Master Project Presentation

Internet accessible embedded system for distance measuring

Author: Dilyana Panayotova
Technical University - Sofia, branch Plovdiv,
Department Electronics and Automation

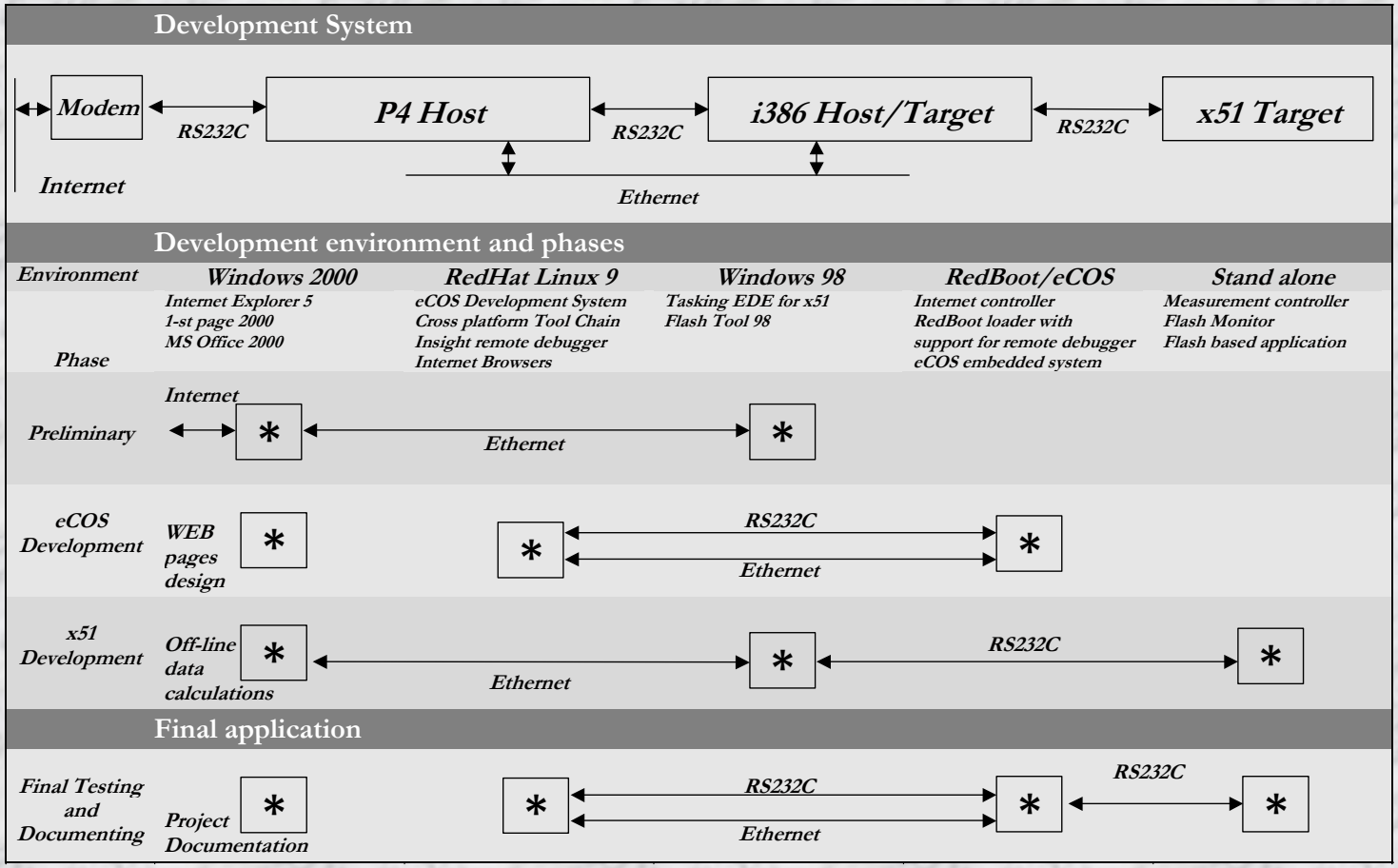
Supervisor:
M.Sc. Dr. Valentin Mollov
Head of CE Department

Consultant:
Christo Radev
CS/IT Expert

Referee:

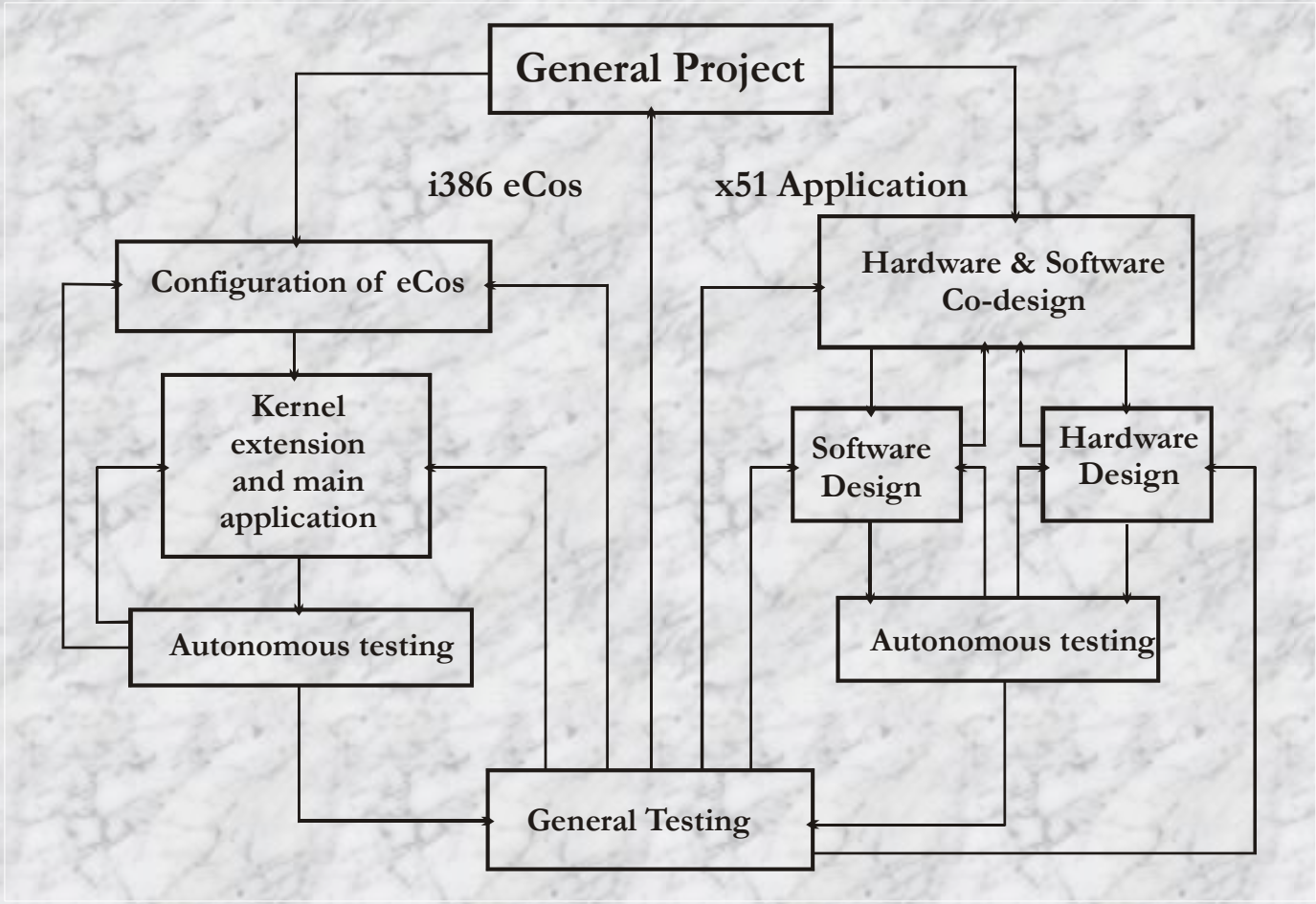
Introduction

Functional schematic of Development System in all phases



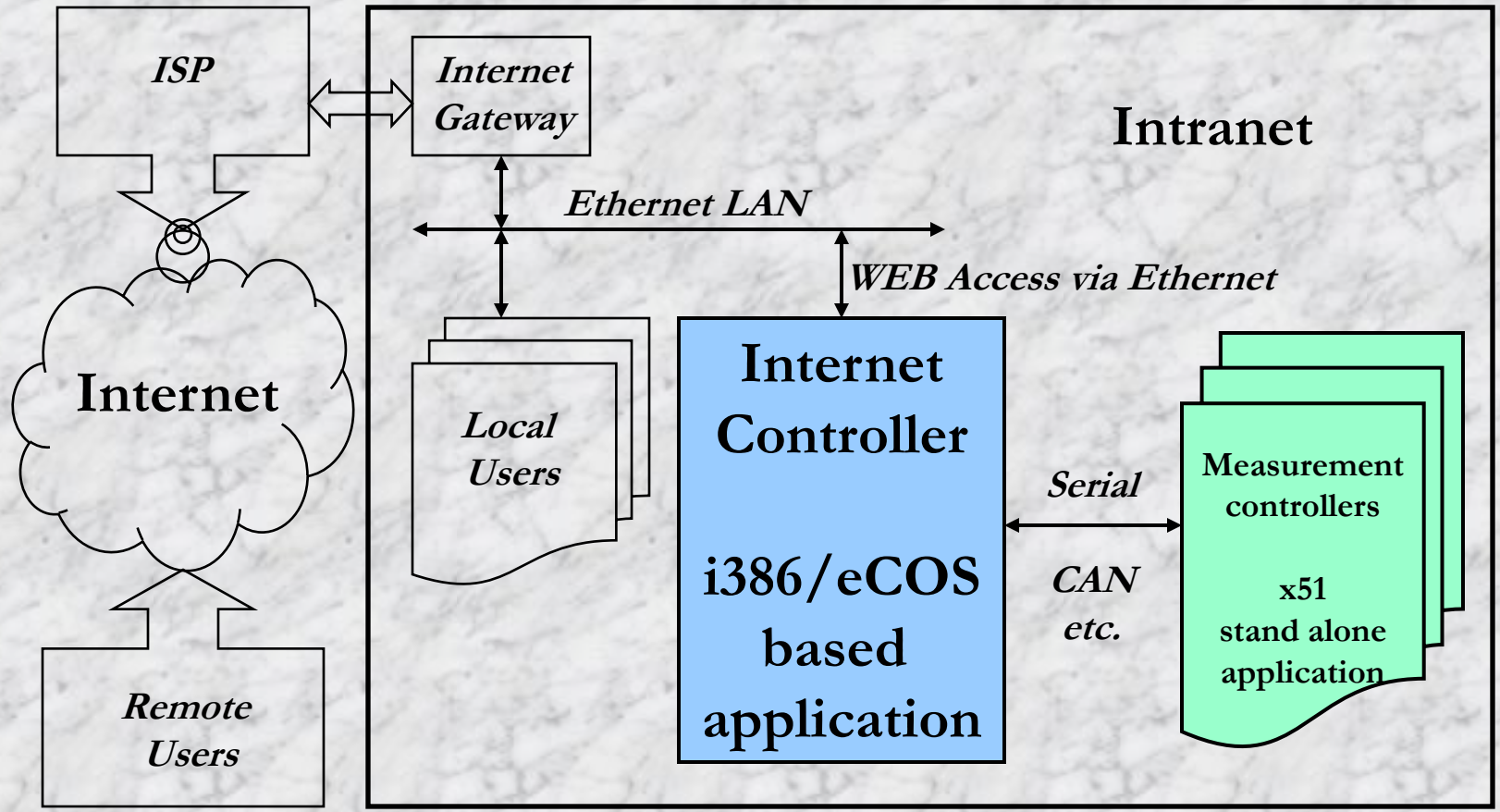
Introduction

Methodology schematic of development process



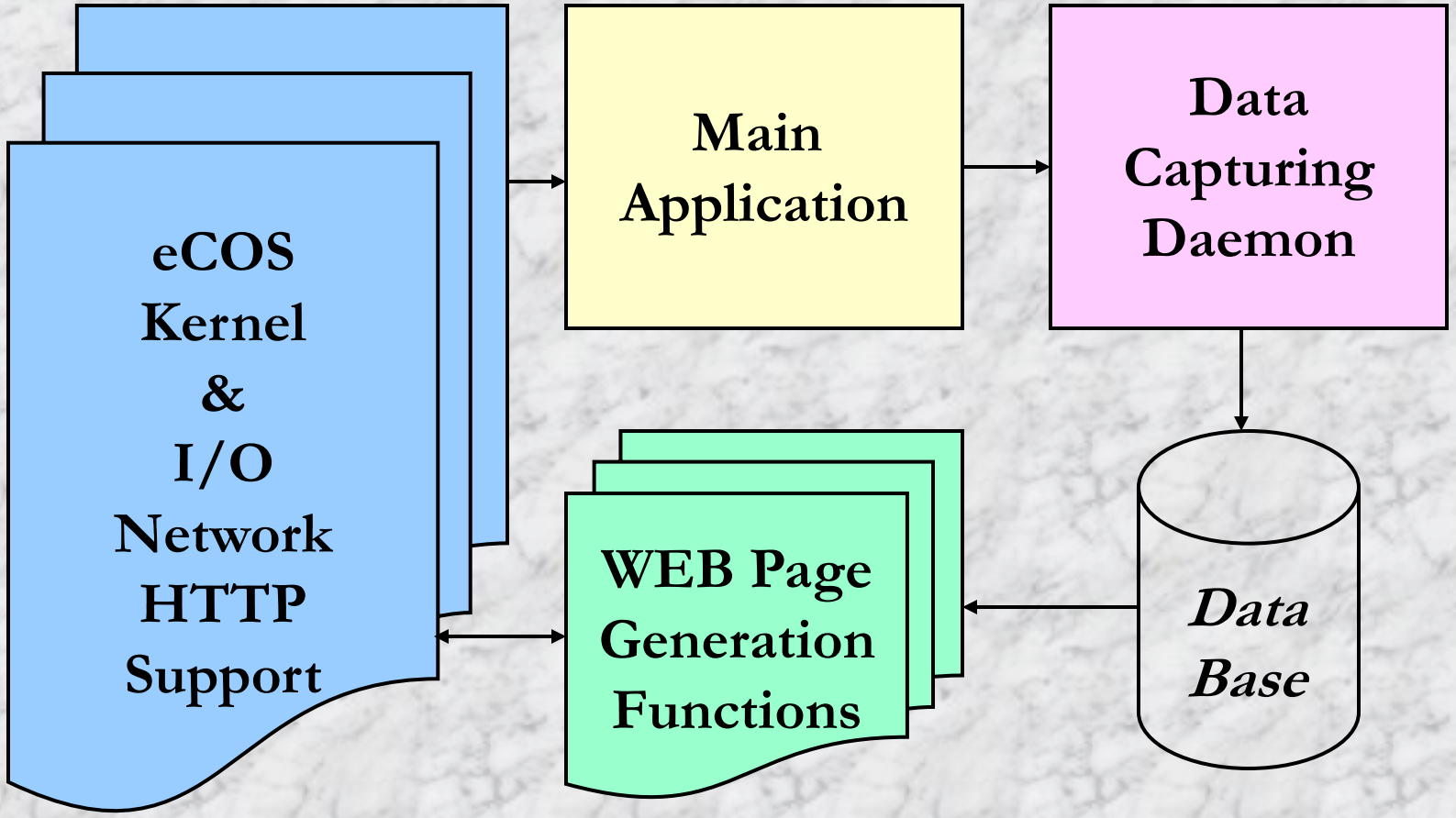
Measurement System

General idea description



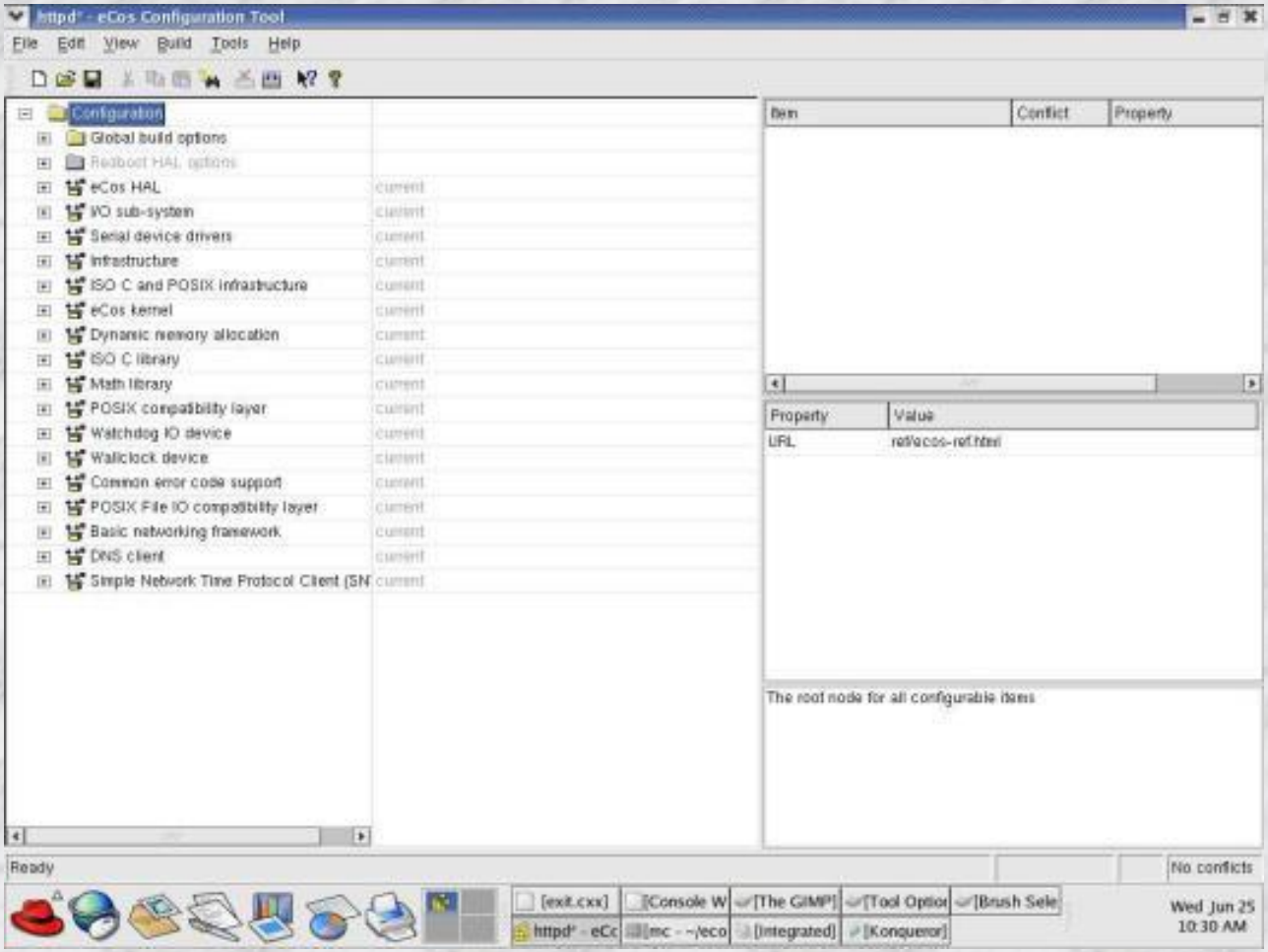
Internet Controller

i386/**eCOS Application – Modules Schematic**



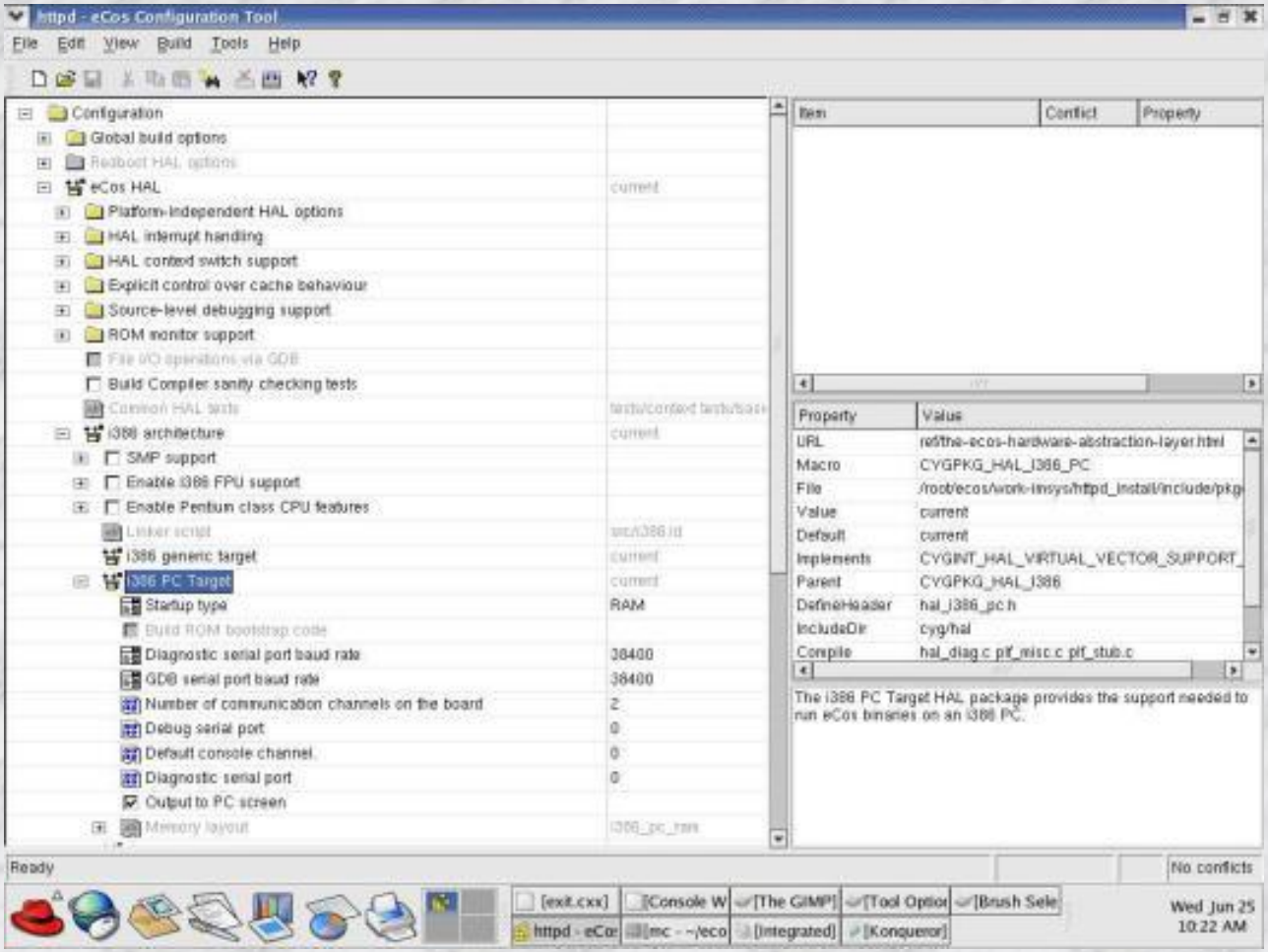
Internet Controller

eCOS Configuration – All modules



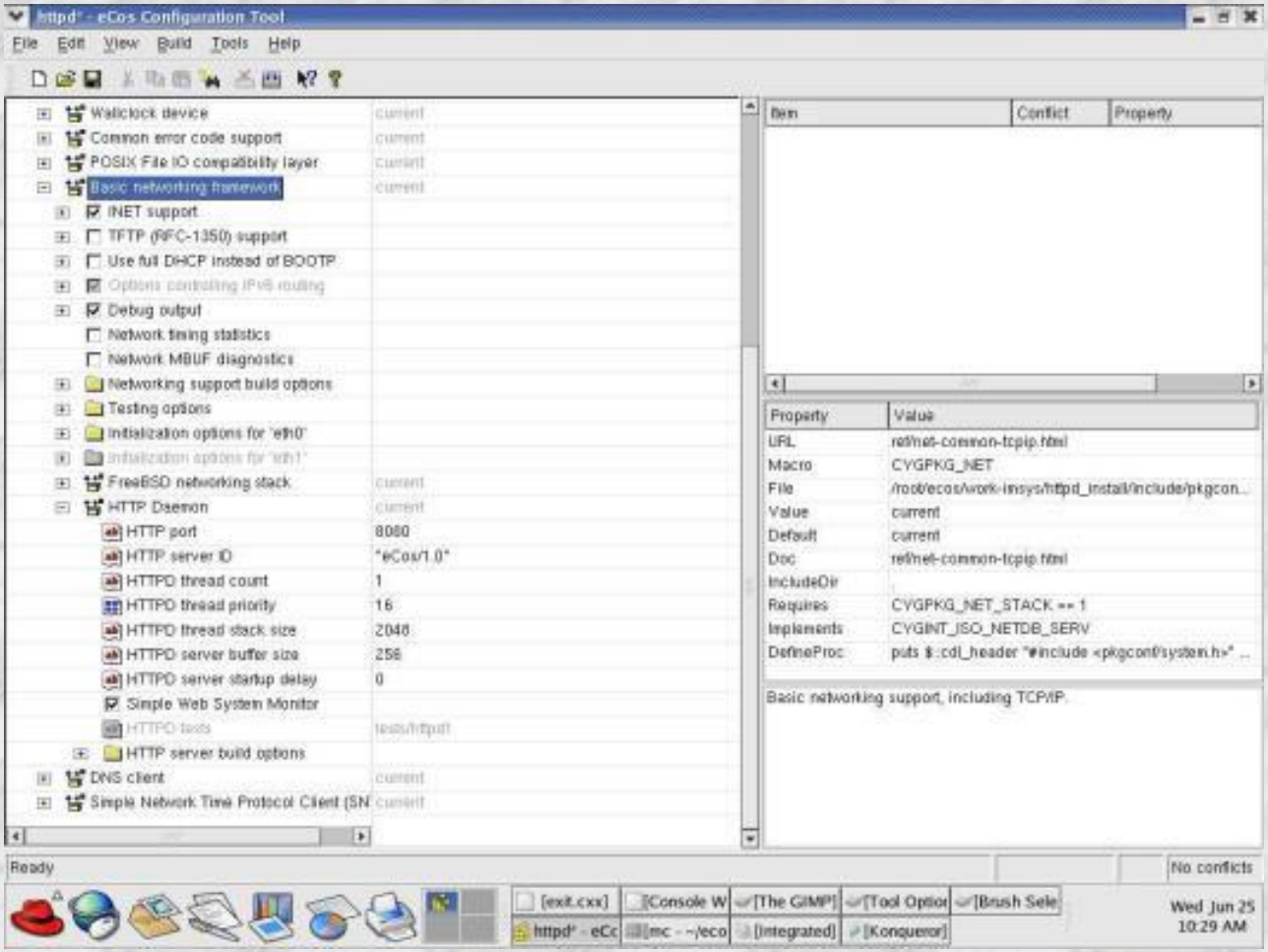
Internet Controller

eCOS Configuration – i386 PC Target



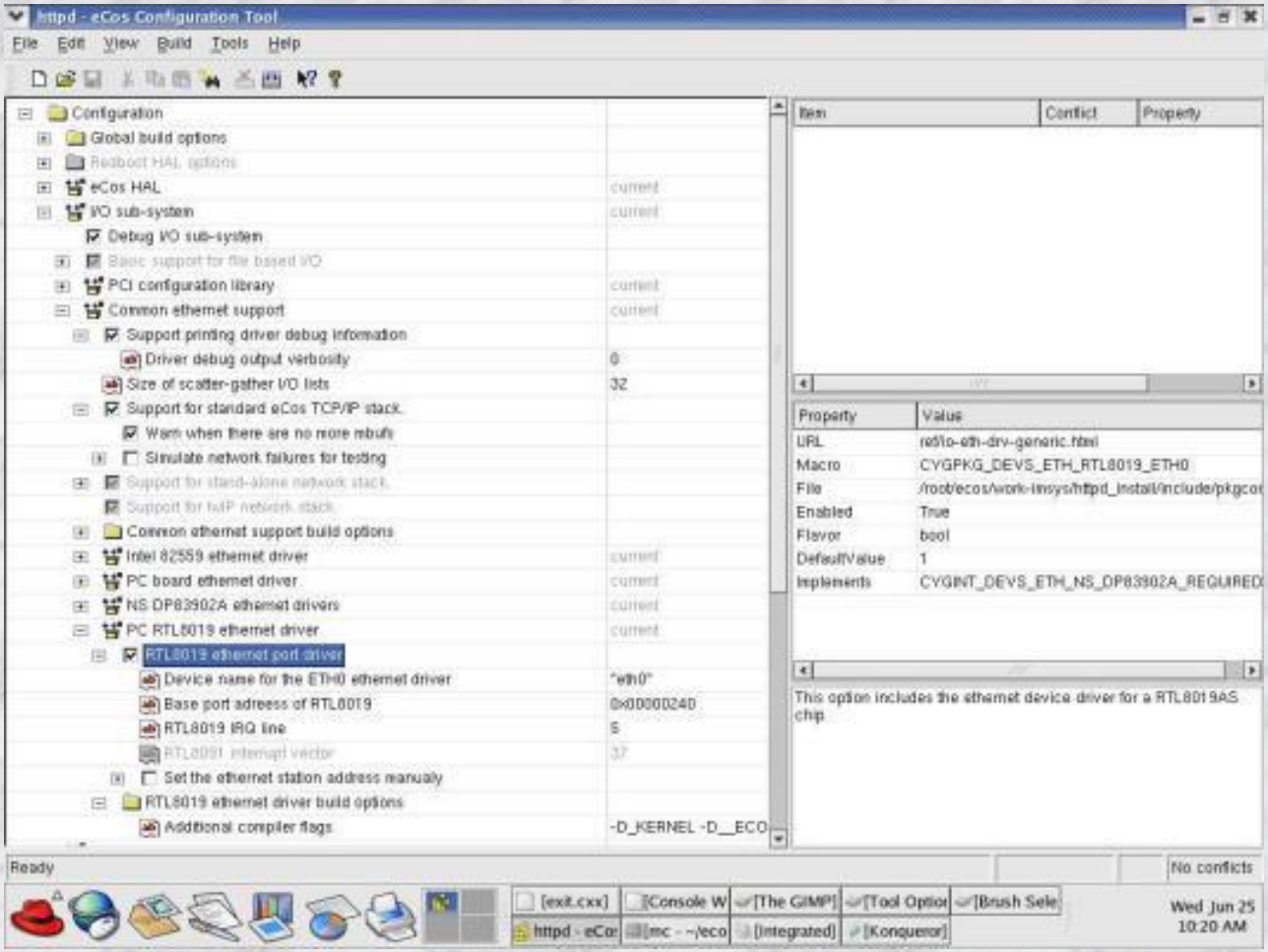
Internet Controller

eCOS Configuration – HTTP Daemon



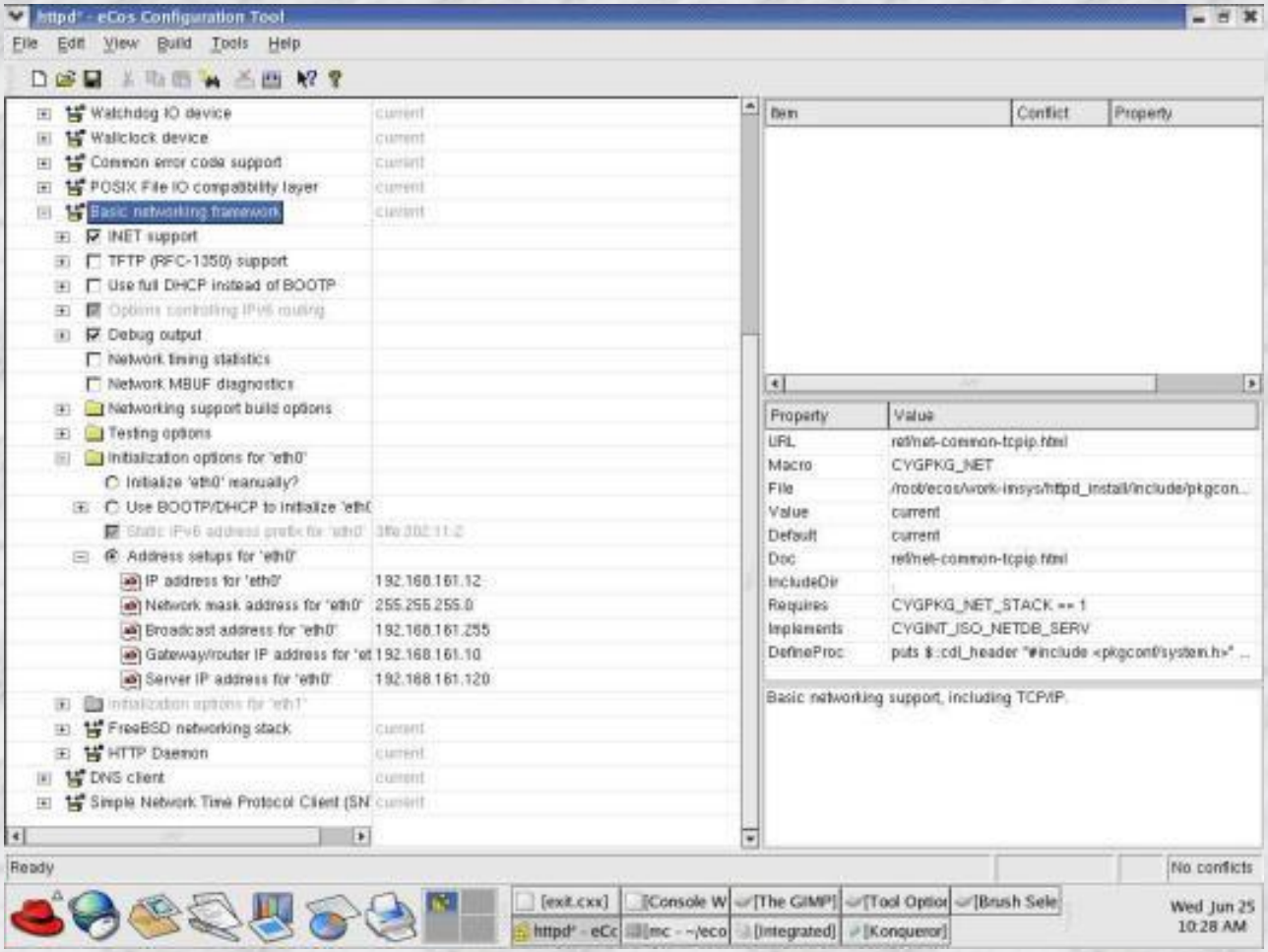
Internet Controller

eCOS Configuration – PC RTL8019 Ethernet Driver



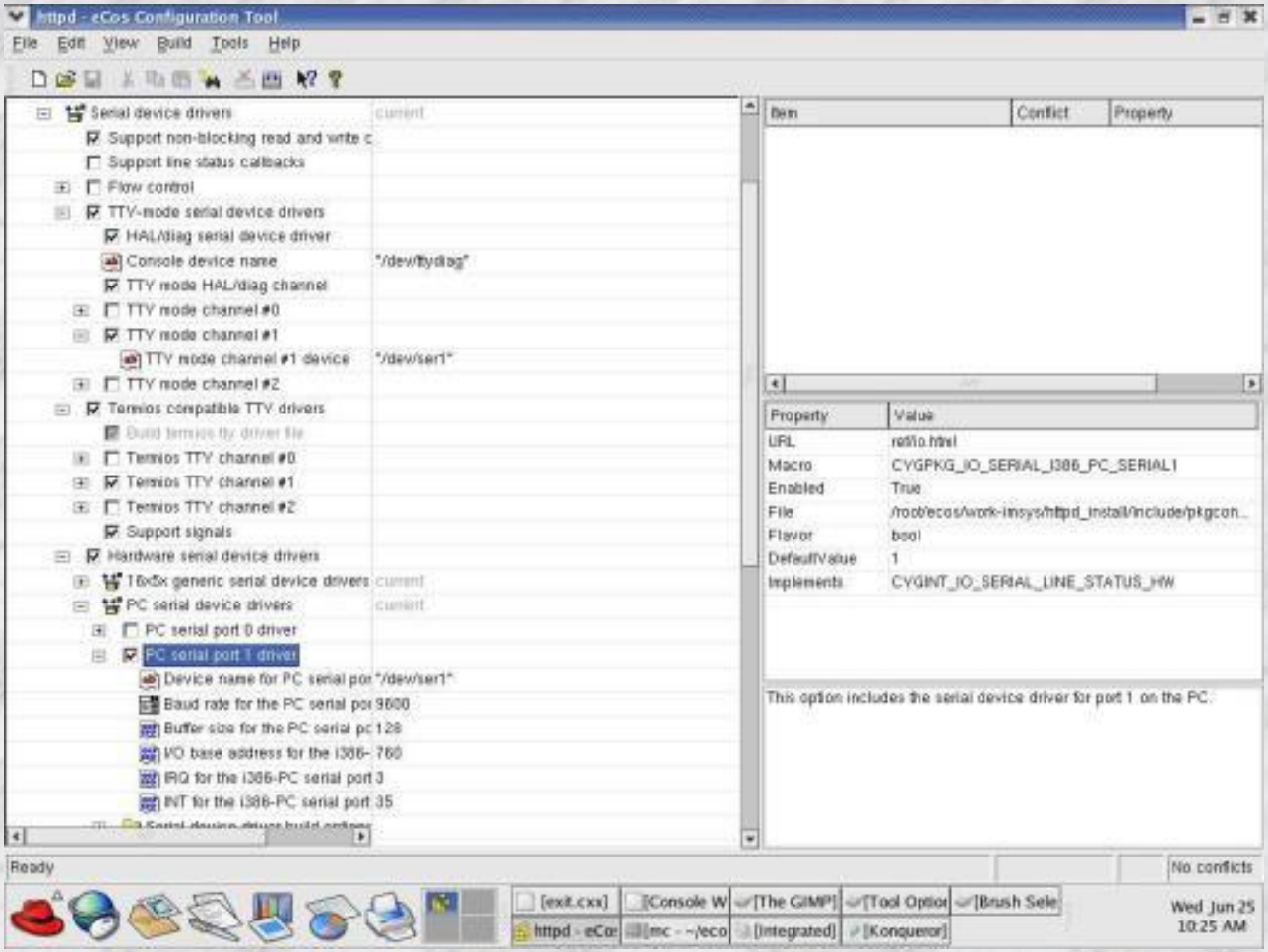
Internet Controller

eCOS Configuration – Initialization options for eth0



Internet Controller

eCOS Configuration – PC serial port 1 driver



Internet Controller

eCOS Application – Main and Daemon modules

```
/* Global definitions */
#define NTHREADS 1
#define STACKSIZE (CYGNUM_HAL_STACK_SIZE_TYPICAL + 4096)

/* STATICS */
static cyg_handle_t thread[NTHREADS];
static cyg_thread thread_obj[NTHREADS];
static char stack[NTHREADS][STACKSIZE];

/* Static structures for data base */
static char  mes_point_name [16];
static char  data_chanel_types [3][16];
static char  data_chanel_units [3][16];
static float data_chanel_value [3][20];

/* use mutex to protect calls to the C library */
cyg_mutex_t datalock;

/* Static functions definition - used in main */
static void serial_daemon(CYG_ADDRESS data)

/* Application main function */

int main(void)
{
    int i;

    /* Init network interfaces */
    init_all_network_interfaces();

    /* Init data base */
    sprintf(mes_point_name, "Point name");
    mes_point_channnels = 0;
    for (i=0; i<3; i++) {
      sprintf(data_chanel_types[i], "Channel %d", i);
      sprintf(data_chanel_units[i], "unit %d", i);
    }

    /* Start measurement daemon as a new tread */
    cyg_thread_create  4, serial_daemon,
                       (cyg_addrword_t) 0, "Serial",
                       (void *)stack[0], STACKSIZE,
                       &thread[0], &thread_obj[0]);

    cyg_thread_resume(thread[0]);
    return 0;
}
```

```
/* Static functions definition - used in serial daemon */
static cyg_io_handle_t init_serial(char * serial);
static void init_connection(cyg_io_handle_t handle);
static float get_data(cyg_io_handle_t handle, int chan);


/* Measurement daemon function running in tread Serial*/

static void serial_daemon(CYG_ADDRESS data)
{
  int delay = 200;
  char serial [] = "/dev/tty1"
  cyg_io_handle_t handle;

  handle = init_serial(serial);
  init_connection(handle);
  for (;;) {  /* Endless daemon cycle  */
    cyg_thread_delay(delay);      /* Tread delay */
    for (i=0; i<3; i++) {
      cyg_mutex_lock(&datalock);  /* Lock shared data base */
      {
        for (j=19; j > 0; j--)
          data_chanel_value[i][j] = data_chanel_value[i][j - 1];
        data_chanel_value[i][0] = get_data(handle, i);
      }
      cyg_mutex_unlock(&datalock);
    }
  }
}


/*    Define the gif file as a byte array,
      then define the data structure
      and table entry to allow it to be fetched by the client.
*/

static cyg_uint8 ecos_green_gif[] = {71, 73, 70, 56, 57, 97, 2, 0, 12, 0,
       128, 0, 0, 0, 255, 0, 255, 255, 255, 33, 254, 21, 67, 114, 101, 97,
       116, 101, 100, 32, 119, 105, 116, 104, 32, 84, 104, 101, 32, 71, 73,
       77, 80, 0, 44, 0, 0, 0,   0, 2, 0, 12, 0, 0, 2, 4, 132, 143, 169,
       87, 0, 59};

CYG_HTTPD_DATA(cyg_measure_green_data,
               "image/gif", sizeof(ecos_green_gif), ecos_green_gif);

CYG_HTTPD_TABLE_ENTRY(cyg_measure_green, "/measure/green.gif",
                      cyg_httpd_send_data, &cyg_measure_green_data);
```

Internet Controller

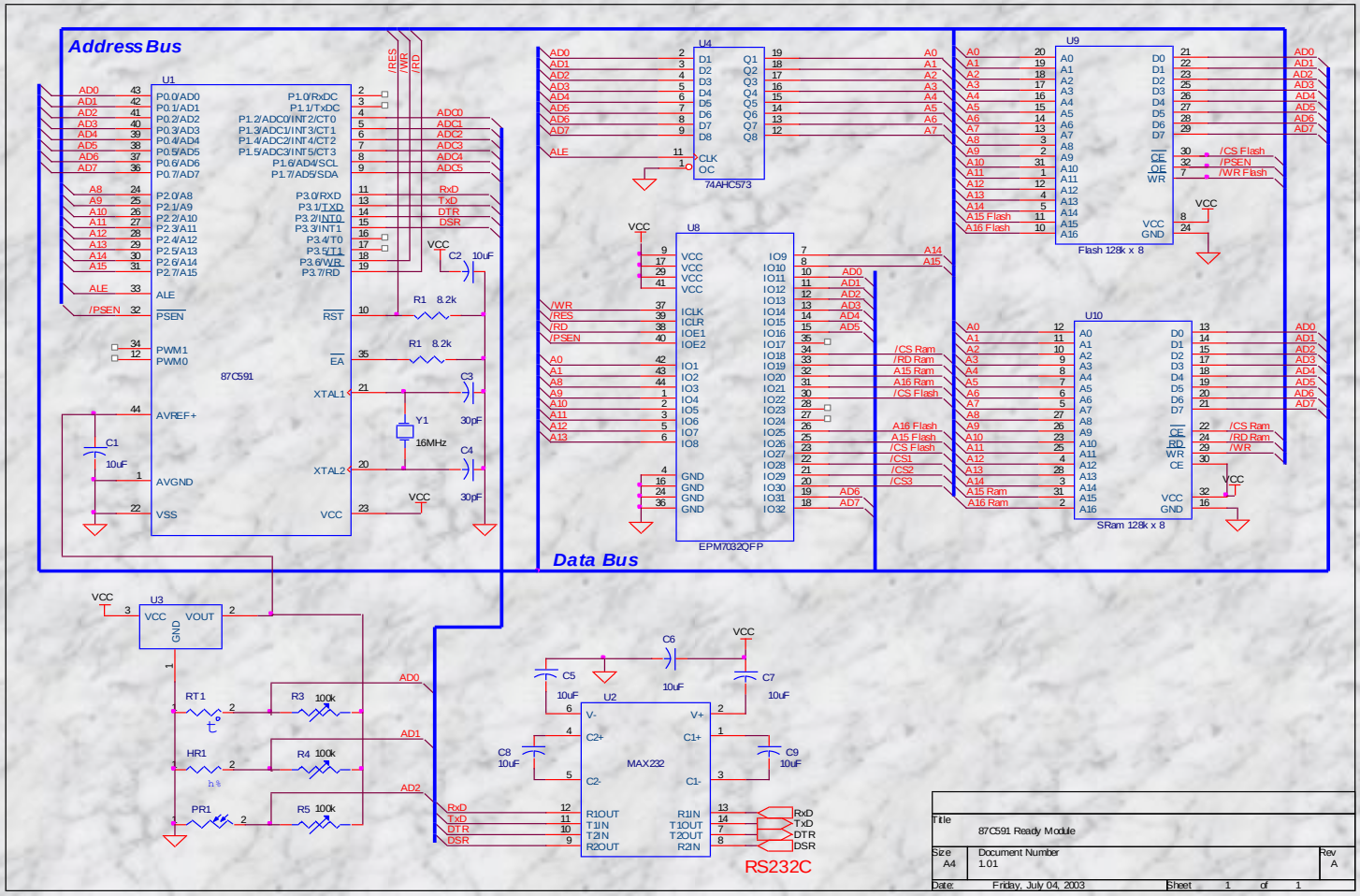
eCOS Application – WEB Page generation modules (index)

```
/* Draw table Function */
static void draw_table( FILE *client)
{
  int i;
  char point[120];
  html_table_begin( client, " width=\"80\%\" border=2 cellpadding=\"4\"" );
  {
        html_table_row_begin(client, "" ); {
          sprintf( point, "<h2>Measurement point: <b><i>%s #1 - %d \
                           channels</i></b></h2>",
                           mes_point_name, mes_point_channnels);
          html_url( client, point, "/measure/history.html?chan=0");
        }
        html_table_row_end( client );
        html_table_row_begin(client, "" );
        {
            html_table_begin( client, "width=\"80\%\" border=1 cellpadding=\"4\"" );
            {
              for (i=0; i<3; i++) {
                html_table_row_begin(client, "" );
                {
                  fprintf( client, "<td><a href=\"/measure/history.html?chan=%d\"> \
                                    <h2><i>  %s </i></h2></a></td>",
                                     i+1, data_chanel_types[i]);
                  cyg_mutex_lock(&datalock);
                  {
                    fprintf( client, "<td align=right><h2> %6.0f </h2></td>", \
                                     data_chanel_value[i][0]);
                  }
                  cyg_mutex_unlock(&datalock);
                  fprintf( client, "<td><h2> %s </h2></td>", data_chanel_units[i]);
                }
                html_table_row_end( client );
              }
            }
            html_table_end( client );
        }
        html_table_row_end( client );
  }
  html_table_end( client );
}

/* Static text used in WEB Page generation */

static char measure_index_blurb[] =
"<p>This is i386/eCOS & x51 based Internet Enabled Distributed Measurement System.  \
 <p>Simple web monitor shows the temperature, humidity and lighting at the point of
measure.";
```

```
/* Generate function for main page of Measurement System */

static cyg_bool cyg_measure_index( FILE * client, char *filename,
                                   char *formdata, void *arg )
{
    html_begin(client);
    fprintf(client, "<META HTTP-EQUIV=REFRESH CONTENT=5>");
    html_head(client,"Internet Enabled Distributed Mesurement System",
"");
    html_body_begin(client,"");
    {
        html_heading(client, 2, "Internet Enabled Distributed
Mesurement System" );
        draw_navbar(client);
        fputs( measure_index_blurb, client );
        draw_table(client);
    }
    html_body_end(client);
    html_end(client);
    return 1;
}
/* Main page declaration macroces */

CYG_HTTPD_TABLE_ENTRY( cyg_measure_entry,
                       "/measure",
                       cyg_measure_index,
                       NULL );

CYG_HTTPD_TABLE_ENTRY( cyg_measure_index_entry,
                       "/measure/index.html",
                       cyg_measure_index,
                       NULL );
```

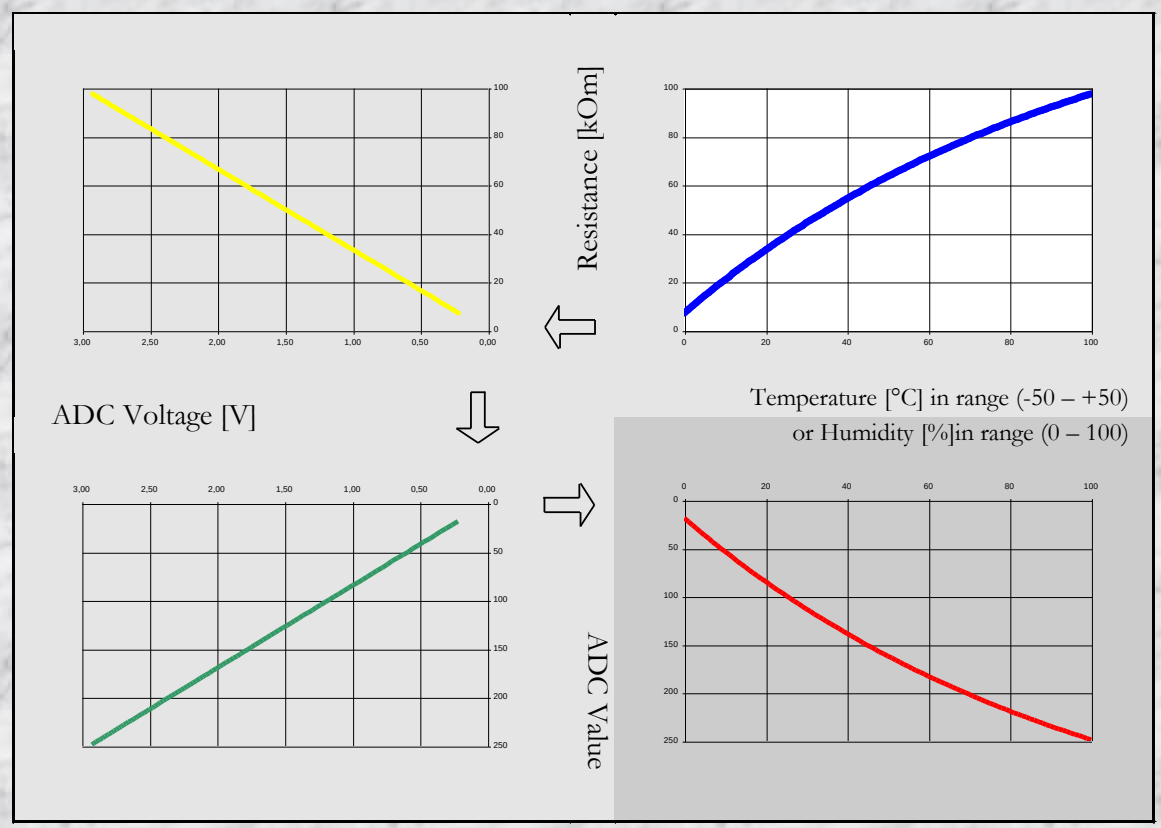
Measurement Controller

Hardware schematic design



Measurement Controller

Calculation mechanism



Preliminary phase calculations

Real-time calculations

Measurement Controller

Software implementation

```
#include <reg592.sfr>   /* register set of 8052 controller */
#include <stdio.h>            /* standard I/O functions          */
#include "SerInit.h"    /* automatic baud rate detection   */

#pragma romstring             /* rom resided data                */
char textA [] [] = {"Temperature",  /* channel identification */
                    "Humidity",
                    "Lighting"};
char textB [] [] = {"degree C\0",   /* unit identification    */
                    "Percent\0",  "Relative\0"};

/* ADC values calculated for following channels at points:
   temperature {-50, -45, ..., 45, 50} degrees Celsius and
   humidity {0, 5, ..., 95, 100}                              */
char values [][] = {{18, 12}, {36, 23}, {53, 39}, {69, 57}, {84, 74}, { 99,  90},
                    {112, 105}, {126, 120}, {138, 133}, {150, 146}, {161, 159},
                    {172, 171}, {182, 182}, {192, 193}, {201, 203}, {210, 212},
                    {219, 222}, {226, 230}, {234, 239}, {241, 247}, {248, 248}};

#pragma ramstring             /* rom resided data                */
int get_ADC (int chan);
int get_value(int chan);

void main (void)        /* execution starts here             */
{
  char numc = '3';            /* number of measured channels       */
  char comm;                  /* command char                      */
  char scom;                  /* sub command char                  */
  int  chan;
  SerInit();                  /* call automatic baudrate detection */
                        /* send space char to continue       */
  printf ("iMSys\r");
  while (1)
  {
    comm = getchar();
    if ( (comm > 'b') && (comm < 'f') ) {
      scom = getchar();
      chan = (int) scom - 0x30;
      if ((chan < 0) || (chan > 2)) chan = 0;
    }
    if      (comm == ' ') printf("iMSys\r");
    else if (comm == 'a') printf("iMSys\r);
    else if (comm == 'b') printf("%c\r", numc);
    else if (comm == 'c') printf("%s\r", textA [chan]);
    else if (comm == 'd') printf("%s\r", textB [chan]);
    else if (comm == 'e') printf("%d\r", get_value(chan));
    else            printf("Error\r");
  }
}
```

```
/* Function to calculate real values for requested value */

int get_value(int chan)
{
      int adc_value, i, up_value, dn_value, value;
      adc_value = get_ADC(chan);
      if (chan == 2) return (adc_value * 100 / 256);
      for (i=0; i<21; i++) {
            if (adc_value > values[chan][i]) continue;
            dn_value = values[chan][i];
            up_value = values[chan][i+1];
            break;
      }
      value = i * 5 + (up_value - dn_value) / (value -
dn_value);
      if (chan == 0) value -= 50;
      return (value);
}

/* Function to get ADC values for requested value */

int get_ADC (int chan)
{
   int poti;
   char cnum;
   cnum = 0xf0 + (chan && 0x3);
   ADCON &= cnum;       /* select channel on P1.0 .. P1.2 */
   ADCON |= 0x08;       /* set ADCS to start conversion */
   while(!(ADCON == (ADCON | 0x10))); /* wait for int flag */
   ADCON &= 0xEF;                   /* clear int flag */
   poti = (0xFF - ADCH);
   return (poti);                   /* return 0-100 */
}
```

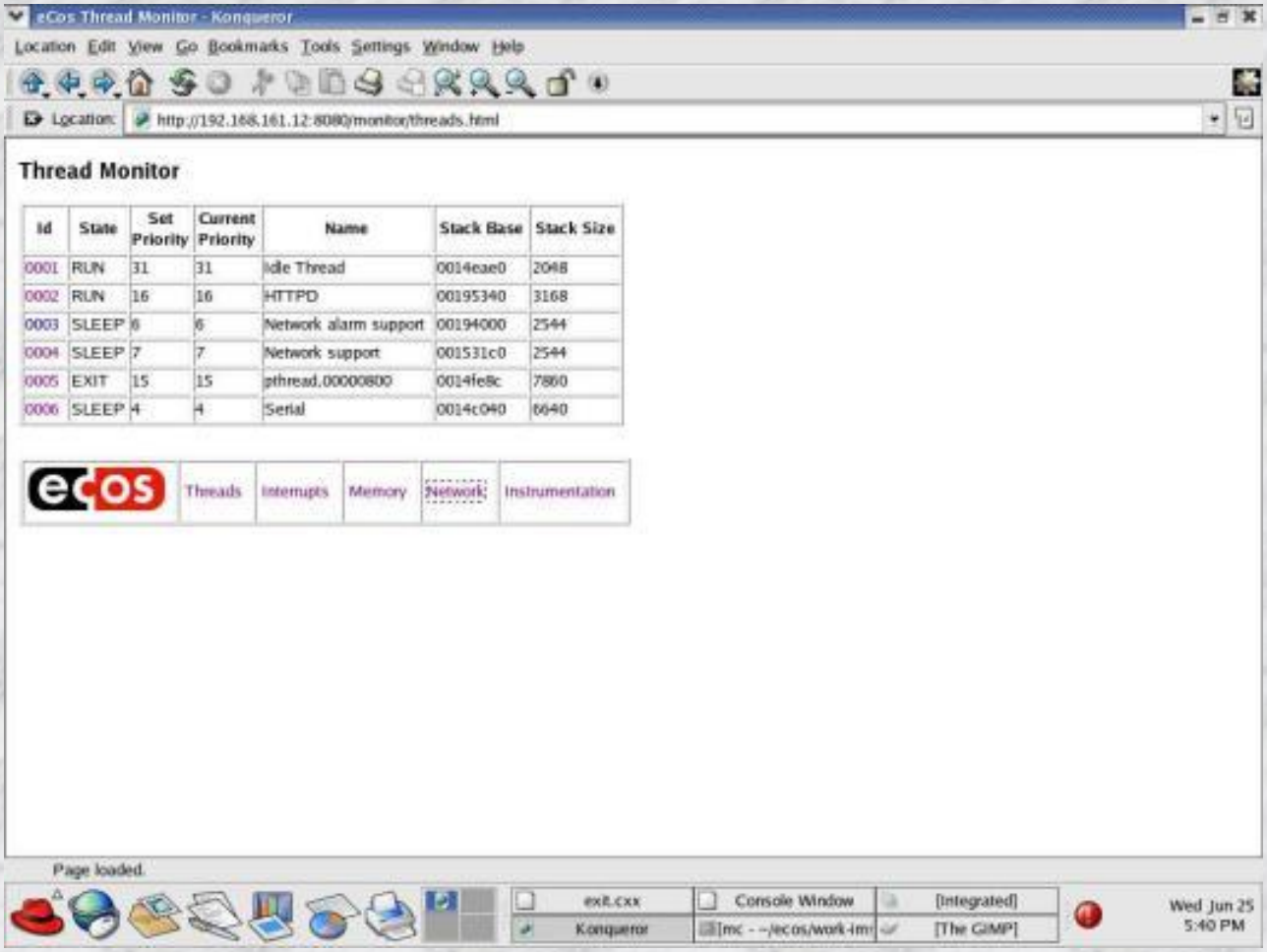
System Description

Description of processes – start up and interaction

Process	*Client machine*	*Internet controller*	*Measurement controller*
Start up			
	Start OS Start Browser	Initialize Kernel; Initialize IO subsystem Initialize Networking Initialize HTTPD Start Measuring process	Initialize processor settings Initialize Serial device
Measuring process start up			
		Initialize Serial connection Connect to Measurement controller	Connect to Internet controller Init serial communication parameters
		Send "get properties" command →	Wait for receiving command If (get properties command)
		Get channel properties ←	Send channel properties
		Store channel properties to DB	
Measuring process			
		Wait for 2 seconds Send "get data" command →	Wait for receiving command If (get data command)
			Measure requested channel Calculate measured value
		Get channel data ←	Send channel data
		Store data to DB	
HTTPD process			
	Send HTTP request →	Wait for HTTP request Call page generation routine	
		Get data and properties from DB Generate requested page	
	Show Page ←	Send requested page	

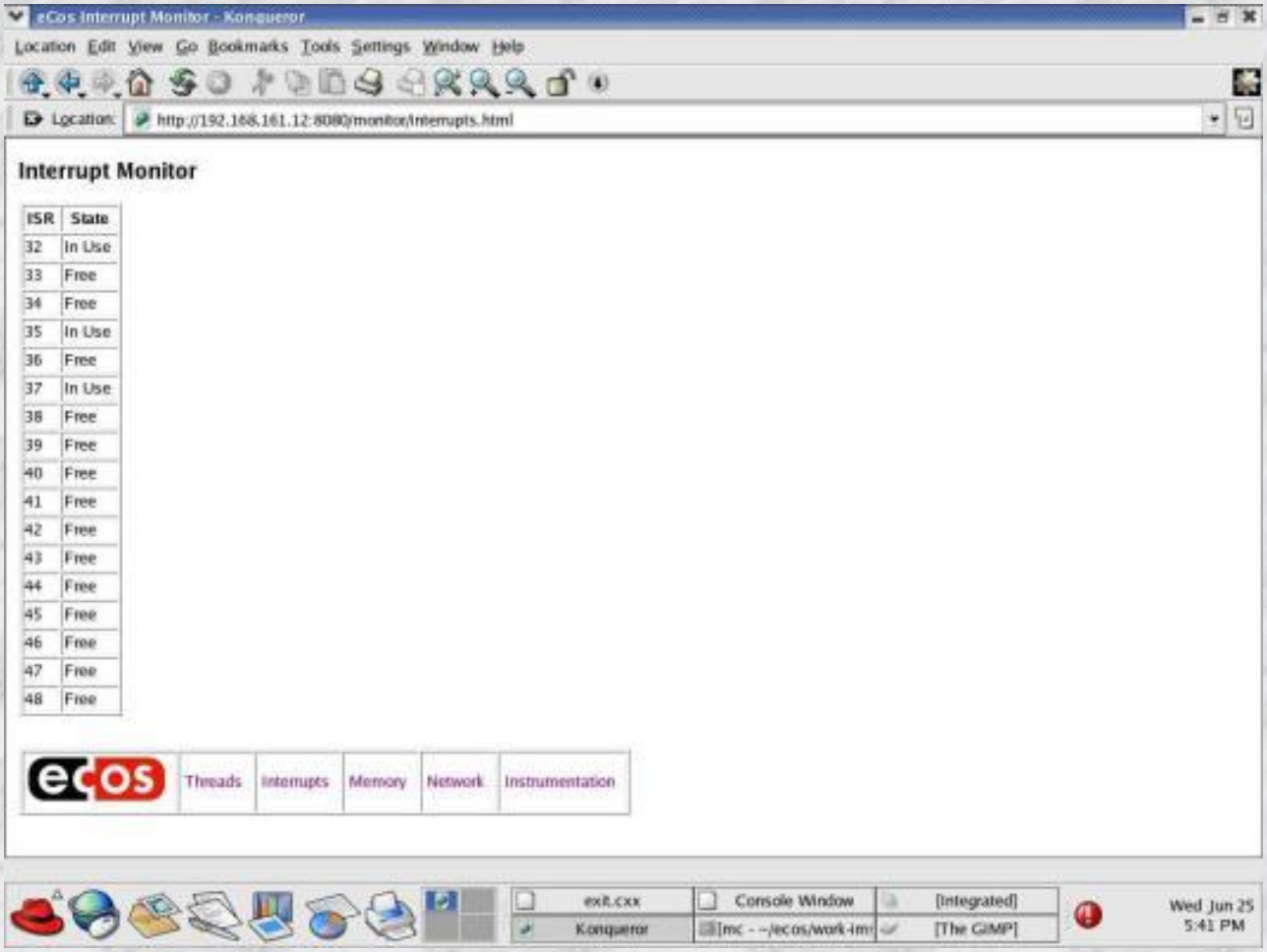
System Description

System Monitor – Threads



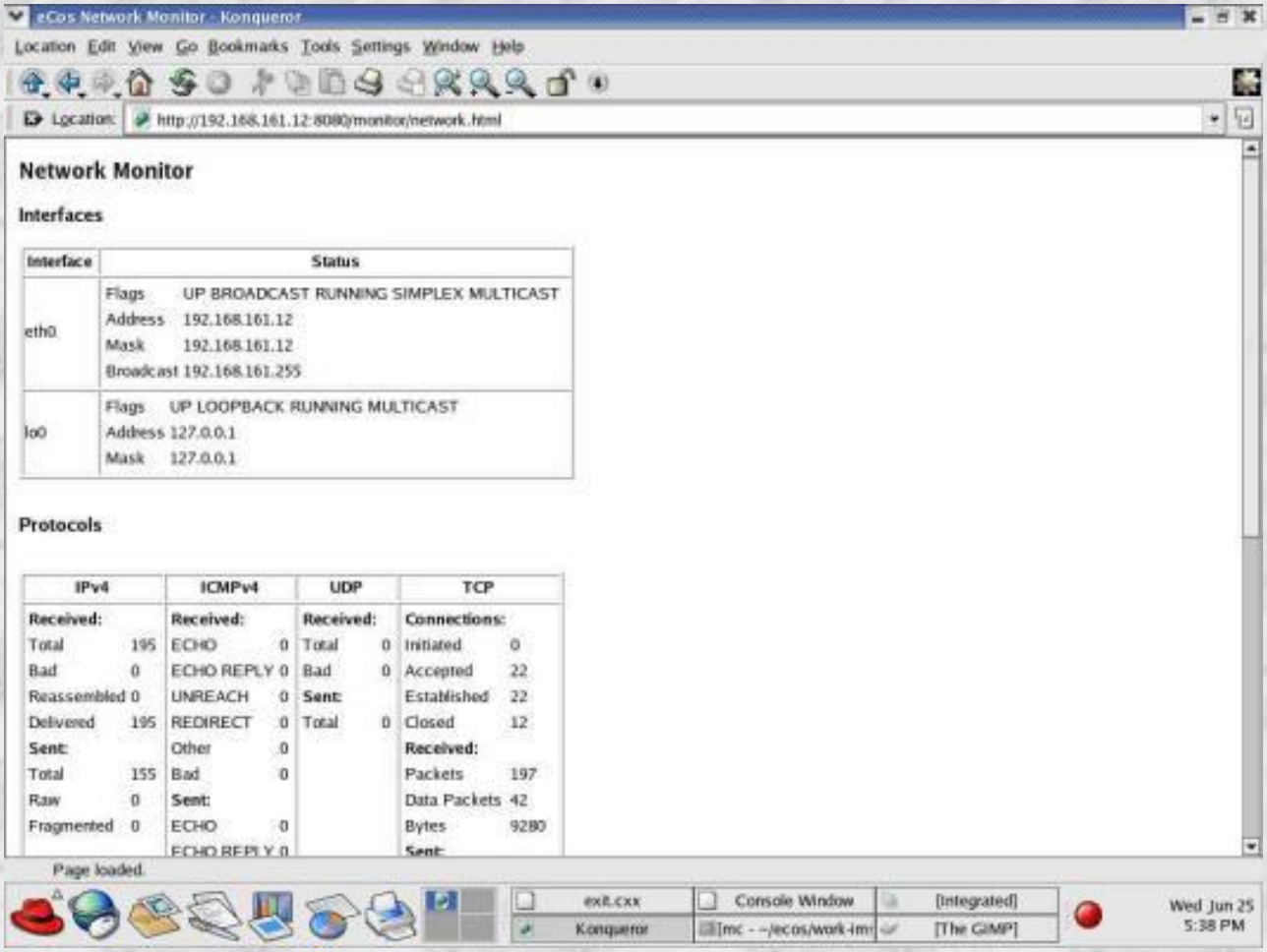
System Description

System Monitor – Interrupts



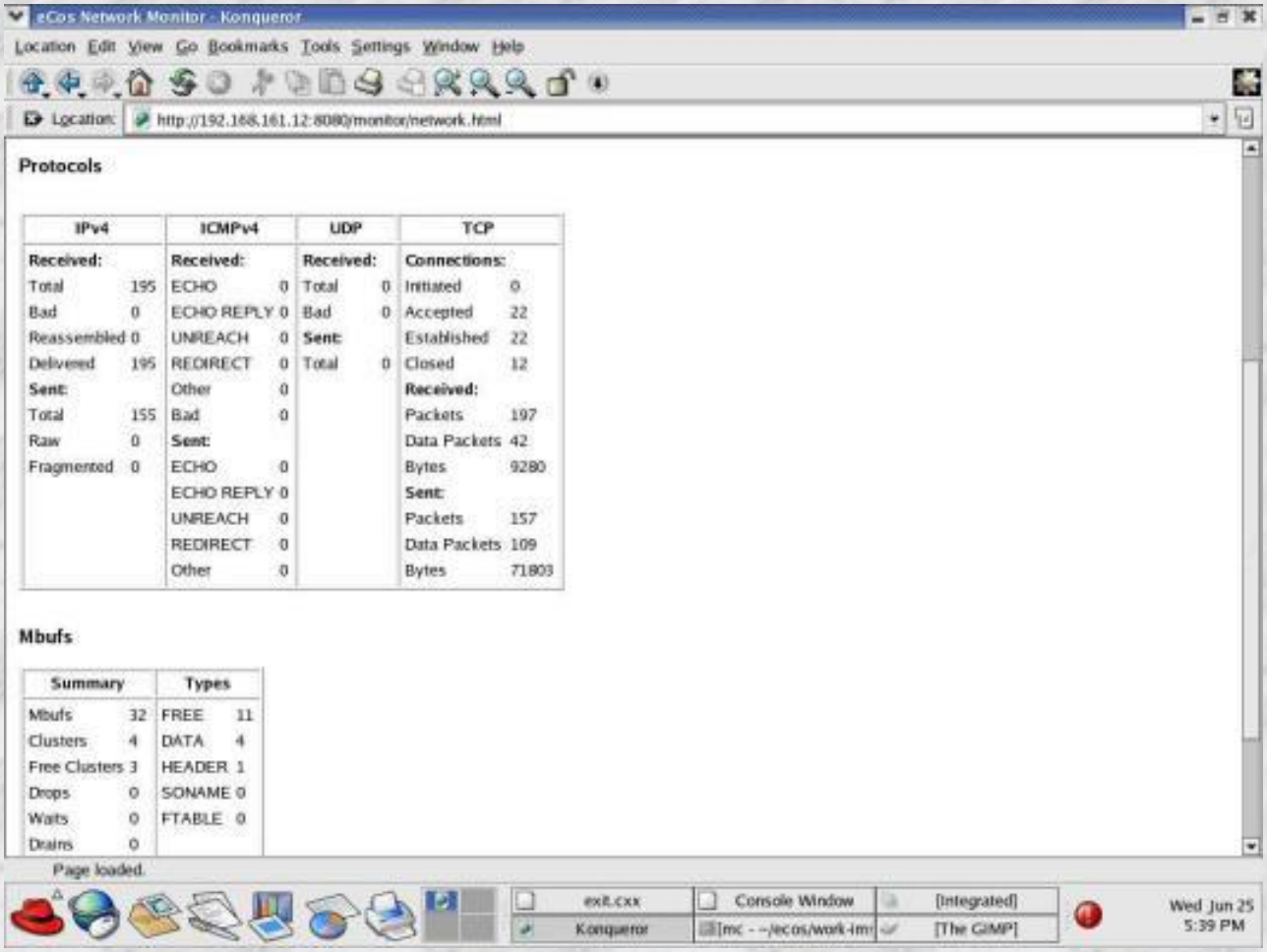
System Description

System Monitor – Network



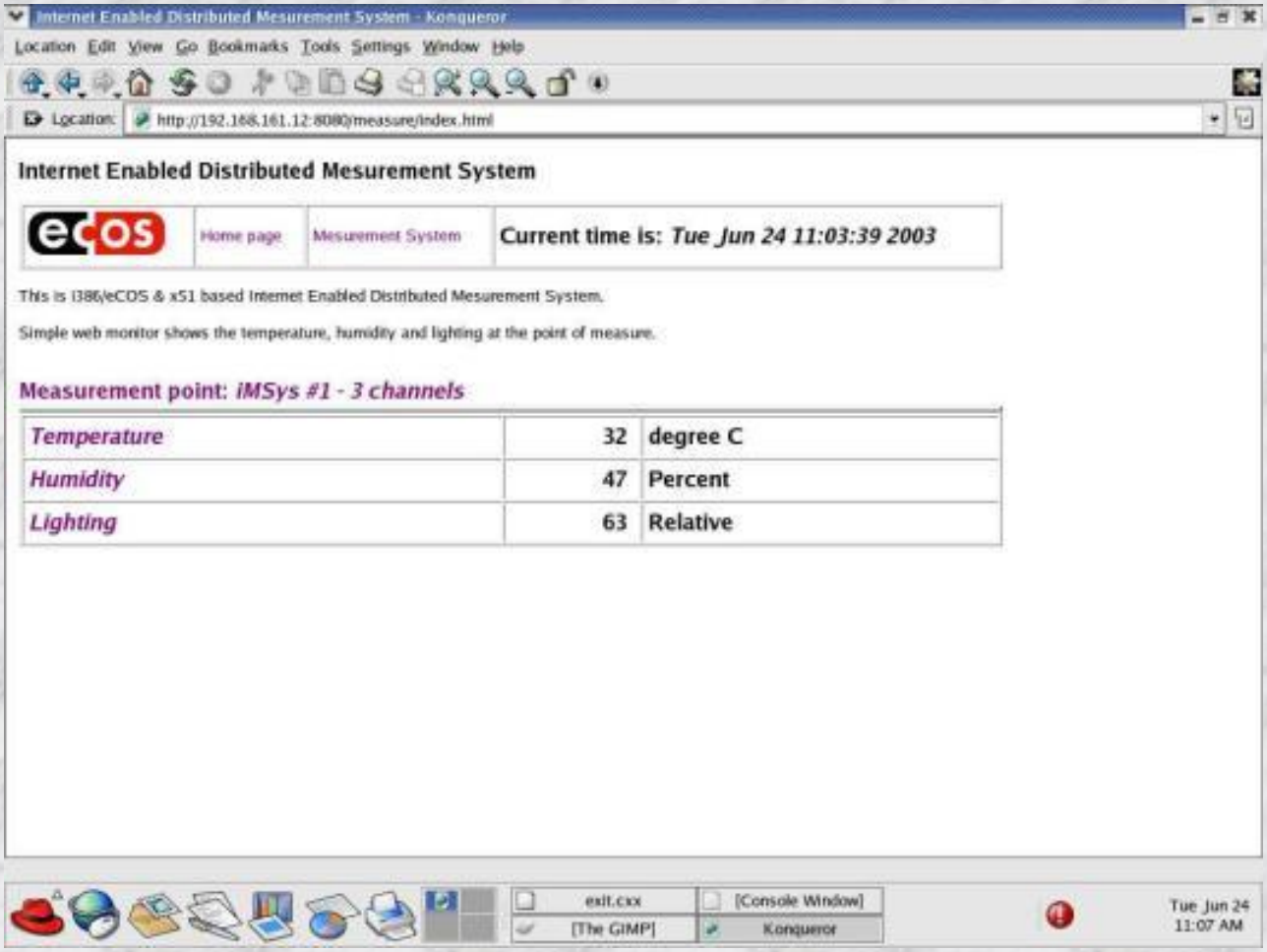
System Description

System Monitor – Protocols



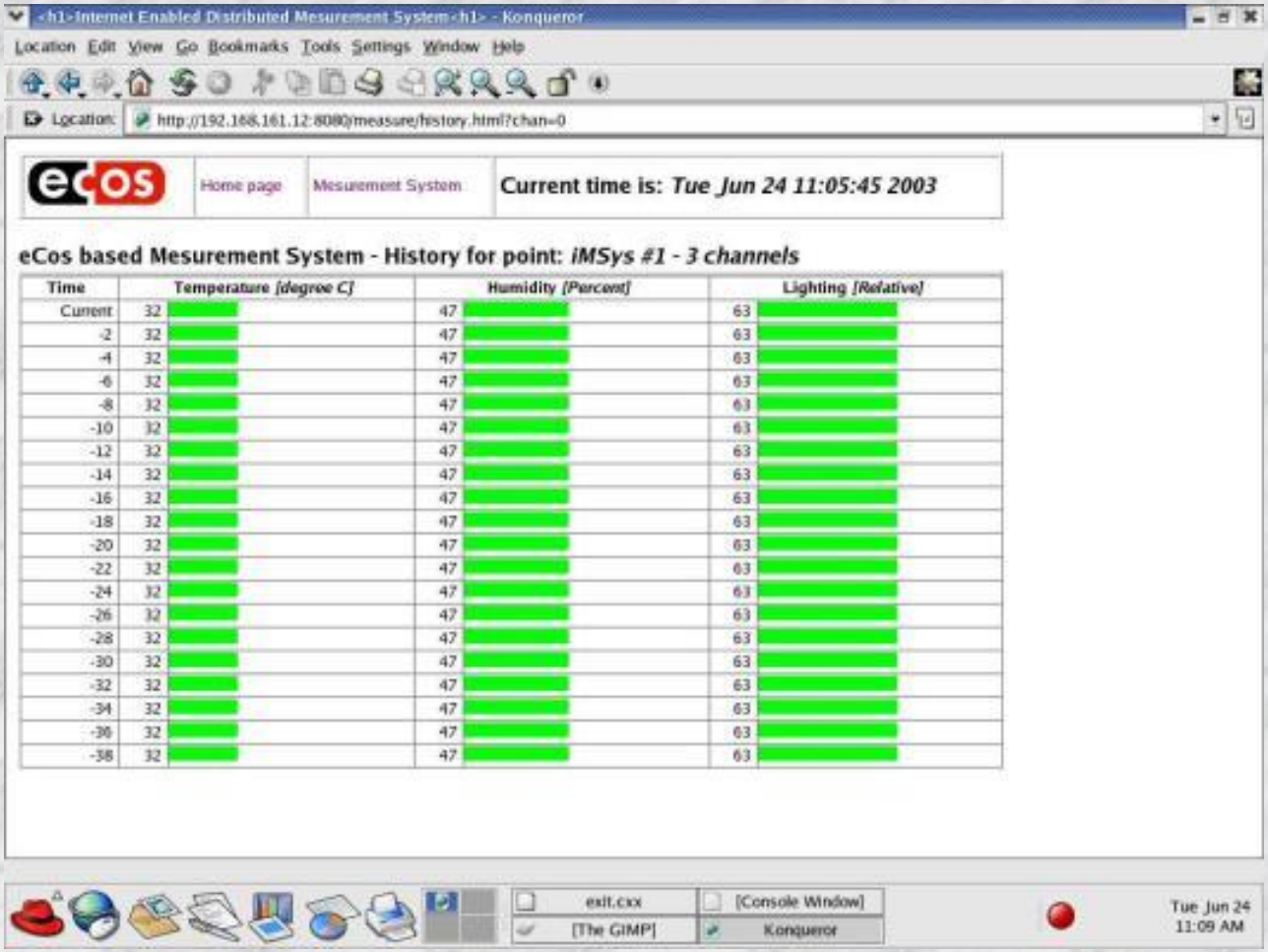
System Description

Measurement System – Main page



System Description

Measurement System – All channels history



System Description

Measurement System – Temperature channel history