Compile the kernel and modules, and install the modules to where they can be accessible by ***insmod(8), rmmod(8),*** and ***modprobe(8)*** (usually /lib/modules):

make dep
make bzImage // approx. 30 minutes, go have coffee.
make modules
make modules_install
cd arch/i386/boot
ls

ensure that bzImage is there.

cp bzImage /boot/rtzImage
ls /boot

ensure that rtzImage is there.
Install the new boot image

cd /etc
vi lilo.conf (or your favorite editor)

Add the following lines to /etc/lilo.conf by editing the file with your favorite editor (you only need to do this once):
NOTE: simple instructions using vi
use down and right arrow until bottom right corner
press i to start inserting
right arrow then enter
add the following lines in. (use tab after the first line)

image=/boot/rtzImage
label=rtlinux
read-only
root=/dev/hda1

Note: replace /dev/hda1 in the above with your bootable partition. To find out which partition to use, take a look at the other existing entries for root= in your /etc/lilo.conf file. It is imperative that you specify the correct partition, otherwise, your entire Linux filesystem may be corrupted when you reboot.
Comment out default=linux

#default=linux
replace it with
default=rtlinux

Esc, then **:wq** to save and quit

/sbin/lilo

check the output of the **/sbin/lilo** Command. If any errors are displayed, then you need to fix your /etc/lilo.conf file. *Do not proceed with the next steps until all errors have been resolved!*
Restart the computer: reboot

/sbin/shutdown -r now

or
CTRL+ALT+ DEL
Load the RTLinux kernel: At the LILO: prompt, press "Shift" or "Tab". This will give you a listing of the available kernels.
Enter:
rtlinux

The RTLinux kernel should boot. If Kuzdu runs to setup the hardware, keep the Configuration, or remove configuration if the hardware has been removed.
Login: root
Password: ventana

Compiling RTLinux

- The next step is to compile RTLinux properly.
  cd /usr/src/rtlinux/rtlinux-3.1
  ln -s /usr/src/linux linux
  (This creates a symbolic link to tell rtlinux where to find linux)
- Configure RTLinux:
  make config OR ***make menuconfig*** OR ***make xconfig***
  (Again, we will usually use make menuconfig).
  Select the following and de-select all others:

Support Options -->
[*] Posix Standard IO
[*] Posix priority protection
[*] Dev Mem Support
[*] Enable Debugging
[*] rtl_printf uses printk
[*] Floating point Support
[*] RTLinux Debugger

Drivers -->
[*] Shared Memory Driver (we don't use it, but could be useful in future)
[*] Serial Port Driver

- Compile RTLinux:
  make dep
  make
  make devices

Post Installation and Running RTLinux Programs

To be able to run any programs, you must first load the rtlinux modules. To do so, type:
cd /usr/src/rtlinux/rtlinux-3.1
./scripts/insrtl
lsmod
(you should see the following rt kernel modules loaded)
rtl_sched
rtl_fifo
rtl_posixio
rtl_time
rtl

mount /mnt/floppy
cp /mnt/floppy/ventana_sub.tgz .

NOTE: if above did not work and rtlinux complains of bad filesystem it is most likely that the floppy is of a fat system type.
mount -t msdos /dev/fd0 /mnt/floppy
cp /mnt/floppy/ventan~1.tgz .
mv ventan~1.tgz ventana_sub.tgz

NOTE: DOS changes the filename, therefore ventan~1.tgz may be of a different name. Look for the file with the tgz extension.

tar -zxvf ventana_sub.tgz
cp rtl.mk ./ventana_sub (your rtl.mk file would be different then ours, say yes to overwrite.)
cd ./ventana_sub
ls
make

This should compile the program. If you end up with telemetryCom.o and rt_serial.o, the compilation has succeeded. NOTE!!!! YOU WILL NOT BE ABLE TO RUN THE SOFTWARE AS THE ADDRESS SPACE IS ARCHITECTURE DEPENDENT. IE. PLEASE REFERENCE THE VENTANA SUB HARDWARE ARCHITECTURE FOR ADDRESS SPACE INFORMATION.

- The image rtzImage is our kernel image needed for the sub. We also need to create a root filesystem and automatic script that will load the modules automatically and start running Ventana.

(Next: creating the root filesystem, downloading/ installing the root filesystem and kernel, and overview of rtlinux programming environment.)

- You can also try running the examples. To do so, simply go to the appropriate directory under /usr/src/rtlinux/rtlinux-3.1/examples and follow the instructions in the corresponding README file.

About rtlinux

you can obtain help from your peers via the rtl@rtlinux.org mailing list for which you can un/subscribe to via http://www.rtlinux.org/mailing_lists.html .
FSM Labs further provides Commercial support, development, and
training. Please contact FSM Labs at
business@fsmlabs.com
or visit their website at
http://www.fsmlabs.com

Appendix G Software Listings

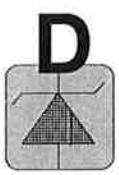
IO Element

```
#ifndef IO_ELEMENT_H
#define IO_ELEMENT_H
#include <cmath>
#include <iostream>
#include <memory>
#include <string>

enum IOTYPE { none = 0,
    topDigIn, topDigOut, topAnalogIn, topAnalogOut, subDigIn,
subDigOut, subAnalogIn, subAnalogOut, button_exist, slidebar_exist };

using namespace std;

/************** global variables - help out with grouping ***************/

struct Graphics {
   string type_;
   int x_;
   int y_;
   string type2_;
   int x2_;
   int y2_;
   int color_;
   int vga_screen_;
 };


struct DigitalInput {
   unsigned int group;
   unsigned int port;
   unsigned char channel;
};

struct Output_List {

   Output_List *next;
   Output_List *prev;
   int out_surveh;
   int out_diganal;    // used for
   int out_elem_num;
   int out_board;      // used for bottom output only
   int out_digital_value;
   float out_analog_value;
   float out_analog_raw;
};

struct Bar_Detail {

   float min;
   float max;
   float average;
   float yrange;
   float rrange;

};


class IOElement {

// IOElement is encapsulated to prevent corruption in case of newer
//classes contain them.  The only way to interact with this is through
// this interface. Any changes that should be made should derive this
//class and extend  the functionally and/or variables.
public:
   IOElement()                    throw();
   ~IOElement() throw();
   void make(   int    surfVehicle,
                int    digAnalog,
                int    element_num,
                string element_name,
                string element_calculation,
                int    default_level,
                bool   enable_channel,
                bool   auto_calibrate_enable,
                bool   enable_graphics,
                int    digital_value,
                float  analog_value,
                float  analog_raw,
                float  analog_offset,
                float  p_gain,
                float  i_gain,
                float  d_gain,
          string graphic_type,
                int    color,
                int    x,
                int    y,
                int    vga_screen )        throw(bad_alloc);

   string   name()              { return element_name_; };
   int      num()               { return element_num_; };
   int      getIOtype()         { return iotype_; };
   bool     isChannelenabled() { return enable_channel_; };

   DigitalInput digital_input_info();
   void     updateDigital_value(int digVal) { digital_value_ = digVal; };

   unsigned int getAnaChannel() { return element_num_; };
   void     updateAnalogValue(float value) { analog_value_ = value; };
   void     updateAnalogRaw(float raw)   { analog_raw_ = raw; };

   void     getAllInfo(int, int, int, int, char line[]);
   void     getAllInfo(char retString[]);
   void     getAllSubAnaInfo(char retString[]);
   void     test();

   int      get_digital_value() { return digital_value_; };
   float    get_analog_value() { return analog_value_; };
   float    get_analog_raw() { return analog_raw_; };
   void     get_graph_info( char graph_string[] );

   Output_List      *next_out;
   string           out_string;   // output string board#|elem#|...



/*****************************************
 * Output purposes
 *****************************************/
```

```
  int     iotype_;
  int     element_num_;
  string  element_name_;
  string  element_calculation_;
  int     default_level_;
  bool    enable_channel_;
  bool    auto_calibrate_enable_;
  bool    enable_graphics_;
  float   analog_value_;
  int     digital_value_;
  float   analog_raw_;
  int     input_io_point_;
  int     output_io_point_;
  float   analog_gain_;
  float   analog_offset_;
  float   p_gain_;
  float   i_gain_;
  float   d_gain_;
  Graphics        graphics_;
  DigitalInput    digital_input_;

  /* helper functions - internal to class */
  int findIOtype(int surfVehicle, int digAnalog);
  void findDigitalInfo();
  int typeToSD(int, int&, int&);

  Bar_Detail      bar_detail;

};

extern IOElement io_element;

#endif
```

IO Element Control

```
#ifndef IO_ELEMENT_CONTROL_H
#define IO_ELEMENT_CONTROL_H
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include "io_element.h"
#include "./analog_interface/analog_in.h"
#include "./digital_interface/digital_in.h"

typedef vector<IOElement> topDigInIOList;
typedef vector<IOElement> subDigInIOList;
typedef vector<IOElement> topAnalogInIOList;
typedef vector<IOElement> subAnalogInIOList;
typedef vector<IOElement>::iterator digIterator;
typedef vector<IOElement>::iterator anaIterator;

struct Button
{
        Button *next;       // pointer to the next node
        string name;
        string type;
        string color;
        int board;

        int elem_num;
        int state;
        int x;
        int y;
        int x2;
        int y2;
        int width;
};

struct Slidebar
{
        Slidebar *next;     // pointer to the next node
        string name;
        string color;
        int board;
        int elem_num;
        float value;
        int x;
        int y;
        int x2;
        int y2;
};

int findIOtype(int, int);       // helper function for "getIOElement()"

class IOElementControl {
public:
  IOElementControl()
  ~IOElementControl()
  bool pushElement(IOElement element)
  digIterator getIOElement(int, int, int, digIterator)  throw();
  int     getIOElement(int, int, int)                   throw();
  int     readFromChannel(int, int, int)                throw();
  bool enableChannel(IOElement element)                 throw();
  bool    retIOString(int, int, int, int, char line[])  throw();
  bool    retDigitalObjString(char objString[], int)    throw();
  bool    retAnalogObjString(char objString[], int)     throw();
  bool    retSubDigitalObjString(char objString[], int)         throw();
  bool    retSubAnalogObjString(char objString[], int)          throw();
  int     retDigSize();
  int     retAnaSize();
  int     retSubDigSize();
  int     retSubAnaSize();
  bool    readVoltage(int, int, int, float&, float&)    throw();
  void    dump();
  void    clean_db();
  char*   show_connection_existence(int type );

  // ---------- OUTPUT ----------

  Output_List     *curr_out;  // can traverse the list of output nodes
  Output_List     *curr_prev;         // for deletion purposes
  void    make_output ( int   surfVehicle, int digAnalog, int
element_num, char output_string[] );
  int     check_io_output_db  (int out_board, int out_elem_num );
  int     set_out_string( IOElement *some_ptr, Output_List *ptr );
  string  display_output( int surfVehicle, int digAnalog, int
element_num);
  // delete all output nodes
  int     delete_output ( Output_List *ptr );    int
                  delete_single_output ( int type, Output_List *ptr );
  void    dump_output();
  void    test_output(int type);
  void    test_output_helper(Output_List *ptr);


  int     send_output_down( int surfVehicle, int digAnalog, int
element_num, int digitalVal, float analogVal);
  int     get_graph_info (int index, char graph_string[] );
  int     setting_bar_detail (int elem_num,  float min, float max, float
ave, float yrange, float rrange );
  int     getting_bar_detail (int elem_num,  char bar_string[] );

  // ---------- BUTTON ----------
```



```
  Button  *but_ptr;   // pointer to traverse the button nodes
  Button  *first_but;  // pointer to the first node of button node

  int       check_button_db (string name, int board, int elem_num);
  int       add_to_button_db ( string name, string type, int out_board,
int out_elem_num, int x, int y, string color, int state, int screen );
  int       remove_button_from_db (int board, int elem_num, int
screen);
  void      display_button_db ();
  char*     send_alternate_but_down (int board, int elem_num);
  int       send_momentary_but_down (int board, int elem_num);
  int       get_but_state(int board, int elem_num);
  int       set_but_state(int board, int elem_num, int state);
  bool      ret_button_db_string (char objString[], Button *ptr);
  int       bcount;
  char*     get_button_string(int index);

  // ---------- SLIDEBAR ----------

  Slidebar  *slide_ptr;  // pointer to traverse the slidebar nodes
  Slidebar  *first_slide; // pointer to the first node of slidebar node
                          // count the number of slidebar

  int       check_slidebar_db (string name, int board, int elem_num);
  int       add_to_slidebar_db ( string name, int out_board, int
out_elem_num, float value, int x, int y, string color, int screen );
  int       remove_slidebar_from_db (int board, int elem_num, int
screen);
  void      display_slidebar_db ();
  float     get_slide_value(int board, int elem_num);
  bool      ret_slide_db_string (char objString[], Slidebar *ptr);
  int       set_slide_value(int board, int elem_num, float value);
  int       scount;
  char*     get_slidebar_string(int index);

  // ---------- Detritus ----------
  int  Detritus_make();
  int  Detritus_select(int selection);
  void Detritus_initialize();
  char*     get_element_name(int surveh, int diganal, int elem_num);

  //-----------------------------------------------------------

   clean_all_ioelem();
  char*     get_button_connection(string name);
  char*     get_slide_connection(string name);

  //-----------------------------------------------------------
  topDigInIOList              topdigitalInList;
  subDigInIOList              subdigitalInList;
  topAnalogInIOList           topanalogInList;
  subAnalogInIOList           subanalogInList;
  topDigInIOList::iterator    currDigIn;
  subDigInIOList::iterator    currSubDigIn;
  topAnalogInIOList::iterator         currAnaIn;
  subAnalogInIOList::iterator         currSubAnaIn;


 bool inDataBase(IOElement element);

  int count;

  /********** Detritus structures **********/

  struct Output_List Detritus[16];

  /*****************************************************************/
};

extern IOElementControl ioElementControl;
#endif
```

Alarm

```
#ifndef ALARM_H
#define ALARM_H
#include <iostream>
#include <fstream>
#include <string>
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include "errors.h"

struct Digital_Alarm
{
        string    name;
        string    descr;
    int status;                 // On = 1, Off = 0

};

struct Analog_Alarm
{
        string    name;
        string    descr;
    int status;                 // On = 1, Off = 0

};

struct Other_Alarm
{
        string    name;
        string    descr;
    int status;                 // On = 1, Off = 0

};

class Alarm {

  public:
        Alarm();
        ~Alarm();

  string get_alarm_description (int type);
  int write_alarm_history();

  Alarm   *alarm_ptr;
  Alarm   *first_alarm;

  Alarm   *next;
  string  name;
  string  descr;

  Digital_Alarm     digital_alarm[35];
  Analog_Alarm      analog_alarm[50];
  Other_Alarm       other_alarm[100];


  void      set_io_alarm_descr();
  int       add_alarm_to_db(int diganal, int elem_num, string name);
// set the alarm name from IO ELEMENT
int         check_alarm_db(string name);
  void      display_alarm_on();
```

```
  void    display_digital_alarm();
  void    display_analog_alarm();

  // setting the alarm of each type to be on
  int     set_dig_alarm(int elem_num);
  int     set_ana_alarm(int elem_num);
  int     set_other_alarm(int index);
  void    reset_alarm();

  // might not be needed here.....remove later
  int     check_ana_status (int elem_num);
  int     check_dig_status (int elem_num);
  int     check_other_status (int index);

  string  get_ana_alarm_name (int elem_num);
  string  get_dig_alarm_name (int elem_num);
  string  get_other_alarm_name (int index);
  string  get_alarm_descr (string name);
  void    clear_alarm_list();


  int ana_count;
  int dig_count;
  int other_count;
  int alarm_status;
  string alarm_name;

};

extern Alarm alarm_obj;

#endif
```

Depth

```
  ndef DEPTH_H
#define DEPTH_H
#include <iostream>
#include <string>
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include "errors.h"
#define MAXCOUNT1       100
#define MAXCOUNT2       50
#define MAXCOUNT3       25


extern void depth_update_request(int arr);
extern void* depth_head_routine(void* args);
extern void* depth_routine1(void* args);
extern void* depth_routine2(void* args);
extern void* depth_routine3(void* args);
extern void* head_routine(void* args);

class Tel_Depth {

  public :

        Tel_Depth();
        ~Tel_Depth();

        void update_depth_hist(int which_array);
        void update_head_hist();
        float ret_curr_depth();
        float ret_curr_head();
        float ret_curr_pitch();
        float ret_curr_roll();
        int set_curr_depth(float depth);
        int set_curr_head(float depth);
        int set_curr_pitch(float depth);
        int set_curr_roll(float depth);
        bool get_depth_hist(int which_array, char hist[]);
        bool get_head_hist(char hist[]);
  private :
        float array1[MAXCOUNT1];    /* 2 minutes */
        float array2[MAXCOUNT2];    /* 30 minutes */
        float array3[MAXCOUNT3];    /* 120 minutes */
        float headArray[MAXCOUNT2]; /* 50 headings hist */
        float curr_depth;
        float curr_heading;
        float curr_pitch;
        float curr_roll;
        int count1;
        int count2;
        int count3;
        int headCount;

};

extern Tel_Depth td;                                    // global object

#endif
```

Graphic

```
#ifndef GRAPHIC_H
#define GRAPHIC_H
#include <iostream>
#include <fstream>
#include <string>
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include "errors.h"

struct Compass {
      string name;
      int x, y;
};



struct Vehicle {
      string name;
      int x, y;

};

struct Light {
      string name;
      int x;            // x position for light window
      int y;            // y position for light window

      // position for each camera
      int x1, y1, x2, y2, x3, y3,  x4, y4, x5, y5, x6, y6, x7, y7, x8, y8, x9,
y9, x10, y10;

};
```

```
class Graphic {

        public:
        Graphic();
        ~Graphic();

        Compass comp1;
        Compass comp2;
        Vehicle vehicle1;
        Vehicle vehicle2;
        Light light1;
        Light light2;

        int set_light_db ( int screen, int xpos, int ypos,
        int x1, int x2, int x3, int x4, int x5, int x6, int x7, int x8, int x9,
int x10,
        int y1, int y2, int y3, int y4, int y5, int y6, int y7, int y8, int y9,
int y10 );

        int set_graphic_info (string type, string name, int x, int y, int
screen);
        int  get_light_string ( int screen , char graphic_string[]);
        int  get_compass_string ( char graphic_string[] );
        int  get_vehicle_string ( char graphic_string[] );

};

extern Graphic graph;

#endif
```

Telemetry String

```
#ifndef TELEMETRY_STRING_H
#define TELEMETRY_STRING_H

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/signal.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>
#include <stdio.h>
#include <string>
#include <pthread.h>
#include <math.h>

#include <ctype.h>
#include "errors.h"

extern int start_threading (void);
extern void *telem_string_routine (void *arg);
extern void tty_server_request (int operation);

/* this is the structure with information to be sent to the
 * mbari logging system.
 */
struct LoggingStringOut_ {
        long    milli_bar;  /* depth */
        int     not_CTD_power;     /* GESDAC_1_0_DAC */
        int     in_water_switch;     /* GESDAC_1_1_DAC */
        int     GESDAC_1_2_DAC;
        int     GESDAC_1_3_DAC;
        int     GESDAC_2_6_ADC;
        int     GESDAC_2_7_ADC;
        int     GESDAC_2_8_ADC;
        int     GESDAC_2_9_ADC;
        int     savonious_counts;
        int     dive_num;
        float   heading;
        float   pitch_fb;
        float   roll_fb;
        float   altitude_lp;
        float   pilot_turn_stick;
        float   pilot_axial_stick;
        float   pilot_lateral_stick;
        float   pilot_vertical_stick;
        float   depth;
        float   sony_camera_tilt_fb;
        float   sony_camera_shoulder_fb;
};

/* this is the structure with information to be received from the
 * mbari logging system. (12 member)*/
struct LoggingStringIn_ {
        float port_thruster_ext_in;
        float stbd_thruster_ext_in;
        float lateral_thruster_ext_in;
        float spare_0_ext_in;
        float spare_1_ext_in;
        float spare_2_ext_in;
        float spare_3_ext_in;
        float spare_4_ext_in;
        float spare_5_ext_in;
        float spare_6_ext_in;
        float spare_7_ext_in;
        int input_ctr_disable;
};
/* structure with information to be sent to the ARL Vision Control
System */
struct VisionControlStringOut_ {
        long milli_bar;
        int GESDAC_1_0_DAC;
        int GESDAC_1_1_DAC;
        int GESDAC_1_2_DAC;
        int GESDAC_1_3_DAC;
        int GESADC_2_6_ADC;
        int GESADC_2_7_ADC;
        int GESADC_2_8_ADC;
        int teleos_control;
        int savonious_counts;
        int dive_num;


        float heading;
        float pitch_fb;
        float roll_fb;
        float altitude_lp;
        float SIT_camera_tilt_fb;       /* pilot_turn_stick */
        float SIT_camera_pan_fb;        /* pilot_axial_stick */
        float pilot_lateral_stick;
        float pilot_vertical_stick;
        float depth;
        float sony_camera_tilt_fb;
        float sony_camera_shoulder_fb;

};

/* structure with info be be received and processed
 * from the ARL Vision Control System    (13 member)
 */
struct VisionControlStringIn_ {
        float port_thruster_ext_in;
        float stbd_thruster_ext_in;
        float lateral_thruster_ext_in;
        float vertical_thruster_ext_in;
```

```
        float spare_0_ext_in;
        float spare_1_ext_in;
        float spare_2_ext_in;
        float spare_3_ext_in;
        float spare_4_ext_in;
        float spare_5_ext_in;
        float spare_6_ext_in;
        float spare_7_ext_in;
        int input_ctr_disable;
} ;

struct TmsControlStringIn_ {
        char mode_1[80];
        char mode_2[80];
        float val_1;
        float val_2;
        char unit_1[80];
        char unit_2[80];
};

class Telemetry_String {
public:
  Telemetry_String()          throw();
  ~Telemetry_String()         throw();

  void updateOutFields(int protocolChoice);
  char* concat();
  int ReceivePacket(int fd, char* line);
  void parse_and_set_string( char some_string[] );
  void lineout(int fd, char* line);
  void setup_serial (int fd, int baud_choice, int nStopBits);
    LoggingStringOut_ LoggingStringOut;
  LoggingStringIn_ LoggingStringIn;
  VisionControlStringOut_ VisionControlStringOut;
  VisionControlStringIn_ VisionControlStringIn;
  TmsControlStringIn_ TmsControlStringIn;


extern Telemetry_String telstring;

#endif
```

#include <sys/types.h>

Telemetry

```
#ifndef TELEMETRY_H
#define TELEMETRY_H
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/signal.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <math.h>
#include "telemetry_string.h"

/*****************************************************************/

extern float sub_analog_in[64];          /* sub analog coming in */
extern int tty_send_request (char* cmd );
extern void *tty_recv_routine (void *arg);


/*****************************************************************/

using namespace std;

class Telemetry {
public:
        Telemetry()                     throw();
        ~Telemetry()                    throw();
        bool parse_and_set_values ();

};

extern Telemetry telemetryObj;
#endif
```

IO Poll

```
#ifndef IO_POLL_H
#define IO_POLL_H
#include "./digital_interface/digital_in.h"
#include "./analog_interface/analog_in.h"
#include "./io_element_control.h"
#include <cmath>
#include <stdio.h>
#include <memory>
#include <string>
#include <assert.h>
#include <pthread.h>

void tsetup_serial(int);
void tlineout(int, char*);
char* tconcat(char str[]);

void    initialize();
void*   pollDigital(void* args);
void    pollAnalog();

#endif
```

Digital In

```
#ifndef _DIGITAL_IN_H
        #define _DIGIAL_IN_H


int readDigitalChannel(int group, int port, unsigned char channel);
bool initializePort(int group);
bool writeControlPort(int group);

#endif
```

Analog In

```
#ifndef __ANALOG_IN_C_
#define __ANALOG_IN_C_

//unsigned int BASEADDRESS = 0x0300;

void write_port_data(unsigned char current[3]);
void startConversion(int fd);
int readVolt(int fd, unsigned int channel, float& voltage, float& raw);


#endif
```

Index

,
,JOYSTICK 22
1
1/4 WAVE DAMPING 16
A
ADDRESS 24
ALT-F1 20
ANALOG INTERFACE CARDS 22
ANALOG OUTPUT CARDS 24
AUTO DEPTH/ALTITUDE 16
AUTO HEADING 16
AUTO PILOTS 16
AUTOPILOTS 16
B
BAUD 23
C
CHANGE THE TIME 7
CHANGING THE GAIN 9
CHANGING THE OFFSET 9
COM 1 23
COMPUTER 22
CONSOLE WIRING DIAGRAM 26
CP PROBE INTERFACE 24
D
DATA BITS 23
AGNOSTIC PAGE 5
DIAGNOSTIC PAGE 9
DIGITAL INPUTS 22
DISPLAYED DATA 5
E
ERROR PAGE 20
F
F6 7
FLOPPY DISK DRIVE 22
H
HARD DRIVE 22
HELP PAGE 5
I
INITIAL PAGE 5
L
LOGGING DATA 9
O
OPERATORS SCREEN 7
OPTIMUM GAIN 16
P
PARALLEL PORT 22
PARITY 23
PERIPHERAL EQUIPMENT 22
PILOTS PAGE 5
PRINT THE CONFIGURATION FILE 7
PRINTER 22
Q
QUITING THE PROGRAM 7
R
RAM 22
RS-232 23
S
SERIAL PORT 22
SERIAL PORT 23
STOP BIT 23
SUBSEA CONTROL 24
SURFACE CONTROL 22
SWITCH CONFIGURATIONS 27
SWITCHES 22
T
TELEMETRY 24
TELEMETRY ERROR 26
TELEMETRY ISOLATION DRIVER 24
TELEMETRY STATUS AND ALARM 7
TEXT\: 7
TIME\: 7
V
VGA CARD 22