Experiment #1
Timing

Introduction

To understand and optimize the performance of a real-time system, it can be useful to time some of the VxWorks and application functions. VxWorks provides a number of timing facilities to help with this task.
The VxWorks execution timer can time any subroutine or group of subroutines. To time very fast subroutines, the timer can also repeatedly execute a group of functions until the time of a single iteration is known with reasonable certainty.

Objectives

The following are the primary objectives of this experiment:

- To demonstrate how to time a single subroutine using the VxWorks *timex()* routine.

Description

The *timex()* routine times a single execution of a specified function with up to eight integer arguments to be passed to the function. When execution us complete, *timex()* routine displays the execution time and a margin of error in miliseconds. If the execution was so fast relative to the clock rate that the time is meaningless(error > 50%), a warning message will appear. In such cases, use *timexN()* which will repeatedly execute the function until the time of a single iteration is known with reasonable certainty.

1. Syntax

void *timex(FUNCPTR function_name, int arg1, .., int arg8)*

Note: the first argument in *timex()* routine is a pointer to the function to be timed.

2. Example

This small example has two subroutines. The first subroutine "timing" makes a call to *timex()* with the function name "printit" which is the subroutine to be timed. The arguments are all NULL, so no parameters are being passed to "printit".

The second subroutine, "printit", which is being timed iterates 200 times while printing its task id(using *taskIdSelf()*) and the increment variable "i".

```
-------------------------------------------------------------------------------
#include "vxWorks.h"  /* Always include this as the first thing in every program */
#include "timexLib.h"
#include "stdio.h"

#define ITERATIONS 200

int printit(void);


void timing() /* Function to perform the timing */
{
FUNCPTR function_ptr = printit; /* a pointer to the function "printit" */
timex(function_ptr,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL); /* Timing the "print"
function */
}

int printit(void) /* Function being timed */
{
int i;
for(i=0; i < ITERATIONS; i++)  /* Printing the task id number and the increment variable
"i" */
          printf("Hello, I am task %d and is i = %d\n",taskIdSelf(),i);
return 0;
}
-------------------------------------------------------------------------------
```

Procedures

1. Copy the source code in the example and compile it.

2. Load the object file onto the target machine.

3. Run the example by executing "timing" on the WindSh.

Note: Make sure you have redirected I/O, otherwise you won't see the results of the printf commands.

Follow On Experiment

Experiment 1. Vary the number of ITERATIONS in the loop(300,400,500,600,700) and note the changes in execution speed.
Experiment 2. Decrease the number of the iteration to 5, note what happens(you should get a warning message). Write a program to get the timing in this case.

Additional Information

The timings measure the execution time of the routine body, without the usual subroutine entry and exit code. Also, the time required to set up the arguments and call the routines is not included in the reported times. This is because the timing routines automatically calibrate themselves by timing the invocation of a null routine, and thereafter subtracting that constant overhead.
Refer to the VxWorks Programmer's Manual and Reference Manual(timexLib).