

# RTX Software Design Report

Clement Hoang

20531116

c8hoang@uwaterloo.ca

David Su

20516776

dysu@uwaterloo.ca

Cole Vander Veen

20503626

cgvander@waterloo.ca

Peter Li

20531116

y648li@uwaterloo.ca

Winter 2016

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Design Description</b>	<b>6</b>
2.1	Data Structures . . . . .	6
2.2	Global Variables . . . . .	6
2.3	Memory Management . . . . .	6
2.3.1	Memory Structure . . . . .	6
2.3.2	Requesting Memory Blocks . . . . .	8
2.3.3	Releasing Memory Blocks . . . . .	8
2.4	Processor Management . . . . .	9
2.4.1	Process Control Structures . . . . .	9
2.4.2	Process Queues . . . . .	9
2.4.3	Process Scheduling . . . . .	9
2.5	Process Priority Management . . . . .	9
2.5.1	Get Process Priority . . . . .	9
2.5.2	Set Process Priority . . . . .	9
2.6	Interprocess Communication . . . . .	9
2.6.1	Message Structure . . . . .	9
2.6.2	Sending Messages . . . . .	9
2.6.3	Receiving Messages . . . . .	9
2.6.4	Delayed Send . . . . .	9
2.7	Interrupts and I-Processes . . . . .	9
2.7.1	UART I-Process . . . . .	9
2.7.2	Timer I-Process . . . . .	10
2.8	System Processes . . . . .	10
2.8.1	Null Process . . . . .	10
2.8.2	CRT Process . . . . .	10
2.9	User Processes . . . . .	10
2.9.1	Wall Clock Process . . . . .	10
2.9.2	Set Priority Process . . . . .	10
2.9.3	Stress Test Processes . . . . .	10
2.10	Initialization . . . . .	10
2.11	Testing . . . . .	10
<b>3</b>	<b>Lessons Learned</b>	<b>11</b>
3.1	Source Control and Code Management . . . . .	11

<b>4</b>	<b>Team Dynamics and Individual Responsibilities</b>	<b>12</b>
4.1	adsfadsf . . . . .	12
<b>5</b>	<b>Timing and Analysis</b>	<b>13</b>

# List of Algorithms

1	k_request_memory_block . . . . .	8
2	The memory release function . . . . .	8

# List of Figures

2.1	Memory Layout . . . . .	7
-----	-------------------------	---

# Chapter 1

## Introduction

The purpose

# Chapter 2

## Design Description

### 2.1 Data Structures

- `MemBlock`: a node that represents a block of memory, with its size defined by the global, `BLOCK_SIZE`
- `MemQueue`: a data structure which models the physical memory available to the OS, using a linked list of `MemBlock` nodes.
- `PCB`
- `PCBQ`
- `envelope`
- `gp_pcbs`

### 2.2 Global Variables

- `memQueue`:
- `gp_pcbs`:
- `gp_stack`
- `p_end`
- `numOfBlocks`

### 2.3 Memory Management

#### 2.3.1 Memory Structure

dsfdasfdsafdsafdsafdsaf

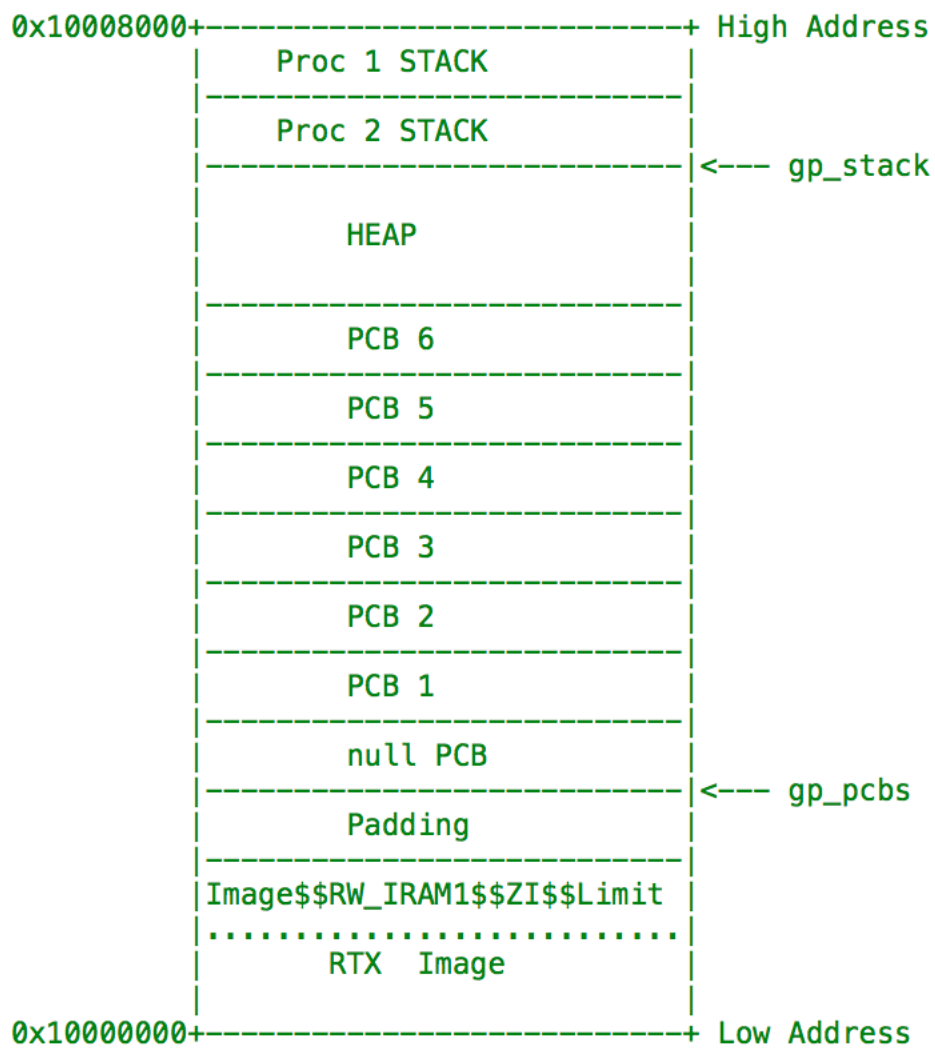


Figure 2.1: Memory Layout



### 2.3.2 Requesting Memory Blocks

```
int k_request_memory_block(void);
```

describe input, output, effects

---

**Algorithm 1** k.request\_memory\_block

---

```
1: procedure REQUEST_MEMORY_BLOCK
2:   while heap is full do
3:     block the current process
4:   end while
5:   update the free space list
6:   return the address of the top of the block
7: end procedure
```

---

### 2.3.3 Releasing Memory Blocks

```
int k_release_memory_block(void* memory_block);
```

describe input, output, effects

---

**Algorithm 2** The memory release function

---

```
1: procedure RELEASE_MEMORY_BLOCK(*memory_block)
2:   if this block is the top block of the heap then
3:     modify heap header node (never gets overwritten)
4:   end if
5:   if there is free space immediately beneath this block then
6:     combine them by increasing this block's length
7:   else this block becomes a new block node, is added to the list
8:   end if
9:   if there is free space immediately beneath this block then
10:    combine them by increasing this block's length
11:   end if
12:   if a process is blocked on memory then
13:     unblock that process, release the processor
14:   end if
15: end procedure
```

---

## **2.4 Processor Management**

### **2.4.1 Process Control Structures**

DFASFAFD

### **2.4.2 Process Queues**

fsadfasdfadsf

### **2.4.3 Process Scheduling**

sdfasdfasfdasf

## **2.5 Process Priority Management**

### **2.5.1 Get Process Priority**

asdfadsfasf

### **2.5.2 Set Process Priority**

dsfasdfasfsfdf

## **2.6 Interprocess Communication**

### **2.6.1 Message Structure**

dsfadsfadsfdasfdafs

### **2.6.2 Sending Messages**

adsfdsafasdfasf

### **2.6.3 Receiving Messages**

dsfafasfdasf

### **2.6.4 Delayed Send**

sdfasfasfd

## **2.7 Interrupts and I-Processes**

### **2.7.1 UART I-Process**

dsfadsfadsfadsf

### **2.7.2 Timer I-Process**

sdfasfdafd

## **2.8 System Processes**

### **2.8.1 Null Process**

sdfdasfafadsf

### **2.8.2 CRT Process**

sdfdsfafaf

## **2.9 User Processes**

### **2.9.1 Wall Clock Process**

sdfasdfafadf

### **2.9.2 Set Priority Process**

dsfasdfasdfadsf

### **2.9.3 Stress Test Processes**

dfdasfasdfads

## **2.10 Initialization**

dasfasfasfd

## **2.11 Testing**

dfadsfasdf

# Chapter 3

## Lessons Learned

### 3.1 Source Control and Code Management

sdfdsafsadf

# Chapter 4

## Team Dynamics and Individual Responsibilities

### 4.1 adsfadsf

dfasfasdf

# Chapter 5

## Timing and Analysis