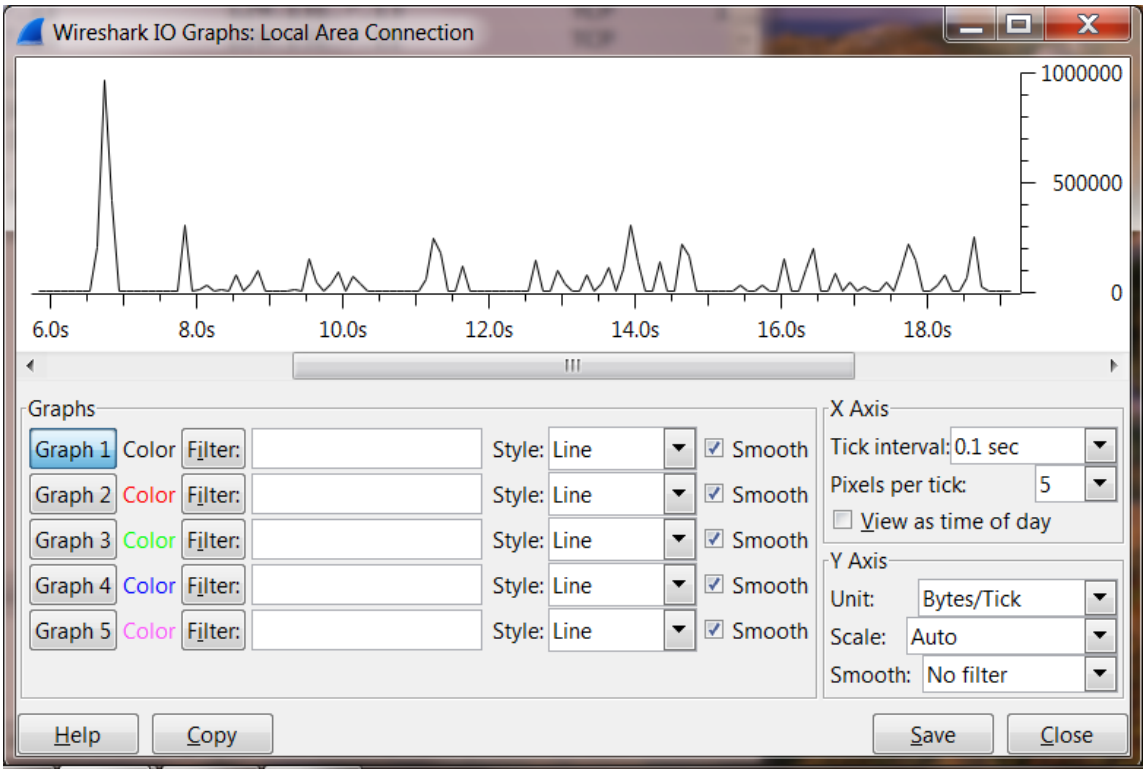
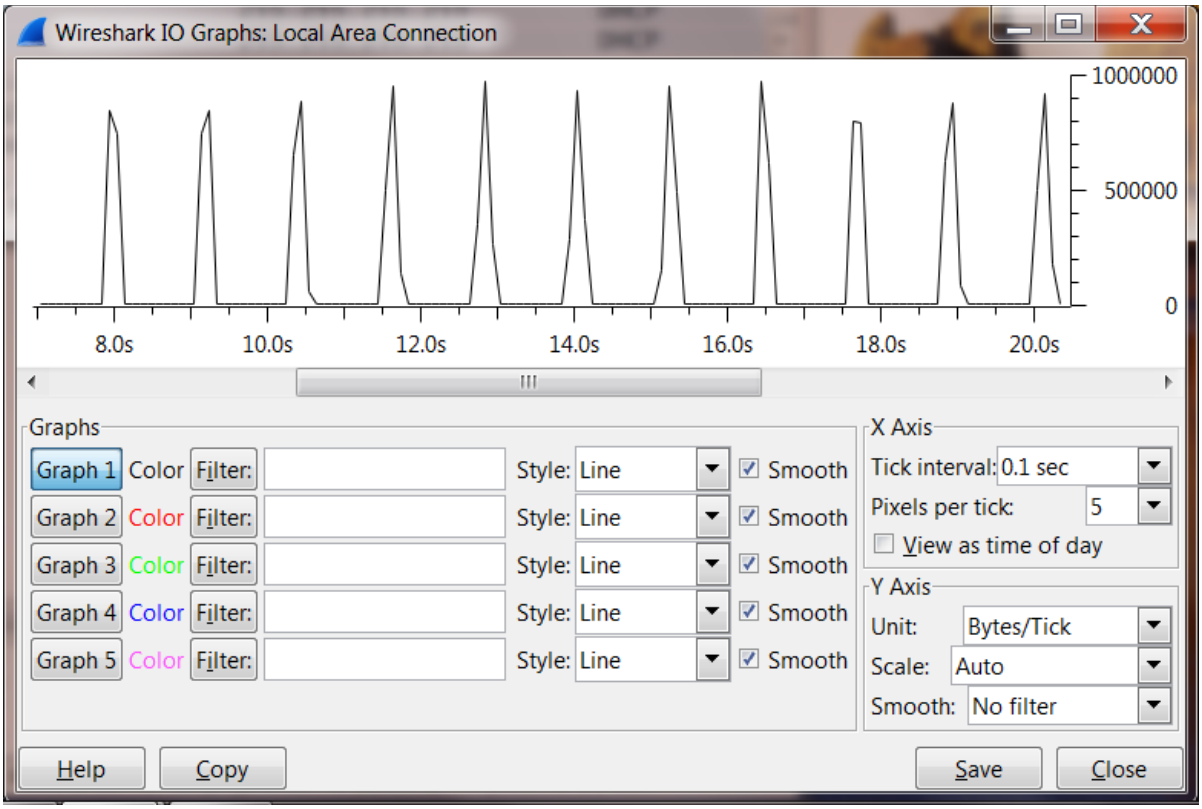
Cyclone V Dev board "changingArray.c"




```
root@socfpga_cyclone5:~/altera# ./changingArray
pthreadRecordRet 0
RECORD THREAD STARTED record status 0
cycloneVserver  Server process 444 set to use SIGUSR1 and SIGUSR2
cycloneVserver  port Number we are trying to open 1025
cycloneVserver  Connection has been made to unknown
cycloneVserver  Ready to read a byte request
cycloneVserver  -- request received from client 0x55--->>>>
START RECORDING
cycloneVserver  Ready to read a byte request
Enet 1.000000e+00
Enet 4.000000e+00
Enet 3.000000e+00
Enet 6.000000e+00
Enet 5.000000e+00
Enet 4.000000e+00
Enet 3.000000e+00
Enet 4.000000e+00
Enet 1.000000e+01
Enet 4.000000e+00
cycloneVserver  -- request received from client 0xaa--->>>>
STOP RECORDING
cycloneVserver  Ready to read a byte request
cycloneVserver  -- request received from client 0xbb--->>>>
CLOSE CONNECTION
 CLOSE REQUESTED .... connection will close and re-open for listening
cycloneVserver  Server process 444 set to use SIGUSR1 and SIGUSR2
cycloneVserver  port Number we are trying to open 1025
```

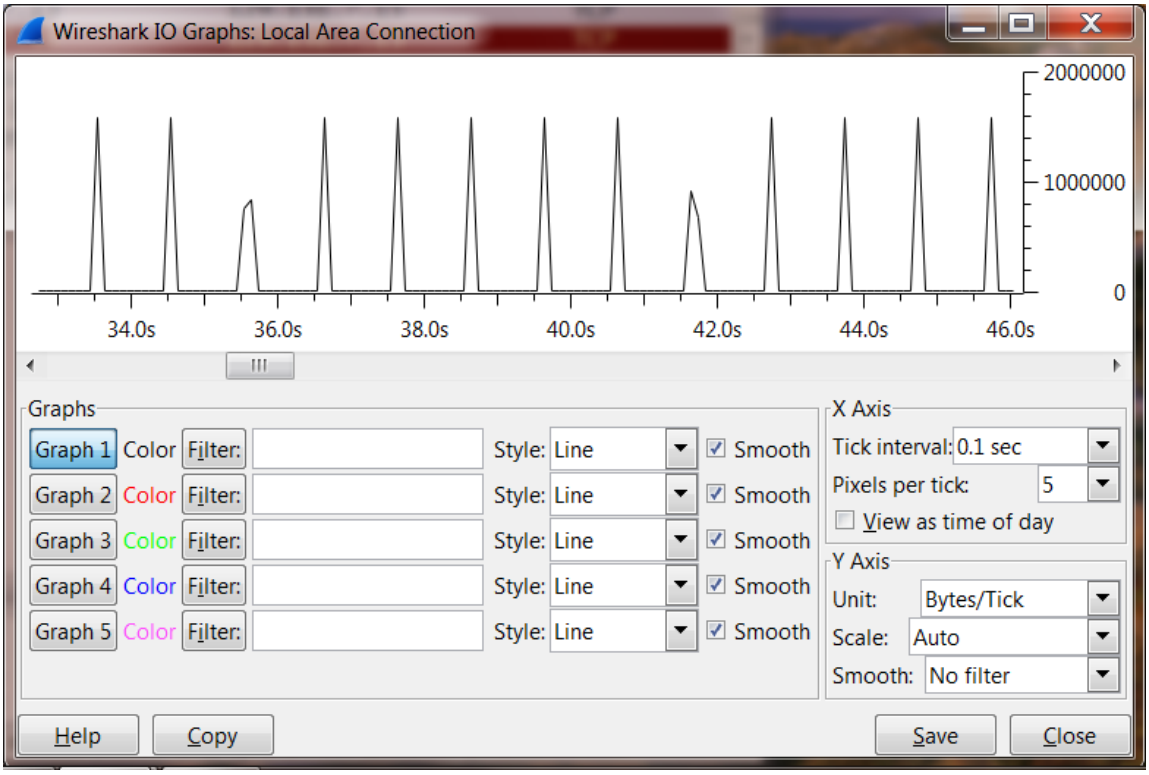
Cyclone V Dev Board “staticArray.c”




```
root@socfpga_cyclone5:~/altera# ./staticArray
pthreadRecordRet 0
RECORD THREAD STARTED record status 0
cycloneVserver  Server process 507 set to use SIGUSR1 and SIGUSR2
cycloneVserver  port Number we are trying to open 1025
cycloneVserver  Connection has been made to unknown
cycloneVserver  Ready to read a byte request
cycloneVserver  -- request received from client 0x55--->>>>
START RECORDING
cycloneVserver  Ready to read a byte request
Enet 0.000000e+00
.Enet 0.000000e+00
.Enet 0.000000e+00
.Enet 1.000000e+00
.Enet 0.000000e+00
.Enet 0.000000e+00
.Enet 0.000000e+00
.Enet 0.000000e+00
.Enet 0.000000e+00
.Enet 0.000000e+00
.Enet 0.000000e+00
.Enet 0.000000e+00
.Enet 0.000000e+00
.Enet 1.000000e+00
.Enet 0.000000e+00
.cycloneVserver  -- request received from client 0xaa--->>>>
STOP RECORDING
cycloneVserver  Ready to read a byte request
```

FOR COMPARISON →

Linux Laptop – "changingArray.c"



The output was like the Cyclone V "staticArray.c" code.

THE CODE

changingArray.c

```
// FILENAME:         changingArray.c
/*****************************************************************************
 ***************
 *
 * Created by:      Janet Estabridis
 *
 *
 * Date created:   1 May 2015
 * Date revised:   1 May 2015
 * Reason for revision:
 *  1 May 2015
 *
 *
 *----------------------------------------------------------------------------
 *
 * File Description:
 *
 * It utilizes ethernet uici.c & uici.h functions
 *
 * It acts as a server and opens a port number --  1025
 *
```

```
 * Initialization:
 *
 * Inputs:
 *   THIS IS STARTED ON THE COMMAND LINE
 *   ./changingArray
 *
 *   sends information via e-net - to waiting Windows GUI
 *
 * Outputs:
 *
 *=========================================================================
 *************************************************************************/

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

#include <limits.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>  //for child processes WNOHANG
#include <fcntl.h>  //O_WRONLY, O_RDONLY ...
#include <stdint.h>
#include <errno.h>
#include <sys/stat.h>
#include <asm/param.h>
#include <pthread.h>
#include <sched.h>

#include <string.h>

#include <time.h>

#include "uici.h"  //=== e-net stuff

#include "cycloneVserver.h"

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

/*************************************************************************
 *
 *  Global Data
 *
 *************************************************************************/


static unsigned int    totalBytesSaved = 0;

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

#define CURRENT_USLEEP_VALUE 1000000  //


#define WORDS_IN_BUFFER 375000

unsigned int separateBuffer[WORDS_IN_BUFFER];

#define EXIT    0
#define BYTE    unsigned char

char *systemCommand;

//////////////////////////////
///////////////
extern char *environ;   ///commented out when added #define _GNU_SOURCE
// ============ GLOBALS


char                temp;
short               configurationReceived = -1;
```

```
short               softwareStarted      = -1;

pid_t               dataRecordServerPid  = 0;
pid_t               waitreturnpid     = 0;


u_port_t portnumber;
int      listenfd;
int      communfd;

char     remote[MAX_CANON];   //used for e-net connection

char     sysBuf[100];

struct threadParams
{
       int record;
       int misc;
       int threadActivated;
};


//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

//------------- ETHERNET STUFF
void usr1handler(int s)
{
   fprintf(stderr, "cycloneVserver  SIGUSR1 signal caught by server\n");
}

void usr2handler(int s)
{
   fprintf(stderr, "cycloneVserver  SIGUSR2 signal caught by server\n");
}

void installusrhandlers()
{
    struct  sigaction  newact;
    newact.sa_handler = usr1handler;      /* set the new usr1 handler */
    sigemptyset(&newact.sa_mask);    /* no additional signals blocked */
    newact.sa_flags = 0;            /* nothing special on the options */
    if (sigaction(SIGUSR1, &newact, (struct sigaction *)NULL) == -1) {
       perror("cycloneVserver  Could not install SIGUSR1 signal handler");
       return;
    }
    newact.sa_handler = usr2handler;      /* set the new usr2 handler */
    if (sigaction(SIGUSR2, &newact, (struct sigaction *)NULL) == -1) {
       perror("cycloneVserver  Could not install SIGUSR2 signal handler");
       return;
    }
    fprintf(stderr,
       "cycloneVserver  Server process %ld set to use SIGUSR1 and SIGUSR2\n",
       (long)getpid());
}



//----------------- END ETHERNET STUFF
/////////////////////////////////////////////////////////////////////////////////
#define OFFSET  0x0400000
void *threadRecord ( void * threadArgs )
{

    struct threadParams* tArgs = (struct threadParams*)threadArgs;
    ssize_t  byteswritten;

    int                        whileCount = 0;
    int                        i;
    time_t Estart, Eend;
    double Eseconds = 0.0;
```

```
   printf("RECORD THREAD STARTED record status %d \n", tArgs->record );

   i = 0;
    while (1)
    {

if (tArgs->record)
{


//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

////***************************** ETHERNET OUT
               Estart = time(NULL);
               byteswritten = u_write( communfd, separateBuffer, WORDS_IN_BUFFER*4 );

               Eend = time(NULL);
               Eseconds = difftime(Eend, Estart );
               printf("Enet %e \n", Eseconds );
               if (byteswritten != WORDS_IN_BUFFER*4 )
               {
                       fprintf(stderr,"RECORD  Error writing  %ld bytes written\n",
(long)byteswritten);
                       usleep(100000);
                       break;
               }
               totalBytesSaved      += byteswritten;

               usleep(CURRENT_USLEEP_VALUE);

               /// change the buffer
                for( i=0; i< WORDS_IN_BUFFER; i++ )
                       separateBuffer[i] = i + whileCount;


////****************************  END ETHERNET OUT




//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       whileCount++;
} /// RECORD
    }   // END WHILE (1)


    pthread_exit(1);

}
/////////////////////////////////////////////////////////////////////////////////
int main(int argc, char *argv[])
{
     // =========== Variables for used with shared memory

  BYTE                loopControl = 1;   //EXIT = 0

//=====================================  RECORD Thread
  pthread_t           pthreadRecord;
  int                 pthreadRecordRet;
  struct threadParams recordThreadArgs;          // for recording status --->

  ssize_t             bytesread;
  //ssize_t             byteswritten;
  char                bufin[100];              //used for e-net configuration --  from client
  int                 i;            // loop variable for delay
  int                 j;        recordThreadArgs.record      = 0;    // variable in display data
  //int                 len;
```

```
/// THREAD PRIORITY STUFF
   int nMaxPriority;
   struct sched_param tSchedParam;
   int  nResult;
/// end THREAD Priority Stuff


    //==================  ETHERNET CONNECTION
/*
 *  This is a server which opens a connection to a
 *  port number and listens for a request.  It then
 *  opens a communication port and writes to the connection
 *  until the connection is terminated.
 */

      //==================  START RECORD THREAD ==================
    recordThreadArgs.record    = 0;
    recordThreadArgs.misc      = 0;

    if ( recordThreadArgs.threadActivated != 1 )     //if 1st time create thread
    {
                  recordThreadArgs.threadActivated = 1;
                  pthreadRecordRet = pthread_create ( &pthreadRecord, NULL, (void
*)threadRecord,(void *) &recordThreadArgs );

                 printf("pthreadRecordRet %d \n", pthreadRecordRet);
    }
  /// Make HIGHEST priority thread
  ////printf("MAKE THE LED THREAD THE LOWEST PRIORITY\n");
 /*  nMaxPriority = sched_get_priority_max( SCHED_FIFO );
   tSchedParam.sched_priority        =  (nMaxPriority-5);
   nResult = pthread_setschedparam(pthreadRecord,  SCHED_FIFO,  &tSchedParam);
   if( nResult != 0)
   {
      perror("Error: Setting priority of RECORD THREAD");
   }
   printf("thread priority %d \n", nMaxPriority-10 );
*/
      //==================  END START RECORD THREAD ==============

RESTART:
    installusrhandlers();
    portnumber          = (u_port_t)CENTRAL_MAIN_PORT_NUMBER;

    fprintf(stderr,"cycloneVserver  port Number we are trying to open %d \n", portnumber );

    if ((listenfd = u_open(portnumber)) < 0) {
       u_error("cycloneVserver  Unable to establish a port connection");
       exit(1);
    }



//================================== MAIN LOOP -- LOOP FOREVER ====================
//=================================================================================

while (1)
  {

    //=================  ETHERNET
    if ((communfd = u_listen(listenfd, remote)) < 0) {
       u_error("cycloneVserver Failure to listen on server");
       exit(1);
    }
    fprintf(stderr, "cycloneVserver  Connection has been made to %s\n", remote);
    //when we get the above message then the client (analysis computer) has
    //opened a socket.

    //================== END ETHERNET -- MAKING CONNECTION WITH THE CLIENT

    loopControl = !EXIT;
```

```
  while (  loopControl != EXIT )
    {

    //Read a byte to know what well be doing ---
    //wait for 1 byte request from client
    fprintf(stderr,"cycloneVserver  Ready to read a byte request \n");

    usleep(1000000);   // we don't need to process requests that quickly --> 23 April 2015

    bytesread = u_read(communfd, bufin, 1 );

    if (bytesread <= 0) //they have closed the connection
      {

      loopControl = EXIT;

      fprintf(stderr,"cycloneVserver  --->  CLOSED e-net connection bytesread %d \n", bytesread);
      for (i=0; i<10000; i++)
        for (j=0; j<10000; j++)
          ; //delay a bit so the e-net has time to close

      }
    else
      {

       fprintf(stderr,"cycloneVserver  -- request received from client 0x%x--->>>>
\n",bufin[0]);

  //AFTER WE GET THE CONFIGURATION WE CAN RUN

        switch((BYTE) bufin[0] )
          {
          case 0xBB:           // CLOSE ENET CONNECTION
            printf("CLOSE CONNECTION \n");
            fprintf(stderr," CLOSE REQUESTED .... connection will close and re-open for listening
\n");
            u_close(listenfd);
            u_close(communfd);
            goto RESTART;
            break;
          case 0xAA:           // STOP RECORDING
            recordThreadArgs.record    = 0;
            printf("STOP RECORDING \n");
            break;

          case 0x55:   // start record -- SO I NEED TO START SENDING DATA !!! - another task
            printf("START RECORDING \n");
            recordThreadArgs.record    = 1;

            break;

          default:
            fprintf(stderr,"cycloneVserver  Unknown command \n");
            break;
          }

        }//end else of if-then-else



      }// ++++++++++++++++++ END THE WHILE LOOP -- loopControl



  }//end while (1)
  fprintf(stderr," DATA RECORD  --  HALTED  .... connection will close and re-open \n");
  u_close(listenfd);
  u_close(communfd);
  goto RESTART;
//========================================== MAIN LOOP -- LOOP FOREVER END ====================
```

```
//=============================================================================================
  return 0;

}
```

staticArray.c

```
// FILENAME:         staticArray.c
/*****************************************************************************
 ***************
 *
 * Created by:      Janet Estabridis
 *
 *
 * Date created:   1 May 2015
 * Date revised:   1 May 2015
 * Reason for revision:
 *  1 May 2015
 *
 *
 *---------------------------------------------------------------------------
 *
 *
 * File Description:
 *
 * It utilizes ethernet uici.c & uici.h functions
 *
 * It acts as a server and opens a port number --  1025
 *
 * Initialization:
 *
 * Inputs:
 *   THIS IS STARTED ON THE COMMAND LINE
 *   ./changingArray
 *
 *   sends information via e-net - to waiting Windows GUI
 *
 * Outputs:
 *
 *===========================================================================
 ****************************************************************************/

///////////
//////////////////////////#define _GNU_SOURCE

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

#include <limits.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>  //for child processes WNOHANG
#include <fcntl.h>  //O_WRONLY, O_RDONLY ...
#include <stdint.h>

#include <errno.h>

#include <sys/stat.h>
#include <asm/param.h>
#include <pthread.h>
#include <sched.h>

#include <string.h>



#include <time.h>

#include "uici.h"  //=== e-net stuff
```

```
#include "cycloneVserver.h"

//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////

/***************************************************************************
 *
 *  Global Data
 *
 ***************************************************************************/

static unsigned int    totalBytesSaved = 0;


//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////

#define CURRENT_USLEEP_VALUE 1000000  //


#define WORDS_IN_BUFFER 375000

unsigned int separateBuffer[WORDS_IN_BUFFER];


#define EXIT    0
#define BYTE    unsigned char

char *systemCommand;

/////////////////////////////
///////////////
extern char *environ;   ///commented out when added #define _GNU_SOURCE
// ============ GLOBALS


char                temp;
short               configurationReceived = -1;
short               softwareStarted       = -1;
pid_t               dataRecordServerPid  = 0;
pid_t               waitreturnpid     = 0;


u_port_t portnumber;
int      listenfd;
int      communfd;

char     remote[MAX_CANON];   //used for e-net connection

char     sysBuf[100];

struct threadParams
{
       int record;
       int misc;
       int threadActivated;
};


//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

//------------- ETHERNET STUFF
void usr1handler(int s)
{
   fprintf(stderr, "cycloneVserver  SIGUSR1 signal caught by server\n");
}

void usr2handler(int s)
{
   fprintf(stderr, "cycloneVserver  SIGUSR2 signal caught by server\n");
```

```
}

void installusrhandlers()
{
    struct  sigaction  newact;
    newact.sa_handler = usr1handler;      /* set the new usr1 handler */
    sigemptyset(&newact.sa_mask);    /* no additional signals blocked */
    newact.sa_flags = 0;            /* nothing special on the options */
    if (sigaction(SIGUSR1, &newact, (struct sigaction *)NULL) == -1) {
       perror("cycloneVserver  Could not install SIGUSR1 signal handler");
       return;
    }
    newact.sa_handler = usr2handler;      /* set the new usr2 handler */
    if (sigaction(SIGUSR2, &newact, (struct sigaction *)NULL) == -1) {
       perror("cycloneVserver  Could not install SIGUSR2 signal handler");
       return;
    }
    fprintf(stderr,
       "cycloneVserver  Server process %ld set to use SIGUSR1 and SIGUSR2\n",
       (long)getpid());
}



//----------------- END ETHERNET STUFF
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
#define OFFSET  0x0400000
void *threadRecord ( void * threadArgs )
{

    struct threadParams* tArgs = (struct threadParams*)threadArgs;
    ssize_t  byteswritten;

    int                        whileCount = 0;
    int                        i;
    time_t Estart, Eend;
    double Eseconds = 0.0;

   printf("RECORD THREAD STARTED record status %d \n", tArgs->record );

   i = 0;

   for( i=0; i< WORDS_IN_BUFFER; i++ )
        separateBuffer[i] = 0;
    while (1)
    {

if (tArgs->record)
{

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

////***************************** ETHERNET OUT
               Estart = time(NULL);
               byteswritten = u_write( communfd, separateBuffer, WORDS_IN_BUFFER*4 );

               Eend = time(NULL);
               Eseconds = difftime(Eend, Estart );
               printf("Enet %e \n", Eseconds );
               if (byteswritten != WORDS_IN_BUFFER*4 )
               {
                       fprintf(stderr,"RECORD  Error writing  %ld bytes written\n",
(long)byteswritten);
                       usleep(100000);
                       break;
               }
               totalBytesSaved      += byteswritten;
```

```
                usleep(CURRENT_USLEEP_VALUE);

                /// change the buffer
                for( i=0; i< WORDS_IN_BUFFER; i++ )
                {
                        if (i==0) printf("."); fflush(stdout);
                        separateBuffer[i] = 0x12345678;
                }

////****************************  END ETHERNET OUT



//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       whileCount++;
} /// RECORD
    }   // END WHILE (1)
    pthread_exit(1);

}


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int main(int argc, char *argv[])
{
     // =========== Variables for used with shared memory

  BYTE                loopControl = 1;   //EXIT = 0

//====================================  RECORD Thread
  pthread_t           pthreadRecord;
  int                 pthreadRecordRet;
  struct threadParams recordThreadArgs;          // for recording status --->

  ssize_t             bytesread;
  //ssize_t             byteswritten;
  char                bufin[100];              //used for e-net configuration --  from client
  int                 i;            // loop variable for delay
  int                 j;        recordThreadArgs.record      = 0;    // variable in display data
  //int                 len;
/// THREAD PRIORITY STUFF
   int nMaxPriority;
   struct sched_param tSchedParam;
   int  nResult;
/// end THREAD Priority Stuff

    //==================  ETHERNET CONNECTION
/*
 *  This is a server which opens a connection to a
 *  port number and listens for a request.  It then
 *  opens a communication port and writes to the connection
 *  until the connection is terminated.
 */

      //==================  START RECORD THREAD ==================
    recordThreadArgs.record    = 0;
    recordThreadArgs.misc      = 0;

    if ( recordThreadArgs.threadActivated != 1 )     //if 1st time create thread
    {
                  recordThreadArgs.threadActivated = 1;
                  pthreadRecordRet = pthread_create ( &pthreadRecord, NULL, (void
*)threadRecord,(void *) &recordThreadArgs );

                 printf("pthreadRecordRet %d \n", pthreadRecordRet);
    }
  /// Make HIGHEST priority thread
  ////printf("MAKE THE LED THREAD THE LOWEST PRIORITY\n");
```

```
 /*  nMaxPriority = sched_get_priority_max( SCHED_FIFO );
   tSchedParam.sched_priority        =  (nMaxPriority-5);
   nResult = pthread_setschedparam(pthreadRecord,  SCHED_FIFO,  &tSchedParam);
   if( nResult != 0)
   {
      perror("Error: Setting priority of RECORD THREAD");
   }
   printf("thread priority %d \n", nMaxPriority-10 );
*/
      //==================  END START RECORD THREAD ==============

RESTART:
    installusrhandlers();
    portnumber          = (u_port_t)CENTRAL_MAIN_PORT_NUMBER;

    fprintf(stderr,"cycloneVserver  port Number we are trying to open %d \n", portnumber );

    if ((listenfd = u_open(portnumber)) < 0) {
       u_error("cycloneVserver  Unable to establish a port connection");
       exit(1);
    }



//================================== MAIN LOOP -- LOOP FOREVER ====================
//=================================================================================

while (1)
  {

    //=================  ETHERNET
    if ((communfd = u_listen(listenfd, remote)) < 0) {
       u_error("cycloneVserver Failure to listen on server");
       exit(1);
    }
    fprintf(stderr, "cycloneVserver  Connection has been made to %s\n", remote);
    //when we get the above message then the client (analysis computer) has
    //opened a socket.

    //================== END ETHERNET -- MAKING CONNECTION WITH THE CLIENT

    loopControl = !EXIT;

  while (  loopControl != EXIT )
    {

    //Read a byte to know what well be doing ---
    //wait for 1 byte request from client
    fprintf(stderr,"cycloneVserver  Ready to read a byte request \n");

    usleep(1000000);   // we don't need to process requests that quickly --> 23 April 2015

    bytesread = u_read(communfd, bufin, 1 );

    if (bytesread <= 0) //they have closed the connection
      {

      loopControl = EXIT;

      fprintf(stderr,"cycloneVserver  --->  CLOSED e-net connection bytesread %d \n", bytesread);
      for (i=0; i<10000; i++)
        for (j=0; j<10000; j++)
          ; //delay a bit so the e-net has time to close

      }
    else
      {

       fprintf(stderr,"cycloneVserver  -- request received from client 0x%x--->>>>
\n",bufin[0]);
```

```
  //AFTER WE GET THE CONFIGURATION WE CAN RUN

        switch((BYTE) bufin[0] )
          {
          case 0xBB:           // CLOSE ENET CONNECTION
            printf("CLOSE CONNECTION \n");
            fprintf(stderr," CLOSE REQUESTED .... connection will close and re-open for listening
\n");
            u_close(listenfd);
            u_close(communfd);
            goto RESTART;
            break;
          case 0xAA:           // STOP RECORDING
            recordThreadArgs.record    = 0;
            printf("STOP RECORDING \n");
            break;

          case 0x55:   // start record -- SO I NEED TO START SENDING DATA !!! - another task
            printf("START RECORDING \n");
            recordThreadArgs.record    = 1;

            break;

          default:
            fprintf(stderr,"cycloneVserver  Unknown command \n");
            break;
          }

         }//end else of if-then-else



       }// ++++++++++++++++++ END THE WHILE LOOP -- loopControl



  }//end while (1)
  fprintf(stderr," DATA RECORD  --  HALTED  .... connection will close and re-open \n");
  u_close(listenfd);
  u_close(communfd);
  goto RESTART;
//========================================== MAIN LOOP -- LOOP FOREVER END ====================
//=============================================================================================
  return 0;

}
```