**Average Initialization Time**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N/B | 1 | 2 | 4 | 8 | 10 |
| 20 | 0.000669 | 0.000745 | 0.000624 | 0.000626 | 0.000723 |
| 40 | 0.000718 | 0.000653 | 0.000686 | 0.000656 | 0.000668 |
| 80 | 0.000683 | 0.000626 | 0.00077 | 0.000703 | 0.000698 |
| 160 | 0.000619 | 0.000799 | 0.000639 | 0.000603 | 0.000792 |
| 320 | 0.000642 | 0.000624 | 0.000643 | 0.000655 | 0.000666 |
|  |  |  |  |  |  |

**Average Transmission Time**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N/B | 1 | 2 | 4 | 8 | 10 |
| 20 | 0.000417 | 0.000478 | 0.000461 | 0.000527 | 0.00059 |
| 40 | 0.000563 | 0.000564 | 0.000505 | 0.0005 | 0.000563 |
| 80 | 0.000743 | 0.000651 | 0.000649 | 0.00063 | 0.000629 |
| 160 | 0.001489 | 0.001414 | 0.00113 | 0.000964 | 0.001114 |
| 320 | 0.001913 | 0.001545 | 0.001511 | 0.00149 | 0.001537 |

**Standard Deviation for Transmission Time**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N/B | 1 | 2 | 4 | 8 | 10 |
| 20 | 0.000615 | 0.000771 | 0.000705 | 0.001026 | 0.00091 |
| 40 | 0.000759 | 0.00085 | 0.000671 | 0.000814 | 0.000897 |
| 80 | 0.000864 | 0.000772 | 0.000923 | 0.001192 | 0.000793 |
| 160 | 0.001886 | 0.003374 | 0.001596 | 0.001375 | 0.001521 |
| 320 | 0.001969 | 0.001444 | 0.00153 | 0.001645 | 0.001739 |

Histogram of average transmission time



Average Initialization and Transmission Time for Keil Board

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **B** | **Average Initialization Time** | **Average Transmission Time** |
| 20 | 1 | 0.000145 | 0.009253 |
| 20 | 2 | 0.000145 | 0.009253 |
| 20 | 4 | 0.000145 | 0.009253 |
| 20 | 8 | 0.000145 | 0.009253 |
| 20 | 10 | 0.000145 | 0.009253 |
| 40 | 1 | 0.000145 | 0.018930 |
| 40 | 2 | 0.000145 | 0.018930 |
| 40 | 4 | 0.000145 | 0.018930 |
| 40 | 8 | 0.000145 | 0.018930 |
| 40 | 10 | 0.000145 | 0.018930 |
| 80 | 1 | 0.000145 | 0.038284 |
| 80 | 2 | 0.000145 | 0.038284 |
| 80 | 4 | 0.000145 | 0.038284 |
| 80 | 8 | 0.000145 | 0.038284 |
| 80 | 10 | 0.000145 | 0.038284 |
| 160 | 1 | 0.000145 | 0.078019 |
| 160 | 2 | 0.000145 | 0.078019 |
| 160 | 4 | 0.000145 | 0.078019 |
| 160 | 8 | 0.000145 | 0.078019 |
| 160 | 10 | 0.000145 | 0.078019 |
| 320 | 1 | 0.000145 | 0.158198 |
| 320 | 2 | 0.000145 | 0.158198 |
| 320 | 4 | 0.000145 | 0.158198 |
| 320 | 8 | 0.000145 | 0.158198 |
| 320 | 10 | 0.000145 | 0.158198 |

Keil Discussion

* When conducing 5 trials for the (N,B) pair for the keil board testing, we discovered virtually no change in results for initialization and transmission time. Therefore, on the keil board atleast, the program does not need to be run multiple times for a given pair to compute average time. This is because, on average since there are only the tasks we created as part of the lab being executed at a given time, unlike linux, where the OS was managing other remote users and tasks as well, it can be safely assumed that the impact to average transmission time would be relatively minimal resulting in similar results for average initialization and transmission time.

Discussion of different N,B pairs on Linux and Keil.

* The general idea of having the B values is to specify the number of messages the mailbox/message queue can hold at a given time. On the Keil board with increasing the value of N and B at the same time, the value of Transmission Time increased. But this effect is significantly negligible on the x86 platform as processor runs at a much faster clock speed than compared to the ARM clock 100 Mhz.

**Appendix**

/\*----------------------------------------------------------------------------

\* RL-ARM - RTX

\*----------------------------------------------------------------------------

\* Name: MAILBOX.C

\* Purpose: RTX example program

\*----------------------------------------------------------------------------

\* This code is part of the RealView Run-Time Library.

\* Copyright (c) 2004-2011 KEIL - An ARM Company. All rights reserved.

\*---------------------------------------------------------------------------\*/

#include <LPC17xx.h>

#include "uart\_polling.h"

#include <RTL.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

OS\_TID tsk1; /\* assigned identification for task 1 \*/

OS\_TID tsk2; /\* assigned identification for task 2 \*/

OS\_MUT g\_mut\_uart;

OS\_TID g\_tid = 255;

const U32 N = 20;

const U32 B = 2;

/\*

U32 sendTaskCtr = 0;

U32 recvTaskCtr = 0;

\*/

U32 msg\_counter\_rec = 0;

U32 msg\_counter\_send = 0;

//int N\_arr[] = { 20, 40, 80, 160, 320 };

//int B\_arr[] = { 1, 2, 4, 8, 10 };

typedef struct { /\* Message object structure \*/

U32 number; /\* Random number \*/

} T\_NUM;

os\_mbx\_declare (MsgBox,320); /\* Declare an RTX mailbox \*/

\_declare\_box (mpool,sizeof(T\_NUM),320);/\* Dynamic memory pool \*/

\_\_task void send\_task (void);

\_\_task void rec\_task (void);

//\_\_task void updater\_task (void);

\_\_task void init(void);

U32 time\_a, time\_b, time\_c;

/\*----------------------------------------------------------------------------

\* Task 1: RTX Kernel starts this task with os\_sys\_init (send\_task)

\*---------------------------------------------------------------------------\*/

\_\_task void send\_task (void) {

T\_NUM \*random\_num\_tx;

//Get Time A

time\_a = os\_time\_get();

// fork the child process

tsk2 = os\_tsk\_create (rec\_task, 0); /\* start task 2 \*/

os\_mbx\_init (MsgBox, sizeof(MsgBox));/\* initialize the mailbox \*/

os\_mut\_wait(g\_mut\_uart, 0xFFFF);

printf("Mailbox Created\n");

os\_mut\_release(g\_mut\_uart);

//Get Time B

time\_b = os\_time\_get() ;

while (msg\_counter\_send < N)

{

random\_num\_tx = \_alloc\_box(mpool); /\* Allocate a memory for the message \*/

if( random\_num\_tx == NULL )

{

os\_mut\_wait(g\_mut\_uart, 0xFFFF);

printf("\_alloc\_box failed because of no mem available!\n");

os\_mut\_release(g\_mut\_uart);

exit(1);

}

random\_num\_tx->number = (U32) rand() % 9;

os\_mbx\_send (MsgBox, random\_num\_tx, 0xffff); /\* Send the message to the mailbox \*/

os\_mut\_wait(g\_mut\_uart, 0xFFFF);

printf("[send\_task (%d) ]: Sent %u\n", msg\_counter\_send, random\_num\_tx->number);

msg\_counter\_send++;

os\_mut\_release(g\_mut\_uart);

// os\_dly\_wait (100);

}

//msg\_counter\_send = 0;

os\_tsk\_delete\_self (); /\* We are done here, delete this task \*/

}

/\*----------------------------------------------------------------------------

\* Task 2: RTX Kernel starts this task with os\_tsk\_create (rec\_task, 0)

\*---------------------------------------------------------------------------\*/

\_\_task void rec\_task (void) {

T\_NUM \*random\_num\_rx;

while( msg\_counter\_rec < N )

{

os\_mbx\_wait (MsgBox, (void \*\*)&random\_num\_rx, 0xffff); /\* wait for the message \*/

os\_mut\_wait(g\_mut\_uart, 0xFFFF);

printf("[rec\_task (%d)]: Received %u\n", msg\_counter\_rec, random\_num\_rx->number);

msg\_counter\_rec += 1;

os\_mut\_release(g\_mut\_uart);

if( \_free\_box (mpool, random\_num\_rx) ) /\* free memory allocated for message \*/

{

os\_mut\_wait(g\_mut\_uart, 0xFFFF);

printf("\_free\_box failed because memory couldn't be freed\n");

os\_mut\_release(g\_mut\_uart);

exit(1);

}

}

//Get Time C

time\_c = os\_time\_get();

os\_mut\_wait(g\_mut\_uart, 0xFFFF);

printf("Time to initialize system: %0.6f\n", (float)(((float)time\_b - time\_a)/1000000) );

printf("Time to transmit data: %0.6f\n", (float)(((float)time\_c - time\_b)/1000000) );

os\_mut\_release(g\_mut\_uart);

// os\_dly\_wait(10);

os\_tsk\_delete\_self (); /\* We are done here, delete this task \*/

}

/\*

\_\_task void updater\_task (void) {

int i = 0;

int j = 0;

for(; i < 5; i++)

{

for(; j < 5; j++)

{

os\_mut\_wait(g\_mut\_uart, 0xFFFF);

N = N\_arr[i];

B = B\_arr[j];

printf("N = %d\t B = %d\n", N, B);

os\_mut\_release(g\_mut\_uart);

os\_dly\_wait(100000);

}

}

}

\*/

\_\_task void init(void)

{

os\_mut\_init(&g\_mut\_uart);

os\_tsk\_create(send\_task, 0);

// os\_tsk\_create(updater\_task, 10);

os\_tsk\_delete\_self();

}

/\*----------------------------------------------------------------------------

\* Main: Initialize and start RTX Kernel

\*---------------------------------------------------------------------------\*/

int main (void) { /\* program execution starts here \*/

SystemInit(); /\* initialize the LPC17xx MCU \*/

uart0\_init(); /\* initilize the first UART \*/

printf("Calling os\_sys\_init()...\n");

\_init\_box (mpool, sizeof(mpool), /\* initialize the 'mpool' memory for \*/

sizeof(T\_NUM)); /\* the membox dynamic allocation \*/

os\_sys\_init(init); /\* initilize the OS and start the first task \*/

}

/\*----------------------------------------------------------------------------

\* end of file

\*---------------------------------------------------------------------------\*/