4. Demonstration Programs

The purpose of the demonstration programs is to demonstrate software, or software techniques, on the MachZ integrated development system. The development system provides a general-purpose platform for trying out things with the MachZ. The development system is convenient in that the PCI in ISA slots are available to try out peripherals, and because the development system comes up and running (right out of the box) with software already on hard disk.

Wherever possible, these demonstration programs are already installed on the hard disk. In addition, the source and binary files appear on the accompanying CD. In the same manner that we annotated the schematic of the board, we provide some annotation and notes on the software source code, and we provide instructions on how to build the software.

There are many different ways to build and test software. In general, I build software on a host system which runs close to a GHz and has a very high-definition video display. Some software is built using a command line interface and some sort of a batch or makefile. Other software is built using some sort of IDE (integrated development environment). Testing can be done on the target or perhaps with some sort of umbilical cable using a debugger on the host.

In the case of VxWorks, many developers do virtually everything on the host using Tornado 2. The Tornado 2 environment allows individual modules to be rebuilt and downloaded to the target while the target remains running.

As we evolve this set of demonstration programs for your use, you will see in this and in subsequent editions of the quick start guide an increasing variety of demonstrations and development techniques.

4.1. VxWorks Shell Demo

The shell demo is the simplest possible demo we have for VxWorks. It is generated from one directory, and it brings up a VxWorks shell to which you can add your own C program as a task. A trivial "hello world" program is in there as a starting point.

The VxWorks demos currently load off the DOS partition. You may want to upgrade your DOS partition (see ‘Using CD ROM Drive from DOS’ on page 28), but that is not necessary for the demo to operate. You may also wish to set the default boot of your IDS to DOS rather than Linux. See ‘Set the Boot Default to DOS’ on page 28.

4.1.1. Running the Shell Demo

The demo software is pre-installed on the MachZ Integrated Development System in the DOS partition:

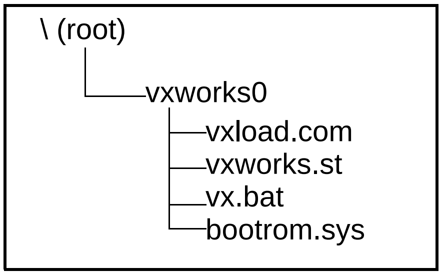


Figure 4-5 VxWorks Shell Demo

To run the program, change to the vxworks0 directory and type vx. Here is what it does:

```
1          del \vxworks.*
2          copy vxworks.st \
3          vxload bootrom.sys
```

vx.bat in line 1 removes any previous vxworks.* file from the root (as both demos put their own version in the root). In line 2 it copies the vxworks.st image to the root, and in line 3 it invokes VxLoad. VxLoad is a DOS program which will load a VXworks executable from the

DOS file system[1]. We use VxLoad to load bootrom.sys. In turn, bootrom.sys loads vxworks.st from the root of the hard disk. How does bootrom.sys know to do that? See the DEFAULT BOOT LINE just below!

If you get the error message the error message "Image memory is occupied - try to reduce system space" -- make sure that there is "nothing" in config.sys or autoexec.bat which will consume memory.

When you run the demo, VxWorks will come up in the shell. Once VxWorks is up, you should be able to ping the "e" address of 192.168.200.144. See the line below from VxWorks CONFIG.H:

```
#undef DEFAULT_BOOT_LINE \

"ata=0,0(0,0)host:/ata0/vxWorks.st
h=192.168.200.129 e=192.168.200.144 u=target tn=target pw=target o=elPci"
```

The shell command **hostShow** will show you your target and host IP addresses. They will be as above.

You may use the following VxWorks shell control characters and commands. Note that the commands are case sensitive, and that with the help commands you need to finish the scroll list <CR> or quit them Q<CR>.

Table 4-9: VxWorks Shell

Command	Description
CTRL+C	Abort and restart the shell.
CTRL+X	Reboot (trap to the ROM monitor).
help	print list of shell commands
i	list current tasks
debHelp	print debugger help info
netHelp	print network help info

Besides the shell commands, we have provided the classic "hello world" task. In this case it is:

```
#include <stdio.h>
void charlie_task (void);
int sam;
void charlie_task (void) {
printf ("Hello There");
}
```

Figure 4-6 source file charlie.c

You can run this task from the shell by entering **sp charlie_task** -- and you can see the address of the symbol charlie_task by typing **lkup "charlie_task"** or **lkup "sam"**. You will note that charlie_task is a text symbol and sam is a bss (block starting with symbol) or data symbol. In the next section you can see how to rebuild (or modify and rebuild) these files.

4.1.2. Rebuilding the Shell Demo

The shell demo is simple in that the target files BOOTROM.SYS and VXWORKS.ST are both built from within the same directory. It is perhaps simplistic in that it does not show off the VxWork Tornado Integrated Development Environment. That said, here’s how to do it:

Install Tornado (we will use the compiler and editor, but not the IDE) on your host development system. Create a directory in the root called ataboot (you may use your own name). Then copy into that directory the contents of the IDS CD folder ataboot: VxWorks Demos DOS Bootable\VxWorks Shell Demo\ataboot.

Rebuilding BOOTROM.SYS

To rebuild bootrom.sys, execute the file makbootunc.bat. This file contains:

```
call \Tornado\host\x86-win32\bin\torvars
make clean
make bootrom_uncmp
copy bootrom_uncmp a:\bootrom.sys
```

[1] There are many ways of getting a VxWorks image into memory. In a typical target system, VxLoad is *not* the way to go. However, it serves our purpose nicely on the IDS.

In line 1 we execute torvars.bat which sets up the path for the Tornado tools. If you do this in a DOS window, after a while your environment string gets rather long as you keep calling torvars.bat.

Rebuilding VXWORKS.ST

To rebuild vxworks.st, execute the file makst.bat. This file contains:

```
1    call \Tornado\host\x86-win32\bin\torvars
2    rem make clean
3    make vxWorks.st
4    copy vxWorks.st a:\
```

The makefile contains a macro (a define) for MACH_EXTRA as follows:

MACH_EXTRA = charlie.o # crc 08-09-00

This statement will cause charlie.c to be compiled and included in the vxworks image. You can also compile charlie.c by typing **make charlie.o** in the ataboot directory (once torvars has been called). You do not have to open the tornado IDE to do any of this.

4.2. VxWorks Menued Demo

The VxWorks Menued Demo features a "real" application program and also uses the Tornado IDE (project facility) to build the VxWorks image. The VxWorks demo program itself currently uses a text menu, but a future demo will make use of the Zinc Application to provide a graphics interface.

When you run text-mode menu can select desired items and perform specific actions. Included are:

- Ping demo
- Network transfer speed measuring tests
- FTP server demo
- RAM-disk performance measuring test
- Hard disk performance measuring test
- Information about running tasks
- Stop running test processes
- Exit to VxWorks Shell

The user is able to run multiple demo instances or performance tests concurrently as separate tasks and thus see the performance impact to whole system. Not all items can be run as concurrent tasks, further features are described in the "Using the Demo Software" and "Test menu items in detail" chapters.

Required Target Hardware

The standard MachZ Integrated Development system is needed for running this VxWorks demo program. This includes:

- 3Com905TX 10/100 network card (required for network tests)
- Hard disk with 10-100 Mbytes of free space (needed for HD performance tests)

Required Host Hardware/SW

You will only need a host computer if you decide to modify the VxWorks Menued demo. To do this, you will need to use the Tornado Tools. You are entitled to a 60 day free evaluation of those tools with the purchase of the Integrated Development System (see WindRiverReadMe.PDF).

4.2.1. Running the Menued Demo

The demo software is pre-installed on the MachZ Integrated Development System in the DOS partition:

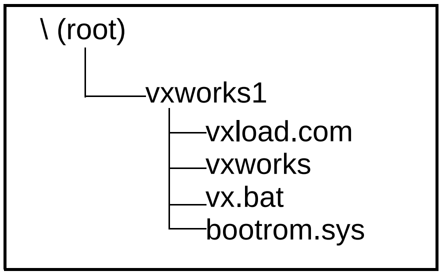


Figure 4-7 VxWorks Shell Demo

To run the program, change to the vxworks1 directory and type vx. Here is what it does:

```
1         del \vxworks.*
2         copy vxworks \
3         vxload bootrom.sys
```

Using the Menued Demo

The source code of the demo follows.

When vxworks demo is successfully loaded by bootrom and succesfully starts, it first asks user for the desired RAM-disk size. See Line 78 on page 40. Press Enter key to allocate 32 Mbytes for RAM-disk or enter any other amount in kilobytes. When the RAM-disk is created as "/RAMDISK" device, a corresponding message is displayed.

Then the clock watchdog is started which is used for measuring elapsed time during performance tests. In addition the hard disk is attached as "/ATA" device. Finally, the IDS VxWorks Demonstration Program main menu is presented to user. The items in menu are:

1. PING test
2. Net Receiver Test
3. Net Sender Test
4. Net Loopback Test
5. FTP Server Test
6. Hard disk performance test
7. RAM-disk performance test

i Show running tasks info

d Show info about available devices

s Stop running processes

q Exit to Shell

An "Enter option (h for help): " - prompt is presented to user and user should select corresponding number or letter and press the Enter-key.

When some of the tests are executed, the output results will appear after certain intervals on the bottom of the screen. Other lines on screen will be scrolled up and finally off the screen. When multiple processes are running and user wants to start additional test-tasks or end some, then the user should just type in the appropriate number or letter followed by Enter.

The "**s**" menuitem allows you to stop all currently running network tests and performance tests.

The "**i**" menuitem shows info about currently running tasks and their status.

The "**d**" menuitem shows info about all defined devices (serial ports, RAM-disks, block devices) in the system.

The "**q**" menuitem ends the IDS Demonstration Program and exits to the VxWorks interactive Shell.

The IDS demo program can the be restarted only by rebooting the system. To do this, press CTRL-X at the shell prompt.

Demo Menu Items In Detail

1. PING Test

Ping is just meant for testing network connectivity between different machines. The user is prompted for the target computer's IP address and repetition count. The Ping to different machines can be executed multiple times and thus multiple Ping tasks are spawned in the VxWorks environment. The Ping test is able to run concurrently with all other test items. The Ping test cannot be terminated by pressing "s" in user menu so please be cautious with entering the ping repetition count.

2. Net Receiver Test

This item starts a network listener task for a certain TCP-IP network port on the test machine. The user is prompted for a port number. When the sender task is also executed somewhere in the network for this IDS computer and directed to the same port, then the listener task prints out a network transfer rate every 10 seconds. There is no output in the case when there is no network traffic. The network sender task can be launched using the menu item "3".

You may run multiple receiver tasks for different port numbers on the same machine.

The receiver tasks can be running concurrently with all other test item tasks.

3. Net Sender Test

This starts a network sender task which sends packets to certain destination machine's certain port. The machine IP address and port number are asked from user. When there is no network receiver task launched on target computer for the same port number then the sender task will also exit with corresponding error message (connect failed).

There can be running multiple sender tasks on the same machine with different target IP-s or even for different ports on the same machine.

The sender tasks can be running concurrently with all other test items except FTP server test.

When sender task is running on the IDS machine for example, then FTP file transfer from remote host to this IDS machine is not possible.

4. Net Loopback test

This menu item is actually a combination of two previous items. It prompts user for a desired port number and then starts both the network receiver task and sender task on the same machine (actually for IP address 127.0.0.1 which is localhost) for the same port. There can be also multiple concurrent network loopback tests running in the system and they can be running concurrently with all other test items except FTP server. When the sender task is running on this machine, then FTP file transfer from remote host to this IDS machine is not possible.

5. FTP Server Test

The FTP server is actually running on the system as soon as the demo is started. When you select "5" from the main menu, instructions are displayed which describe how to do a FTP file transfer from a remote machine to this test machine's RAMDISK.

During network loopback tests, net sender tests, or disk performance tests, the FTP server tasks do no respond because of lower proirity of FTPServer tasks.

Instead of the proposed "/RAMDISK" directory the user can also do a "cd /ATA" on remote computer's FTP client prompt and thus transfer a test file also to IDS test machine's hard disk.

6. Hard Disk Performance Test
7. RAM-Disk Performance Test

The menuitems "6" and "7" use actually the same subroutine for performing disk access transfer rate measuring, only in case of "6" the test files will be created on hard disk and in case "7" the test files are created on RAM-disk.

The user is prompted for a test file size in kilobytes and the test repetition count.

After each read or write cycle to the target device the read or write transfer rate is displayed on screen.

There can be multiple simultaneous disk transfer tests running and they can run simultaneously with all test items except FTP test. During disk tests the FTP server tasks are in a "pending" state because of their lower priority.

4.2.2. Building Menued Demo Software

The demo software itself includes binaries and source code for building needed Board Support Package (bootrom.sys) and for building demo program (vxworks) using the WindRiver Tornado 2.0 IDE. (In part 4.1, we used the compiler and other GNU tools installed with the IDE, but we did not use the IDE.)

Rebuilding BOOTROM.SYS

BOOTROM.SYS and VXWORKS are built from different directories. BOOTROM.SYS is built from the DOS command line (as in the previous demo), but VXWORKS is built using the Tornado Project Facility.

That said, here’s how to do it:

Install Tornado (we will use the compiler and editor, but not the IDE) on your host development system. Create a directory in the root called ataboot1 (you may use your own name). Then copy into that directory the contents of the IDS CD folder ataboot1: VxWorks Demos DOS Bootable\VxWorks Menued Demo\ataboot1.

To rebuild bootrom.sys, execute the file makboot.bat. This file contains:

```
call \Tornado\host\x86-win32\bin\torvars
make clean
make bootrom_uncmp
copy bootrom_uncmp bootrom.sys
```

Before building BOOTROM.SYS, you may set appropriate IP addresses for host and target in \config.h. There in config.h are defined in multiple boot lines, edit the one which is not commented (undef) out. If you want to use the over-the-net boot, comment the ata boot line and uncomment in the net boot line.

Copy the new bootrom.sys to the IDS hard disk into directory VxWorks1.

Rebuilding VXWORKS

Build vxworks with Tornado Project facility. The first step is to copy some files from the IDS CD to your host system. Look on the IDS CD for VxWorks Demos DOS Bootable\VxWorks Menued Demo\Tornado\target\proj. Copy the files from the ...Tornado\target\proj folder on the CD to the Tornado\target\proj folder on your host. Then click on the wsp file.[1]

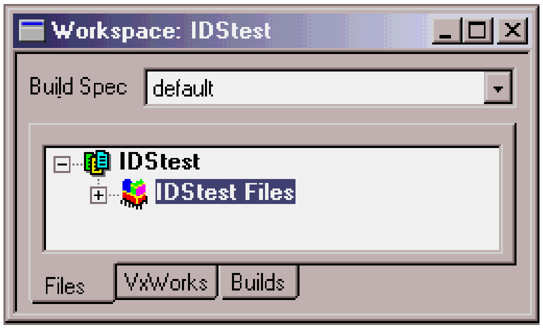


Figure 4-8 IDStest: Files

You can expand the file list with the [+] key, and if you subsequently click right on any file you get a menu of actions you can perform on the file. This is called a "context" menu in that it represents things that you might want to do in the current "context".

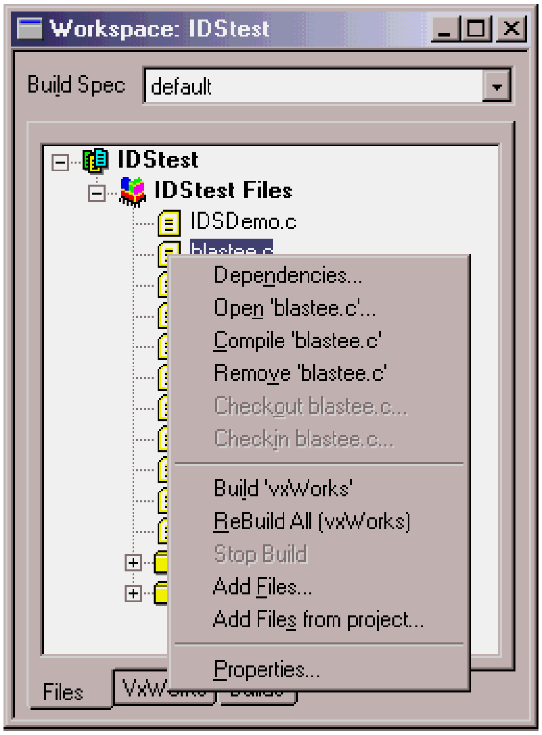


Figure 4-9 File Context Menu

[1] If you hit F1 and get the Tornado help it says: "The workspace window divides your project information into three categories: Files, VxWorks, and Builds. Move between the three categories by using the tab controls at the bottom of the workspace window.

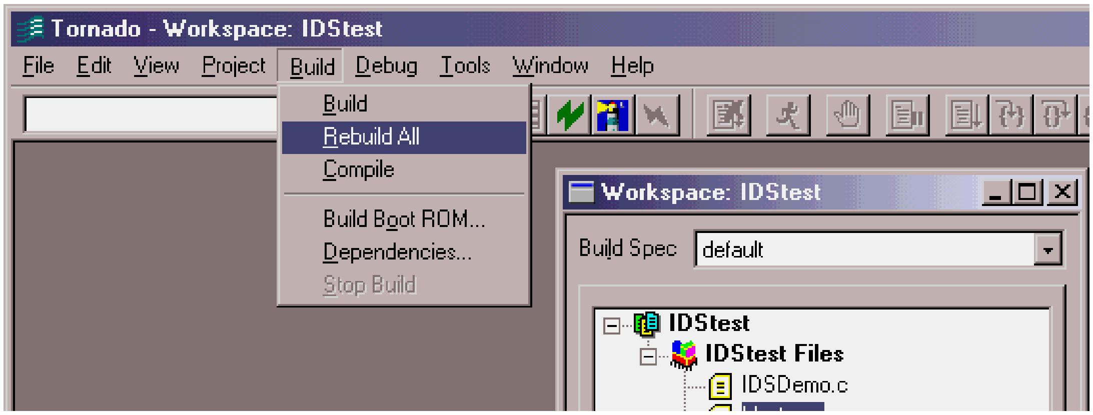
The Files view displays information about the files associated with the projects in the workspace.

The VxWorks view displays information about the operating system components that may be included in VxWorks or bootable application projects. This view is empty for downloadable application projects.

The Builds view displays information about the builds specifications defined for projects in the workspace."

Tornado Build ® Rebuild All

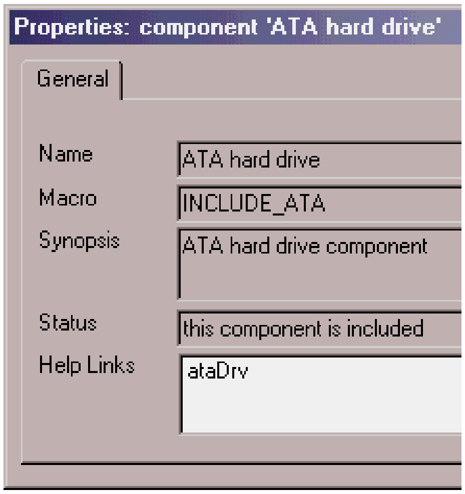
If you click "Build" on the Tornado Menu Bar, you will see a pull-down which also allows you to rebuild the project.



Build Spec: default

Figure 4-10 Tornado - Build - Rebuild All

Prior to a build, you can go to the VxWorks tab of the Workspace:IDStest panel and expand the list to see which components of VxWorks are included in the build. You will note that the ATA hard drive component is enabled, and that the IDE hard drive is not. If you click right on ATA hard drive, you will see the component properties. note that "macro" or include for this is INCLUDE_ATA. That is the philosophy of the project tool: the project tool sets the necessary includes in the configuration files for you, and it checks for dependencies.



Name: ATA hard drive

Macro: INCLUDE_ATA

Synopsis: ATA hard drive component

Status: this component is included

Help Links: ataDrv

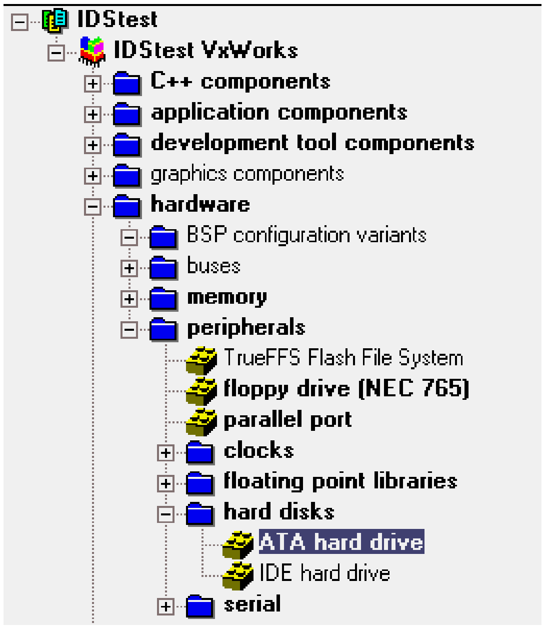


Figure 4-11 IDStest: VxWorks

If you execute Rebuild all, you will see the build output on the screen, and at the end of that you will see:

```
ld386 -X -N -e _sysInit -Ttext 00120000 \
          dataSegPad.o partialImage.o ctdt.o symTbl.o -o vxWorks
C:\TORNADO\host\x86-win32\bin\vxsize 386 -v 00020000 00120000 vxWorks
vxWorks: 705920(t) + 85912(d) +  34012(b) = 825844
Done.
```

In this example, the total size of VxWorks is 825,844 bytes comprised of (t) text, (d) data, and (b) bss. Bss represents uninitialized data and stands for block starting with symbol.

If you now look into \Tornado\target\proj\IDStest\default, you will find the object files created and the vxWorks file. Copy the vxWorks file, which is a vXworks image, to the vxworks1 directory of your IDS.

Optional Network Boot Feature

Many Wind River developers set up their BOOTROM.SYS so that it will get the vxworks image from the host over the LAN. In that way, you do not have to copy the VxWorks file to the target via a floppy.

If you built your bootrom.sys for network boot, then after the bootrom.sys is executed, it automatically loads the vxworks file from host computer over the network and then launches it on the IDS. In order the loading to succeed, the Wind River "FTP server" must be running on your host computer. The "FTP server" software is installed along with Tornado 2 installation and it can be found in the Start menu under "Programs\Tornado2". Run the FTP server and check the "User Rights" option under the "Security" menu. Add a new user named "target" with password "target" and set it's home directory to be D:\TORNADO\TARGET\PROJ\IDSTEST\DEFAULT where d:\tornado is your Tornado2 main installation directory.

Make sure that before building a bootrom.sys for network booting also the vxworks-file path corresponds fully to your build path. The example network boot line in config.h is:

```
#define DEFAULT_BOOT_LINE
"elPci(0,0)host:/tornado/target/proj/idstest/default/vxWorks h=192.168.100.34 e=192.168.100.15 u=target tn=target pw=target o=elPci f=8"
```

In this case the path to vxworks is /tornado/target/proj/idstest/default/vxWorks. The "h" parameter in boot line describes the remote host's (the development host) IP address and "e" parameter defines the demo-machine's IP address.

4.2.3. IDS Menu Demo Main Source File IDS Demo.C

```
/* this IDS Demo software main body file

Version History
0.0.1  14.08.00 RaJ  First Draft

*/
#include "stdio.h"
#include "taskLib.h"
#include "pingLib.h"
#include "shellLib.h"
#include "kernelLib.h"
#include "usrLib.h"
#include "wdLib.h"

int makeRamDisk (int sizeK, char * RDname);
int blastee(int port, int size, int blen);    /* port,2000,16000 */
int blaster(char * destAddr,int port, int size, int blen);    /* port,2000,16000 */

int DiskRWTest(char * diskname,int fsize,int repcnt);

extern int blasteeStop;
extern int blasterStop;

#define BUF_SIZE 1024

int writeNum=0;
int testfilecnt=0;

int exitflag=FALSE;
int stopflag=FALSE;

unsigned long clkticks=0;

WDOG_IDclkWd=NULL;
void clkWdFn(int);


/*****************************************************************************/
void demoHelp(void)
   {
   printf("\n***************************************************************\n");
   printf("IDS VxWorks Demonstration Program v.1.0 by RaJ %s
      %s\n",__DATE__,__TIME__);
   printf("***************************************************************\n\n");
```

```

    printf("        1. PING test\n");
    printf("        2. Net Receiver Test\n");
    printf("        3. Net Sender Test\n");
    printf("        4. Net Loopback Test\n");
    printf("        5. FTP server test\n");
    printf("        6. Hard disk performance test\n");
    printf("        7. RAM-disk performance test\n\n");
    /*
    printf("        8. Graphics demo\n");
    printf("        9. Internet Browser demo\n\n");
    */

    printf("        i  Show running tasks info\n");
    printf("        d  Show info about available devices\n");
    printf("        s  Stop running processes\n");
    printf("        q  Exits to Shell\n\n");
    }

/******************************************************************************/
int IDSDemo(void)
    {
    int j,pnum;
    int ret,filesize,rdsize=0,repcnt;
    int taskcnt=0,tid;
    char ch;
    char buf[255];
    char nbuf[80];
    char abuf[80];
    char sourcedisk[80];

    /* enable round-robin scheduling */
    kernelTimeSlice(1); /* 1 tick per task */

    /* create RAM-DISK */
    for(j=0;j<3;j++)
        {
            printf("Enter desired RAM-disk size in kbytes (press Enter for 32M): ");
        gets(buf);
        if(buf[0]==0)
            rdsize=32768;
        else
            sscanf(buf,"%d",&rdsize);
        ret=makeRamDisk(rdsize*1.2,"RAMDISK");
        if(ret==0)
             break;
        /* else try again for 2 times, then fail */
```

```
      }

if (clkWd == NULL && (clkWd = wdCreate ()) == NULL)
      {
      printf ("cannot create CLK watchdog\n");
      exit (1);
      }

sysClkRateSet(100);
wdStart (clkWd, 1, clkWdFn, clkWd);
printf("Clock watchdog started, rate set to 100 ticks/sec.\n");

/* connect to ATA disk */
ret=usrAtaConfig(0,0,"/ATA");
if(ret!=OK)
   {
   printf("Attaching to ATA disk failed!\n");
   }
else
   {
   printf("ATA disk attached OK.\n");
   }

/* kill possibly running shell task */
/*
tid=taskNameToId("tShell");
taskDelete(tid);
*/

demoHelp();

/* stay in demo as long as needed */
while(exitflag==FALSE)
   {
   printf("\nEnter option (h for help): ");
   gets(buf);
   ch=buf[0];

   /* now begin selection processing */
   switch(ch)
      {
      case ’h’:
      case ’H’:
               demoHelp();
         break;

      case ’1’:
         printf("\n**************** PING TEST *********************\n");
         printf("Enter IP to ping (press Enter for 127.0.0.1): ");
         gets(buf);
         if(buf[0]==0)
            {
            sprintf(buf,"127.0.0.1");
            }

         printf("Enter number of times to ping (press Enter for 10): ");
         gets(abuf);
         if(abuf[0]==0)
            {
            pnum=10;
            }
         else
            {
            sscanf(abuf,"%d",&pnum);
            }

         taskcnt++;
         sprintf(nbuf,"task%d",taskcnt);
         taskSpawn(nbuf,150,0,10000,ping,buf,pnum,0,0,0,0,0,0,0,0);
         printf("Ping task running for %d times...\n",pnum);
         break;

      case ’2’:
         printf("\n************* Net Receiver Test ***************\n");
         printf("Enter desired PORT nr (press Enter for 2000): ");
         gets(buf);
         if(buf[0]==0)
            {
            pnum=2000;
            }
         else
            {
            sscanf(buf,"%d",&pnum);
            }

         taskcnt++;
         sprintf(nbuf,"task%d",taskcnt);
         taskSpawn(nbuf,200,0,10000,blastee,pnum,2000,16000,0,0,0,0,0,0,0);
         printf("Receiver task running...\n");
         break;

      case ’3’:
         printf("\n************* Net Sender Test *****************\n");
         printf("Enter desired PORT nr (press Enter for 2000): ");
```

```
        gets(buf);
        if(buf[0]==0)
            {
            pnum=2000;
            }
        else
            {
            sscanf(buf,"%d",&pnum);
            }

        printf("Enter receiver machine’s IP address (press Enter for 127.0.0.1): ");
        gets(buf);
        if(buf[0]==0)
            {
            sprintf(buf,"127.0.0.1");
            }

        taskcnt++;
        sprintf(nbuf,"task%d",taskcnt);
        taskSpawn(nbuf,200,0,10000,blaster,buf,pnum,2000,16000,0,0,0,0,0,0);
        printf("Sender task running...\n");
        break;

    case ’4’:
        printf("\n************* Net Loopback Test ***************\n");
        printf("Enter desired PORT nr (press Enter for 2000): ");
        gets(buf);
        if(buf[0]==0)
            {
            pnum=2000;
            }
        else
            {
            sscanf(buf,"%d",&pnum);
            }

        taskcnt++;
        sprintf(nbuf,"task%d",taskcnt);
        printf("Starting receiver for port %d...\n",pnum);
        taskSpawn(nbuf,200,0,10000,blastee,pnum,2000,16000,0,0,0,0,0,0,0);

        sprintf(buf,"127.0.0.1");

        taskcnt++;
        sprintf(nbuf,"task%d",taskcnt);
        printf("Starting sending to %s:%d\n",buf,pnum);
        taskSpawn(nbuf,200,0,10000,blaster,buf,pnum,2000,16000,0,0,0,0,0,0);
        printf("Loopback test running...\n");
        break;

    case ’5’:
        printf("\n************ FTP Server Test ****************\n");
        printf(" FTP Server is already running, maximum space in /RAMDISK is %d kbytes.\n\n",rdsize);
        ifAddrGet("elPci0",buf);
        printf(" This machine’s IP address is %s\n\n",buf);
        printf(" Establish a FTP connection from remote machine to this \n \
        test machine by issuing ’FTP %s’ command from remote machine’s \n \
        command line. Enter ’target’ as user and ’target’ as password.\n \
        If successfully logged in, issue commands ’bin’ to set binary \n \
        transfer mode and ’cd /RAMDISK’ to set target directory on this \n \
        test machine.\n",buf);
       printf(" Use ’lcd local_needed_dir’ command to set remote machine’s working \n \
        directory, use ’put file_name’ to send files from remote machine to \n \
        this test machine and use ’get filename’ to transfer files from this \n \
        test machine to remote one. Use ’bye’ to log out, ’dir’ to get info \n \
        about files in test machine’s RAMDISK.\n");
        break;

    case ’6’:
        printf("\n******** Hard-disk Performance Test ***********\n");
        printf("Enter size of test file in kbytes (press Enter for 10M): ");
        gets(buf);
        if(buf[0]==0)
            {
            filesize=10*1024;
            }
        else
            {
            sscanf(buf,"%d",&filesize);
            }

        printf("Enter number of repetitions (0=forever, press Enter for 10): ");
        gets(buf);
        if(buf[0]==0)
            {
            repcnt=10;
            }
        else
            {
            sscanf(buf,"%d",&repcnt);
            }

        testfilecnt++;

        sprintf(nbuf,"tstRW%d",testfilecnt);
        taskSpawn(nbuf,200,0,10000,DiskRWTest,"ATA",filesize,repcnt,0,0,0,0,0,0,0,0);
        printf("Hard-disk performance test running...\n");
        break;


    case ’7’:
        printf("\n******** RAM-disk Performance Test ************\n");
       printf("Enter size of test file in kbytes (max %d, press Enter for 32M): ",rdsize);
        gets(buf);
        if(buf[0]==0)
            {
            filesize=32*1024;
            }
        else
            {
            sscanf(buf,"%d",&filesize);
            }


        printf("Enter number of repetitions (0=forever, press Enter for 10): ");
        gets(buf);
        if(buf[0]==0)
            {
            repcnt=10;
            }
        else
            {
            sscanf(buf,"%d",&repcnt);
            }

        testfilecnt++;

        sprintf(nbuf,"tstRW%d",testfilecnt);
        taskSpawn(nbuf,200,0,10000,DiskRWTest,"RAMDISK",filesize,repcnt,0,0,0,0,0,0,0,0);
        printf("RAM-disk performance test running...\n");
        break;

        case ’s’:
        case ’S’:
        printf("\n******** Stopping Running Processes ***********\n");
                blasteeStop=TRUE;
        blasterStop=TRUE;
```

```
                stopflag=TRUE;
                break;

                case ’q’:
                case ’Q’:
                printf("\n*********** Exiting to Shell *****************\n");

            exitflag=TRUE;
                stopflag=TRUE;

                blasteeStop=TRUE;
                blasterStop=TRUE;
                shellInit(10000,TRUE);
                break;

            case ’i’:
            case ’I’:
                printf("\n******** Info about running tasks *************\n");
                i(0);
                break;

            case ’d’:
            case ’D’:
                printf("\n********** Info about devices *****************\n");
                devs();
                break;

            default:
                        break;
            }
        } /* while(not exit) */

    wdCancel(clkWd);
    wdDelete(clkWd);

    return(0);
    }

/*******************************************************************/
void clkWdFn (int parm)
    {
    clkticks++;

    if(clkticks%6000==0)
        logMsg("Uptime %ld minutes.\n",clkticks/6000);

    if(exitflag==FALSE)
```

```
      {
      wdStart (clkWd, 1, clkWdFn, 0);
      }
   else
      {
      wdCancel(clkWd);
      wdDelete(clkWd);
      logMsg("Clock WDOG stopped.\n");
      }
   }


/******************************************************************/
int DiskRWTest(char * diskname,int fsize,int repcnt)
   {
   FILE * fp;
   char fname[80];
   int i,j,ret,kbytes;
   unsigned long startticks;
   float transfer;
   char *buf1;
   char *buf2;

   stopflag=FALSE;

   if(repcnt==0)
      repcnt=2000000000;

   sprintf(fname,"/%s/t%07d",diskname,clkticks);

   buf1=(char *)malloc(2048);
   if(buf1==NULL)
      {
      printf("Memory allocation for buf1 failed!\n");
      return(-1);
      }

   /* initialize 1K buf */
   for(j=0;j<1024;j=j+8)
         {
         sprintf(buf1+j,"%07ld",j);
         }

   buf2=(char *)malloc(2048);
   if(buf2==NULL)
      {
      printf("Memory allocation for buf2 failed!\n");
```

```
    free(buf1);
    return(-1);
    }


writeNum=0;

for(i=0;i<repcnt;i++)
    {
    rm(fname);

    startticks=clkticks;

    if(stopflag==TRUE)
        break;

    fp=fopen(fname,"wb");
    if(fp==NULL)
        {
        printf("\n%s open for write failed!\n",fname);
        free(buf1);
        free(buf2);
        return(-1);
        }

    for (kbytes=0;kbytes<fsize;kbytes++) /* kbytes loop */
        {
        if(stopflag==TRUE)
            break;

            ret=fwrite(buf1,BUF_SIZE,1,fp);
            if(ret<1)
                    {
                    printf("\nWrite to %s error!\n",fname);
            fclose(fp);
            free(buf1);
            free(buf2);
            return(-1);
                    }

        writeNum+=BUF_SIZE;

        } /* for fsize of kbytes */

    fclose(fp);

    if((clkticks-startticks)>0)
```

```
            transfer=((float)fsize*1024)*(float)sysClkRateGet()/(float)(clkticks-startticks);
        else
            transfer=0.0;
        printf("%s WRITE test %d done: %10.0f bytes/sec\n",fname,i+1,transfer);

        /********** now reading test *****************************/
        startticks=clkticks;

        if(stopflag==TRUE)
            break;

        fp=fopen(fname,"rb");
        if(fp==NULL)
            {
            printf("\n%s open for read failed!\n",fname);
            free(buf1);
            free(buf2);
            return(-1);
            }

        for (kbytes=0;kbytes<fsize;kbytes++) /* kbytes loop */
            {
            if(stopflag==TRUE)
                break;

                ret=fread(buf2,BUF_SIZE,1,fp);
                if(ret<1)
                        {
                        printf("\n%s reading error at %d kbytes!\n",fname,kbytes);
                fclose(fp);
                free(buf1);
                free(buf2);
                return(-1);
                        }

            } /* for fsize of kbytes */

        fclose(fp);

        if((clkticks-startticks)>0)
            transfer=((float)fsize*1024)*(float)sysClkRateGet()/(float)(clkticks-startticks);
        else
            transfer=0.0;
        printf("%s READ  test %d done: %10.0f bytes/sec\n",fname,i+1,transfer);

        }/* for repcnt */
```

```
free(buf1);
free(buf2);

rm(fname);

printf("DiskRWTest for %s ended.\n",fname);
return(0);
}
```