%float.confirm
          script = Mission1
          input = AUV_Depth
          value = SURFACE1
          range = 0.3
          step = 7 thread = 1


/////////////////////////////////////////////////////////////
////////////////
// STEP 8
// Disable the data logging

%int.set
          script = Mission1
          output = MissionStep
          value = 8
          step=8 thread=1
```

```
%int.set
        script = Mission1
        output = M1LogEnable
        value = FALSE
        step=8 thread=1
%int.set
        script = Mission1
        output = AUV_CTD_PumpOn
        value = FALSE
        step=8 thread=1

%timed.wait
        script=Mission1
        trigger=TimeTick
        interval=1000
        step=8 thread=1

//////////////////////////////////////////////////////////////
////////////////
// STEP 9
// Put the AUV into IDLE mode

%int.set
        script = Mission1
        output = MissionStep
        value = 9
        step=9 thread=1

%int.set
        script = Mission1
        output = TelemModeSelect
        value = IDLE
        step=9 thread=1


// ******************** Data Logging For Mission 1
********************

// ********************** TIMESTAMP TIGGER
**************************
%copy
        enable = M1LogEnable
        input = SYS_TwoHzTrigger
        output = M1LogTrigger

// *********************** DATA LOG COMPONENT
*************************

%data.log
  name                  = data1
  path                  = "M1LOG.DAT"
  enable                = TRUE
  time_stamp_trigger    = M1LogTrigger
  level                 = LOW
  time_stamp_byte       = 254
  escape_byte           = 27
  byte_count_output     = M1LogSize

// *************************** ALIASES
*******************************

@data1.log.int.data     enable=TRUE trigger=M1LogTrigger
@data1.log.float.data   enable=TRUE trigger=M1LogTrigger
@data1.log.double.data  enable=TRUE trigger=M1LogTrigger

// ************************ LOG ITEM LIST
****************************

// Vehicle Sensor Feedbacks and Control Items
%data1.log.float.data   input = HeadingSetpoint
%data1.log.float.data   input = AUV_CompassHeading
%data1.log.int.data     input = HeadingControl_RPM

%data1.log.int.data     input = VelocitySetpoint

%data1.log.float.data   input = DepthSetpoint
%data1.log.float.data   input = AUV_Depth
%data1.log.int.data     input = VerticalControl_RPM

%data1.log.float.data   input = AltitudeSetpoint
%data1.log.float.data   input = AUV_Altitude
%data1.log.int.data     input = AUV_AltSignalStrength

%data1.log.float.data   input = AUV_Pitch
%data1.log.float.data   input = AUV_Roll

// Motor Feedbacks
%data1.log.int.data      input = AUV_Left_RPM_Fb
%data1.log.int.data      input = AUV_Right_RPM_Fb
%data1.log.int.data      input = AUV_VertLeft_RPM_Fb
%data1.log.int.data      input = AUV_VertRight_RPM_Fb
%data1.log.int.data      input = AUV_Left_PWM_Fb
%data1.log.int.data      input = AUV_Right_PWM_Fb
%data1.log.int.data      input = AUV_VertLeft_PWM_Fb
%data1.log.int.data      input = AUV_VertRight_PWM_Fb

// AUV Mode and Status Items
%data1.log.float.data    input = AUV_BatteryVoltage
%data1.log.int.data      input = AUV_Leaking
%data1.log.int.data      input = AUV_Mode
%data1.log.int.data      input = MissionStep
%data1.log.int.data      input = LocalStep

// Payload Items
%data1.log.float.data    input = AUV_Conductivity
%data1.log.float.data    input = AUV_Temperature
%data1.log.float.data    input = AUV_Pressure
%data1.log.int.data      input = AUV_StandardConductivity
%data1.log.int.data      input = AUV_CTD_PumpOn
```

mission2.csp

```
/*
$Log: mission2.csp $
# Revision 1.6  1997/05/10  16:22:02  PURL
# Peter H: checked in to re-establish software on the new
hard drive
#
# Revision 1.5  1996/11/22  16:48:50  COUSTEAU
# Peter H: Added the CTD variables and control
#
# Revision 1.4  1996/09/18  20:37:43  PURL
# Peter H and Kevin M: Loon Lake Mission for Sept 19,1996
#
# Revision 1.3  1996/08/30  20:38:49  PURL
# Peter H: Pool trials
#
# Revision 1.2  1996/08/24  11:11:08  COUSTEAU
# Peter H: Added new logging items, changed the enable and
log trigger
#
# Revision 1.1  1980/01/04  03:19:48  PURL
# Initial revision
#
*/
// Mission 2 : Loon Lake Trial
//
// The following is the mission script for MISSION2
//
// This is the long range mission in Loon Lake 900m out, 900m
back.
//
/////////////////////////////////////////////////////////////
/////////////////

%define                  name=OUT_HEADING2        value=330
%define                  name=BACK_HEADING2       value=173
%const.float             name=OUT_HEADING2
value=OUT_HEADING2
%const.float             name=BACK_HEADING2
value=BACK_HEADING2
%define                  name=SHALLOW2            value=0.25
%define                  name=DEEP2               value=1.5
%define                  name=SURFACE2            value=0.0
%const.float             name=SHALLOW2
value=SHALLOW2
%const.float             name=DEEP2               value=DEEP2
%const.float             name=SURFACE2
value=SURFACE2

/////////////////////////////////////////////////////////////
//////////////
%global.script
          enable=GoMission2
          layer=HIGH
          repeat=FALSE
          name=Mission2

/////////////////////////////////////////////////////////////
/////////////////
// STEP 1
// Wait for the serial link tether to be disconnected and the
vehicle
// placed into the water.  This whole procedure should not
take more
// than 45 seconds

%int.set
          script = Mission2
          output = MissionStep
          value = 1
          step=1 thread=1
%int.set
          script = Mission2
          output = M2LogEnable
          value = TRUE
          step=1 thread=1
%int.set
          script = Mission2
          output = WatchDogValue
          value = 250
          step=1 thread=1
%timed.wait
          script=Mission2
          trigger=TimeTick
          interval=45000
          step=1 thread=1

/////////////////////////////////////////////////////////////
/////////////////
// STEP 2
// Initialize the heading setpoint to the desired heading
// Initialize the depth setpoint to SHALLOW2
// Initialize the velocity setpoints to zero
// Wait until the AUV is on the desired heading and depth
then move to the
// next step

%int.set
          script = Mission2
          output = MissionStep
          value = 2
          step=2 thread=1
%int.set
          script = Mission2
          output = MotorMode
          value = MOTORS_CONTROLLED
          step=2 thread=1
%float.set
          script = Mission2
          output = MissionHeadingSetpoint
          value = OUT_HEADING2
          step=2 thread=1
%int.set
          script = Mission2
          output = MissionVelocitySetpoint
          value = 0
          step=2 thread=1
%float.set
          script = Mission2
          output = MissionDepthSetpoint
          value = SHALLOW2
          step=2 thread=1

%float.confirm
          script = Mission2
          input = AUV_CompassHeading
          value = OUT_HEADING2
          range = 3.0
          step=2 thread=1
%float.confirm
          script = Mission2
          input = AUV_Depth
          value = SHALLOW2
          range = 0.1
          step=2 thread=1

/////////////////////////////////////////////////////////////
///////////////
// STEP 3
// Start up the thrusters and travel at 90% speed for 1875
seconds which
// will take us approximately 750m at 0.4m/s or 656.25m at
0.35m/s.

%int.set
          script = Mission2
          output = MissionStep
          value = 3
          step=3 thread=1
%int.set
          script = Mission2
          output = MissionVelocitySetpoint
          value = 4000
          step=3 thread=1

%timed.wait
          script=Mission2
          trigger=TimeTick
          interval=60000
          step=3 thread=1

/////////////////////////////////////////////////////////////
/////////////////
// STEP 4
// Change the depth setpoint to SURFACE2 and go to the
surface for the
// turnaround and marking of the halfway point in the
mission.
//

%int.set
          script = Mission2
          output = MissionStep
          value = 4
          step=4 thread=1
%float.set
          script = Mission2
          output = MissionDepthSetpoint
          value = SURFACE2
          step=4 thread=1
```

```
%int.set
        script = Mission2
        output = MissionVelocitySetpoint
        value = 0
        step=4 thread=1
%float.confirm
        script = Mission2
        input = AUV_Depth
        value = SURFACE2
        range = 0.1
        step=4 thread=1

/////////////////////////////////////////////////////////////////
////////////////
// STEP 5
//
// PURL has now reached the surface and is going to turn
around and wait
// for a while before going to the deep part of the mission
and returning
// home to the start.
//

%int.set
        script = Mission2
        output = MissionStep
        value = 5
        step=5 thread=1
%float.set
        script = Mission2
        output = MissionHeadingSetpoint
        value = BACK_HEADING2
        step=5 thread=1

%float.confirm
        script = Mission2
        input = AUV_CompassHeading
        value = BACK_HEADING2
        range = 3.0
        step=5 thread=1
%timed.wait
        script=Mission2
        trigger=TimeTick
        interval=30000
        step=5 thread=1

/////////////////////////////////////////////////////////////////
////////////////
// STEP 6
// Go down to the deep depth for the return trip to the
start.

%int.set
        script = Mission2
        output = MissionStep
        value = 6
        step=6 thread=1
%float.set
        script = Mission2
        output = MissionDepthSetpoint
        value = DEEP2
        step=6 thread=1
%float.confirm
        script = Mission2
        input = AUV_Depth
        value = DEEP2
        range = 0.1
        step=6 thread=1

/////////////////////////////////////////////////////////////////
////////////////
// STEP 7
// Start up the thrusters and travel at 90% speed for 1875
seconds which
// will take us approximately 750m at 0.4m/s or 656.25m at
0.35m/s.

%int.set
        script = Mission2
        output = MissionStep
        value = 7
        step=7 thread=1
%int.set
        script = Mission2
        output = MissionVelocitySetpoint
        value = 4000
        step=7 thread=1

%timed.wait
        script=Mission2
        trigger=TimeTick
        interval=60000
        step=7 thread=1

/////////////////////////////////////////////////////////////////
////////////////
// STEP 8
// Change the depth setpoint to SURFACE2 and go to the
surface for the
// end of the mission.
//

%int.set
        script = Mission2
        output = MissionStep
        value = 8
        step=8 thread=1
%float.set
        script = Mission2
        output = MissionDepthSetpoint
        value = SURFACE2
        step=8 thread=1
%int.set
        script = Mission2
        output = MissionVelocitySetpoint
        value = 0
        step=8 thread=1
%float.confirm
        script = Mission2
        input = AUV_Depth
        value = SURFACE2
        range = 0.1
        step=8 thread=1

/////////////////////////////////////////////////////////////////
////////////////
// STEP 9
// Disable the data logging

%int.set
        script = Mission2
        output = MissionStep
        value = 9
        step=9 thread=1
%int.set
        script = Mission2
        output = M2LogEnable
        value = FALSE
        step=9 thread=1
%timed.wait
        script=Mission2
        trigger=TimeTick
        interval=1000
        step=9 thread=1


/////////////////////////////////////////////////////////////////
////////////////
// STEP 10
// Put the AUV into IDLE mode

%int.set
        script = Mission2
        output = MissionStep
        value = 10
        step=10 thread=1
%int.set
        script = Mission2
        output = TelemModeSelect
        value = IDLE
        step=10 thread=1



// ********************* Data Logging For Mission 2
********************

// ********************** TIMESTAMP TIGGER
**************************
%copy
        enable = M2LogEnable
        input = SYS_TwoHzTrigger
        output = M2LogTrigger

// *********************** DATA LOG COMPONENT
*************************

%data.log
  name                  = data2
  path                  = "M2LOG.DAT"
  enable                = TRUE
  time_stamp_trigger    = M2LogTrigger
```

```
  level                 = LOW
  time_stamp_byte       = 254
  escape_byte           = 27
  byte_count_output     = M2LogSize

// ***************************** ALIASES
********************************

@data2.log.int.data     enable=TRUE trigger=M2LogTrigger
@data2.log.float.data   enable=TRUE trigger=M2LogTrigger
@data2.log.double.data  enable=TRUE trigger=M2LogTrigger

// ************************** LOG ITEM LIST
****************************

// Vehicle Sensor Feedbacks and Control Items
%data2.log.float.data   input = HeadingSetpoint
%data2.log.float.data   input = AUV_CompassHeading
%data2.log.int.data     input = HeadingControl_RPM

%data2.log.int.data     input = VelocitySetpoint

%data2.log.float.data   input = DepthSetpoint
%data2.log.float.data   input = AUV_Depth
%data2.log.int.data     input = VerticalControl_RPM

%data2.log.float.data   input = AltitudeSetpoint
%data2.log.float.data   input = AUV_Altitude
%data2.log.int.data     input = AUV_AltSignalStrength

%data2.log.float.data   input = AUV_Pitch
%data2.log.float.data   input = AUV_Roll

// Motor Feedbacks
%data2.log.int.data      input = AUV_Left_RPM_Fb
%data2.log.int.data      input = AUV_Right_RPM_Fb
%data2.log.int.data      input = AUV_VertLeft_RPM_Fb
%data2.log.int.data      input = AUV_VertRight_RPM_Fb
%data2.log.int.data      input = AUV_Left_PWM_Fb
%data2.log.int.data      input = AUV_Right_PWM_Fb
%data2.log.int.data      input = AUV_VertLeft_PWM_Fb
%data2.log.int.data      input = AUV_VertRight_PWM_Fb

// AUV Mode and Status Items
%data2.log.float.data    input = AUV_BatteryVoltage
%data2.log.int.data      input = AUV_Leaking
%data2.log.int.data      input = AUV_Mode
%data2.log.int.data      input = MissionStep
%data2.log.int.data      input = LocalStep

// Payload Items
%data2.log.float.data    input = AUV_Conductivity
%data2.log.float.data    input = AUV_Temperature
%data2.log.float.data    input = AUV_Pressure
%data2.log.int.data      input = AUV_StandardConductivity
%data2.log.int.data      input = AUV_CTD_PumpOn
```

mission3.csp

```
/*
$Log: mission3.csp $
# Revision 1.7  1997/05/10  16:22:02  PURL
# Peter H: checked in to re-establish software on the new
hard drive
#
# Revision 1.6  1996/11/22  16:48:50  COUSTEAU
# Peter H: Added the CTD variables and control
#
# Revision 1.5  1996/09/20  16:10:40  PURL
# Peter H: Mission3 Sawtooth of Loon Lake
#
# Revision 1.4  1996/09/18  20:37:43  PURL
# Peter H and Kevin M: Loon Lake Mission for Sept 19,1996
#
# Revision 1.3  1996/08/30  20:38:49  PURL
# Peter H: Pool trials
#
# Revision 1.2  1996/08/24  11:11:08  COUSTEAU
# Peter H: Added new logging items, changed the enable and
log trigger
#
# Revision 1.1  1980/01/04  03:20:32  PURL
# Initial revision
#
*/
// Mission 3 : Loon Lake Trial
//
// The following is the mission script for Mission3
//
// A SAWTOOTH mission there and back again.
//
//////////////////////////////////////////////////////////////
/////////////////

%define                 name=OUT_HEADING3       value=338.0
//340.0 //130.0
%define                 name=BACK_HEADING3      value=155.0
//320.0
%define                 name=TURN_HEADING3      value=70.0
//220.0
%const.float            name=OUT_HEADING3
value=OUT_HEADING3
%const.float            name=BACK_HEADING3
value=BACK_HEADING3
%const.float            name=TURN_HEADING3
value=TURN_HEADING3
%define                 name=SHALLOW3           value=10.0
%define                 name=DEEP3              value=20.0
%define                 name=SURFACE3           value=0.5
%const.float            name=SHALLOW3
value=SHALLOW3
%const.float            name=DEEP3              value=DEEP3
%const.float            name=SURFACE3
value=SURFACE3
%define                 name=LOW_ALT3           value=5.0
%define                 name=HIGH_ALT3          value=5.0
%const.float            name=LOW_ALT3
value=LOW_ALT3
%const.float            name=HIGH_ALT3
value=HIGH_ALT3

//////////////////////////////////////////////////////////////
//////////////
%global.script
          enable=GoMission3
          layer=HIGH
          repeat=FALSE
          name=Mission3

//////////////////////////////////////////////////////////////
/////////////////
// STEP 1
// Wait for the serial link tether to be disconnected and the
vehicle
// placed into the water.  This whole procedure should not
take more
// than 60 seconds

%int.set
          script = Mission3
          output = MissionStep
          value = 1
          step=1 thread=1

%int.set
          script = Mission3
          output = M3LogEnable
          value = TRUE
          step=1 thread=1

%int.set
          script = Mission3
          output = AUV_CTD_PumpOn
          value = TRUE
          step=1 thread=1
%int.set
          script = Mission3
          output = WatchDogValue
          value = 250
          step=1 thread=1

%timed.wait
          script=Mission3
          trigger=TimeTick
          interval=60000
          step=1 thread=1

//////////////////////////////////////////////////////////////
/////////////////
// STEP 2
// Initialize the heading setpoint to the desired heading
// Initialize the depth setpoint to the desired depth
// Initialize the velocity setpoints to zero
// Wait until the AUV is on the desired heading and depth
then move to the
// next step

%int.set
          script = Mission3
          output = MissionStep
          value = 2
          step=2 thread=1

%int.set
          script = Mission3
          output = MotorMode
          value = MOTORS_CONTROLLED
          step=2 thread=1

%float.set
          script = Mission3
          output = MissionHeadingSetpoint
          value = OUT_HEADING3
          step=2 thread=1

%int.set
          script = Mission3
          output = MissionVelocitySetpoint
          value = 0
          step=2 thread=1

%float.set
          script = Mission3
          output = MissionDepthSetpoint
          value = SHALLOW3
          step=2 thread=1

%float.set
          script = Mission3
          output = MissionAltitudeSetpoint
          value = LOW_ALT3
          step=2 thread=1

%int.set
          script = Mission3
          output = AUV_FollowingBottom
          value = FALSE
          step=2 thread=1

%float.confirm
          script = Mission3
          input = AUV_CompassHeading
          value = OUT_HEADING3
          range = 3.0
          step=2 thread=1

!float.confirm
          script = Mission3
          input = AUV_Depth
          value = SHALLOW3
          range = 0.3
          step=2 thread=1

!float.confirm
          script = Mission3
          input = AUV_Altitude
          value = LOW_ALT3
          range = 0.5
```

```
        step=2 thread=1

/////////////////////////////////////////////////////////////////
/////////////////
// STEP 3
// Begin the sawtoothing and continue sawtoothing for 1800
seconds
// (approximately 1100m)
//

%int.set
        script = Mission3
        output = MissionStep
        value = 3
        step=3 thread=1

%int.set
        script = Mission3
        output = MissionVelocitySetpoint
        value = 4000
        step=3 thread=1

// **********************************
// There is a local script in step 3 that repeats the
sawtooth indefinitely
// **********************************

%timed.wait
        script=Mission3
        trigger=TimeTick
        interval=3000000
        step=3 thread=1

//-----------------------------------------------------------
-------------
// LOCAL SCRIPT IN STEP 3
// SAWTOOTH
//-----------------------------------------------------------
-------------

%local.script
        layer=HIGH
        repeat=TRUE
        name=OutSawtooth
        script=Mission3
        step=3 thread=2

// ----------- Step 1 Local

%int.set
        script = OutSawtooth
        output = LocalStep
        value = 1
        step=1 thread=1

%float.set
        script = OutSawtooth
        output = MissionDepthSetpoint
        value = DEEP3
        step=1 thread=1

%float.confirm
        script = OutSawtooth
        input = AUV_Depth
        value = DEEP3
        range = 0.3
        step=1 thread=1
%timed.wait
        script = OutSawtooth
        trigger=TimeTick
        interval=500
        step=1 thread=1

%float.confirm
        script = OutSawtooth
        input = AUV_Altitude
        value = LOW_ALT3
        range = 0.5
        step=1 thread=2
%timed.wait
        script = OutSawtooth
        trigger=TimeTick
        interval=500
        step=1 thread=2

// ----------- Step 2 Local

%int.set
        script = OutSawtooth
        output = LocalStep
        value = 2
        step=2 thread=1

%float.set
        script = OutSawtooth
        output = MissionDepthSetpoint
        value = SHALLOW3
        step=2 thread=1

%float.confirm
        script = OutSawtooth
        input = AUV_Depth
        value = SHALLOW3
        range = 0.3
        step=2 thread=1
%timed.wait
        script = OutSawtooth
        trigger=TimeTick
        interval=500
        step=2 thread=1


/////////////////////////////////////////////////////////////////
////////////////
// STEP 4
//
// PURL has reached the halfway point in the mission.  It is
going to
// turn around and then move to the next step.
//

%int.set
        script = Mission3
        output = MissionStep
        value = 4
        step=4 thread=1

%float.set
        script = Mission3
        output = MissionHeadingSetpoint
        value = TURN_HEADING3
        step=4 thread=1

%int.set
        script = Mission3
        output = MissionVelocitySetpoint
        value = -4000
        step=4 thread=1

%float.set
        script = Mission3
        output = MissionDepthSetpoint
        value = SURFACE3
        step=4 thread=1

%float.confirm
        script = Mission3
        input = AUV_CompassHeading
        value = TURN_HEADING3
        range = 30.0
        step=4 thread=1

/////////////////////////////////////////////////////////////////
///////////////
// STEP 5
// Complete the turn and then move to the next step

%int.set
        script = Mission3
        output = MissionStep
        value = 5
        step=5 thread=1

%float.set
        script = Mission3
        output = MissionHeadingSetpoint
        value = BACK_HEADING3
        step=5 thread=1

%float.confirm
        script = Mission3
        input = AUV_CompassHeading
        value = BACK_HEADING3
        range = 3.0
        step=5 thread=1


/////////////////////////////////////////////////////////////////
///////////////
// STEP 6
// Finish going to the surface.
```

```
%int.set
         script = Mission3
         output = MissionStep
         value = 6
         step=6 thread=1

%int.set
         script = Mission3
         output = MissionVelocitySetpoint
         value = 0
         step=6 thread=1

%float.confirm
         script = Mission3
         input = AUV_Depth
         value = SURFACE3
         range = 0.5
         step=6 thread=1


///////////////////////////////////////////////////////////////
///////////////
// STEP 7
// Set the velocity setpoint and begin sawtoothing again.
// Travel back for 1800 seconds at 4000 RPM shaft speed

%int.set
         script = Mission3
         output = MissionStep
         value = 7
         step=7 thread=1

%int.set
         script = Mission3
         output = MissionVelocitySetpoint
         value = 4000
         step=7 thread=1

// ************************************
// There is a local script here for sawtoothing on the way
back
// ************************************

%timed.wait
         script=Mission3
         trigger=TimeTick
         interval=3000000
         step=7 thread=1

//-----------------------------------------------------------
-------------
// LOCAL SCRIPT IN STEP 7
// BACKSAWTOOTH
//-----------------------------------------------------------
-------------

%local.script
         layer=HIGH
         repeat=TRUE
         name=BackSawtooth
         script=Mission3
         step=7 thread=2

// ----------- Step 1 Local

%int.set
         script = BackSawtooth
         output = LocalStep
         value = 1
         step=1 thread=1

%float.set
         script = BackSawtooth
         output = MissionDepthSetpoint
         value = SHALLOW3
         step=1 thread=1

%float.confirm
         script = BackSawtooth
         input = AUV_Depth
         value = SHALLOW3
         range = 0.2
         step=1 thread=1
%timed.wait
         script = BackSawtooth
         trigger=TimeTick
         interval=500
         step=1 thread=1

// ----------- Step 2 Local

%int.set
         script = BackSawtooth
         output = LocalStep
         value = 2
         step=2 thread=1

%float.set
         script = BackSawtooth
         output = MissionDepthSetpoint
         value = DEEP3
         step=2 thread=1

%float.confirm
         script = BackSawtooth
         input = AUV_Depth
         value = DEEP3
         range = 0.2
         step=2 thread=1
%timed.wait
         script = BackSawtooth
         trigger=TimeTick
         interval=500
         step=2 thread=1


%float.confirm
         script = BackSawtooth
         input = AUV_Altitude
         value = LOW_ALT3
         range = 0.5
         step=2 thread=2
%timed.wait
         script = BackSawtooth
         trigger=TimeTick
         interval=500
         step=2 thread=2


///////////////////////////////////////////////////////////////
///////////////
// STEP 8
// Come to the surface for the end of the mission

%int.set
         script = Mission3
         output = MissionStep
         value = 8
         step=8 thread=1

%int.set
         script = Mission3
         output = MissionVelocitySetpoint
         value = 0
         step=8 thread=1

%float.set
         script = Mission3
         output = MissionDepthSetpoint
         value = SURFACE3
         step=8 thread=1

%float.confirm
         script = Mission3
         input = AUV_Depth
         value = SURFACE3
         range = 0.3
         step=8 thread=1

///////////////////////////////////////////////////////////////
////////////////
// STEP 9
// Disable the data logging and stop PURL

%int.set
         script = Mission3
         output = MissionStep
         value = 9
         step=9 thread=1

%int.set
         script = Mission3
         output = M3LogEnable
         value = FALSE
         step=9 thread=1

%int.set
         script = Mission3
         output = AUV_CTD_PumpOn
         value = FALSE
```

```
          step=9 thread=1

%timed.wait
          script=Mission3
          trigger=TimeTick
          interval=1000
          step=9 thread=1

//////////////////////////////////////////////////////////////
////////////////
// STEP 10
// Put the AUV into IDLE mode

%int.set
          script = Mission3
          output = MissionStep
          value = 10
          step=10 thread=1

%int.set
          script = Mission3
          output = TelemModeSelect
          value = IDLE
          step=10 thread=1


// ******************** Data Logging For Mission 3
********************

// ********************** TIMESTAMP TIGGER
**************************
%copy
          enable = M3LogEnable
          input = SYS_TwoHzTrigger
          output = M3LogTrigger

// *********************** DATA LOG COMPONENT
*************************

%data.log
  name                    = data3
  path                    = "M3LOG.DAT"
  enable                  = TRUE
  time_stamp_trigger      = M3LogTrigger
  level                   = LOW
  time_stamp_byte         = 254
  escape_byte             = 27
  byte_count_output       = M3LogSize


// **************************** ALIASES
********************************

@data3.log.int.data     enable=TRUE trigger=M3LogTrigger
@data3.log.float.data   enable=TRUE trigger=M3LogTrigger
@data3.log.double.data  enable=TRUE trigger=M3LogTrigger

// ************************* LOG ITEM LIST
*****************************

// Vehicle Sensor Feedbacks and Control Items
%data3.log.float.data   input = HeadingSetpoint
%data3.log.float.data   input = AUV_CompassHeading
%data3.log.int.data     input = HeadingControl_RPM

%data3.log.int.data     input = VelocitySetpoint

%data3.log.float.data   input = DepthSetpoint
%data3.log.float.data   input = AUV_Depth
%data3.log.int.data     input = VerticalControl_RPM

%data3.log.float.data   input = AltitudeSetpoint
%data3.log.float.data   input = AUV_Altitude
%data3.log.int.data     input = AUV_AltSignalStrength

%data3.log.float.data   input = AUV_Pitch
%data3.log.float.data   input = AUV_Roll

// Motor Feedbacks
%data3.log.int.data      input = AUV_Left_RPM_Fb
%data3.log.int.data      input = AUV_Right_RPM_Fb
%data3.log.int.data      input = AUV_VertLeft_RPM_Fb
%data3.log.int.data      input = AUV_VertRight_RPM_Fb
%data3.log.int.data      input = AUV_Left_PWM_Fb
%data3.log.int.data      input = AUV_Right_PWM_Fb
%data3.log.int.data      input = AUV_VertLeft_PWM_Fb
%data3.log.int.data      input = AUV_VertRight_PWM_Fb

// AUV Mode and Status Items
%data3.log.float.data    input = AUV_BatteryVoltage
%data3.log.int.data      input = AUV_Leaking
%data3.log.int.data      input = AUV_Mode
%data3.log.int.data      input = MissionStep
%data3.log.int.data      input = LocalStep

// Payload Items
%data3.log.float.data    input = AUV_Conductivity
%data3.log.float.data    input = AUV_Temperature
%data3.log.float.data    input = AUV_Pressure
%data3.log.int.data      input = AUV_StandardConductivity
%data3.log.int.data      input = AUV_CTD_PumpOn
```

mission4.csp

```
/*
$Log: mission4.csp $
# Revision 1.8  1997/05/10  16:22:02  PURL
# Peter H: checked in to re-establish software on the new
hard drive
#
# Revision 1.7  1996/11/22  16:48:50  COUSTEAU
# Peter H: Added the CTD variables and control
#
# Revision 1.6  1996/09/27  15:24:10  PURL
# Peter H: Mission demonstrated for NSERC
#
# Revision 1.5  1996/09/20  16:11:17  PURL
# Peter H: Mission4 Bottom following Loon Lake
#
# Revision 1.4  1996/09/18  20:37:43  PURL
# Peter H and Kevin M: Loon Lake Mission for Sept 19,1996
#
# Revision 1.3  1996/08/30  20:38:49  PURL
# Peter H: Pool trials
#
# Revision 1.2  1996/08/24  11:11:08  COUSTEAU
# Peter H: Added new logging items, changed the enable and
log trigger
#
# Revision 1.1  1980/01/04  03:21:00  PURL
# Initial revision
#
*/
// Mission 4 : Loon Lake Trial
//
// The following is the mission script for MISSION4
//
//
////////////////////////////////////////////////////////////////
/////////////////

%define               name=OUT_HEADING4      value=130.0
%define               name=BACK_HEADING4     value=320.0
%define               name=TURN_HEADING4     value=220.0
%const.float          name=OUT_HEADING4
value=OUT_HEADING4
%const.float          name=BACK_HEADING4
value=BACK_HEADING4
%const.float          name=TURN_HEADING4
value=TURN_HEADING4
%define               name=SHALLOW4          value=0.5
%define               name=DEEP4             value=10.0
%define               name=SURFACE4          value=0.25
%const.float          name=SHALLOW4
value=SHALLOW4
%const.float          name=DEEP4             value=DEEP4
%const.float          name=SURFACE4
value=SURFACE4
%define               name=LOW_ALT4          value=5.0
%define               name=HIGH_ALT4         value=10.0
%const.float          name=LOW_ALT4
value=LOW_ALT4
%const.float          name=HIGH_ALT4
value=HIGH_ALT4

////////////////////////////////////////////////////////////////
//////////////
%global.script
        enable=GoMission4
        layer=HIGH
        repeat=FALSE
        name=Mission4

////////////////////////////////////////////////////////////////
/////////////////
// STEP 1
// Wait for the serial link tether to be disconnected and the
vehicle
// placed into the water.  This whole procedure should not
take more
// than 45 seconds

%int.set
        script = Mission4
        output = MissionStep
        value = 1
        step=1 thread=1

%int.set
        script = Mission4
        output = M4LogEnable
        value = TRUE
        step=1 thread=1
%int.set
        script = Mission4
        output = WatchDogValue
        value = 250
        step=1 thread=1

%timed.wait
        script=Mission4
        trigger=TimeTick
        interval=45000
        step=1 thread=1

////////////////////////////////////////////////////////////////
/////////////////
// STEP 2
// Initialize the heading setpoint to the desired heading
// Initialize the depth setpoint to SHALLOW4
// Initialize the altitude setpoint
// Initialize the velocity setpoints to zero

%int.set
        script = Mission4
        output = MissionStep
        value = 2
        step=2 thread=1

%int.set
        script = Mission4
        output = MotorMode
        value = MOTORS_CONTROLLED
        step=2 thread=1

%float.set
        script = Mission4
        output = MissionHeadingSetpoint
        value = OUT_HEADING4
        step=2 thread=1

%int.set
        script = Mission4
        output = MissionVelocitySetpoint
        value = 0
        step=2 thread=1

%float.set
        script = Mission4
        output = MissionDepthSetpoint
        value = DEEP4
        step=2 thread=1

%float.set
        script = Mission4
        output = MissionAltitudeSetpoint
        value = LOW_ALT4
        step=2 thread=1

%int.set
        script = Mission4
        output = AUV_FollowingBottom
        value = TRUE
        step=2 thread=1

%float.confirm
        script = Mission4
        input =AUV_CompassHeading
        value = OUT_HEADING4
        range = 3.0
        step=2 thread=1

!float.confirm
        script = Mission4
        input = AUV_Depth
        value = SHALLOW4
        range = 0.2
        step=2 thread=1

!float.confirm
        script = Mission4
        input = AUV_Altitude
        value = LOW_ALT4
        range = 0.5
        step=2 thread=1

////////////////////////////////////////////////////////////////
///////////////
// STEP 3
// Go out for 500 seconds

%int.set
        script = Mission4
```

```
          output = MissionStep
          value = 3
          step=3 thread=1

%int.set
          script = Mission4
          output = MissionVelocitySetpoint
          value = 4500
          step=3 thread=1

%timed.wait
          script=Mission4
          trigger=TimeTick
          interval=500000
          step=3 thread=1

/////////////////////////////////////////////////////////////////
/////////////////
// STEP 4
//
// Turn PURL around
//

%int.set
          script = Mission4
          output = MissionStep
          value = 4
          step=4 thread=1

%float.set
          script = Mission4
          output = MissionHeadingSetpoint
          value = TURN_HEADING4
          step=4 thread=1

%int.set
          script = Mission4
          output = MissionVelocitySetpoint
//        value = 0
          value = -4000
          step=4 thread=1

%float.set
          script = Mission4
          output = MissionDepthSetpoint
          value = SURFACE4
          step=4 thread=1

%float.confirm
          script = Mission4
          input = AUV_CompassHeading
          value = TURN_HEADING4
          range = 30.0
          step=4 thread=1

/////////////////////////////////////////////////////////////////
/////////////////
// STEP 5
//
// Finish turning PURL around and move to next step
//

%int.set
          script = Mission4
          output = MissionStep
          value = 5
          step=5 thread=1

%float.set
          script = Mission4
          output = MissionHeadingSetpoint
          value = BACK_HEADING4
          step=5 thread=1

%float.confirm
          script = Mission4
          input = AUV_CompassHeading
          value = BACK_HEADING4
          range = 3.0
          step=5 thread=1

%float.confirm
          script = Mission4
          input = AUV_Depth
          value = SURFACE4
          range = 0.3
          step=5 thread=1

/////////////////////////////////////////////////////////////////
////////////////
// STEP 6
// Go back for 500 seconds

%int.set
          script = Mission4
          output = MissionStep
          value = 6
          step=6 thread=1

%float.set
          script = Mission4
          output = MissionDepthSetpoint
          value = DEEP4
          step=6 thread=1

%float.set
          script = Mission4
          output = MissionVelocitySetpoint
          value = 4500
          step=6 thread=1

%timed.wait
          script=Mission4
          trigger=TimeTick
          interval=500000
          step=6 thread=1

/////////////////////////////////////////////////////////////////
////////////////
// STEP 7
// Return PURL to the surface
//

%int.set
          script = Mission4
          output = MissionStep
          value = 7
          step=7 thread=1

%int.set
          script = Mission4
          output = MissionVelocitySetpoint
          value = 0
          step=7 thread=1

%float.set
          script = Mission4
          output = MissionDepthSetpoint
          value = SURFACE4
          step=7 thread=1

%float.confirm
          script = Mission4
          input = AUV_Depth
          value = SURFACE4
          range = 0.2
          step =7 thread = 1

/////////////////////////////////////////////////////////////////
////////////////
// STEP 8
// Disable the data logging

%int.set
          script = Mission4
          output = MissionStep
          value = 8
          step=8 thread=1

%int.set
          script = Mission4
          output = M4LogEnable
          value = FALSE
          step=8 thread=1

%timed.wait
          script=Mission4
          trigger=TimeTick
          interval=1000
          step=8 thread=1

/////////////////////////////////////////////////////////////////
////////////////
// STEP 9
// Put the AUV into IDLE mode

%int.set
          script = Mission4
```

```
          output = MissionStep
          value = 9
          step=9 thread=1

%int.set
          script = Mission4
          output = TelemModeSelect
          value = IDLE
          step=9 thread=1


// ********************* Data Logging For Mission 4
*******************

// ********************** TIMESTAMP TIGGER
**************************
%copy
          enable = M4LogEnable
          input = SYS_TwoHzTrigger
          output = M4LogTrigger

// *********************** DATA LOG COMPONENT
*************************

%data.log
  name                  = data4
  path                  = "M4LOG.DAT"
  enable                = TRUE
  time_stamp_trigger    = M4LogTrigger
  level                 = LOW
  time_stamp_byte       = 254
  escape_byte           = 27
  byte_count_output     = M4LogSize



// **************************** ALIASES
********************************

@data4.log.int.data     enable=TRUE trigger=M4LogTrigger
@data4.log.float.data   enable=TRUE trigger=M4LogTrigger
@data4.log.double.data  enable=TRUE trigger=M4LogTrigger

// ************************* LOG ITEM LIST
*****************************

// Vehicle Sensor Feedbacks and Control Items
%data4.log.float.data   input = HeadingSetpoint
%data4.log.float.data   input = AUV_CompassHeading
%data4.log.int.data     input = HeadingControl_RPM

%data4.log.int.data     input = VelocitySetpoint

%data4.log.float.data   input = DepthSetpoint
%data4.log.float.data   input = AUV_Depth
%data4.log.int.data     input = VerticalControl_RPM

%data4.log.float.data   input = AltitudeSetpoint
%data4.log.float.data   input = AUV_Altitude
%data4.log.int.data     input = AUV_AltSignalStrength

%data4.log.float.data   input = AUV_Pitch
%data4.log.float.data   input = AUV_Roll

// Motor Feedbacks
%data4.log.int.data      input = AUV_Left_RPM_Fb
%data4.log.int.data      input = AUV_Right_RPM_Fb
%data4.log.int.data      input = AUV_VertLeft_RPM_Fb
%data4.log.int.data      input = AUV_VertRight_RPM_Fb
%data4.log.int.data      input = AUV_Left_PWM_Fb
%data4.log.int.data      input = AUV_Right_PWM_Fb
%data4.log.int.data      input = AUV_VertLeft_PWM_Fb
%data4.log.int.data      input = AUV_VertRight_PWM_Fb

// AUV Mode and Status Items
%data4.log.float.data    input = AUV_BatteryVoltage
%data4.log.int.data      input = AUV_Leaking
%data4.log.int.data      input = AUV_Mode
%data4.log.int.data      input = MissionStep
%data4.log.int.data      input = LocalStep

// Payload Items
%data4.log.float.data    input = AUV_Conductivity
%data4.log.float.data    input = AUV_Temperature
%data4.log.float.data    input = AUV_Pressure
%data4.log.int.data      input = AUV_StandardConductivity
%data4.log.int.data      input = AUV_CTD_PumpOn
```

mission5.csp

```
/*
$Log: mission5.csp $
*/
// Mission 5 : Simple test mission Purl II
//PURL will dive to 1 meter, travel on course 200 for 60 sec.
and surface
// The following is the mission script for MISSION5
//
//
///////////////////////////////////////////////////////////////
////////////////

%define                 name=HEADING5          value=154.0
//130.0
%define                 name=BACK_HEADING5     value=180.0
//320.0
%define                 name=TURN_HEADING5     value=180.0
//220.0
%const.float            name=HEADING5
value=HEADING5
%const.float            name=BACK_HEADING5
value=BACK_HEADING5
%const.float            name=TURN_HEADING5
value=TURN_HEADING5
%define                 name=DEPTH5            value=0.5
%define                 name=BACK_DEPTH5       value=0.5
%define                 name=SURFACE5         value=0.00
%const.float            name=DEPTH5        value=DEPTH5
%const.float            name=BACK_DEPTH5
value=BACK_DEPTH5
%const.float            name=SURFACE5         value=SURFACE5
%define                 name=LOW_ALT5          value=0.0
%define                 name=HIGH_ALT5         value=5.0
%const.float            name=LOW_ALT5
value=LOW_ALT5
%const.float            name=HIGH_ALT5
value=HIGH_ALT5

///////////////////////////////////////////////////////////////
//////////////
%global.script
          enable=GoMission5
          layer=HIGH
          repeat=FALSE
          name=Mission5

///////////////////////////////////////////////////////////////
/////////////////
// STEP 1
// Wait for the serial link tether to be disconnected and the
vehicle
// placed into the water.  This whole procedure should not
take more
// than 10 seconds

%int.set
          script = Mission5
          output = MissionStep
          value = 1
          step=1 thread=1
%int.set
          script = Mission5
          output = M5LogEnable
          value = FALSE
          step=1 thread=1
%int.set
          script = Mission5
          output = AUV_CTD_PumpOn
          value = FALSE
          step=1 thread=1
%int.set
          script = Mission5
          output = WatchDogValue
          value = 250
          step=1 thread=1

%timed.wait
          script=Mission5
          trigger=TimeTick
          interval=10000
          step=1 thread=1

///////////////////////////////////////////////////////////////
/////////////////
// STEP 2
// Initialize the heading setpoint to the desired heading
// Initialize the depth setpoint to the desired depth
// Initialize the altitude setpoint to the desired altitude
and set either
// bottom following mode or depth following mode
// Initialize the velocity setpoints to zero

%int.set
          script = Mission5
          output = MissionStep
          value = 2
          step=2 thread=1

// Enable the motors
%int.set
          script = Mission5
          output = MotorMode
          value = MOTORS_CONTROLLED
          step=2 thread=1

// Set the intitial heading, depth, velocity and altitude
setpoints.
// Set Bottom Following to TRUE for altitude following, and
FALSE for
// depth following.
// These parameters should be initialised at the beginning of
every mission
%float.set
          script = Mission5
          output = MissionHeadingSetpoint
          value = HEADING5
          step=2 thread=1
%int.set
          script = Mission5
          output = MissionVelocitySetpoint
          value = 0
          step=2 thread=1
%float.set
          script = Mission5
          output = MissionDepthSetpoint
          value = DEPTH5
          step=2 thread=1
%float.set
          script = Mission5
          output = MissionAltitudeSetpoint
          value = LOW_ALT5
          step=2 thread=1
%int.set
          script = Mission5
          output = AUV_FollowingBottom
          value = FALSE
          step=2 thread=1

// if desired heading and depth have been reached then move
to the
// next step
%float.confirm
          script = Mission5
          input = AUV_CompassHeading
          value = HEADING5
          range = 3.0
          step=2 thread=1

%float.confirm
          script = Mission5
          input = AUV_Depth
          value = DEPTH5
          range = 0.2
          step=2 thread=1

// if desired heading and altitude have been reached then
move to the
// next step
!float.confirm
          script = Mission5
          input = AUV_CompassHeading
          value = HEADING5
          range = 3.0
          step=2 thread=2

!float.confirm
          script = Mission5
          input = AUV_Altitude
          value = LOW_ALT5
          range = 0.5
          step=2 thread=2

///////////////////////////////////////////////////////////////
////////////////
// STEP 3
// Go out for a predetermined time

%int.set
```

```
        script = Mission5
        output = MissionStep
        value = 3
        step=3 thread=1

!int.set
        script = Mission5
        output = MissionVelocitySetpoint
        value = 4000
        step=3 thread=1

!timed.wait
        script=Mission5
        trigger=TimeTick
        interval=30000
        step=3 thread=1

/////////////////////////////////////////////////////////////
////////////////
// STEP 4
//
//

!int.set
        script = Mission5
        output = MissionStep
        value = 4
        step=4 thread=1

!float.set
        script = Mission5
        output = MissionHeadingSetpoint
        value = TURN_HEADING5
        step=4 thread=1

!float.set
        script = Mission5
        output = MissionDepthSetpoint
        value = BACK_DEPTH5
        step=4 thread=1

!float.set
        script = Mission5
        output = MissionAltitudeSetpoint
        value = LOW_ALT5
        step=4 thread=1

!int.set
        script = Mission5
        output = MissionVelocitySetpoint
//      value = 0
        value = -4000
        step=4 thread=1

!float.confirm
        script = Mission5
        input = AUV_CompassHeading
        value = TURN_HEADING5
        range = 30.0
        step=4 thread=1

!float.confirm
        script = Mission5
        input = AUV_Depth
        value = BACK_DEPTH5
        range = 0.2
        step=4 thread=1

!float.confirm
        script = Mission5
        input = AUV_Altitude
        value = LOW_ALT5
        range = 0.5
        step=4 thread=1

/////////////////////////////////////////////////////////////
////////////////
// STEP 5
// Change the heading setpoint to BACK_HEADING5
// Wait until the AUV reaches the desired heading, then move
to the next step

!int.set
        script = Mission5
        output = MissionStep
        value = 5
        step=5 thread=1

!float.set
        script = Mission5
        output = MissionHeadingSetpoint
        value = BACK_HEADING5
        step=5 thread=1

!float.confirm
        script = Mission5
        input = AUV_CompassHeading
        value = BACK_HEADING5
        range = 3.0
        step=5 thread=1

!float.confirm
        script = Mission5
        input = AUV_Depth
        value = BACK_DEPTH5
        range = 0.2
        step=5 thread=1

!float.confirm
        script = Mission5
        input = AUV_Altitude
        value = LOW_ALT5
        range = 0.5
        step=5 thread=1

/////////////////////////////////////////////////////////////
////////////////
// STEP 6
// Go back for a predetermined time  (1800 sec)

!int.set
        script = Mission5
        output = MissionStep
        value = 6
        step=6 thread=1

!int.set
        script = Mission5
        output = MissionVelocitySetpoint
        value = 4000
        step=6 thread=1

!timed.wait
        script=Mission5
        trigger=TimeTick
        interval=1800000
        step=6 thread=1

/////////////////////////////////////////////////////////////
////////////////
// STEP 7
// Stop the vehicle and return to the surface

!int.set
        script = Mission5
        output = MissionStep
        value = 7
        step=7 thread=1

!int.set
        script = Mission5
        output = MissionVelocitySetpoint
        value = 0
        step=7 thread=1

!float.set
        script = Mission5
        output = MissionDepthSetpoint
        value = SURFACE5
        step = 7 thread=1

!float.confirm
        script = Mission5
        input = AUV_Depth
        value = SURFACE5
        range = 0.3
        step = 7 thread = 1


/////////////////////////////////////////////////////////////
////////////////
// STEP 8
// Disable the data logging

!int.set
        script = Mission5
        output = MissionStep
        value = 8
        step=8 thread=1

!int.set
        script = Mission5
```

```
          output = M1LogEnable
          value = FALSE
          step=8 thread=1
%int.set
          script = Mission5
          output = AUV_CTD_PumpOn
          value = FALSE
          step=8 thread=1
%timed.wait
          script=Mission5
          trigger=TimeTick
          interval=1000
          step=8 thread=1

/////////////////////////////////////////////////////////////
////////////////
// STEP 9
// Put the AUV into IDLE mode

%int.set
          script = Mission5
          output = MissionStep
          value = 9
          step=9 thread=1

%int.set
          script = Mission5
          output = TelemModeSelect
          value = IDLE
          step=9 thread=1


// ******************** Data Logging For Mission 1
********************

// ********************** TIMESTAMP TIGGER
**************************
%copy
          enable = M5LogEnable
          input = SYS_TwoHzTrigger
          output = M5LogTrigger

// *********************** DATA LOG COMPONENT
*************************


%data.log
  name                  = data5
  path                  = "M5LOG.DAT"
  enable                = TRUE
  time_stamp_trigger    = M5LogTrigger
  level                 = LOW
  time_stamp_byte       = 254
  escape_byte           = 27
  byte_count_output     = M5LogSize

// **************************** ALIASES
*******************************

@data5.log.int.data     enable=TRUE trigger=M5LogTrigger
@data5.log.float.data   enable=TRUE trigger=M5LogTrigger
@data5.log.double.data  enable=TRUE trigger=M5LogTrigger

// ************************* LOG ITEM LIST
*****************************

// Vehicle Sensor Feedbacks and Control Items
%data5.log.float.data   input = HeadingSetpoint
%data5.log.float.data   input = AUV_CompassHeading
%data5.log.int.data     input = HeadingControl_RPM

%data5.log.int.data     input = VelocitySetpoint

%data5.log.float.data   input = DepthSetpoint
%data5.log.float.data   input = AUV_Depth
%data5.log.int.data     input = VerticalControl_RPM

%data5.log.float.data   input = AltitudeSetpoint
%data5.log.float.data   input = AUV_Altitude
%data5.log.int.data     input = AUV_AltSignalStrength

%data5.log.float.data   input = AUV_Pitch
%data5.log.float.data   input = AUV_Roll

// Motor Feedbacks
%data5.log.int.data      input = AUV_Left_RPM_Fb
%data5.log.int.data      input = AUV_Right_RPM_Fb
%data5.log.int.data      input = AUV_VertLeft_RPM_Fb
%data5.log.int.data      input = AUV_VertRight_RPM_Fb
%data5.log.int.data      input = AUV_Left_PWM_Fb
%data5.log.int.data      input = AUV_Right_PWM_Fb
%data5.log.int.data      input = AUV_VertLeft_PWM_Fb
%data5.log.int.data      input = AUV_VertRight_PWM_Fb

// AUV Mode and Status Items
%data5.log.float.data    input = AUV_BatteryVoltage
%data5.log.int.data      input = AUV_Leaking
%data5.log.int.data      input = AUV_Mode
%data5.log.int.data      input = MissionStep
%data5.log.int.data      input = LocalStep

// Payload Items
%data5.log.float.data    input = AUV_Conductivity
%data5.log.float.data    input = AUV_Temperature
%data5.log.float.data    input = AUV_Pressure
%data5.log.int.data      input = AUV_StandardConductivity
%data5.log.int.data      input = AUV_CTD_PumpOn
```

mission6.csp

```
/*
$Log: mission6 . csp $
*/
// Mission 6 : CTD Collection
//Purpose: to collect CTD data over the sill at Loon Lake to
determine if
//there is is a upwelling of coldwater from the main lake
into the arm.
// There is a thermistor chain about 100 meters in front of
the arm in the
// main lake. Purl has to start near the chain, dive and
travel on a heading
// towards the sill maintaining a two meter bottom following
altitude. At a
//specific time over the sill a course correction has to be
made to allow
//PURL to continue down the arm as far as possible at the 2
metre altitude.
// At the end of the run time PURL stops and surfaces.

///////////////////////////////////////////////////////////////
////////////////

%define             name=HEADING6          value=180.0
%define             name=NEW_HEADING6      value=155.0
%define             name=TURN_HEADING6     value=155.0
%const.float        name=HEADING6
value=HEADING6
%const.float        name=NEW_HEADING6
value=NEW_HEADING6
%const.float        name=TURN_HEADING6
value=TURN_HEADING6
%define             name=DEPTH6            value=10.0
%define             name=BACK_DEPTH6       value=10.0
%define             name=SURFACE6          value=0.0
%const.float        name=DEPTH6            value=DEPTH6
%const.float        name=BACK_DEPTH6
value=BACK_DEPTH6
%const.float        name=SURFACE6
value=SURFACE6
%define             name=LOW_ALT6          value=1.25
%define             name=HIGH_ALT6         value=3.00
%const.float        name=LOW_ALT6
value=LOW_ALT6
%const.float        name=HIGH_ALT6
value=HIGH_ALT6

///////////////////////////////////////////////////////////////
//////////////
%global.script
        enable=GoMission6
        layer=HIGH
        repeat=FALSE
        name=Mission6

///////////////////////////////////////////////////////////////
/////////////////
// STEP 1
// Wait for the serial link tether to be disconnected and the
vehicle
// placed into the water.  This whole procedure should not
take more
// than 60 seconds

%int.set
        script = Mission6
        output = MissionStep
        value = 1
        step=1 thread=1
%int.set
        script = Mission6
        output = M6LogEnable
        value = TRUE
        step=1 thread=1
%int.set
        script = Mission6
        output = AUV_CTD_PumpOn
        value = TRUE
        step=1 thread=1
%int.set
        script = Mission6
        output = WatchDogValue
        value = 250
        step=1 thread=1

%timed.wait
        script=Mission6
        trigger=TimeTick
        interval=60000
        step=1 thread=1

///////////////////////////////////////////////////////////////
/////////////////
// STEP 2
// Initialize the heading setpoint to the desired heading
// Initialize the depth setpoint to the desired depth
// Initialize the altitude setpoint to the desired altitude
and set either
// bottom following mode or depth following mode
// Initialize the velocity setpoints to zero

%int.set
        script = Mission6
        output = MissionStep
        value = 2
        step=2 thread=1

// Enable the motors
%int.set
        script = Mission6
        output = MotorMode
        value = MOTORS_CONTROLLED
        step=2 thread=1

// Set the intitial heading, depth, velocity and altitude
setpoints.
// Set Bottom Following to TRUE for altitude following, and
FALSE for
// depth following.
// These parameters should be initialised at the beginning of
every mission
%float.set
        script = Mission6
        output = MissionHeadingSetpoint
        value = HEADING6
        step=2 thread=1
%int.set
        script = Mission6
        output = MissionVelocitySetpoint
        value =0
        step=2 thread=1
%float.set
        script = Mission6
        output = MissionDepthSetpoint
        value = DEPTH6
        step=2 thread=1
%float.set
        script = Mission6
        output = MissionAltitudeSetpoint
        value = LOW_ALT6
        step=2 thread=1
%int.set
        script = Mission6
        output = AUV_FollowingBottom
        value = TRUE
        step=2 thread=1

// if desired heading and depth have been reached then move
to the
// next step
%float.confirm
        script = Mission6
        input = AUV_CompassHeading
        value = HEADING6
        range = 3.0
        step=2 thread=1

%float.confirm
        script = Mission6
        input = AUV_Depth
        value = DEPTH6
        range = 0.2
        step=2 thread=1

// if desired heading and altitude have been reached then
move to the
// next step
%float.confirm
        script = Mission6
        input = AUV_CompassHeading
        value = HEADING6
        range = 3.0
        step=2 thread=2

%float.confirm
        script = Mission6
        input = AUV_Altitude
        value = LOW_ALT6
        range = 0.5
```

```
          step=2 thread=2

//////////////////////////////////////////////////////////////
////////////////
// STEP 3
// Go out for a predetermined time following the bottom at
1.25 meters alt

%int.set
          script = Mission6
          output = MissionStep
          value = 3
          step=3 thread=1

%int.set
          script = Mission6
          output = MissionVelocitySetpoint
          value = 4000
          step=3 thread=1

%timed.wait
          script=Mission6
          trigger=TimeTick
          interval=180000
          step=3 thread=1

//////////////////////////////////////////////////////////////
////////////////
// STEP 4
// After the predetermined time at HEADING6, change heading
to TURN_HEADING6

%int.set
          script = Mission6
          output = MissionStep
          value = 4
          step=4 thread=1

%float.set
          script = Mission6
          output = MissionHeadingSetpoint
          value = TURN_HEADING6
          step=4 thread=1

%float.set
          script = Mission6
          output = MissionDepthSetpoint
          value = BACK_DEPTH6
          step=4 thread=1

%float.set
          script = Mission6
          output = MissionAltitudeSetpoint
          value = LOW_ALT6
          step=4 thread=1

%int.set
          script = Mission6
          output = MissionVelocitySetpoint
//        value = 0
          value =-4000
          step=4 thread=1

%float.confirm
          script = Mission6
          input = AUV_CompassHeading
          value = TURN_HEADING6
          range = 5.0
          step=4 thread=1

!float.confirm
          script = Mission6
          input = AUV_Depth
          value = DEPTH6
          range = 0.2
          step=4 thread=1

!float.confirm
          script = Mission6
          input = AUV_Altitude
          value = LOW_ALT6
          range = 0.5
          step=4 thread=1

//////////////////////////////////////////////////////////////
////////////////
// STEP 5
// Change the heading setpoint to NEW_HEADING6
// Wait until the AUV reaches the desired heading, then move
to the next step

%int.set
          script = Mission6
          output = MissionStep
          value = 5
          step=5 thread=1

%float.set
          script = Mission6
          output = MissionHeadingSetpoint
          value = NEW_HEADING6
          step=5 thread=1

%float.confirm
          script = Mission6
          input = AUV_CompassHeading
          value = NEW_HEADING6
          range = 5.0
          step=5 thread=1

!float.confirm
          script = Mission6
          input = AUV_Depth
          value = BACK_DEPTH6
          range = 0.2
          step=5 thread=1

!float.confirm
          script = Mission6
          input = AUV_Altitude
          value = LOW_ALT6
          range = 0.5
          step=5 thread=1

//////////////////////////////////////////////////////////////
////////////////
// STEP 6
// Go on new heading for a predetermined time 10 min or 600
sec or 600000 ms

%int.set
          script = Mission6
          output = MissionStep
          value = 6
          step=6 thread=1

%int.set
          script = Mission6
          output = MissionVelocitySetpoint
          value = 4000
          step=6 thread=1

%timed.wait
          script=Mission6
          trigger=TimeTick
          interval=600000
          step=6 thread=1

//////////////////////////////////////////////////////////////
///////////////
// STEP 7
// Stop the vehicle and return to the surface

%int.set
          script = Mission6
          output = MissionStep
          value = 7
          step=7 thread=1

%int.set
          script = Mission6
          output = MissionVelocitySetpoint
          value = 0
          step=7 thread=1

%float.set
          script = Mission6
          output = MissionDepthSetpoint
          value = SURFACE6
          step = 7 thread=1

%float.confirm
          script = Mission6
          input = AUV_Depth
          value = SURFACE6
          range = 0.3
          step = 7 thread = 1


//////////////////////////////////////////////////////////////
///////////////
// STEP 8
```

```
// Disable the data logging

%int.set
         script = Mission6
         output = MissionStep
         value = 8
         step=8 thread=1

%int.set
         script = Mission6
         output = M6LogEnable
         value = FALSE
         step=8 thread=1
%int.set
         script = Mission6
         output = AUV_CTD_PumpOn
         value = FALSE
         step=8 thread=1
%timed.wait
         script=Mission6
         trigger=TimeTick
         interval=1000
         step=8 thread=1

/////////////////////////////////////////////////////////////
/////////////////
// STEP 9
// Put the AUV into IDLE mode

%int.set
         script = Mission6
         output = MissionStep
         value = 9
         step=9 thread=1

%int.set
         script = Mission6
         output = TelemModeSelect
         value = IDLE
         step=9 thread=1


// ******************** Data Logging For Mission 1
********************

// ********************** TIMESTAMP TIGGER
**************************
%copy
         enable = M6LogEnable
         input = SYS_TwoHzTrigger
         output = M6LogTrigger

// *********************** DATA LOG COMPONENT
*************************


%data.log
  name                   = data6
  path                   = "M6LOG.DAT"
  enable                 = TRUE
  time_stamp_trigger     = M6LogTrigger
  level                  = LOW
  time_stamp_byte        = 254
  escape_byte            = 27
  byte_count_output      = M6LogSize

// **************************** ALIASES
*******************************

@data6.log.int.data     enable=TRUE trigger=M6LogTrigger
@data6.log.float.data   enable=TRUE trigger=M6LogTrigger
@data6.log.double.data  enable=TRUE trigger=M6LogTrigger

// ************************* LOG ITEM LIST
*****************************

// Vehicle Sensor Feedbacks and Control Items
%data6.log.float.data   input = HeadingSetpoint
%data6.log.float.data   input = AUV_CompassHeading
%data6.log.int.data     input = HeadingControl_RPM

%data6.log.int.data     input = VelocitySetpoint

%data6.log.float.data   input = DepthSetpoint
%data6.log.float.data   input = AUV_Depth
%data6.log.int.data     input = VerticalControl_RPM

%data6.log.float.data   input = AltitudeSetpoint
%data6.log.float.data   input = AUV_Altitude
%data6.log.int.data     input = AUV_AltSignalStrength
%data6.log.float.data   input = AUV_Pitch
%data6.log.float.data   input = AUV_Roll

// Motor Feedbacks
%data6.log.int.data      input = AUV_Left_RPM_Fb
%data6.log.int.data      input = AUV_Right_RPM_Fb
%data6.log.int.data      input = AUV_VertLeft_RPM_Fb
%data6.log.int.data      input = AUV_VertRight_RPM_Fb
%data6.log.int.data      input = AUV_Left_PWM_Fb
%data6.log.int.data      input = AUV_Right_PWM_Fb
%data6.log.int.data      input = AUV_VertLeft_PWM_Fb
%data6.log.int.data      input = AUV_VertRight_PWM_Fb

// AUV Mode and Status Items
%data6.log.float.data    input = AUV_BatteryVoltage
%data6.log.int.data      input = AUV_Leaking
%data6.log.int.data      input = AUV_Mode
%data6.log.int.data      input = MissionStep
%data6.log.int.data      input = LocalStep

// Payload Items
%data6.log.float.data    input = AUV_Conductivity
%data6.log.float.data    input = AUV_Temperature
%data6.log.float.data    input = AUV_Pressure
%data6.log.int.data      input = AUV_StandardConductivity
%data6.log.int.data      input = AUV_CTD_PumpOn
```

log.csp

```
/*
$Log: log.csp $
# Revision 1.3  1996/09/18  20:37:43  PURL
# Peter H and Kevin M: Renamed a log item to
AUV_VertThrust_RPM
#
# Revision 1.2  1996/08/24  10:38:22  PURL
# Peter H: Added data logging items and changed the enable
and trigger
#
# Revision 1.1  1980/01/04  03:14:59  PURL
# Initial revision
#
*/
// LOG.CSP
//
// This file performs the general logging that can be turned
off and on
// by the surface computer.  It is generally used for
debugging or measuring
// the performance of a particular action (i.e. the step
response of the
// vehicle while turning )
//
%copy
        enable = TRUE
        input = TelemEnableLogging
        output = AUV_IsLogging

// *********************** TIMESTAMP TIGGER
***************************
%copy
        enable = TelemEnableLogging
        input = SYS_TwoHzTrigger
        output = LogTrigger

// ************************ DATA LOG COMPONENT
**************************

%data.log
  name                  = data
  path                  = "LOG.DAT"
  enable                = TRUE
  time_stamp_trigger    = LogTrigger
  level                 = LOW
  time_stamp_byte       = 254
  escape_byte           = 27
  byte_count_output     = LogSize

// **************************** ALIASES
*********************************

@data.log.int.data      enable=TRUE trigger=LogTrigger
@data.log.float.data    enable=TRUE trigger=LogTrigger
@data.log.double.data   enable=TRUE trigger=LogTrigger

// ************************* LOG ITEM LIST
*****************************

// Vehicle Sensor Feedbacks and Control Items
%data.log.float.data   input = HeadingSetpoint
%data.log.float.data   input = AUV_CompassHeading
%data.log.int.data     input = HeadingControl_RPM

%data.log.int.data     input = VelocitySetpoint

%data.log.float.data   input = DepthSetpoint
%data.log.float.data   input = AUV_Depth
%data.log.int.data     input = VerticalControl_RPM

%data.log.float.data   input = AltitudeSetpoint
%data.log.float.data   input = AUV_Altitude
%data.log.int.data     input = AUV_AltSignalStrength

%data.log.float.data   input = AUV_Pitch
%data.log.float.data   input = AUV_Roll

// Motor Feedbacks
%data.log.int.data      input = AUV_Left_RPM_Fb
%data.log.int.data      input = AUV_Right_RPM_Fb
%data.log.int.data      input = AUV_VertLeft_RPM_Fb
%data.log.int.data      input = AUV_VertRight_RPM_Fb
%data.log.int.data      input = AUV_Left_PWM_Fb
%data.log.int.data      input = AUV_Right_PWM_Fb
%data.log.int.data      input = AUV_VertLeft_PWM_Fb
%data.log.int.data      input = AUV_VertRight_PWM_Fb

// AUV Mode and Status Items
%data.log.float.data    input = AUV_BatteryVoltage
%data.log.int.data      input = AUV_Leaking
%data.log.int.data      input = AUV_Mode
%data.log.int.data      input = MissionStep
%data.log.int.data      input = LocalStep
```

Appendix Three

	Calc	Max		Calc	Max
Vtx (dBm)	-14	-14	Vrx (dBm)	-37	-37
Etx (dBm)	-12	-12	Erx (dBm)	-30	-30

Reflection Source	Back Reflection (dB)			Attenuation (dB)		
	Calc	Max	Min	Calc	Min	Max
Splitter/Coupler	-55	-55	-60	-3.4	-3	-3.6
WD1315U	-55	-55	-55	-0.6	0	-0.8
Termination	-60	-60	-60	0	0	0
FC/APC	-60	-60	-68	-0.3	0	-0.5
FC/PC	-50	-50	-56	-0.2	0	-0.2
FO Cable	None	None	None	-0.25	-0.15	-0.25

Back Reflection Relative To a 0 dB input

	(mW)	(dB)	(mW)^0.5	%	32.83997	43.85294	1923.08
Splitter	3.16E-06	-55	0.001778	28.80704	36.83997	69.50222	4830.558
Termination	2.09E-07	-66.8	0.000457	7.40455			
WD1315U	6.61E-07	-61.8	0.000813	13.16736			
Cable Connector	1.58E-07	-68	0.000398	6.449093			
Cable Connector	1.38E-07	-68.6	0.000372	6.018644			
Cable Connector	1.2E-07	-69.2	0.000347	5.616926			
WD1315U	3.39E-07	-64.7	0.000582	9.429717			
Splitter	2.95E-07	-65.3	0.000543	8.800324			
Video Connector	1.95E-07	-67.1	0.000442	7.153172			
Ethernet Connector	1.95E-07	-67.1	0.000442	7.153172			
TOTALS	5.47E-06		0.006173	100			

Total Incoherent Pwr	5.47E-06
Incoherent dBm	-52.6181
Total Coherent Pwr	3.81E-05
Coherent dBm	-44.19

Video Transmission

Power Tx (dBm)	Attenuation (dB)	Power Rx (dBm)	Max Sensitivity (dBm)
-14	-9.35	-23.35	-37

SIR 32.83997

Ethernet Transmission

Power Tx (dBm)	Attenuation (dB)	Power Rx (dBm)	Max Sensitivity (dBm)
-12	-9.35	-21.35	-30

SIR 36.83997