ECE 6680 Embedded Computing
Lab 8: Real time scheduling using RMA
Cameron L. Burroughs

Introduction:

The purpose of this lab is to perform real-time Rate-Monatomic Analysis, or RMA, on an Inertial Navigation System (INS). The RMA performed on this INS must account for both overhead and direct/push-through blocking. Each task on this system had an overhead of 0.153μs and time constraints following Table 1. The INS runs on a Motorola MC68302 microcontroller and a Linux real-time kernel. The estimated resource usage for each task is given in Table 2.

Feature	**Period (ms)**
Compute attitude data	10.56
Compute velocity data	40.96
Compute position data	350.00
Display data	100.00
Run-time Built-In Test (BIT)	285.00
Compose attitude message	61.44
Compose navigation message	165.00
Compose test message	700.00

Table 1: Task Time Constraints

Task	**Run time (ms)**	**Result table usage (ms)**	**I/O channel usage (ms)**	**Disk usage (ms)**
attitude	1.30	0.20	-	2.00
velocity	4.70	0.20	-	3.00
position	3.00	0.20	-	3.00
display	23.00	0.30	-	-
runtime BIT	10.00	-	-	1.00
att message	9.00	-	3.00	-
nav message	38.30	-	6.00	-
test message	2.00	-	2.00	-

Table 2: Estimated Resource Usage

Implementation:

In order to schedule each task, a priority level was given based on each task's period. The takes with the shortest period, computation of attitude data, was given the highest priority, while the task with the longest period, composing a test message, was given the lowest priority. From

there, the maximum direct and push-through blocking of each task was calculated for each resource. Tables 3-5 display the calculated max blocking for each resource; Result Table, I/O, and Disk. From this point, the total blocking across all resources was calculated for each task by calculating the sum of the max blocking for each resource. Total Blocking for each task can be seen in Table 6.

$$Total\ Blocking = MaxBlocking_{ResultTable} + MaxBlocking_{I/O} + MaxBlocking_{Disk}$$

Task	Priority	Time using resource	Max blocking (direct)	Max blocking (push-through)	Max blocking
Compute attitude data	1	0.2	0.3	0	0.3
Compute velocity data	2	0.2	0.3	0.3	0.3
Compose attitude message	3	-	0	0.3	0.3
Display Data	4	0.3	0.2	0.2	0.2
Compose navigation message	5	-	0	0.2	0.2
Run-Time Built-In Test (BIT)	6	-	0	0.2	0.2
Compute position data	7	0.2	0	0	0
Compose Test Message	8	-	0	0	0

Table 3: Result Table Usage and Blocking

Task	Priority	Time using resource	Max blocking (direct)	Max blocking (push-through)	Max blocking
Compute attitude data	1	-	0	0	0
Compute velocity data	2	-	0	6	6
Compose attitude message	3	3.00	6.00	2	6
Display Data	4	-	0	2	2
Compose navigation message	5	-	0	2	2
Run-Time Built-In Test (BIT)	6	6.00	2.00	0	2
Compute position data	7	-	0	0	0
Compose Test Message	8	2.00	0	0	0

Table 4: I/O Usage and Blocking

Task	Priority	Time using resource	Max blocking (direct)	Max blocking (push-through)	Max blocking
Compute attitude data	1	2	3	0	3
Compute velocity data	2	3	3	3	3
Compose attitude message	3	-	0	3	3
Display Data	4	-	0	3	3
Compose navigation message	5	-	0	3	3
Run-Time Built-In Test (BIT)	6	1	3	3	3

Compute position data	7	3	0	0	0
Compose Test Message	8	-	0	0	0

Table 5: Disk Usage and Blocking

Task	Total blocking
Compute attitude data	3.3
Compute velocity data	9.3
Compose attitude message	9.3
Display Data	5.2
Compose navigation message	5.2
Run-Time Built-In Test (BIT)	5.2
Compute position data	0
Compose Test Message	0

Table 6: Total Blocking

Following the theorem shown in Figure 1, The total max blocking, overhead, and runtime for each task were used to determine if that task was schedulable or not. If each task on the system is schedulable, the entire system is schedulable according to the RMA Analysis. A program was created to determine the exact k and l values, following the theorem, that produce a schedulable task for the system. The result of this program can be seen in Figure 2. As you can see, each task was scheduled, and the INS system passed the RMA. This is a schedulable system.

A set of tasks is schedulable if:

$\forall$ i=1...n

let k=1...i

let l=1... $\left\lfloor \frac{T_i}{T_k} \right\rfloor$

total max blocking

for some (k,l) $\sum_{j=1}^{i-1} \left((R_j + O) \left\lceil \frac{lT_k}{T_j} \right\rceil \right) + R_i + B_i \leq lT_k$

Figure 1: Schedulable Task Theorem

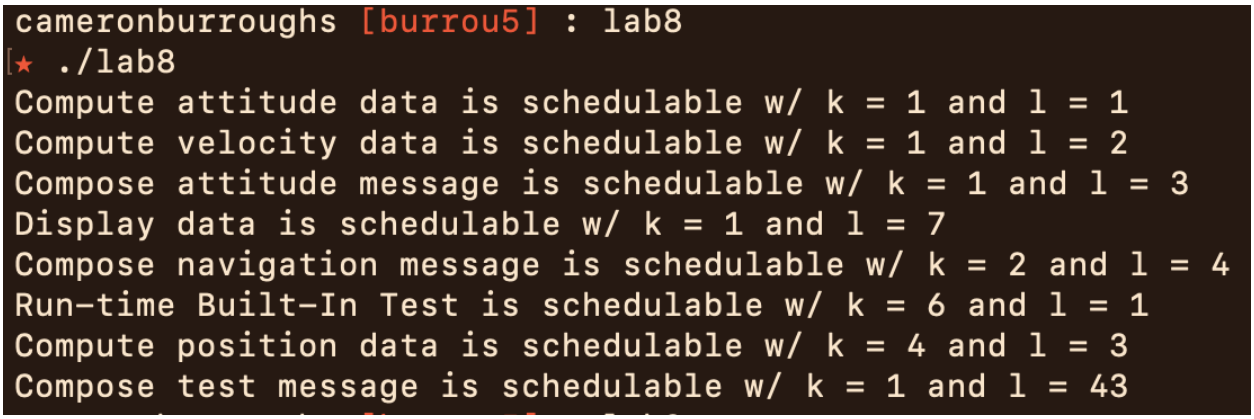

Figure 2: Result of Scheduled Tasks.

Code Implementation:

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>



int main(int argc, char** argv)
{
    // feature table
    char feature[8][100] = {
        "Compute attitude data",
        "Compute velocity data",
        "Compose attitude message",
        "Display data",
        "Compose navigation message",
        "Run-time Built-In Test",
        "Compute position data",
        "Compose test message"
    };


    float period[8] = {10.56, 40.96, 61.44, 100, 165, 285, 350, 700};
    float runtime[8] = {1.30, 4.70, 9.00, 23.00, 38.30, 10.00, 3.00,2.00};
    float blocking[8] = {3.3, 9.3, 9.3, 5.2, 5.2, 5.2, 0.0, 0.0};

    int i,k,l,j;
    float sum;

    for(i = 0; i < 8; i++) {
        for(k = 0; k < i+1; k++) {
            for(l = 0; l < floor(period[i]/period[k]); l++) {
                sum = 0;
                for(j = 0; j < i; j++) {
                    sum += (runtime[j] + 0.153) * ceil(((l+1)*period[k])/period[j]);
                }

                sum += runtime[i] + 0.153 + blocking[i];

                if(sum < period[k]*(l+1)) {
                    printf("%s is schedulable w/ k = %d and l = %d\n", feature[i], k+1, l+1);
                    break;
                }

            }
            if(l < floor(period[i]/period[k])) {
                break;
            }
        }
        if(k == i+1){
            printf("%s not scheduable\n", feature[i]);
        }
    }
    return 0;


}

```