Design and development of a three-phase transducer for real and reactive power, using line current and voltage inputs.

This could be useful for plotting performance charts for synchronous and induction machines.

This project has potential to develop into a graduate project.

Dr. Iain Collings

IBC1. Communication channel modelling

An important limitation to both mobile communications systems and teleconferencing systems is that the transmission channel varies with time. Modelling and measuring communication channels are important in order to generate algorithms which combat problems due to this time variation (or fading). This project aims to measure a communication channel, analyse the measurements, and use computer simulations to model the channel. Equalisation algorithms will also be developed to cancel the channel effects.

IBC2. High speed communication system modelling

With the realisation that fibre-to-the-home networks are not going to be installed in the foreseeable future, there is significant interest in high speed digital transmission over the 'last mile' of copper. Standards such as ADSL, VDSL, HFC and others are being proposed and implemented by telecommunications and pay-TV companies around the world. This project will use simulations and analysis to investigate the relative benefits and possibilities of each technology, and compare them to current modems operating over standard telephone lines. It will suit students with good mathematical and programming skills.

IBC3. Automatic speech recognition system

When requesting information from some modern telephone systems, it is no longer necessary to enter requests and data using the key-pad. Automatic speech recognition allows the customer to speak their requests to a computer which literally recognises the words and performs the request. The aim of this project is to develop an automatic speech recognition systems. this will involve recording speech, and analysing the measured waveforms. There are a number of possible approaches to the signal processing problem. This project will make use of hidden Markov model techniques. It will build on work done by previous 4th year project students, and suit students with good mathematical and programming skills. Check out http://www.ee.mu.oz.au/staff/iainc/www/ugradprojs.html

IBC4. Image processing for blood vessel profileometry in the human eye

Medical research into the human eye has indicated that blood vessels on the retina can be important for certain eye conditions. Lasers are used by biomedical engineers to measure the surface of the retina. Unfortunately, laser light is reflected by more than just these blood vessels, so they are hard to see in the returned image. This project aims to use signal processing techniques to model, simulate, and analyse the laser reflected images. Actual measured data will also be used in order to see what can be done with real eyes. One important problem will be that the laser images are taken at video rate, so that means lots of data needs to be processed in real time, so a doctor can make a diagnosis on the spot. This project will build on work done by previous 4th year students. Check out http://www.ee.mu.oz.au/staff/iainc/www/ugradprojs.html

IBC5. Hardware and software for the radio-astronomy project

In conjunction with the School of Physics, we are continuing to develop the hardware and software necessary to use the 3.5m antenna located on the roof of the Redmond Barry building as an instrument for experiments in radio-astronomy. At present, the antenna is under computer control, and much of the microwave hardware has been constructed. But much remains to be done! Working closely with Dr. Matthew Bailes from the School of Physics, planned developments for 1997 include: writing low-level interface software for a one-bit digitiser, commissioning and calibration of the microwave-frequency components, consolidation of the computer-controlled positioning system, writing code to support radio astronomy experiments,

18/3/98

Done.

- controller software done but crashes occ.
  - 2> runs in 2 modes;
    - ① Real time - controls motors
    - ② Simulation - sim to screen with real time
- c:\unimelb\unimel.exe is the program
- c:\unimel\project is source code
- when 'unimel' running talks to:
  - key board
  - P.L.C. - ant. cont.
  - linux serial lines
- 'unimel' does not use bios, not worth ~~one~~ porting to another O/S
- ~~[illegible]~~ only way to dissable TCP/IP is to take out card
- Problems
  - UNOS - no longer under development
    - soln - QNX
    - LINUX RT ② if Time

Sergei:

Java client sends info to linux server
User creates a/c - linux
Bookings - linux
take commands from active user, all others observing
linux - secretary
unos - drives antenna in RT
linux should know all that unos does
send test packets & perform task

Neil
understand test packet
take info from tracking algo. & use to control unos
take feed back from unos & update Sergei & things

PO

18/3/98

for 19/3/98
Tasks to be completed

- Simon
  1/ computers & when
  2/ Keys Collings
  3/ try & log into tower from roof
  establish if Network works
  4/ Expected arrival 'encoders'
  5/ Thuja to copy 'cinema' code to local machine
  6/ 'parser.c' serial link

1:30 Simon
2:00 Simon

- Computer's For Project

8/3/98

Server

Backup Device ??

400 • Proc - 200 MMX
240 • Mem - 128Mb SD RAM MIN (If possible more)
100 • MB - INTEL TX
300 • ~~6235~~ HD - 4.3GB SCSI (req. SCSI cont.)
/ • CACHE - Pipeline Burst 512Kb
60 • ~~2MB Sirrus V46~~ ~~S3~~ 2MB S3 Verge
20 • 16bit S/C ~~or Ethernet Card~~
100 • 16x CD-Rom x1
• 3½ Drive x1
335 • Monitor 15in (Acer)
1925
200
$2100 Inc tax

UNOS
350 • Proc - 166MMX
120 • Mem - 64Mb SDRAM
380 • HD - 4.3Gb IDE
• MB - Intel Tx
520 • rest as above
1370
200
1500 Inc

Linux
• Lic for 1 Win 95
• lic for 1 MS Visual C

19/3 For next 3 weeks

Sajith
- Java applet - GI for sys
  click button send message to linux serve

Theyo
- Intel.c - tracking alg for Intel level sw
- Startrw.c - tracks stars
- Developing Alg for tracking - high level

Vinosh
- Set up home page which is accessible from other locations.
- back up slip driver

22/3 Server

Proc - 200 MMX

Mem - 128 MB (SD RAM)

Mother Board - Intel TX

Hard Drive - 4.3GB SCSI

CACHE - Pipline Burst 512kb

Video Card - 2MB S3 Verge

Sound card - 16 bit

CD ROM - 16X

3½ Drive - 1 HD

Monitor - 15in Acer

Backup Device - ZIP DRIVE ⇐ should be portable for use on several PC's

Operating System - LINUX

Cards - Ethernet x1

- x1

Add. Software -

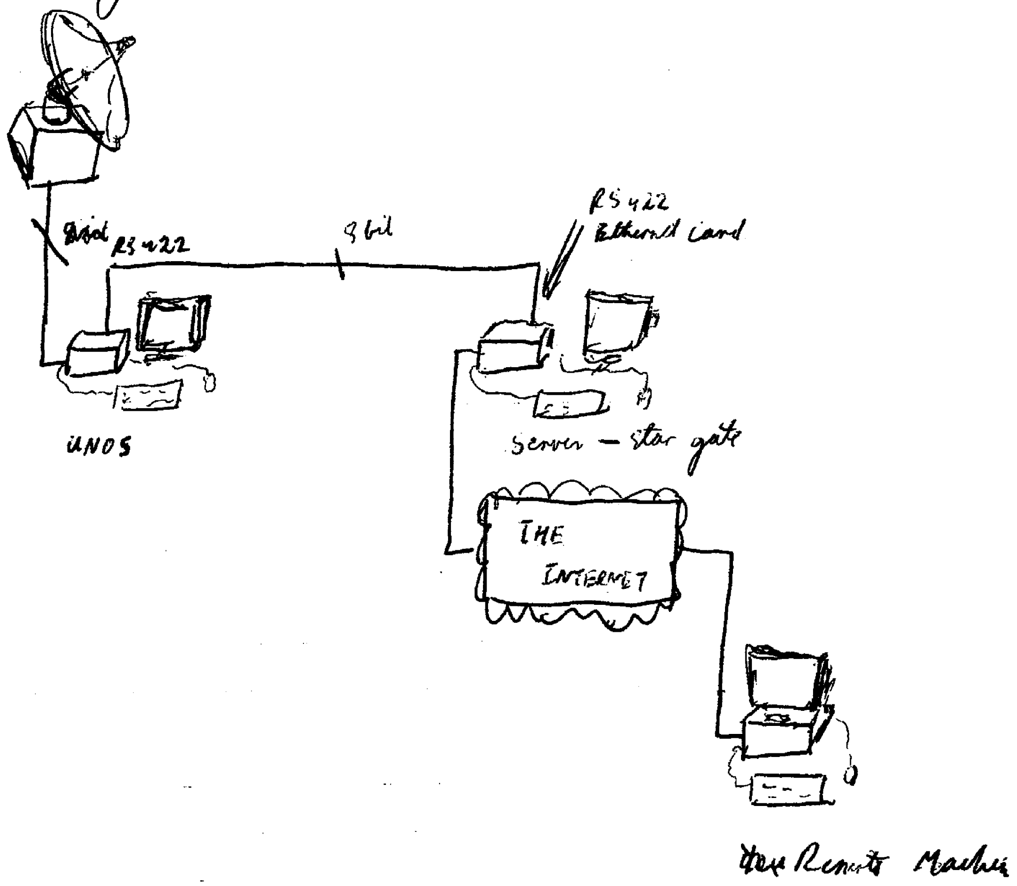
1/4/98

SLIDES FOR PRESENTATION

① Project Name
Our Names
Co-ordinators names
Running Home Page Since –

Nice back ground like front of lab book

②



Future steps
③ Different Sections
① UNOS – • Tracking ~~Satt~~ Satellites
• Communicating with Server
• Presentation of antenna control user interface – super user

Thaya

② Server – • Communicating with UNOS PC
• Setup home page
• Reliability / Efficiency
• Communicating with Internet
• User accounts

Niroshan

Sajee

(3) The ~~Intranet~~ Remote machine - • Setup user interface for internet control of dish
- Communication with server
- Stability
- Protection / Authorisation.

(4) The Internet - God Help Us!!

3 Current Setup

(1) UNOS - simulation / ~~off~~ real ~~dish~~ movement of dish
- reliable control system
- communicate with server (unstable!!)
- dosish user interface

(2) Server - • Linux 1.X (older!!)
- No web server (i.e. no internet access, only remote)
- communicate with UNOS (unstable!!)
- remote log in only (not working at this stage, but did)

(3) The remote ~~[illegible]~~ machine - ??
- Nothing, can just log into server
- no feed back of antenna movement ~~stop~~.

(5) Who's doing what at this Stage?

Thaya → Tracking algorithms

~~Sajee~~ Nirochan → ~~User interface via~~ also setting up server home page maintenance

Sajee → Netscape → user interface

(6) Gif image.

(7) Question? Name Server, Criticisms of GUI, General Questions.

7/4/98

Dead Lines !!

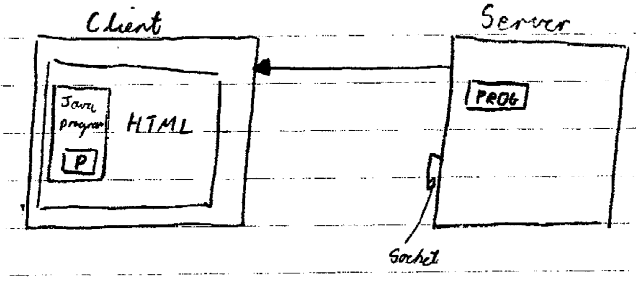
Me.
Get client to send understandable msg to server.

Sajee.
Java complete
No mods to final p GUI
Running on Netscape
- Get Vamsi: Basketball + ~~3~~ 2 Mars Bars

Thaya.
Tracking Algorithms
Kalman filter - Why?
How to use in our case?
"Understand Unimel Code!!"

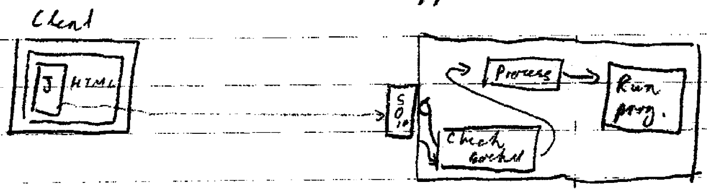
22/4/98

Presentation for 23/4 3pm



Problem ?
Running java applet -> must run on ~~server~~ client,
but if you click options how will
server know ?

Sol^n -> write to a socket/port



Ref, UNIX Network Programing
Steven Richards

Other methods/sol^n -> CGI-BIN, using perl

PRO -> • runs from server
• robust / portable
CON -> • script ∴ slow
• as it runs from server
band width limited,

Most probably implemented for data base section
of home page: * bookings
* help
* FAQ
* misc

22/4/98 See remote Command Execution notes for ref
Richard Steven
'UNIX Network Programming'

- rshd implemented in Linux
- need to work out more about sockets, & how rshd will determine socket add, & communicat to it.
- security

31/4/98 Sort through Unimel code.

Format:

path/File name	About/header file	#include files

header files, just mention 'header file' + path.

Would like at some stage to know connections of files, so if a function anywhere is modified, we know where exactly these effects will take place. Place this on web page in a restricted access area.

22/5/98

Server - Client

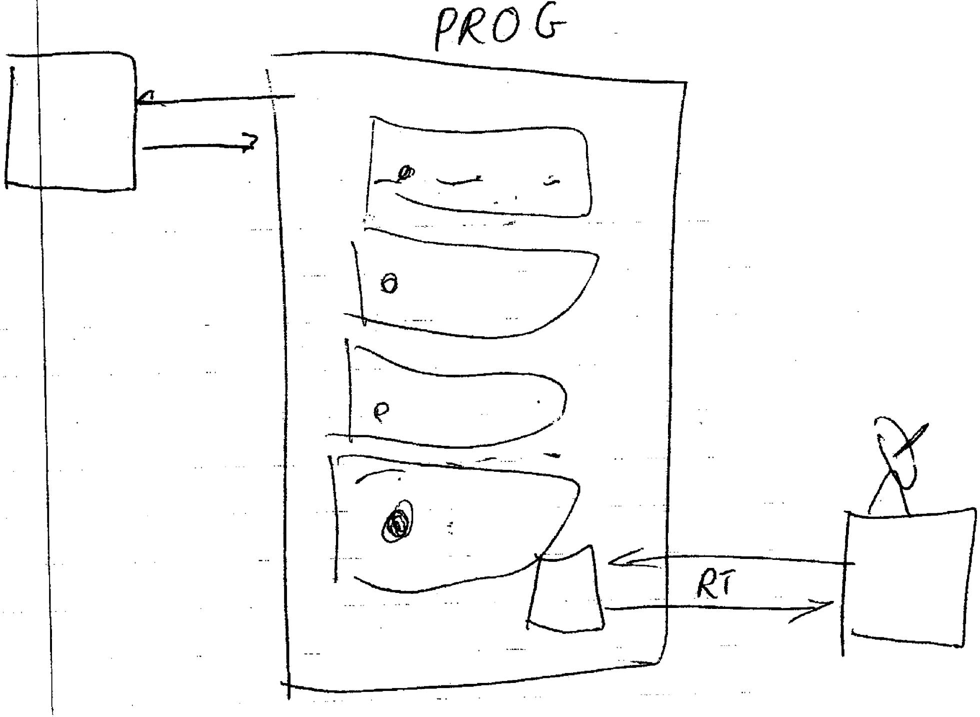
- communication app
- capable of sending message from server to client
- password protection using perl is running
  currently password is stored in program, future would like to read from file located in HD.
- Cannot compile source for Unimel - 800 errors
  - segments of code missing!!

Futures:

- Get dish running by D-Day - August
- Look into web cam for dish.

Me:

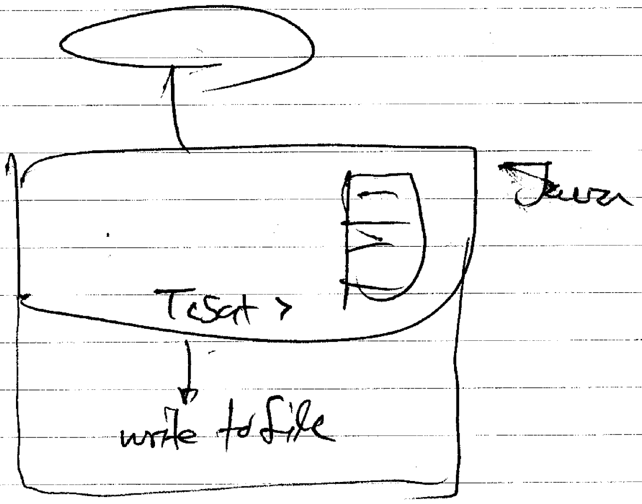
Serial connection - get this tested & running!!



18/07/1998

Grp meeting

External user



Client & Server

Track Sat	T	<sat>
Track Star	R	<star>
Manual	M	<Az> <El>
Stop	S	
Book	B	<date & time>
Download	D	
Veiw Dish	V	
Logout	L	
E		

Server to Client

Active
Passive
Dish Down
Dish Up
Valid
Invalid

No data to dish - down loading

Booking → Confirm
    ↳ Reject

Server down

az-el

23 July 1998

Network Config

IP : 128. 250. 78. 176

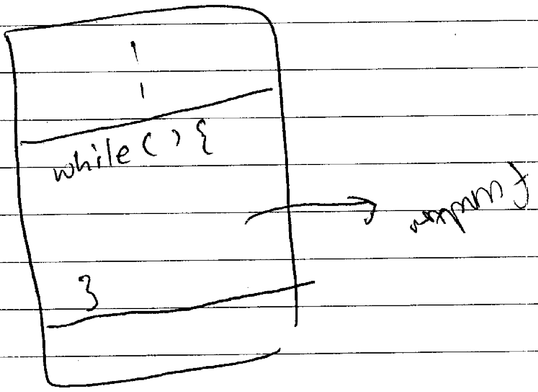
Name : ???. ee. mu. oz. au

windows

subnet mask :	255. 255. 255. 128
WINS	128. 250. 80. 5
Gateway	128. 250. 78. 129

DNS lookup :	128. 250. 80. 1
	128. 250. 20. 2
	128. 250. 1. 21

Domain Suffix :	ee. mu. oz. au unimelb. edu. au



Proxy

http:// www.unimelb.edu.au/cgi-bin/proxy.pac

FTP archive

ftp:// www.unimelb.edu.au

Network Configurator

Names

Host names : aishworya.ee.mu.oz.au
Domain : ee.mu.oz.au

Search for hostnames in add domain

Nameservers
128.250.80.1
128.250.20.2

HOSTS

IP	Name	Nicknames
127.0.0.1	localhost	localhost.localdom
128.250.78.176	aishworya.ee.mu.oz.au	aishwarya

INTERFACES

Interface	IP	proto	atboot	active
Lo	127.0.0.1	none	Y	active
eth0	128.250.78.176	none	Y	active

Routing

Def Gateway 128.250.79.129

Def Gateway Device eth0

23/7/98 What needs to be done to bring stargate On-line?

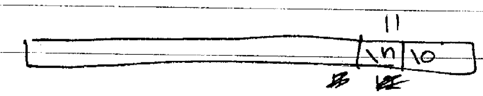
① Setup aishwarya to ~~rlo~~ network

② Server Side

Interface Java applet to server ✓
server to C code

Imbed 'C' code in Java server code as a shared object library

③ Control Side
Talk via a serial communication RS-422 cable from server via C code to controller (rt-linux) to port to UNO

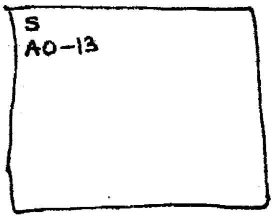


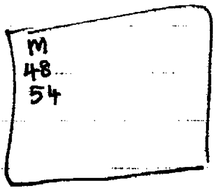
* strlen()
||
12 satna f —

commands:

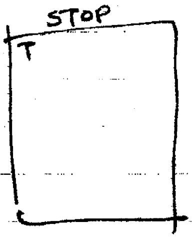
SATTRACK	'S'
STARTRACK	'R'
MANUALTRACK	'M'
STOP	'T'

sattrack

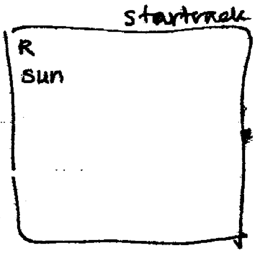




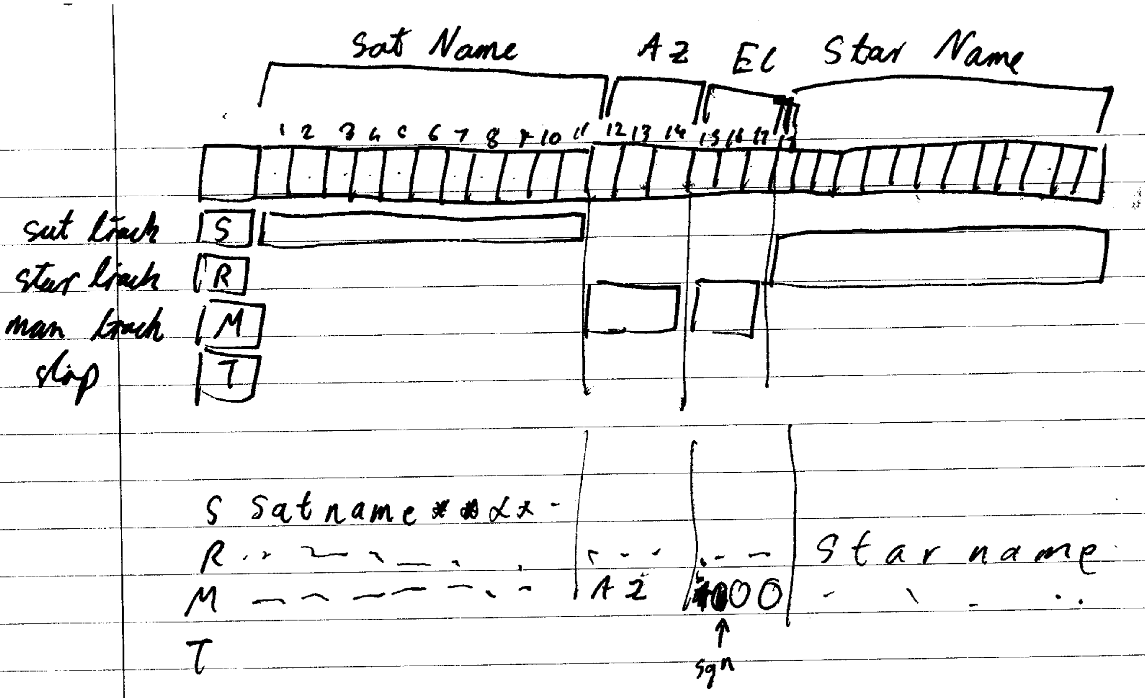
STOP



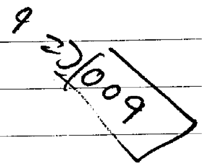
startrack



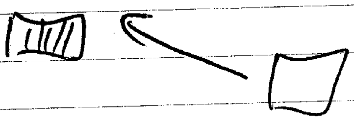
```
int  send_to_serial (char buf[], int len);
int  receive_from_serial (char buff[], int *len);
```



*az:el\n
*000:000\n
123456789



a = get azimuth
e = get elevation
S: satname = satellite name
R: starname = star name
T = stop track
K = start track

Char * 

Seq

21/8/98

Real Time Processing

- Sequencer
- PLC - POLL

Background Tasks

- RS-422 - Server to aish
- RS-422 - aish to PLC
- PC-14 - aish to encoder
- PC-72 - aish to DAC

Radio Telescope Safety Declaration

I have today, 11/9/98, been instructed in the hazards associated with working on the Radio Telescope. These include death through falling or exposure to 415VAC.

I have been instructed in the current Emergency Procedure for the Radio Telescope.

I have been issued with uncontrolled copies of the Radio Telescope Emergency Procedure and Rules of Conduct.

I understand that these Rules of Conduct are to ensure a safe and efficient working environment and that breach of these Rules will result in section 13.1 of University Statutes being invoked.

I understand that the University shall take all reasonable precautions to minimise exposure to hazards.

Signature C Rajadurai 11/9/98

Name Niroshan Carrol Rajadurai

Student Number 25608

RADIO TELESCOPE

EMERGENCY PROCEDURE

On discovery of a fire:

1. All personnel in the area should be alerted.

2. Switch off the mains power immediately at the fuse box.

3. If appropriate, attempt to extinguish the fire using the extinguisher.

4. If fire cannot IMMEDIATELY be extinguished, evacuate the area, closing all doors when all personnel have left the area.

5. Alert Security on 46666.

6. Ensure that all personnel evacuated are present at the assembly point, outside the main doors of the Redmond Barry building, near the lifts.

7. Liase with the Emergency Services when they arrive.

In case of injury:

8. If appropriate, switch off the mains power at the fuse box.

9. If appropriate, administer first aid.

10. If appropriate, contact Security on 46666.

11. Inform the Antenna Manager on 48870, or the Engineering Manager on 44489.

Simon Russell
Antenna Manager
30/1/98

Reviewed 8/9/98

RADIO TELESCOPE

RULES OF CONDUCT

1. No student is to work in the antenna room or on the antenna platform alone.

2. No individual is to work in the antenna room or on the antenna platform whilst an electrical storm is in progress.

3. No individual is to work on the antenna platform during high winds.

4. No modification shall be made to the Radio Telescope system without the express permission of the Antenna Manager.

5. Please do not leave the antenna room in an untidy condition.

6. Any equipment failures should be reported to the Antenna Manager.

7. Telephone numbers:

Antenna Manager	Simon Russell	48870
Academic Supervisors	Dr Iain Collings	46701
	Vaughan Clarkson	45167
Equipment Workshop		46763
Electronics Workshop		47689
Network Administrator	Paul Dwerryhouse	46782
Electrical Workshop		46673
Dept. General Office		46791
Security		46666

Reviewed 8/9/98

Radio Telescope Safety Declaration

I have today, ………………, been instructed ~~in~~ on the hazards associated ~~with~~ while working on the Radio Telescope. These include death through falling or exposure to 415VAC. The University is providing adequate control measures such as fall protection and back up systems *

I have been instructed in the current Emergency Procedure for the Radio Telescope.

I have been issued with uncontrolled copies of the Radio Telescope Emergency Procedure and Rules of Conduct.

I understand that these Rules of Conduct are to be adhered to by ~~responsibilities of~~ the student to ensure ~~a safe and efficient~~ a working environment that is ~~to ensure~~ safety without any risks and that breach of these Rules will result in section 13.1 of University Statutes being invoked.

Signature ………………………………

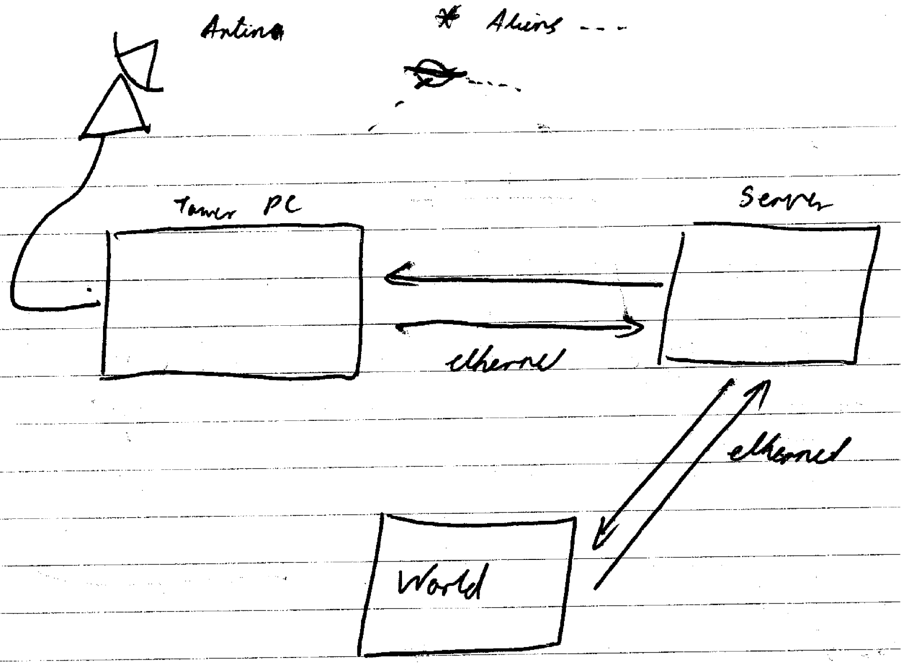
Name ……………………………………

Student Number …………………………

. to control the hazards as far as practicable

*. In the event of an electrical storm, if students are already in the room, facilities will be required to be aware that there is an electrical storm in order that the room can be evacuated

12/9/98



World to Server

Comms:

Java via internet & netscape

Req:

Netscape 4.5 or >

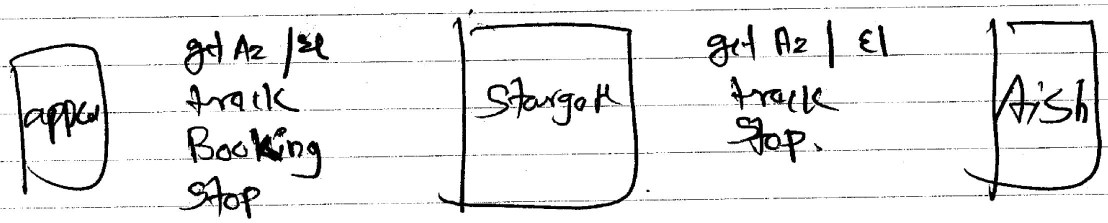
Server _ Aish

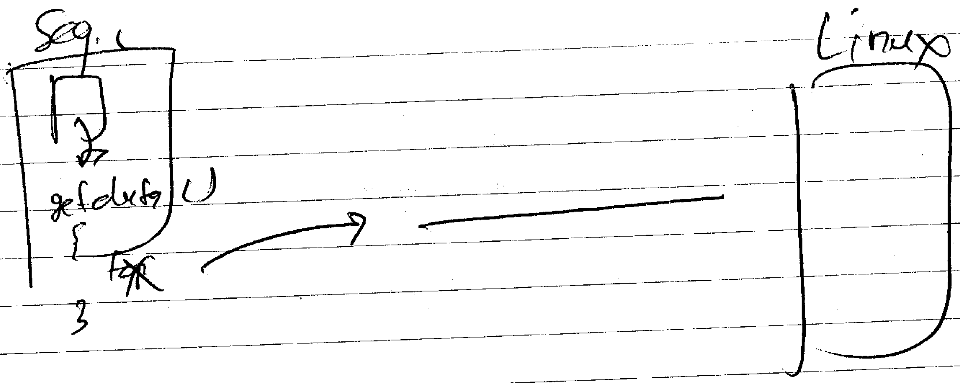
medium: Eth0,

version 4a @ sept 1, 1998

Req : JDK 1.1.6 Vx

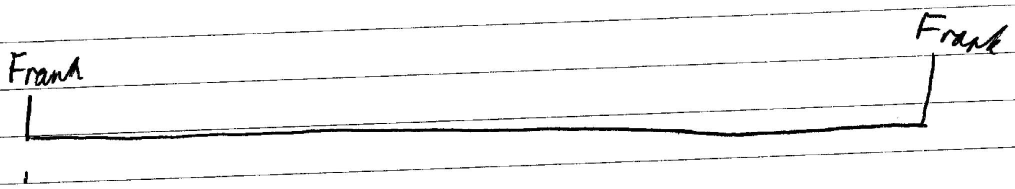
: glibc 2.0.7.x (Red Hat 5.0)

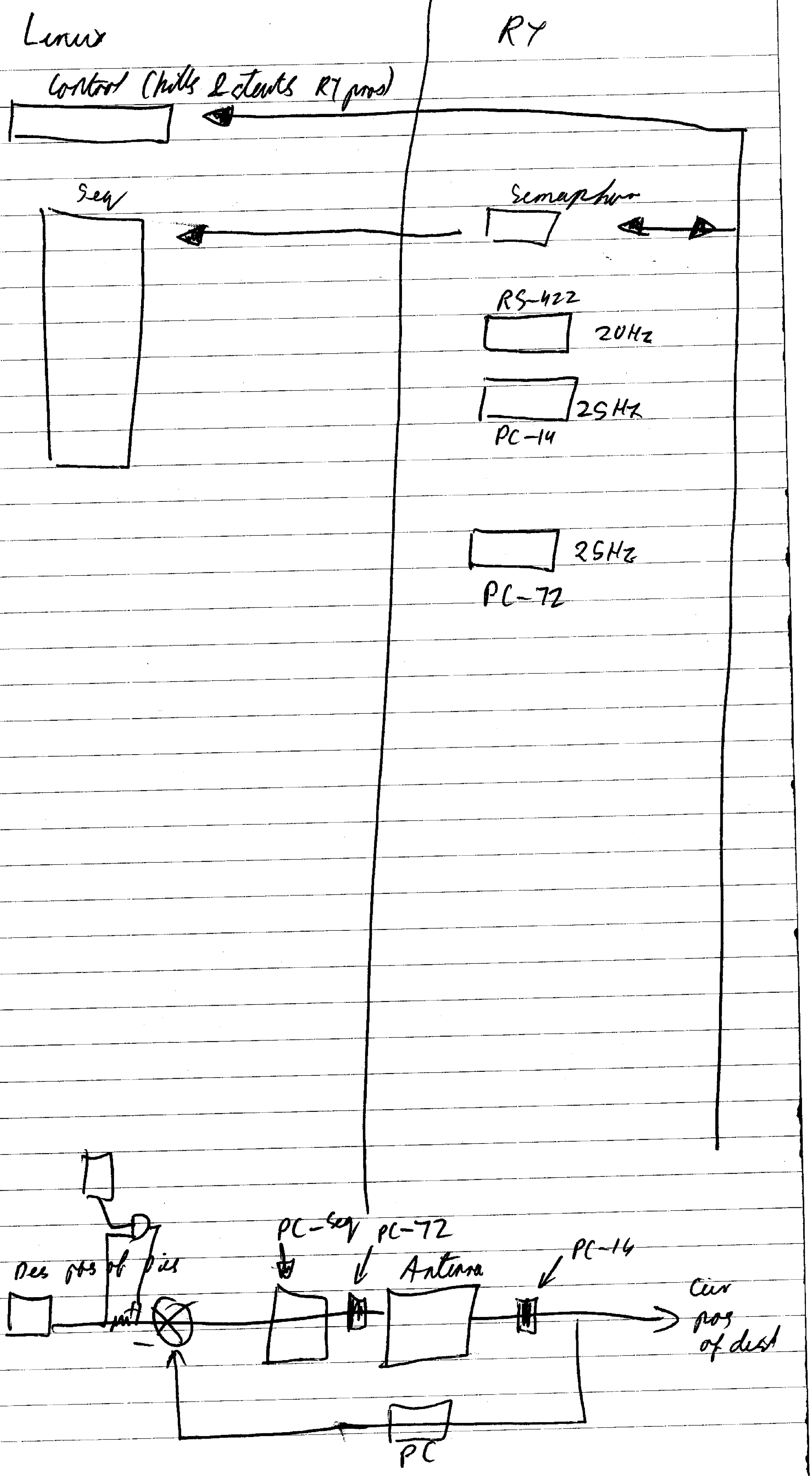




Seq req.

1/ Satellite Data - from file
2/ Commands applet - Java
3/ Commands keyb - keybd




Linux
RT
Control (kills & starts RT proc)
Seq
Semaphore
RS-422
20Hz
25Hz
PC-14
25Hz
PC-72
Des pos of dish
PC-Seq
PC-72
Antenna
PC-14
Cur pos of dish
-
PC

14/9/98

Encoders Tested
and Okay in linux

∴ device drivers for encoders done in Linux

DAC Tested
and okay in linux

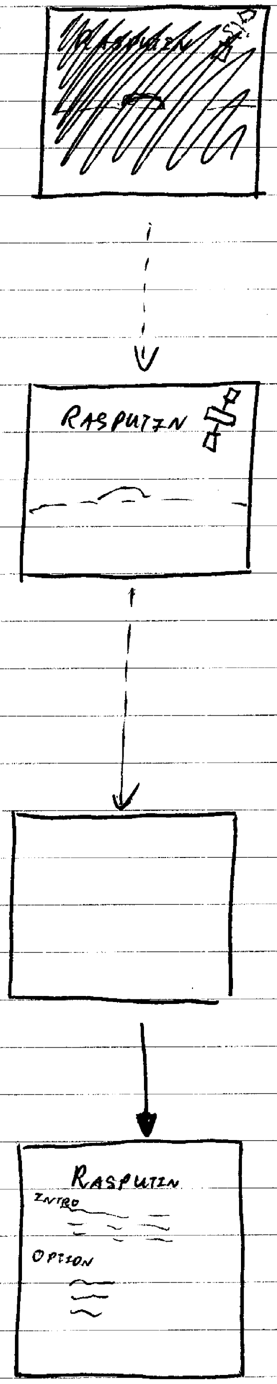
∴ device drivers for DAC done in linux

Web Page Design 15/9/98

Server : Stargate.ee.mu.oz.au
Start page: Stargate.ee.mu.oz.au/index.html
Browser: Netscape 4.06 || Netscape 4.5 PR2 or PR1

/index.html

Java which loads 8 images consequently from Black to White



INTRO PAGE
Splash page
Black White Page which graduates in about 3 or 4 frames to

Front Page
which is about 3 or 4 frames to

White Clear page
after 5 seconds will automatically re-direct to Main Page

www/images/
image 1
↓
image 2
↓
image 3
↓
image 4
↓
image 5
↓
image 6
↓
image 7
↓
image 8

/main_menu.html

Main-menu.
- Will provide users with a brief introduction of the project
- Options:
  1) Use the Dish
  2/ Online Documentation
  3/ ~~Credits/About~~ Requirements
  4/ Credits/About.

23/9/98

RS-422 Card - Old
E 5546

Working Out RC & TR Lines !!

Pin 14 3487
Pin 2 9 Pin

Pin 13 3487
Pin 7 9 Pin

TR DA7

Pin 10 3467
Pin 1 9 Pin

Pin 11 3487
Pin 6 9 Pin

Pa Pin 6 resistor
Pin 14 3486
Pin 4 9 Pin

http://www.perl.org/ (CGI.pm)

Network programming (C)

http://www.lowtek.com/sockets/

Unix Network Programming

W. Richard Stevens

RADIO TELESCOPE

GENERAL INFORMATION

1. No student is to work in the antenna room or on the antenna platform unsupervised by a member of staff.

2. No individual is to work in the antenna room or on the antenna platform whilst an electrical storm is in progress.

3. No individual is to work on the antenna platform during high winds.

4. No modification shall be made to the Radio Telescope system without the express permission of the Antenna Manager.

5. Please do not leave the antenna room in an untidy condition.

6. Any equipment failures should be reported to the Antenna Manager.

7. Telephone numbers:

Antenna Manager	Simon Russell	48870
Academic Supervisor	Dr Iain Collings	46701
Engineering Manager	Vladimir Molotsky	44489
Equipment Workshop		46763
Electronics Workshop		47689
Electrical Workshop		46673
Dept. General Office		46791
Security		46666

Srinath. → email : naga_anjaneya@yahoo.com

RADIO TELESCOPE

EMERGENCY PROCEDURE

On discovery of a fire:

1. All personnel in the area should be alerted.

2. Switch of the mains power immediately at the EMERGENCY STOP button.

3. If appropriate, attempt to extinguish the fire using the extinguisher.

4. If fire cannot IMMEDIATELY be extinguished, evacuate the area, closing all doors when all personnel have left the area.

5. Alert Security on 46666.

6. Ensure that all personnel evacuated are present at the assembly point, outside the main doors of the Redmond Barry building, near the lifts.

7. Liase with the Emergency Services when they arrive.

Incase of injury:

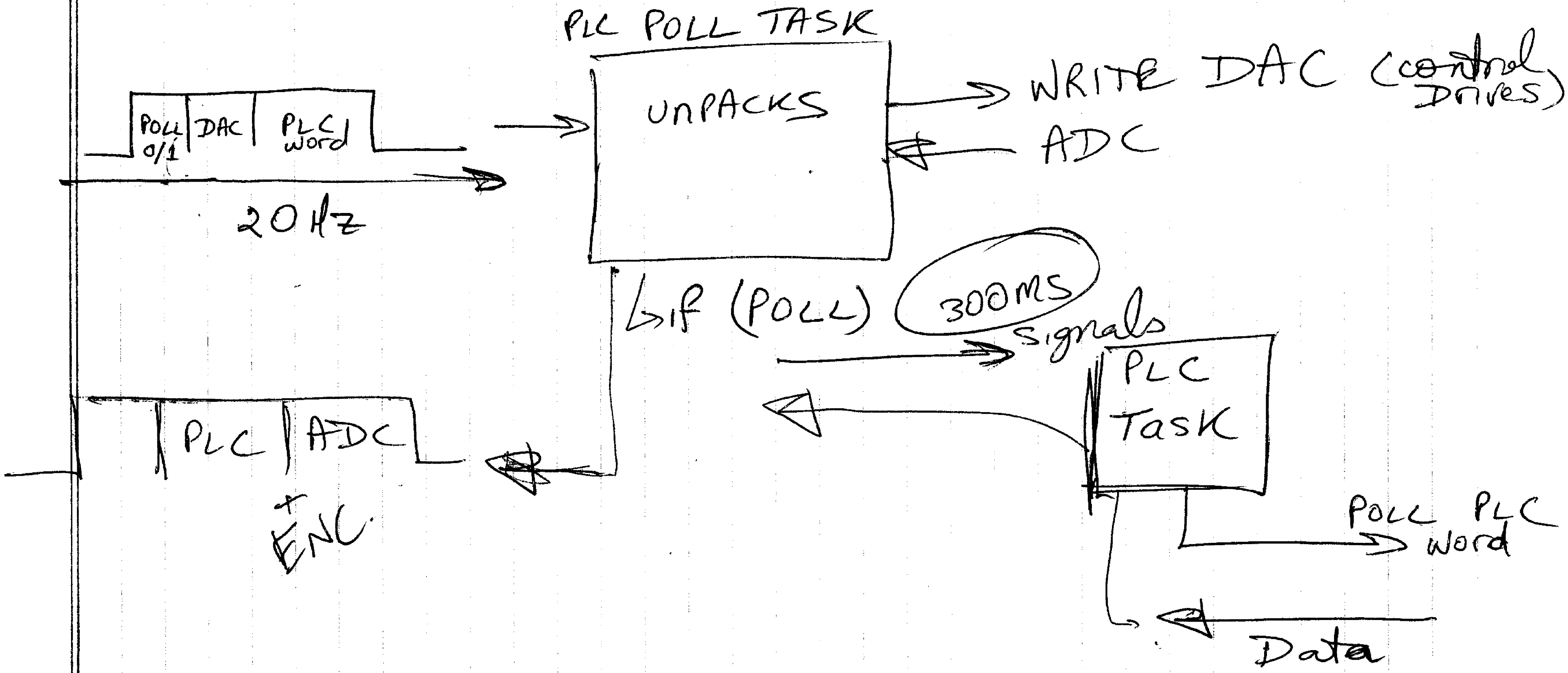
8. If appropriate, switch off the mains power at the EMERGENCY STOP button.

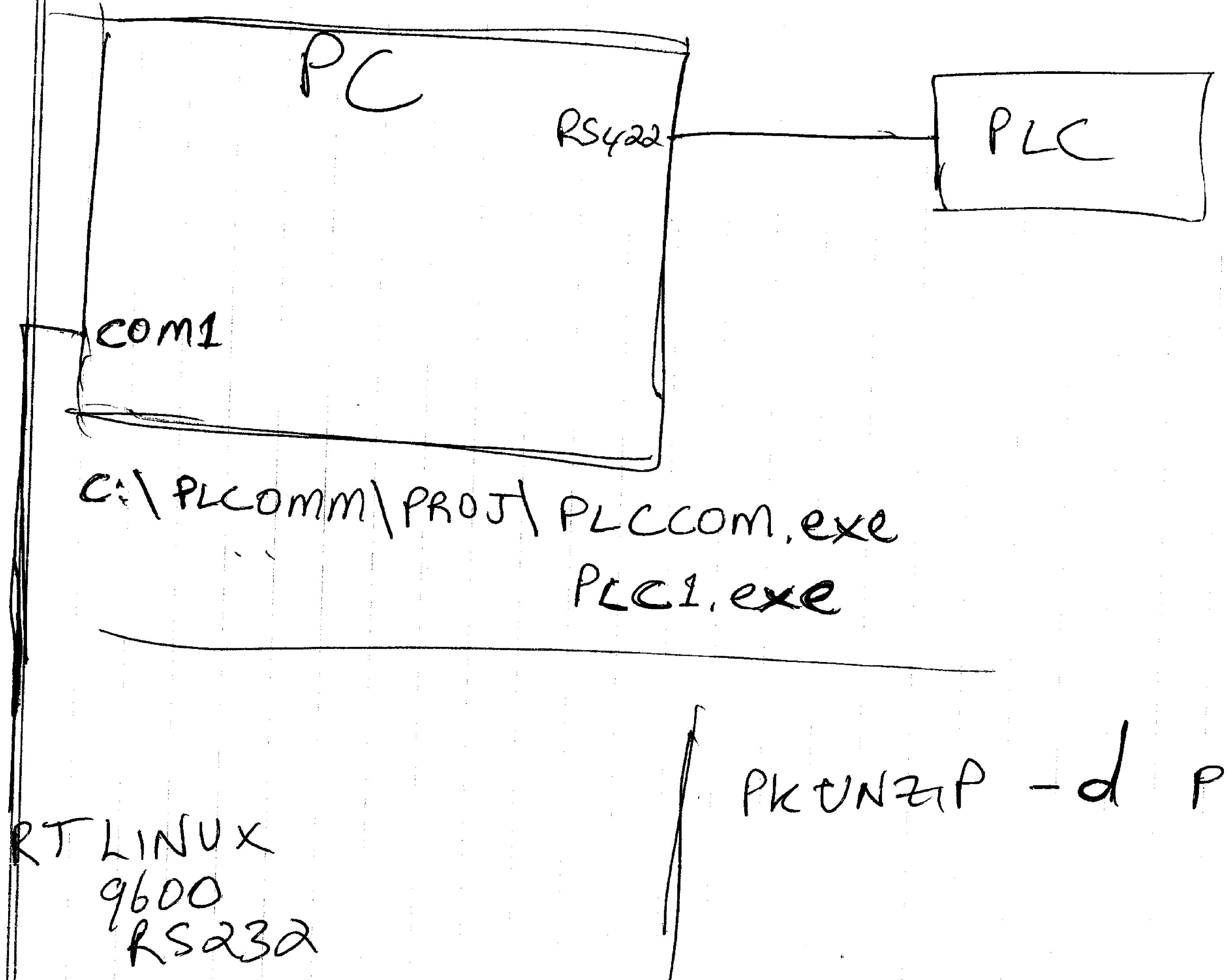
9. If appropriate, administer first aid.

10. If appropriate, contact Security on 46666.

11. Inform the Antenna Manager on 48870, or the Engineering Manager on 44489.

Simon Russell
Antenna Manager
30/1/98


PLC POLL TASK
UNPACKS
WRITE DAC (control Drives)
ADC
POLL 0/1
DAC
PLC word
20Hz
b/f (POLL)
300MS
signals
PLC Task
PLC
ADC
+ ENC
POLL PLC word
Data



C:\PLCOMM\PROJ\PLCCOM.exe

PLC1.exe

```
PKUNZIP -d plc9JUL
```

```
/*
****************************************************************************
PLC.C

Task to communicate to the PLC.

Author:  Len Sciacca
Date: 1994
****************************************************************************
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <conio.h>

#include "general.h"
#include "taskname.h"   // ch_1_tx

#include "unos.h"

#include <unosdef.h>
#include "plc.h"

#include "plcext.h"
#include "prot_ext.h"
#include "main_ext.h"   //LOG_TASK definition
#include "protocol.h"
#include "seq.h"     // Import definisitions, start_high, stop etc
//#include "scrext.h"      /* Import protect_ and unprotect_screen fns  */


#define PLC_Fail      0x01
#define PLC_Trip1     0x02
#define PLC_Trip      0x04
#define PLC_Spare_M99   0x08
#define PLC_24VDC     0x10
#define PLC_AzBrake     0x20
#define PLC_ElBrake     0x40
#define PLC_AzFinalCW   0x80

#define PLC_AzFinalCCW  0x01
#define PLC_ElFinalUp   0x02
#define PLC_ElFinalDwn  0x04
#define PLC_EmergStop   0x08
#define PLC_SmokeDetect 0x10
#define PLC_Door1     0x20
#define PLC_Door2     0x40
#define PLC_CB5_6     0x80

#define PLC_CB4         0x01
#define PLC_CB3         0x02
#define PLC_RUN1      0x04
#define PLC_RUN2      0x08
#define PLC_C1        0x10
#define PLC_C2        0x20


static timer_struc* plc_cycle_timer;
static unsigned int plc_sem;

static unsigned char rx_mess [ 30 ];
static unsigned int rx_mess_length;

static void plc_timer1 ( int * temp );
static char word1, word2, word3;
   // PLC structure contains the actual fault/conditions
   // PLC-bit contains the bit image of the PLC words
static PLC_struct PLC;
static PLC_struct PLC_bit;
static unsigned int task_count= 0;

static char coillower = 0;
static char coilupper = 0;
static char string [ 20 ];

/*
****************************************************************************
```

✓ RS422.

PLC

(ch_1_tx)

UART2_COM_REG_ADDR

CHANNEL2_INTR_NUM.

Server

channel_1

COM1 0x3f8

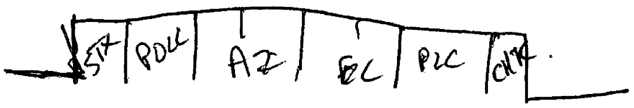
IRQ 4

4+8=12

```
plc_task ( )

 Data is encripted as follows
       Word1:  Bit 0 = Fail    M96
               Bit 1 = Trip1   M97
               Bit 2 = Trip    M98
               Bit 3 = Spare    M99
               Bit 4 = 24VDC OK         M100
               Bit 5 = Az Brake      M101
               Bit 6 = El Brake      M102
               Bit 7 = Az Final CW  M103
      Word2:
               Bit 0 = Az Final CCW M104
               Bit 1 = El Final UP      M105
               Bit 2 = El Final DWN M106
               Bit 3 = Emerg. Stop      M107
               Bit 4 = Smoke Detect M108
               Bit 5 = Door 1        M109
               Bit 6 = Door 2        M110
               Bit 7 = CB5/6         M111
      word 3:
               Bit 0 = CB4           M112
               Bit 1 = CB3           M113
               Bit 2 = RUN 1         M114
               Bit 3 = RUN 2         M115
               Bit 4 = C1            M116
               Bit 5 = C2            M117
               Bit 6 = Spare         M118
               Bit 7 = Spare         M119

*****************************************************************************
*/
void plc_task ( void * Dummy ) {

   int temp, i;
   unsigned int return_status;
   char * status;
   char ch1, ch2;
   FILE *fp;

   enable ( );
   Dummy = Dummy;

   plc_sem = create_semaphore();
   if(plc_sem != 0xffff)
      init_semaphore (plc_sem, 0, 1 );

   /*---- Set up any task dependent variables, incl timers */
   /* Need to run every 2 time ticks if tick = 30Hz */
   plc_cycle_timer = start_timer ( (unsigned char)REPETITIVE,
                                 (unsigned long)10,
                                 (void (*)(int *))plc_timer1, (void*)&temp );

   while ( 1 ) {

      task_count++;
      if (task_count > 500)
         task_count = 0;

      wait ( plc_sem );
      // send a poll to the PLC and wait for a reply.
      // 010c = M
      // 0080 = x0

      // Send poll using form_and_send_message routine (protocol.c)
      form_and_send_message ( "010C03", ch_1_tx, '0' );
      // Now, wait for reply from Message from rx_protocol_task
      status = rcv_mess( rx_mess, &rx_mess_length, 5 );

      // returns a NULL if timeout
      // Only process if ACK is returned and not time-out
      if ( ( status != NULL ) && ( rx_mess [0] != 0x15 ) )
         {
         for (i=0; i < 6; i++ )
            {
            if ( rx_mess [ i ] == 'F' )
               rx_mess [ i ] = 0x3f;
            }
      // Now strip off the data from the PLC reply
      word1 = ((rx_mess [ 0 ]-0x30 )<<4) | ( rx_mess[1]-0x30);
      word2 = ((rx_mess [ 2 ]-0x30 )<<4) | ( rx_mess[3]-0x30);
      word3 = ((rx_mess [ 4 ]-0x30 )<<4) | ( rx_mess[5]-0x30);

      // Now send the set data
      // e.g. 0990 = 9009 => M400
      //form_and_send_message ( "9009", ch_1_tx, '7' );
      // send 1 byte of data to set coils M400 - M407
      // defined by coilupper and coillower
      // Group address for M400 is 0132
      string [ 0 ] = NULL;
      strcat ( string, "013201" );
      ch1 = coillower + 0x30; // convert to ascii
      ch2 = coilupper + 0x30;

      // Now add ch1 and ch2 to string
      strncat ( string, &ch2, 1 );
      strncat ( string, &ch1, 1 );

      form_and_send_message ( string, ch_1_tx, '1' );
      status = rcv_mess( rx_mess, &rx_mess_length, 3 ); // wait for ACK

      // if status ==  NULL then timeout, if rx_mess == NACK then error

      PLC_bit.Fail = (( PLC_Fail & word1 ) == PLC_Fail );
      PLC_bit.Trip1= (( PLC_Trip1 & word1 ) == PLC_Trip1 );
      PLC_bit.Trip = (( PLC_Trip & word1 ) == PLC_Trip );
      PLC_bit.C24VDC = (( PLC_24VDC & word1 ) == PLC_24VDC );
      PLC_bit.AzBrake   = (( PLC_AzBrake & word1 ) == PLC_AzBrake );
      PLC_bit.ElBrake   = (( PLC_ElBrake & word1 ) == PLC_ElBrake );
      PLC_bit.AzFinalCW = (( PLC_AzFinalCW & word1 ) == PLC_AzFinalCW );

      PLC_bit.AzFinalCCW = (( PLC_AzFinalCCW & word2 ) == PLC_AzFinalCCW);
      PLC_bit.ElFinalUp  = (( PLC_ElFinalUp & word2 ) == PLC_ElFinalUp );
      PLC_bit.ElFinalDwn = (( PLC_ElFinalDwn & word2 ) == PLC_ElFinalDwn );
      PLC_bit.EmergStop  = (( PLC_EmergStop & word2 ) == PLC_EmergStop );
      PLC_bit.SmokeDetect  = (( PLC_SmokeDetect & word2 ) == PLC_SmokeDetect );
      PLC_bit.Door1       = ((PLC_Door1 & word2 ) == PLC_Door1 );
      PLC_bit.Door2      = ((PLC_Door2 & word2 ) == PLC_Door2 );
      PLC_bit.CB5_6      = ((PLC_CB5_6 & word2 ) == PLC_CB5_6 );

      PLC_bit.CB4        = ((PLC_CB4 & word3 ) == PLC_CB4 );
      PLC_bit.CB3        = ((PLC_CB3 & word3 ) == PLC_CB3 );
      PLC_bit.RUN1       = ((PLC_RUN1 & word3 ) == PLC_RUN1 );
      PLC_bit.RUN2       = ((PLC_RUN2 & word3 ) == PLC_RUN2 );
      PLC_bit.C1         = ((PLC_C1 & word3 ) == PLC_C1 );
      PLC_bit.C2         = ((PLC_C2 & word3 ) == PLC_C2 );

      //PLC = PLC_bit;
      PLC.Fail = PLC_bit.Fail;
      PLC.Trip1= !PLC_bit.Trip1;
      PLC.Trip = !PLC_bit.Trip;
      PLC.C24VDC = !PLC_bit.C24VDC;
      PLC.AzBrake = PLC_bit.AzBrake;
      PLC.ElBrake = PLC_bit.ElBrake;
      PLC.AzFinalCW = !PLC_bit.AzFinalCW;

      PLC.AzFinalCCW = !PLC_bit.AzFinalCCW;
      PLC.ElFinalUp  = !PLC_bit.ElFinalUp;
      PLC.ElFinalDwn = !PLC_bit.ElFinalDwn;
      PLC.EmergStop  = !PLC_bit.EmergStop;
      PLC.SmokeDetect   = PLC_bit.SmokeDetect;
      PLC.Door1       = PLC_bit.Door1;
      PLC.Door2      = PLC_bit.Door2;
      PLC.CB5_6      = PLC_bit.CB5_6;

      PLC.CB4        = !PLC_bit.CB4;
      PLC.CB3        = !PLC_bit.CB3;
      PLC.RUN1    = PLC_bit.RUN1;
      PLC.RUN2    = PLC_bit.RUN2;
      PLC.C1         = PLC_bit.C1;
      PLC.C2         = PLC_bit.C2;

      } // if
   else
      flush_mbx ( );     // clear mailbox and start afresh

} /* end of infinite while loop */
} /* End of plc_task */


/*
*************************************************************************
plc_timer

   Routine called on the time-out of the timer created in PLC task.
This routine merely signals the sequencer semaphore. The sequencer will then
be scheduled as the next task as it should be the highest priority.

*************************************************************************
*/
void plc_timer1 ( int * temp ) {

   _signal ( plc_sem );
   *temp = 1;     /* avoids warning on compilation only */

} /* End of plc_timer */


/*
*****************************************************************************
ReadPLC ( )

Routine to return pointer to information read by PLC Polls.

*****************************************************************************
*/
void ReadPLC ( PLC_struct * tempPLC ) {

   *tempPLC = PLC;
} // end of ReadPLC

/*
*****************************************************************************
ReadPLCWords ( )

Routine to return pointer to information read by PLC Polls.
These are the raw bytes sent by PLC. User must decipher.
*****************************************************************************
*/
void ReadPLCWords (  char *w1,  char *w2,
                char *w3 ) {
   disable();
   *w1 = word1;
   *w2 = word2;
   *w3 = word3;
   enable();

} // end of ReadPLCWords

/*
*****************************************************************************
WritePLC ( )

Routine to set the two bytes to be sent to the PLC

*****************************************************************************
*/
void WritePLC ( int drive_word ) {

   if ( start_high & drive_word )    // m400
      coillower = coillower | 0x01;
   else
      coillower = coillower & 0x0e;

   if ( stop_closed & drive_word ) // m401
      coillower = coillower | 0x02;
   else
      coillower = coillower & 0x0d;

   if ( poweron & drive_word )      // m402
      coillower = coillower | 0x04;
   else
      coillower = 0; //coillower & 0x0b;
   if ( lights_on & drive_word )  // m403
      coillower = coillower | 0x08;
   else
      coillower = coillower & 0x07;

   if ( reset_high & drive_word ) // m404
      coilupper = coilupper | 0x01;
   else
      coilupper = coilupper & 0x0e;

   coilupper = coilupper | 0x02;    // Watchdog thing

} // end of WritePLC

/*
*****************************************************************************
return_plctask_ctr ( )

   Routine to pass back the plc task counter.
   Used for diagnostic to check the task is alive

Called from screen task when required.

*****************************************************************************
*/
unsigned int return_plctask_ctr (void)
{
   unsigned int   result;

   disable ();
   result = task_count;
   enable ();

   return  ( result );
}   /* end of return_plctask_ctr */
```

```
/*
*****************************************************************************
PLCpoll.C

Task to communicate to the PLC from external PC

Author:  Len Sciacca
Date: 1994/1998

plcpoll_message expects a poll in form,
STX          - 1 byte
POLL         - 1 byte
DAC AZ       - 2 bytes
DAC EL       - 2 bytes
Drive word   - 1 byte
Checksum     - 2 bytes
```

FROM LINUX



```
Total: 9bytes
If there is a timeout ( 1 sec) then a error is flagged.
This routine is called from the plcpoll task
If there is a proper poll, then send back the current data

STX          - 1 byte
AZ Encoder - 2 bytes
EL Encoder - 2 bytes
ADC          - 2 bytes
plc word 1 - 1 byte   (see plc.c)
plc word 2 - 1 byte   (see plc.c)
plc word 3 - 1 byte   (see plc.c)
checksum     - 2 bytes

Total: 12 bytes
*****************************************************************************
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>

#include "general.h"
#include "taskname.h"    // ch_1_tx

#include "unos.h"

#include <unosdef.h>
#include "plc.h"
#include "plcext.h"    // extern ReadPLCWords()

#define NUL 0x00
#define SOH 0x01
#define STX 0x02
#define ETX 0x03
#define EOT 0x04
#define ENQ 0x05
#define ACK 0x06
#define BEL 0x07
#define BS  0x08
#define HT  0x09
#define LF  0x0A
#define VT  0x0B
#define FF  0x0C
#define CR  0x0D
#define SO  0x0E
#define SI  0x0F
static unsigned char plcpoll_message( char *message, int time_out );
static unsigned char rx_mess [ 30 ];
static unsigned int rx_mess_length;

static char word1, word2, word3;
   // PLC structure contains the actual fault/conditions
   // PLC-bit contains the bit image of the PLC words
static PLC_struct PLC;
static PLC_struct PLC_bit;
static unsigned int task_count= 0;
static unsigned int count=0;
static unsigned int checksum_count=0;

static char string [ 20 ];
```

```
/*
****************************************************************************
plcpoll_task ( )

****************************************************************************
*/
void plcpoll_task ( void * Dummy ) {

   int temp, i;
   unsigned int return_status;
   unsigned char status;
   char ch1, ch2;
   char sendbuf [40];
   unsigned char adc[2], azencoder[2], elencoder[2], dacaz[2], dacel[2];
   unsigned char word1, word2, word3, PLCPoll;
   unsigned char checksum, driveword;

   enable ( );
   Dummy = Dummy;

   // Attach to serial channel 0
   //This initialises the rx task with who it needs to send
   // it's data to
   send_mess( (unsigned char *)"?",1, ch_0_rx );

   while ( 1 ) {

      // Wait for Poll from external comms
      // if timeout...send a character down the line, just to tell
      // someone we are alive!
      status = plcpoll_message( rx_mess, 20 );

      // Do a simple test by checking for a character from a terminal
      // and replying with the required data!

      // returns a NULL if timeout
      // Only process if ACK is returned and not time-out
      if ( ( status != NULL ) && ( rx_mess [0] == STX ) )
         {
         checksum=0;
         for (i=0; i < 7; i++ )
            {
            checksum += (unsigned char)rx_mess[ i ];
            }
         // check checksum
         if (checksum==rx_mess[7])
           {
           // Now strip off the data from the PLC reply
           PLCPoll =rx_mess[1];
           dacaz[0]=rx_mess[2];
           dacaz[1]=rx_mess [3];
           dacel[0]=rx_mess[4];
           dacel[1]=rx_mess[5];
           // 1. Unpack PLC Command in "drive_word"
           driveword=rx_mess[6];
           }
         else
         {
         // error
         }


         } // if
      else
         {
         flush_mbx ( );     // clear mailbox and start afresh
         //send_mess( (unsigned char *)"?",1, ch_0_tx );       // send message to
                                                              // anyone
         } // else

      // Read Encoders
      azencoder[0]=0xff;
      azencoder[1]=0xee;
      elencoder[0]=0xdd;
      elencoder[1]=0xcc;
      // Read ADC
      adc[0]=0xaa;
      adc[1]=0xbb;
      // Read PLC
      // Returns the bytes, word1, word2 word3 which the user must decipher
      ReadPLCWords (&word1, &word2, &word3);

      // Pack the reply and send
      sendbuf[0]=STX;
      sendbuf[1]=azencoder[0];
      sendbuf[2]=azencoder[1];
      sendbuf[3]=elencoder[0];
      sendbuf[4]=elencoder[1];
      sendbuf[5]=adc[0];
      sendbuf[6]=adc[1];
      sendbuf[7]=word1;
      sendbuf[8]=word2;
      sendbuf[9]=word3;
      sendbuf[10]=NULL;

      // Make checksum
      checksum = 0;
      for( i = 1; i < (strlen(sendbuf)); i++ )
        {
        checksum += (unsigned char)sendbuf[ i ];
        }
      sendbuf[10]=checksum;
      sendbuf[11]=NULL;

      if ( free_mbx ( ch_0_tx ) > 1 )
        send_mess( (unsigned char *)sendbuf, strlen( sendbuf ), ch_0_tx );

      // Write PLC data, PLC task will send it off when it is ready.
      WritePLC (driveword);

   } /* end of infinite while loop */

} /* End of plcpoll_task */
```

```
/**********************************************************************
* plcpoll_message
* This routine is designed to receive data from a Rx serial handler task,
* and collects the string until a LF is received or an error has occured.
* It then performs checking/stripping as per the Mitsubishi protocol.
* An error code is returned if required. The string will be returned via
* the calling parameter string.
* (N.B. Re-entrant routine.)
*
***********************************************************************/

static unsigned char plcpoll_message( char *message, int time_out )
{
   unsigned int error = 0, readchecksum = 0;
   unsigned char checksum = 0;
   char buf[ 30 ];
   int enddata, i, message_index = 0;
   char EndMessage;
   char * status;

   rx_mess[0]='\0';

   message_index = 0;
   EndMessage = 0;

   i=1;

   count++;
   checksum_count = 0;

   do
   {
      /* Wait for bytes to arrive from the serial handler. */
      status = rcv_mess( &rx_mess[0], &rx_mess_length, time_out );
      // check for timeout
      if (status == NULL)
        {
        rx_mess[0]=NULL;
        EndMessage=1;
        message_index=1;
        error=1;
```

```
     }

   if ( rx_mess[0] == STX )  // hope first is STX
      {
      if (message_index>0)
         EndMessage = 1;
      else {
        message_index = 0;
        i=1;
        message [ message_index ] = rx_mess[0];
        checksum_count = 1;
        message_index++;
        }
      }
   else if ( message_index > 0 )
      {
      message[ message_index ] = rx_mess[0];
      message_index++;
      checksum_count++;

      if ((checksum_count == 9))
         {
         EndMessage = 1;
         message[message_index]=NULL;
         message_index = 0;
         }
      }

   i++;

} while ( !EndMessage );     // DO


if ( error )
   return error;

/* The message passed to this function should start with an STX*/
enddata = strlen( message );      /* N.B. Length does not include NULL. */


switch  ( message[ 0 ] )
{
   case NULL:
            return  (NULL);  // timeout

   case STX:
            /* Calculate the checksum */
            checksum = 0;
            for( i = 1; i < (enddata-2); i++ )
               {
               checksum += (unsigned char)message[ i ];
               }

            /* Now read the checksum and compare */
            strcpy( buf , &message[ enddata-2 ] );

            if ( (sscanf( buf, "%x", &readchecksum ) == 1) &&
                  (((unsigned int)checksum)&0xff) == readchecksum) )
               {
               /* Remove the checksum and the ETX */
               message[ enddata-3 ] = NULL;

               /* Now shift the data area */
               enddata = strlen(message);
               for( i = 1; i <= enddata; i++ )
                  message[i-1] = message[i];

               return 0;
               }
            else
               {
               /* We have a problem, return the error */
               /* Either the scan failed, or the checksum != */
               return  (-1);
               }

   default   : return  (-1);
} // switch
```

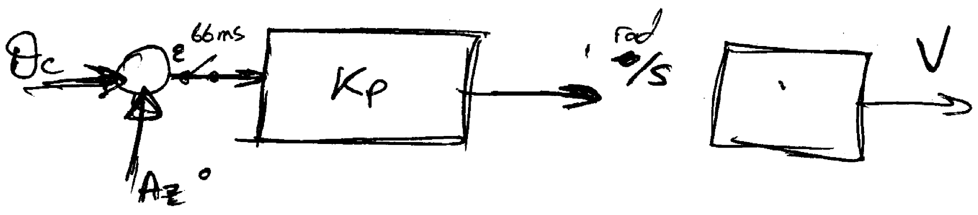
```
}
```

```
/*
****************************************************************************
                Antenna Tracking Control Unit

                    PLC Screen Routines for PC

_author: L. J. Sciacca

_latest: 22-Oct-1990
        22-oct-1990
        29-march-1991 ljs - Clean up of code.
        6-march-1995   sto - this module was created as a separate entity
                    from plc.c by moving all functions related to
                    screen displays. This was done because these routines
                    are called from the screen task, whereas the ones
                    in plc.c are called from the plc task.
        21-march-1996 ksc/tzqh - added the display_function_key function and
                    the draw line functions..
Description.

    These routines are to be called from the screen task to update the
    screen showing PLC variables.


    NOTE: THIS WILL BE THE ONLY PAGE ON THE NEW PLC COMMS SYSTEM
    JULY 1998

****************************************************************************

#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <string.h>

#include "unos.h"
#include "general.h"

#include <unosdef.h>
#include "plc.h"

#include "plcscr.h"
#include "pcscr.h"       /* for line drawings */
#include "kbtask.h"      /* Import page definitions */
#include "seq.h"        /* Import definisitions, start_high, stop etc */
#include "scrext.h"      /* Import protect_ and unprotect_screen fns  */
#include "protocol.h"

extern void ReadPLC ( PLC_struct * PLC );  /* Import from plc.c */

static PLC_struct PLC;
static char display_string [ 100 ];
static int task_count;

/*
****************************************************************************
Init_PLC_Screen

Routine to display template for PLC task variables.

****************************************************************************
*/
static void InitPLC_Screen ( void ) {

int i;

clrscr ( );

protect_screen();

gotoxy(1,1);cprintf(
 "              PLC Screen                    UniMelb Antenna");

pcscr_draw_line ( 79, 1, 2, HOR, 1 );

   gotoxy ( 1, 4 ); cprintf ( "%s", "Fail" );
   gotoxy ( 1, 5 ); cprintf ( "%s" , "Trip1" );
   gotoxy ( 1, 6 ); cprintf ( "%s" , "Trip" );
   gotoxy ( 1, 7 ); cprintf ( "%s" , "C24VDC" );
   gotoxy ( 1, 8 ); cprintf ( "%s" , "AzBrake" );
```

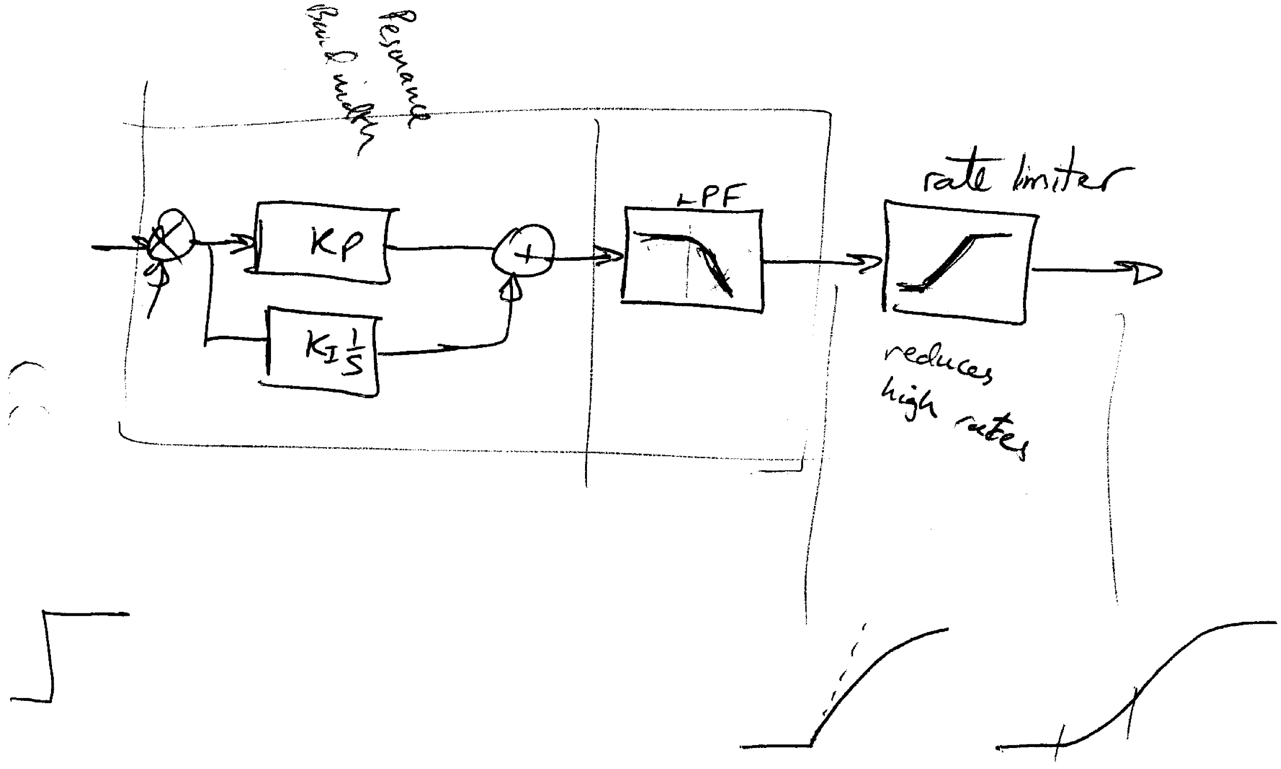
```
      gotoxy ( 1, 9 ); cprintf ( "%s" , "ElBrake" );
      gotoxy ( 1, 10 ); cprintf ( "%s" , "AzFinalCW" );
      gotoxy ( 1, 11 ); cprintf ( "%s" , "AzFinalCCW" );
      gotoxy ( 1, 12 ); cprintf ( "%s" , "ElFinalUp" );
      gotoxy ( 1, 13 ); cprintf ( "%s" , "ElFinalDwn" );
      gotoxy ( 1, 14 ); cprintf ( "%s" , "EmergStop" );
      gotoxy ( 1, 15 ); cprintf ( "%s" , "SmokeDetect" );
      gotoxy ( 1, 16 ); cprintf ( "%s" , "Door1" );
      gotoxy ( 1, 17 ); cprintf ( "%s" , "Door2" );
      gotoxy ( 1, 18 ); cprintf ( "%s" , "CB5_6" );
      gotoxy ( 1, 19 ); cprintf ( "%s" , "CB3" );
      gotoxy ( 1, 20 ); cprintf ( "%s" , "CB4" );
      gotoxy ( 1, 21 ); cprintf ( "%s" , "RUN1" );
      gotoxy ( 1, 22 ); cprintf ( "%s" , "RUN2" );
      gotoxy ( 1, 23 ); cprintf ( "%s" , "C1 - C2" );

for ( i=1; i < 79 ; i++ ) {
          gotoxy ( i, 24 );
          cprintf ( "\xc4" );
          }

//display_function_keys();
//Display the input from the remote!!

unprotect_screen ();
} // end of InitPLC_Screen


/***************************************************************************
PLC_Screen ( )

Routine to display IO stuff on screen

****************************************************************************
*/
void PLC_Screen ( ) {

InitPLC_Screen ( );

display_string [ 0 ] = NULL;

while  ( return_screen_page() == PLC_PAGE )
   {
   update_screen ( 1 );

   ret_protocol_mess ( display_string );

   task_count = return_plctask_ctr ( );

   readPLC ( &PLC );

   protect_screen ();

   gotoxy ( 3, 3 ); cprintf ( "%x %s", display_string[0], display_string );

/* gotoxy ( 40, 3 ); cprintf ("%s ", rx_mess );
   gotoxy ( 60, 3 ); cprintf ("%x", rx_mess[0] ); */

   gotoxy ( 60, 8 ); cprintf ("C: %u", task_count );

/* Checking the messages sent and received by the PLC task */
/* gotoxy ( 60, 4 ); cprintf ("Coils: %x   %x", coillower, coilupper );
     gotoxy ( 60, 5 ); cprintf ("       %s", string );
     gotoxy ( 60, 6 ); cprintf ("%x  %x  %x", rx_mess[2], rx_mess[3], word2 ); */

   gotoxy ( 18,  4 ); cprintf ( "%x", PLC.Fail );
   gotoxy ( 18,  5 ); cprintf ( "%x", PLC.Trip1 );
   gotoxy ( 18,  6 ); cprintf ( "%x", PLC.Trip );
   gotoxy ( 18,  7 ); cprintf ( "%x", PLC.C24VDC );
   gotoxy ( 18,  8 ); cprintf ( "%x", PLC.AzBrake );
   gotoxy ( 18,  9 ); cprintf ( "%x", PLC.ElBrake );
   gotoxy ( 18,  10 ); cprintf ( "%x", PLC.AzFinalCW );
   gotoxy ( 18,  11 ); cprintf ( "%x", PLC.AzFinalCCW );
   gotoxy ( 18,  12 ); cprintf ( "%x", PLC.ElFinalUp );
   gotoxy ( 18,  13 ); cprintf ( "%x", PLC.ElFinalDwn );
   gotoxy ( 18,  14 ); cprintf ( "%x", PLC.EmergStop );
   gotoxy ( 18,  15 ); cprintf ( "%x", PLC.SmokeDetect );
   gotoxy ( 18,  16 ); cprintf ( "%x", PLC.Door1 );
   gotoxy ( 18,  17 ); cprintf ( "%x", PLC.Door2 );
   gotoxy ( 18,  18 ); cprintf ( "%x", PLC.CB5_6 );
   gotoxy ( 18,  19 ); cprintf ( "%x", PLC.CB4 );
   gotoxy ( 18,  20 ); cprintf ( "%x", PLC.CB3 );
   gotoxy ( 18,  21 ); cprintf ( "%x", PLC.RUN1 );
   gotoxy ( 18,  22 ); cprintf ( "%x", PLC.RUN2 );
   gotoxy ( 18,  23 ); cprintf ( "%x", PLC.C1 + PLC.C2 );

   unprotect_screen ();
   }

} /* end of PLC_Screen */
```

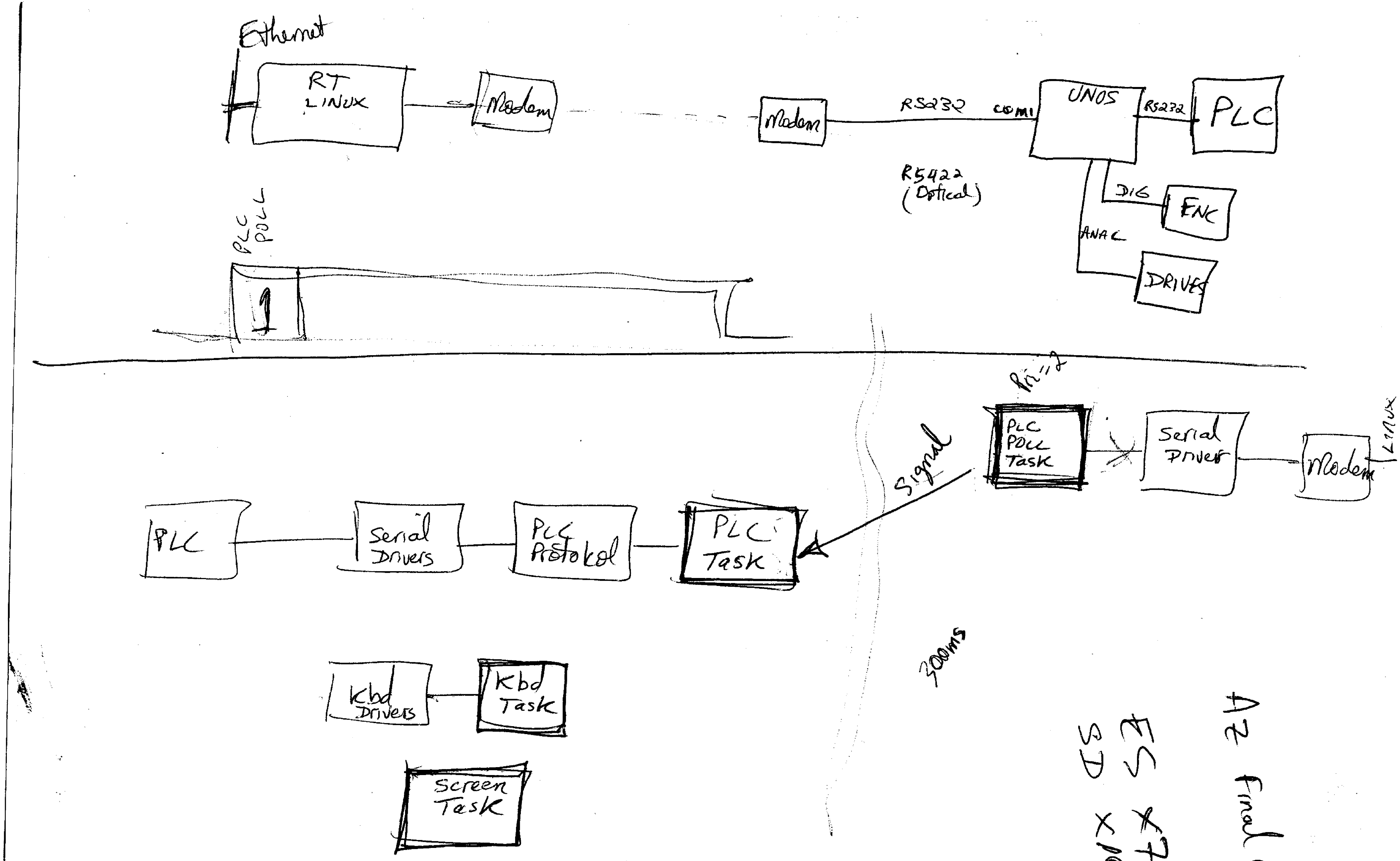
① adaptation_switch = Ø in seq.c for all modes.

②.



0·7
4~~0~~ Hz





Az Final CCW (LL) x4

ES x7
SD x10

```
mkdir /a

mount /dev/hda2 /a

fdisk        p
```

Linux

① Format floppy

```
fdformat /dev/fd0
```

② Data Dump

```
dd if='filename' of='/dev/fd0'
```

③ Kernal

```
rpm -Uvvh --nodeps
```

header
Source

/usr/src

④ /boot

boot img ⇐

⑤ RPM

```
make xconfig ← cpu
make dep
make clean
make ZImage
make modules
make modules-install
cd/boot
cp /usr/src/linux/arch/i386/boot/
                              ZImage .
vi /etc/lilo.conf
/sbin/lilo
/sbin/reboot
```