Runtime Terror OS

R3/R4

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Alarm Struct Reference

Public Attributes

- int **hour**
- int **minute**
- int **second**
- char **message** [85]
- struct Alarm * **next**
- struct Alarm * **prev**

3.1.1 Detailed Description

Definition at line 15 of file userFunctions.h.

The documentation for this struct was generated from the following file:

- modules/R1/userFunctions.h

3.2 context Struct Reference

Public Attributes

- u32int **gs**
- u32int **fs**
- u32int **es**
- u32int **ds**
- u32int **edi**
- u32int **esi**
- u32int **ebp**
- u32int **esp**
- u32int **ebx**
- u32int **edx**
- u32int **ecx**
- u32int **eax**
- u32int **eip**
- u32int **cs**
- u32int **eflags**

3.2.1 Detailed Description

Definition at line 34 of file PCB.h.

The documentation for this struct was generated from the following file:

- modules/R2/PCB.h

3.3 date_time Struct Reference

Public Attributes

- int **sec**
- int **min**
- int **hour**
- int **day_w**
- int **day_m**
- int **day_y**
- int **mon**
- int **year**

3.3.1 Detailed Description

Definition at line 32 of file system.h.

The documentation for this struct was generated from the following file:

- include/system.h

3.4 footer Struct Reference

Public Attributes

- header **head**

3.4.1 Detailed Description

Definition at line 18 of file heap.h.

The documentation for this struct was generated from the following file:

- include/mem/heap.h

3.5 gdt_descriptor_struct Struct Reference

Public Attributes

- u16int **limit**
- u32int **base**

3.5.1 Detailed Description

Definition at line 25 of file tables.h.

The documentation for this struct was generated from the following file:

- include/core/tables.h

3.6 gdt_entry_struct Struct Reference

Public Attributes

- u16int **limit_low**
- u16int **base_low**
- u8int **base_mid**
- u8int **access**
- u8int **flags**
- u8int **base_high**

3.6.1 Detailed Description

Definition at line 32 of file tables.h.

The documentation for this struct was generated from the following file:

- include/core/tables.h

3.7 header Struct Reference

Public Attributes

- int **size**
- int **index_id**

3.7.1 Detailed Description

Definition at line 13 of file heap.h.

The documentation for this struct was generated from the following file:

- include/mem/heap.h

3.8 heap Struct Reference

Public Attributes

- index_table **index**
- u32int **base**
- u32int **max_size**
- u32int **min_size**

3.8.1 Detailed Description

Definition at line 35 of file heap.h.

The documentation for this struct was generated from the following file:

- include/mem/heap.h

3.9 idt_entry_struct Struct Reference

Public Attributes

- u16int **base_low**
- u16int **sselect**
- u8int **zero**
- u8int **flags**
- u16int **base_high**

3.9.1 Detailed Description

Definition at line 8 of file tables.h.

The documentation for this struct was generated from the following file:

- include/core/tables.h

3.10 idt_struct Struct Reference

Public Attributes

- u16int **limit**
- u32int **base**

3.10.1 Detailed Description

Definition at line 18 of file tables.h.

The documentation for this struct was generated from the following file:

- include/core/tables.h

3.11 index_entry Struct Reference

Public Attributes

- int **size**
- int **empty**
- u32int **block**

3.11.1 Detailed Description

Definition at line 22 of file heap.h.

The documentation for this struct was generated from the following file:

- include/mem/heap.h

3.12 index_table Struct Reference

Public Attributes

- index_entry **table** [TABLE_SIZE]
- int **id**

3.12.1 Detailed Description

Definition at line 29 of file heap.h.

The documentation for this struct was generated from the following file:

- include/mem/heap.h

3.13 List Struct Reference

Public Attributes

- Alarm * **head**
- Alarm * **tail**

3.13.1 Detailed Description

Definition at line 24 of file userFunctions.h.

The documentation for this struct was generated from the following file:

- modules/R1/userFunctions.h

3.14 page_dir Struct Reference

Public Attributes

- page_table * **tables** [1024]
- u32int **tables_phys** [1024]

3.14.1 Detailed Description

Definition at line 36 of file paging.h.

The documentation for this struct was generated from the following file:

- include/mem/paging.h

3.15 page_entry Struct Reference

Public Attributes

- u32int **present**: 1
- u32int **writeable**: 1
- u32int **usermode**: 1
- u32int **accessed**: 1
- u32int **dirty**: 1
- u32int **reserved**: 7
- u32int **frameaddr**: 20

3.15.1 Detailed Description

Definition at line 14 of file paging.h.

The documentation for this struct was generated from the following file:

- include/mem/paging.h

3.16 page_table Struct Reference

Public Attributes

- page_entry **pages** [1024]

3.16.1 Detailed Description

Definition at line 28 of file paging.h.

The documentation for this struct was generated from the following file:

- include/mem/paging.h

3.17 param Struct Reference

Public Attributes

- int **op_code**
- int **device_id**
- char * **buffer_ptr**
- int * **count_ptr**

3.17.1 Detailed Description

Definition at line 34 of file mpx_supt.h.

The documentation for this struct was generated from the following file:

- modules/mpx_supt.h

3.18 PCB Struct Reference

Public Attributes

- unsigned char **stack** [MEM1K]
- unsigned char * **stackTop**
- struct PCB * **prev**
- struct PCB * **next**
- char **Process_Name** [10]
- int **Process_Class**
- int **Priority**
- int **ReadyState**
- int **SuspendedState**

3.18.1 Detailed Description

Definition at line 15 of file PCB.h.

The documentation for this struct was generated from the following file:

- modules/R2/PCB.h

3.19 Queue Struct Reference

Public Attributes

- int **count**
- PCB * **head**
- PCB * **tail**

3.19.1 Detailed Description

Definition at line 27 of file PCB.h.

The documentation for this struct was generated from the following file:

- modules/R2/PCB.h

Chapter 4

File Documentation

4.1 include/core/asm.h File Reference

```
#include <system.h>
#include <tables.h>
```

4.2 include/core/interrupts.h File Reference

Functions

- void **init_irq** (void)
- void **init_pic** (void)

4.3 include/core/io.h File Reference

Macros

- #define **outb**(port, data) asm volatile ("outb %%al,%%dx" : : "a" (data), "d" (port))
- #define **inb**(port)

4.3.1 Macro Definition Documentation

4.3.1.1 inb

```
#define inb(
              port )
```

Value:

```
({                          \
    unsigned char r;                    \
    asm volatile ("inb %%dx,%%al": "=a" (r): "d" (port)); \
    r;                              \
})
```

Definition at line 17 of file io.h.

4.4 include/core/serial.h File Reference

Macros

- #define **COM1** 0x3f8
- #define **COM2** 0x2f8
- #define **COM3** 0x3e8
- #define **COM4** 0x2e8

Functions

- int **init_serial** (int device)
- int **serial_println** (const char ∗msg)
- int **serial_print** (const char ∗msg)
- int **set_serial_out** (int device)
- int **set_serial_in** (int device)
- int ∗ **polling** (char ∗buffer, int ∗count)

4.5 include/core/tables.h File Reference

```
#include "system.h"
```

Classes

- struct idt_entry_struct
- struct idt_struct
- struct gdt_descriptor_struct
- struct gdt_entry_struct

Functions

- struct idt_entry_struct **__attribute__** ((packed)) idt_entry
- void **idt_set_gate** (u8int idx, u32int base, u16int sel, u8int flags)
- void **gdt_init_entry** (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void **init_idt** ()
- void **init_gdt** ()

Variables

- u16int **base_low**
- u16int **sselect**
- u8int **zero**
- u8int **flags**
- u16int **base_high**
- u16int **limit**
- u32int **base**
- u16int **limit_low**
- u8int **base_mid**
- u8int **access**

4.6 include/mem/heap.h File Reference

Classes

- struct header
- struct footer
- struct index_entry
- struct index_table
- struct heap

Macros

- #define **TABLE_SIZE** 0x1000
- #define **KHEAP_BASE** 0xD000000
- #define **KHEAP_MIN** 0x10000
- #define **KHEAP_SIZE** 0x1000000

Functions

- u32int **_kmalloc** (u32int size, int align, u32int *phys_addr)
- u32int **kmalloc** (u32int size)
- u32int **kfree** ()
- void **init_kheap** ()
- u32int **alloc** (u32int size, heap *hp, int align)
- heap * **make_heap** (u32int base, u32int max, u32int min)

4.7 include/mem/paging.h File Reference

```
#include <system.h>
```

Classes

- struct page_entry
- struct page_table
- struct page_dir

Macros

- #define **PAGE_SIZE** 0x1000

Functions

- void **set_bit** (u32int addr)
- void **clear_bit** (u32int addr)
- u32int **get_bit** (u32int addr)
- u32int **first_free** ()
- void **init_paging** ()
- void **load_page_dir** (page_dir *new_page_dir)
- page_entry * **get_page** (u32int addr, page_dir *dir, int make_table)
- void **new_frame** (page_entry *page)

4.8 include/string.h File Reference

```
#include <system.h>
```

Functions

- int isspace (const char *c)
- void * memset (void *s, int c, size_t n)
- char * strcpy (char *s1, const char *s2)
- char * strcat (char *s1, const char *s2)
- int strlen (const char *s)
- int strcmp (const char *s1, const char *s2)
- char * strtok (char *s1, const char *s2)
- int atoi (const char *s)

4.8.1 Function Documentation

4.8.1.1 atoi()

```
int atoi (
            const char * s )
```

Description: Convert an ASCII string to an integer

Parameters

s	String

Definition at line 50 of file string.c.

```
51 {
52   int res=0;
53    int charVal=0;
54    char sign = ' ';
55    char c = *s;
56 
57 
58    while(isspace(&c)){ ++s; c = *s;} // advance past whitespace
59 
60 
61    if (*s == '-' || *s == '+') sign = *(s++); // save the sign
62 
63 
64    while(*s != '\0'){
65         charVal = *s - 48;
66     res = res * 10 + charVal;
67     s++;
68 
69    }
70 
71 
72    if ( sign == '-') res=res * -1;
73 
74   return res; // return integer
75 }
```

4.8.1.2 isspace()

```
int isspace (
            const char * c )
```

Description: Determine if a character is whitespace.

Parameters

c	character to check

Definition at line 121 of file string.c.

```
122 {
123   if (*c == ' '  ||
124       *c == '\n' ||
125       *c == '\r' ||
126       *c == '\f' ||
127       *c == '\t' ||
128       *c == '\v'){
129     return 1;
130   }
131   return 0;
132 }
```

4.8.1.3 memset()

```
void* memset (
            void * s,
            int c,
            size_t n )
```

Description: Set a region of memory.

Parameters

s	destination
c	byte to write
n	count

Definition at line 139 of file string.c.

```
140 {
141   unsigned char *p = (unsigned char *) s;
142   while(n--){
143     *p++ = (unsigned char) c;
144   }
145   return s;
146 }
```

4.8.1.4 strcat()

```
char* strcat (
            char * s1,
            const char * s2 )
```

Description: Concatenate the contents of one string onto another.

Parameters

s1	destination
s2	source

Definition at line 108 of file string.c.

```
109 {
110   char *rc = s1;
111   if (*s1) while(*++s1);
112   while( (*s1++ = *s2++) );
113   return rc;
114 }
```

4.8.1.5 strcmp()

```
int strcmp (
              const char * s1,
              const char * s2 )
```

Description: String comparison

Parameters

s1	string 1
s2	string 2

Definition at line 81 of file string.c.

```
82 {
83
84   // Remarks:
85   // 1) If we made it to the end of both strings (i. e. our pointer points to a
86   //    '\0' character), the function will return 0
87   // 2) If we didn't make it to the end of both strings, the function will
88   //    return the difference of the characters at the first index of
89   //    indifference.
90   while ( (*s1) && (*s1==*s2) ){
91     ++s1;
92     ++s2;
93   }
94   return ( *(unsigned char *)s1 - *(unsigned char *)s2 );
95 }
```

4.8.1.6 strcpy()

```
char* strcpy (
              char * s1,
              const char * s2 )
```

Description: Copy one string to another.

Parameters

s1	destination
s2	source

Definition at line 38 of file string.c.

```
39 {
40   char *rc = s1;
41   while( (*s1++ = *s2++) );
42   return rc; // return pointer to destination string
43 }
```

4.8.1.7 strlen()

```
int strlen (
            const char * s )
```

Description: Returns the length of a string.

Parameters

s	input string

Definition at line 26 of file string.c.

```
27 {
28   int r1 = 0;
29   if (*s) while(*s++) r1++;
30   return r1;//return length of string
31 }
```

4.8.1.8 strtok()

```
char* strtok (
             char * s1,
             const char * s2 )
```

Description: Split string into tokens

Parameters

s1	String
s2	delimiter

Definition at line 153 of file string.c.

```
154 {
155   static char *tok_tmp = NULL;
156   const char *p = s2;
157
158   //new string
159   if (s1!=NULL){
160     tok_tmp = s1;
161   }
162   //old string cont'd
163   else {
164     if (tok_tmp==NULL){
165       return NULL;
166     }
167     s1 = tok_tmp;
168   }
169
170   //skip leading s2 characters
171   while ( *p && *s1 ){
```

```
172     if (*s1==*p){
173       ++s1;
174       p = s2;
175       continue;
176     }
177     ++p;
178   }
179
180   //no more to parse
181   if (!*s1){
182     return (tok_tmp = NULL);
183   }
184
185   //skip non-s2 characters
186   tok_tmp = s1;
187   while (*tok_tmp){
188     p = s2;
189     while (*p){
190       if (*tok_tmp==*p++){
191     *tok_tmp++ = '\0';
192     return s1;
193       }
194     }
195     ++tok_tmp;
196   }
197
198   //end of string
199   tok_tmp = NULL;
200   return s1;
201 }
```

4.9 include/system.h File Reference

Classes

- struct date_time

Macros

- #define **NULL** 0
- #define **no_warn**(p) if (p) while (1) break
- #define **asm** __asm__
- #define **volatile** __volatile__
- #define **sti**() asm volatile ("sti"::)
- #define **cli**() asm volatile ("cli"::)
- #define **nop**() asm volatile ("nop"::)
- #define **hlt**() asm volatile ("hlt"::)
- #define **iret**() asm volatile ("iret"::)
- #define **GDT_CS_ID** 0x01
- #define **GDT_DS_ID** 0x02

Typedefs

- typedef unsigned int **size_t**
- typedef unsigned char **u8int**
- typedef unsigned short **u16int**
- typedef unsigned long **u32int**

Functions

- void **klogv** (const char ∗msg)
- void **kpanic** (const char ∗msg)

4.10 kernel/core/interrupts.c File Reference

```
#include <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
```

Macros

- #define **PIC1** 0x20
- #define **PIC2** 0xA0
- #define **ICW1** 0x11
- #define **ICW4** 0x01
- #define **io_wait**() asm volatile ("outb $0x80")

Functions

- void **divide_error** ()
- void **debug** ()
- void **nmi** ()
- void **breakpoint** ()
- void **overflow** ()
- void **bounds** ()
- void **invalid_op** ()
- void **device_not_available** ()
- void **double_fault** ()
- void **coprocessor_segment** ()
- void **invalid_tss** ()
- void **segment_not_present** ()
- void **stack_segment** ()
- void **general_protection** ()
- void **page_fault** ()
- void **reserved** ()
- void **coprocessor** ()
- void **rtc_isr** ()
- void **sys_call_isr** ()
- void **isr0** ()
- void **do_isr** ()
- void **init_irq** (void)
- void **init_pic** (void)
- void **do_divide_error** ()
- void **do_debug** ()
- void **do_nmi** ()
- void **do_breakpoint** ()
- void **do_overflow** ()
- void **do_bounds** ()
- void **do_invalid_op** ()
- void **do_device_not_available** ()
- void **do_double_fault** ()
- void **do_coprocessor_segment** ()

- void **do_invalid_tss** ()
- void **do_segment_not_present** ()
- void **do_stack_segment** ()
- void **do_general_protection** ()
- void **do_page_fault** ()
- void **do_reserved** ()
- void **do_coprocessor** ()

Variables

- idt_entry **idt_entries** [256]

4.11 kernel/core/kmain.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
#include <mem/heap.h>
#include <mem/paging.h>
#include <modules/mpx_supt.h>
#include "modules/R1/comHand.h"
#include "modules/sys_proc_loader.h"
#include "modules/R1/userFunctions.h"
```

Functions

- void **kmain** (void)

4.12 kernel/core/serial.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <core/io.h>
#include <core/serial.h>
```

Macros

- #define **NO_ERROR** 0

Functions

- int **init_serial** (int device)
- int **serial_println** (const char ∗msg)
- int **serial_print** (const char ∗msg)
- int **set_serial_out** (int device)
- int **set_serial_in** (int device)
- int ∗ **polling** (char ∗cmdBuffer, int ∗count)

Variables

- int **serial_port_out** = 0
- int **serial_port_in** = 0

4.13 kernel/core/system.c File Reference

```
#include <string.h>
#include <system.h>
#include <core/serial.h>
```

Functions

- void **klogv** (const char ∗msg)
- void **kpanic** (const char ∗msg)

4.14 kernel/core/tables.c File Reference

```
#include <string.h>
#include <core/tables.h>
```

Functions

- void **write_gdt_ptr** (u32int, size_t)
- void **write_idt_ptr** (u32int)
- void **idt_set_gate** (u8int idx, u32int base, u16int sel, u8int flags)
- void **init_idt** ()
- void **gdt_init_entry** (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void **init_gdt** ()

Variables

- gdt_descriptor **gdt_ptr**
- gdt_entry **gdt_entries** [5]
- idt_descriptor **idt_ptr**
- idt_entry **idt_entries** [256]

4.15 kernel/mem/heap.c File Reference

```
#include <system.h>
#include <string.h>
#include <core/serial.h>
#include <mem/heap.h>
#include <mem/paging.h>
```

Functions

- u32int **_kmalloc** (u32int size, int page_align, u32int *phys_addr)
- u32int **kmalloc** (u32int size)
- u32int **alloc** (u32int size, heap *h, int align)
- heap * **make_heap** (u32int base, u32int max, u32int min)

Variables

- heap * **kheap** = 0
- heap * **curr_heap** = 0
- page_dir * **kdir**
- void * **end**
- void **_end**
- void **__end**
- u32int **phys_alloc_addr** = (u32int)&end

4.16 kernel/mem/paging.c File Reference

```
#include <system.h>
#include <string.h>
#include "mem/heap.h"
#include "mem/paging.h"
```

Functions

- void **set_bit** (u32int addr)
- void **clear_bit** (u32int addr)
- u32int **get_bit** (u32int addr)
- u32int **find_free** ()
- page_entry * **get_page** (u32int addr, page_dir *dir, int make_table)
- void **init_paging** ()
- void **load_page_dir** (page_dir *new_dir)
- void **new_frame** (page_entry *page)

Variables

- u32int **mem_size** = 0x4000000
- u32int **page_size** = 0x1000
- u32int **nframes**
- u32int * **frames**
- page_dir * **kdir** = 0
- page_dir * **cdir** = 0
- u32int **phys_alloc_addr**
- heap * **kheap**

4.17 lib/string.c File Reference

```
#include <system.h>
#include <string.h>
```

Functions

- int strlen (const char *s)
- char * strcpy (char *s1, const char *s2)
- int atoi (const char *s)
- int strcmp (const char *s1, const char *s2)
- char * strcat (char *s1, const char *s2)
- int isspace (const char *c)
- void * memset (void *s, int c, size_t n)
- char * strtok (char *s1, const char *s2)

4.17.1 Function Documentation

4.17.1.1 atoi()

```
int atoi (
            const char * s )
```

Description: Convert an ASCII string to an integer

Parameters

s	String

Definition at line 50 of file string.c.

```
51 {
52   int res=0;
53    int charVal=0;
54    char sign = ' ';
55    char c = *s;
```

```
56
57
58    while(isspace(&c)){ ++s; c = *s;} // advance past whitespace
59
60
61    if (*s == '-' || *s == '+') sign = *(s++); // save the sign
62
63
64    while(*s != '\0'){
65         charVal = *s - 48;
66     res = res * 10 + charVal;
67     s++;
68
69    }
70
71
72    if ( sign == '-') res=res * -1;
73
74   return res; // return integer
75 }
```

4.17.1.2 isspace()

```
int isspace (
             const char * c )
```

Description: Determine if a character is whitespace.

Parameters

c	character to check

Definition at line 121 of file string.c.

```
122 {
123   if (*c == ' '  ||
124       *c == '\n' ||
125       *c == '\r' ||
126       *c == '\f' ||
127       *c == '\t' ||
128       *c == '\v'){
129     return 1;
130   }
131   return 0;
132 }
```

4.17.1.3 memset()

```
void* memset (
             void * s,
             int c,
             size_t n )
```

Description: Set a region of memory.

Parameters

s	destination
c	byte to write
n	count

Definition at line 139 of file string.c.

```
140 {
141   unsigned char *p = (unsigned char *) s;
142   while(n--){
143     *p++ = (unsigned char) c;
144   }
145   return s;
146 }
```

4.17.1.4 strcat()

```
char* strcat (
              char * s1,
              const char * s2 )
```

Description: Concatenate the contents of one string onto another.

Parameters

s1	destination
s2	source

Definition at line 108 of file string.c.

```
109 {
110   char *rc = s1;
111   if (*s1) while(*++s1);
112   while( (*s1++ = *s2++) );
113   return rc;
114 }
```

4.17.1.5 strcmp()

```
int strcmp (
              const char * s1,
              const char * s2 )
```

Description: String comparison

Parameters

s1	string 1
s2	string 2

Definition at line 81 of file string.c.

```
82 {
83
84   // Remarks:
85   // 1) If we made it to the end of both strings (i. e. our pointer points to a
86   //    '\0' character), the function will return 0
87   // 2) If we didn't make it to the end of both strings, the function will
88   //    return the difference of the characters at the first index of
89   //    indifference.
90   while ( (*s1) && (*s1==*s2) ){
91     ++s1;
92     ++s2;
```

```
93   }
94   return ( *(unsigned char *)s1 - *(unsigned char *)s2 );
95 }
```

4.17.1.6 strcpy()

```
char* strcpy (
             char * s1,
             const char * s2 )
```

Description: Copy one string to another.

Parameters

s1	destination
s2	source

Definition at line 38 of file string.c.

```
39 {
40   char *rc = s1;
41   while( (*s1++ = *s2++) );
42   return rc; // return pointer to destination string
43 }
```

4.17.1.7 strlen()

```
int strlen (
             const char * s )
```

Description: Returns the length of a string.

Parameters

s	input string

Definition at line 26 of file string.c.

```
27 {
28   int r1 = 0;
29   if (*s) while(*s++) r1++;
30   return r1;//return length of string
31 }
```

4.17.1.8 strtok()

```
char* strtok (
             char * s1,
             const char * s2 )
```

Description: Split string into tokens

Parameters

s1	String
s2	delimiter

Definition at line 153 of file string.c.

```
154 {
155   static char *tok_tmp = NULL;
156   const char *p = s2;
157
158   //new string
159   if (s1!=NULL){
160     tok_tmp = s1;
161   }
162   //old string cont'd
163   else {
164     if (tok_tmp==NULL){
165       return NULL;
166     }
167     s1 = tok_tmp;
168   }
169
170   //skip leading s2 characters
171   while ( *p && *s1 ){
172     if (*s1==*p){
173       ++s1;
174       p = s2;
175       continue;
176     }
177     ++p;
178   }
179
180   //no more to parse
181   if (!*s1){
182     return (tok_tmp = NULL);
183   }
184
185   //skip non-s2 characters
186   tok_tmp = s1;
187   while (*tok_tmp){
188     p = s2;
189     while (*p){
190       if (*tok_tmp==*p++){
191     *tok_tmp++ = '\0';
192     return s1;
193       }
194     }
195     ++tok_tmp;
196   }
197
198   //end of string
199   tok_tmp = NULL;
200   return s1;
201 }
```

4.18 modules/mpx_supt.c File Reference

```
#include "mpx_supt.h"
#include <mem/heap.h>
#include <string.h>
#include <core/serial.h>
```

Functions

- int **sys_req** (int op_code, int device_id, char *buffer_ptr, int *count_ptr)
- void **mpx_init** (int cur_mod)
- void **sys_set_malloc** (u32int(*func)(u32int))
- void **sys_set_free** (int(*func)(void *))

- void * **sys_alloc_mem** (u32int size)
- int **sys_free_mem** (void *ptr)
- void **idle** ()
- u32int * **sys_call** (context *registers)

Variables

- param **params**
- int **current_module** = -1
- u32int(* **student_malloc**)(u32int)
- int(* **student_free**)(void *)
- PCB * **cop**
- context * **initial**

4.19 modules/mpx_supt.h File Reference

```
#include <system.h>
#include "R2/PCB.h"
```

Classes

- struct param

Macros

- #define **EXIT** 0
- #define **IDLE** 1
- #define **READ** 2
- #define **WRITE** 3
- #define **INVALID_OPERATION** 4
- #define **TRUE** 1
- #define **FALSE** 0
- #define **MODULE_R1** 0
- #define **MODULE_R2** 1
- #define **MODULE_R3** 2
- #define **MODULE_R4** 4
- #define **MODULE_R5** 8
- #define **MODULE_F** 9
- #define **IO_MODULE** 10
- #define **MEM_MODULE** 11
- #define **INVALID_BUFFER** 1000
- #define **INVALID_COUNT** 2000
- #define **DEFAULT_DEVICE** 111
- #define **COM_PORT** 222

Functions

- int **sys_req** (int op_code, int device_id, char *buffer_ptr, int *count_ptr)
- void **mpx_init** (int cur_mod)
- void **sys_set_malloc** (u32int(*func)(u32int))
- void **sys_set_free** (int(*func)(void *))
- void * **sys_alloc_mem** (u32int size)
- int **sys_free_mem** (void *ptr)
- void **idle** ()
- u32int * **sys_call** (context *registers)

4.20 modules/R1/comHand.h File Reference

Functions

- int comHand ()

4.20.1 Function Documentation

4.20.1.1 comHand()

```
int comHand ( )
```

Description: Interprets user input to call the appropriate user functions.

Definition at line 22 of file comHand.c.

```
22                      {
23
24          Help("\0");
25
26          char cmdBuffer[100];
27          int bufferSize = 99;
28          int quit = 0;
29          int shutdown = 0;
30
31          while(quit != 1)    {
32              memset(cmdBuffer, '\0', 100);
33            sys_req(READ, DEFAULT_DEVICE, cmdBuffer, &bufferSize);
34              char* FirstToken = strtok(cmdBuffer, "-");
35          char* SecondToken = strtok(NULL, "-");
36              char* ThirdToken = strtok(NULL, "-");
37              char* FourthToken = strtok(NULL, "-");
38              char* FifthToken = strtok(NULL, "-");
39              if(shutdown == 0)   {
40 /*****************************************************************************
41                  R1 comHand
42 *****************************************************************************/
43                      if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,NULL) == 0)     {
44                          Help("\0");
45                      }
46                      //R1 Commands
47                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"version") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
48                          Help("Version");
49                      }
50                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"getDate") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
51                          Help("GetDate");
52                      }
53                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setDate") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
```

```
54                      Help("SetDate");
55                  }
56                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"getTime") == 0 &&
    strcmp(ThirdToken,NULL) == 0) {
57                      Help("GetTime");
58                  }
59                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setTime") == 0 &&
    strcmp(ThirdToken,NULL) == 0) {
60                      Help("SetTime");
61                  }
62                  // R2 Commands
63                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"suspend") == 0 &&
    strcmp(ThirdToken,NULL) == 0) {
64                      Help("suspend");
65                  }
66                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"resume") == 0 &&
    strcmp(ThirdToken,NULL) == 0)  {
67                      Help("resume");
68                  }
69                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setPriority") == 0 &&
    strcmp(ThirdToken,NULL) == 0) {
70                      Help("setPriority");
71                  }
72                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showPCB") == 0 &&
    strcmp(ThirdToken,NULL) == 0) {
73                      Help("showPCB");
74                  }
75                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showAll") == 0 &&
    strcmp(ThirdToken,NULL) == 0) {
76                      Help("showAll");
77                  }
78                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showReady") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
79                      Help("showReady");
80                  }
81                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showBlocked") == 0 &&
    strcmp(ThirdToken,NULL) == 0) {
82                      Help("showBlocked");
83                  }
84                  // Temporary R2 commands
85                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"createPCB") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
86                      Help("createPCB");
87                  }
88                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"deletePCB") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
89                      Help("deletePCB");
90                  }
91                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"block") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
92                      Help("block");
93                  }
94                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"unblock") == 0 &&
    strcmp(ThirdToken,NULL) == 0) {
95                      Help("unblock");
96                  }
97                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"shutdown") == 0 &&
    strcmp(ThirdToken,NULL) == 0)    {
98                      Help("shutdown");
99                  }
100                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"infinite") == 0 &&
    strcmp(ThirdToken,NULL) == 0)    {
101                     Help("infinte");
102                 }
103                 else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"loadr3") == 0 &&
    strcmp(ThirdToken,NULL) == 0)  {
104                     Help("loadr3");
105                 }
106                 else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"alarm") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
107                     Help("alarm");
108                 }
109                 else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
110                     Help("clear");
111                 }
112
113
114
115
116
117
118                 else if(strcmp(FirstToken,"version") == 0 && strcmp(SecondToken,NULL) == 0)
119                     Version();
120                 else if(strcmp(FirstToken,"clear") == 0 && strcmp(SecondToken,NULL) == 0)
121                     clear();
122
```

```
123                     else if(strcmp(FirstToken,"getDate") == 0 && strcmp(SecondToken,NULL) == 0)
124                         GetDate();
125 
126                     else if(strcmp(FirstToken,"setDate") == 0){
127                         if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
    EdgeCase(FourthToken) == 1 && EdgeCase(FifthToken) == 1)    {
128                             SetDate(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken),
    atoi(FifthToken));
129                         }
130                         else
131                             printf("\x1b[31m""\nERROR: Invalid parameters for setDate \n""\x1b[0m");
132                     }
133                     else if(strcmp(FirstToken,"getTime") == 0 && strcmp(SecondToken,NULL) == 0) //Return
    the current time held by the registers.
134                         GetTime();
135                     else if(strcmp(FirstToken,"setTime") == 0 && strcmp(FifthToken,NULL) == 0){
136                         if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
    EdgeCase(FourthToken) == 1)    {
137                             SetTime(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken));
    //input as Hour-Minute-Seconds
138                         }
139                         else
140                             printf("\x1b[31m""\nERROR: Invalid parameters for setTime \n""\x1b[0m");
141                     }
142 
143 
144 
145 
146 
147 
148      /*****************************************************************************
149                     R2 comHand
150      *****************************************************************************/
151                     else if(strcmp(FirstToken,"suspend") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
152                         Suspend(SecondToken);
153                     }
154                     else if(strcmp(FirstToken,"resume") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
155                         Resume(SecondToken);
156                     }
157                     else if(strcmp(FirstToken,"setPriority") == 0 && strcmp(FourthToken,NULL) == 0 &&
    strcmp(FifthToken,NULL) == 0) {
158                         if(EdgeCase(ThirdToken) == 1)    {
159                             Set_Priority(SecondToken, atoi(ThirdToken));    //input as
    setPriority-Process_Name-Priority
160                         }
161                         else
162                             printf("\x1b[31m""\nERROR: Invalid parameters for setPriority, priority must
    be entered as a integer. \n""\x1b[0m");
163                     }
164                     else if(strcmp(FirstToken,"showPCB") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
165                         Show_PCB(SecondToken);
166                         printf("\n");
167                     }
168                     else if(strcmp(FirstToken,"showAll") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
169                         Show_All();
170                         printf("\n");
171                     }
172                     else if(strcmp(FirstToken,"showReady") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
173                         Show_Ready();
174                         printf("\n");
175                     }
176                     else if(strcmp(FirstToken,"showBlocked") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
177                         Show_Blocked();
178                         printf("\n");
179                     }
180 
181 
182 
183                     /********** R2 Temp Commands **********/
184                     //Removed from active for R3/R4
185                     /*
186                     else if(strcmp(FirstToken,"createPCB") == 0) {
187                         if( strlen(SecondToken) < 11)    {
188                             Create_PCB(SecondToken, atoi(ThirdToken), atoi(FourthToken));
    //input as Process_Name-Priority-Class
189                         }
190                         else
191                             printf("\x1b[31m""\nERROR: Invalid parameters for createPCB, Process_name
    must only contain 10 or fewer characters. \n""\x1b[0m");
192                     }
193                     */
```

```
194                     else if(strcmp(FirstToken,"deletePCB") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
195                         Delete_PCB(SecondToken);
196                     }
197
198
199
200                     //Removed from active for R3/R4
201                     /*
202                     else if(strcmp(FirstToken,"block") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
203                         Block(SecondToken);
204                     }
205                     else if(strcmp(FirstToken,"unblock") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
206                         Unblock(SecondToken);
207                     }
208                     */
209      /*************************************************************************************
210                     R3 comHand
211      *************************************************************************************/
212                     //Removed for R4
213                     /*
214                     else if(strcmp(FirstToken,"yield") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
215                         yield();
216                         printf("\n");
217                     }
218                     else if(strcmp(FirstToken,"loadr3") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
219                         loader();
220                         printf("\n");
221                     }
222                     */
223      /*************************************************************************************
224                     R4 comHand
225      *************************************************************************************/
226                     else if(strcmp(FirstToken,"alarm") == 0) {
227                         if (EdgeCase(ThirdToken) == 1 && EdgeCase(FourthToken) == 1 &&
    EdgeCase(FifthToken) == 1)    {
228                             if (atoi(ThirdToken) < 23 && atoi(FourthToken) < 59 && atoi(FifthToken) <
    59)    {
229                                 loaderalarm(SecondToken, atoi(ThirdToken), atoi(FourthToken),
    atoi(FifthToken));
230                                 printf("\n");   //input as Message-Hour-Minute-Seconds
231                             }
232                             else
233                             printf("\x1b[31m""\nERROR: Invalid parameters for alarm, must be a valid
    time \n""\x1b[0m");
234                         }
235                         else
236                             printf("\x1b[31m""\nERROR: Invalid parameters for alarm \n""\x1b[0m");
237
238                     }
239                     else if(strcmp(FirstToken,"loadr3") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
240                         loader();
241                         printf("\n");
242                     }
243                     else if(strcmp(FirstToken,"infinite") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
244                         loaderinfinite();
245                         printf("\n");
246                     }
247      /*************************************************************************************
248                     shutdown comHand
249      *************************************************************************************/
250                     else if(strcmp(FirstToken,"shutdown") == 0 && strcmp(SecondToken,NULL) == 0){
251                         printf("\x1b[33m""\nAre you sure you want to shutdown? [yes/no]\n""\x1b[0m");
252                         shutdown = 1;
253                     }
254                     else    {
255                         printf("\x1b[31m""\nERROR: Not a valid command \n""\x1b[0m");
256                     }
257                 }
258                 else{
259                     if(strcmp(FirstToken,"yes") == 0 && shutdown == 1)    {
260                         quit = 1;
261                     }
262                     else if(strcmp(FirstToken,"no") == 0){
263                         printf("\x1b[33m""\nShutdown Cancelled\x1b[0m \n");
264                         shutdown = 0;
265                     }
266                     else
267                         printf("\x1b[31m""\nERROR: Please enter \"yes\" or \"no\" \n""\x1b[0m");
268                 }
269                 sys_req(IDLE, DEFAULT_DEVICE, NULL, NULL);
```

```
270         }
271         getReady() -> head = NULL;
272         sys_req(EXIT, DEFAULT_DEVICE, NULL, NULL);
273         return 0;   //shutdown procedure
274     }
```

4.21 modules/R1/userFunctions.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include <core/io.h>
#include "../mpx_supt.h"
#include "userFunctions.h"
#include "../procsr3.h"
#include "../sys_proc_loader.h"
```

Functions

- void **clear** ()
- char * itoa (int num)
- int BCDtoDec (int BCD)
- int DectoBCD (int Decimal)
- void **printf** (char msg[ ])
- int EdgeCase (char *pointer)
- void SetTime (int hours, int minutes, int seconds)
- void GetTime ()
- void SetDate (int day, int month, int millennium, int year)
- void GetDate ()
- void Version ()
- char toLowercase (char c)
- void Help (char *request)
- void Suspend (char *ProcessName)
- void Resume (char *ProcessName)
- void Set_Priority (char *ProcessName, int Priority)
- void Show_PCB (char *ProcessName)
- void Show_All ()
- void Show_Ready ()
- void Show_Blocked ()
- void Create_PCB (char *ProcessName, int Priority, int Class)
- void Delete_PCB (char *ProcessName)
- void Block (char *ProcessName)
- void Unblock (char *ProcessName)
- void **loader** ()
- void **loadr3** (char *name, u32int func)
- void **yield** ()
- void **loaderinfinite** ()
- List * **getList** ()
- void **loaderalarm** (char text[ ], int hours, int minutes, int seconds)

Variables

- List **AlarmList**

4.21.1 Function Documentation

4.21.1.1 BCDtoDec()

```
int BCDtoDec (
              int BCD )
```

Description: Changes binary number to decimal numbers.

Parameters

value	Binary number to be changed to decimal

Definition at line 79 of file userFunctions.c.

```
79                          {
80     return (((BCD»4)*10) + (BCD & 0xF));
81 }
```

4.21.1.2 Block()

```
void Block (
              char * ProcessName )
```

Brief Description: Places a PCD in the blocked state and reinserts it into the correct queue.

Description: Can except a string as a pointer that is the Process Name. The specified PCB will be places in a blocked state and reinserted into the appropriate queue. An error check for a valid name occurs.

Parameters

Process_Name	Character pointer that matches the name of process.

Definition at line 899 of file userFunctions.c.

```
899                                 {
900   PCB* pcb = FindPCB(ProcessName);
901   if (pcb == NULL)  {
902     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
903   }
904   else {
905     if(pcb->ReadyState == BLOCKED)  {
906         printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
907     }
908     else    {
909       RemovePCB(pcb);
910       pcb->ReadyState = BLOCKED;
911       InsertPCB(pcb);
912     }
```

```
913   }
914 }
```

4.21.1.3 Create_PCB()

```
void Create_PCB (
              char * ProcessName,
              int Priority,
              int Class )
```

Brief Description: Calls SetupPCB() and inserts PCB into appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Can accept two integers, Priority and Class. SetupPCB() will be called and the PCB will be inserted into the appropriate queue. An error check for unique and valid Process Name, an error check for valid process class, and an error check for process priority.

Parameters

Process_Name	Character pointer that matches the name of process.
Priority	integer that matches the priority number.
Class	integer that matches the class number.

Definition at line 842 of file userFunctions.c.

```
842                                                                    {
843   if (FindPCB(ProcessName) == NULL) {
844     if(Priority >= 0 && Priority < 10){
845       if(Class == 0 || Class == 1){
846         PCB* pcb = SetupPCB(ProcessName, Class, Priority);
847         InsertPCB(pcb);
848       } else{
849         printf("\x1b[31m""\nERROR: Not a valid Class \n""\x1b[0m");
850       }
851     } else{
852       printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
853     }
854   } else{
855     printf("\x1b[31m""\nERROR: This Process Name already exists \n""\x1b[0m");
856   }
857 }
```

4.21.1.4 DectoBCD()

```
int DectoBCD (
              int Decimal )
```

Description: Changes decimal numbers to binary numbers.

Parameters

Decimal	Decimal number to be changed to binary

Definition at line 86 of file userFunctions.c.

```
86                              {
```

```
87     return (((Decimal/10) << 4) | (Decimal % 10));
88 }
```

4.21.1.5 Delete_PCB()

```
void Delete_PCB (
             char * ProcessName )
```

Brief Description: Removes PCB from appropriate queue and frees all associated memory.

Description: Can except a string as a pointer that is the Process Name. Removes PCB from the appropriate queue and then frees all associated memory. An error check to make sure process name is valid.

Parameters

Process_Name	Character pointer that matches the name of process.

Definition at line 868 of file userFunctions.c.

```
868                                     {
869   PCB* pcb = FindPCB(ProcessName);
870   if (pcb == NULL)  {
871     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
872   }
873   else if(strcmp(pcb->Process_Name,"InfProc") == 0)   {
874         if(pcb->SuspendedState == YES)  {
875             RemovePCB(pcb);
876             FreePCB(pcb);
877         }
878         else
879             printf("\x1b[31m""\nERROR:This process cannot be deleted unless it is in the suspended
       state\n""\x1b[0m");
880   }
881   else if(pcb -> Process_Class == SYSTEM)   {
882       printf("\x1b[31m""\nERROR: System Processes cannot be deleted from the system. \n""\x1b[0m");
883   }
884     else {
885         RemovePCB(pcb);
886         FreePCB(pcb);
887     }
888 }
```

4.21.1.6 EdgeCase()

```
int EdgeCase (
             char * pointer )
```

Description: Compares pointer char to validate if it is a number or not.

Parameters

Compares	pointer char to validate if it is a number or not.

Definition at line 107 of file userFunctions.c.

```
107                                 {
108   int valid = 0;
109   if (strcmp(pointer, "00") == 0) {
```

```
110     valid = 1;
111     return valid;
112   }
113   else if (strcmp(pointer, "0") == 0) {
114     valid = 1;
115     return valid;
116   }
117   else  {
118     int j;
119     valid = 0;
120     for(j = 0; j <= 99; j++)  {
121       if(strcmp(pointer,itoa(j)) == 0)
122         valid = 1;
123     }
124     if(valid == 0)  {
125       return valid;
126     }
127   }
128   return valid;
129 }
```

4.21.1.7 GetDate()

```
void GetDate ( )
```

Description: Returns the full date back to the user in decimal form.

No parameters.

Definition at line 264 of file userFunctions.c.

```
264                 {
265   int check = 2;
266   outb(0x70,0x07);
267     unsigned char day = BCDtoDec(inb(0x71));
268     outb(0x70,0x08);
269     unsigned char month = BCDtoDec(inb(0x71));
270     outb(0x70,0x32);
271     unsigned char millennium = BCDtoDec(inb(0x71));
272     char msg[2] = "-";
273     char msg3[10] = "Date: ";
274     printf(msg3);
275     sys_req(WRITE, COM1, itoa(day), &check);
276     printf(msg);
277     sys_req(WRITE, COM1, itoa(month), &check);
278     printf(msg);
279     sys_req(WRITE, COM1, itoa(millennium), &check);
280   outb(0x70,0x09);
281   if(BCDtoDec(inb(0x71)) == 0){
282     sys_req(WRITE, COM1, "00", &check);
283   }
284   else {
285         unsigned char year = BCDtoDec(inb(0x71));
286         sys_req(WRITE, COM1, itoa(year), &check);
287   }
288     printf("\n");
289 }
```

4.21.1.8 GetTime()

```
void GetTime ( )
```

Description: retrieve and return the time values for hours, minutes, and seconds form the clock register using inb(Port,address).

No parameters.

Definition at line 186 of file userFunctions.c.

```
186                 {
187     int check = 2;
188     int hour;
189     int minute;
190     int second;
191     outb(0x70,0x04);
192     unsigned char hours = inb(0x71);
193     outb(0x70,0x02);
194     unsigned char minutes = inb(0x71);
195     outb(0x70,0x00);
196     unsigned char seconds = inb(0x71);
197     char msg1[2] = ":";
198     char msg2[10] = "Time: ";
199     printf(msg2);
200     hour = BCDtoDec(hours);
201     sys_req(WRITE, COM1, itoa(hour), &check);
202     printf(msg1);
203     minute = BCDtoDec(minutes);
204     sys_req(WRITE, COM1, itoa(minute), &check);
205     printf(msg1);
206     second = BCDtoDec(seconds);
207     sys_req(WRITE, COM1, itoa(second), &check);
208   printf("\n");
209 }
```

4.21.1.9 Help()

```
void Help (
              char * request )
```

Brief Description: Gives helpful information for one of the functions

Description: Can except a string as a pointer, if the pointer is null then the function will print a complete list of avaliable commands to the console. If the pointer is a avaliable commands then instructions on how to use the command will be printed. If the command does not exist then a message explaining that it is not a valid command will be displayed.

Parameters

request	Character pointer that matches the name of the function that you need help with.

Definition at line 318 of file userFunctions.c.

```
318                             {
319     if (request[0] == '\0') {
320         //removed for R3/R4 from active command list
321         //\n createPCB \n block \n unblock
322         printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
323         printf("\n getDate \n setDate \n getTime \n setTime \n version \n suspend \n resume \n
      setPriority \n showPCB \n showAll \n showReady \n showBlocked  \n deletePCB  \n shutdown \n alarm \n
      clear \n loadr3 \n infinte \n\n");
324     }
325     else if (strcmp(request, "GetDate") == 0) {
326         printf("\n getDate returns the current date that is loaded onto the operating system.\n");
327     }
328     else if (strcmp(request, "SetDate") == 0) {
329         printf("\n setDate allows the user to reset the correct date into the system, as follows
      setDate-"BLU"day"RESET"-"BLU"month"RESET"-"BLU"year"RESET".\n Time must be inputed as a two digit
      number, Example 02 or 00");
330     }
331     else if (strcmp(request, "GetTime") == 0) {
332         printf("\n getTime returns the current time as hours, minutes, seconds that is loaded onto the
      operating system.\n");
333     }
334     else if (strcmp(request, "SetTime") == 0) {
335         printf("\n setTime allows the user to reset the correct time into the system, as follows
      setTime-"BLU"hour"RESET"-"BLU"minute"RESET"-"BLU"second"RESET".\n Time must be inputed as a two digit
      number, Example 02 or 00");
336     }
337     else if (strcmp(request, "Version") == 0) {
```

```
338         printf("\n version returns the current operating software version that the system is
      running.\n");
339     }
340     else if (strcmp(request, "infinte") == 0) {
341         printf("\n infinite Loads the infinite process into the ready queue.\n");
342     }
343     else if (strcmp(request, "loadr3") == 0)  {
344         printf("\n loadr3 Loads in all five of the R3 test processes in a suspended state into the
      queue.\n");
345     }
346     else if (strcmp(request, "alarm") == 0)       {
347         printf("\n alarm creates a user specified alarm with a user set message and time
      alarm-MSG-hour-minute-second.\n");
348     }
349     else if (strcmp(request, "clear") == 0)       {
350         printf("\n clear erases the console of all typed commands and refreshes it with just the command
      list.\n");
351     }
352
353   else if(strcmp(request, "shutdown") == 0)   {
354     printf("\n shutdown shuts down the system.\n");
355   }
356
357
358
      /*****************************************************************************************************
359             R2 Commands
360
      *****************************************************************************************************/
361   else if(strcmp(request,"suspend") == 0) {
362         printf("\n Suspend takes in the name of a PCB (suspend-NAME) then places it into the suspended
      state and reinserts it into the correct queue.\n");
363   }
364   else if(strcmp(request,"resume") == 0) {
365         printf("\n Resume takes in the name of a PCB (resume-NAME) then removes it from the suspended
      state and adds it to the correct queue.\n");
366   }
367   else if(strcmp(request,"setPriority") == 0) {
368         printf("\n SetPriority takes in the name of a PCB and the priority (setPrioriry-NAME-PRIORITY)
      it needs to be set to then reinstates the specified PCB into a new location by priority.\n");
369   }
370   else if(strcmp(request,"showPCB") == 0) {
371         printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
      user.\n");
372   }
373   else if(strcmp(request,"showAll") == 0) {
374         printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
      queues.\n");
375   }
376   else if(strcmp(request,"showReady") == 0) {
377         printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
      currently are in the ready state.\n");
378   }
379   else if(strcmp(request,"showBlocked") == 0) {
380         printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
      currently are in the blocked state.\n");
381   }
382
383 /******************************** R2 Temp Commands
      *********************************************************/
384   else if(strcmp(request,"deletePCB") == 0) {
385         printf("\n DeletePCB takes in the process_name (deletePCB-NAME) then deletes it from the queue
      and free's all the memory that was previously allocated to the specified PCB.\n");
386   }
387   //removed for R3/R4 from active command list
388   /*
389   else if(strcmp(request,"createPCB") == 0) {
390         printf("\n CreatePCB takes in the process_name, process_class, and
      process_priority.(createPCB-NAME-PRIORITY-CLASS) Then assigns this new process into the correct
      queue.\n");
391   }
392   else if(strcmp(request,"block") == 0) {
393         printf("\n Block takes in the process_name (block-NAME) then sets it's state to blocked and
      reinserts it back into the correct queue.\n");
394   }
395   else if(strcmp(request,"unblock") == 0) {
396         printf("\n Unblock takes in the process_name (unblock-NAME) then sets it's state to ready and
      reinserts it back into the correct queue.\n");
397   }
398   */
399   else  {
400         printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
      full list of commands.\n""\x1b[0m");
401   }
402 }
```

4.21.1.10 itoa()

```
char* itoa (
            int num )
```

Description: An integer is taken and seperated into individual chars and then all placed into a character array. Adapted from geeksforgeeks.org.

Parameters

num	integer to be put into array Title: itoa Author: Neha Mahajan Date: 29 May, 2017 Availability: https://www.geeksforgeeks.org/implement-itoa/

Definition at line 49 of file userFunctions.c.

```
49                      {
50     int i,j,k,count;
51     i = num;
52     j = 0;
53     count = 0;
54     while(i){ // count number of digits
55         count++;
56         i /= 10;
57     }
58
59     char* arr1;
60     char arr2[count];
61     arr1 = (char*)sys_alloc_mem(count); //memory allocation
62
63     while(num){ // seperate last digit from number and add ASCII
64         arr2[++j] = num%10 + '0';
65         num /= 10;
66     }
67
68     for(k = 0; k < j; k++){ // reverse array results
69         arr1[k] = arr2[j-k];
70     }
71     arr1[k] = '\0';
72
73     return(char*)arr1;
74 }
```

4.21.1.11 Resume()

```
void Resume (
            char * ProcessName )
```

Brief Description: Places a PCD in the not suspended state and reinserts it into the appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Places a PCB in the not suspended state and reinserts it into the appropriate queue. An error check for valid Process Name.

Parameters

Process_Name	Character pointer that matches the name of process.

Definition at line 445 of file userFunctions.c.

```
445                                 {
446     PCB* pcb = FindPCB(ProcessName);
447     if (pcb == NULL)    {
448       printf(RED"\nERROR: Not a valid process name \n"RESET);
449     }
```

```
450     else {
451         if(pcb->SuspendedState == NO)   {
452             printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
453         }
454         else if(pcb -> Process_Class == APPLICATION)    {
455             pcb->SuspendedState = NO;
456         }
457         else
458             printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
459     }
460 }
```

4.21.1.12 Set_Priority()

```
void Set_Priority (
             char * ProcessName,
             int Priority )
```

Brief Description: Sets PCB priority and reinserts the process into the correct place in the correct queue.

Description: Can except a string as a pointer that is the Process Name. Can accept and integer than is the Priority. Sets a PCB's priority and reinserts the process into the correct place in the correct queue. An error check for valid Process Name and an error check for a valid priority 1 - 9.

Parameters

Process_Name	Character pointer that matches the name of process.
Priority	integer that matches the priority number.

Definition at line 472 of file userFunctions.c.

```
472                                                 {
473     PCB* pcb = FindPCB(ProcessName);
474     if (pcb == NULL)    {
475         printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
476     }
477     else if(Priority >= 10){
478         printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
479     }
480     else if(pcb -> Process_Class == APPLICATION) {
481         RemovePCB(pcb);
482         pcb->Priority = Priority;
483         InsertPCB(pcb);
484     }
485     else
486         printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
487 }
```

4.21.1.13 SetDate()

```
void SetDate (
             int day,
             int month,
             int millennium,
             int year )
```

Description: Sets the date register to the new values that the user inputed, all values must be inputed as Set↩ Dime(day, month, millenial, year).

Parameters

day	Integer to be set in the Day position
month	Integer to be set in the Month position
millenial	Integer to be set in the Millenial position
year	Integer to be set in the Year position

Definition at line 217 of file userFunctions.c.

```
217                                                                    {
218   outb(0x70,0x07);
219   int tempDay = BCDtoDec(inb(0x71));
220   outb(0x70,0x08);
221   int tempMonth = BCDtoDec(inb(0x71));
222   outb(0x70,0x32);
223   int tempMillennium = BCDtoDec(inb(0x71));
224   outb(0x70,0x09);
225   int tempYear = BCDtoDec(inb(0x71));
226   cli();
227     outb(0x70,0x07);
228     outb(0x71,DectoBCD (day));
229     outb(0x70,0x08);
230     outb(0x71,DectoBCD (month));
231     outb(0x70,0x32);
232     outb(0x71,DectoBCD (millennium));
233     outb(0x70,0x09);
234     outb(0x71,DectoBCD (year));
235     sti();
236   outb(0x70,0x07);
237   unsigned char newDay = BCDtoDec(inb(0x71));
238   outb(0x70,0x08);
239   unsigned char newMonth = BCDtoDec(inb(0x71));
240   outb(0x70,0x32);
241   unsigned char newMillennium = BCDtoDec(inb(0x71));
242   outb(0x70,0x09);
243   unsigned char newYear = BCDtoDec(inb(0x71));
244   if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
245     printf("Your input was invalid\n");
246     cli();
247         outb(0x70,0x07);
248         outb(0x71,DectoBCD (tempDay));
249         outb(0x70,0x08);
250         outb(0x71,DectoBCD (tempMonth));
251         outb(0x70,0x32);
252         outb(0x71,DectoBCD (tempMillennium));
253         outb(0x70,0x09);
254         outb(0x71,DectoBCD (tempYear));
255         sti();
256   }
257   else
258     printf("Date Set\n");
259 }
```

4.21.1.14 SetTime()

```
void SetTime (
              int hours,
              int minutes,
              int seconds )
```

Description: sets the time register to the new values that the user inputed, all values must be inputed as SetTime(↩ Hours, Minutes, Seconds).

Parameters

hours	Integer to be set in the Hour position
minutes	Integer to be set in the Minutes position
seconds	Integer to be set in the Seconds position

Definition at line 147 of file userFunctions.c.

```
147                                                 {
148   outb(0x70,0x04);
149   unsigned char tempHours = BCDtoDec(inb(0x71));
150   outb(0x70,0x02);
151   unsigned char tempMinutes = BCDtoDec(inb(0x71));
152   outb(0x70,0x00);
153   unsigned char tempSeconds = BCDtoDec(inb(0x71));
154     cli(); //outb(device + 1, 0x00); //disable interrupts
155     outb(0x70,0x04);
156     outb(0x71, DectoBCD(hours));// change to bcd
157     outb(0x70,0x02);
158     outb(0x71, DectoBCD(minutes));
159     outb(0x70,0x00);
160     outb(0x71, DectoBCD(seconds));
161     sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
162   outb(0x70,0x04);
163   unsigned char newHours = BCDtoDec(inb(0x71));
164   outb(0x70,0x02);
165   unsigned char newMinutes = BCDtoDec(inb(0x71));
166   outb(0x70,0x00);
167   unsigned char newSeconds = BCDtoDec(inb(0x71));
168   if(newHours != hours || newMinutes != minutes || newSeconds != seconds){
169     printf("Your input was invalid\n");
170     cli(); //outb(device + 1, 0x00); //disable interrupts
171         outb(0x70,0x04);
172         outb(0x71, DectoBCD(tempHours));// change to bcd
173         outb(0x70,0x02);
174         outb(0x71, DectoBCD(tempMinutes));
175         outb(0x70,0x00);
176         outb(0x71, DectoBCD(tempSeconds));
177         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
178   }
179   else
180     printf("Time Set\n");
181 }
```

4.21.1.15 Show_All()

```
void Show_All ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the ready and blocked queues.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the ready and blocked queues.

Definition at line 568 of file userFunctions.c.

```
568                 {
569     Show_Ready();
570     Show_Blocked();
571 }
```

4.21.1.16 Show_Blocked()

```
void Show_Blocked ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the blocked queue.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the blocked queue.

Definition at line 709 of file userFunctions.c.

```
709                     {
710     if(getBlocked()->head == NULL)  {
```

```
711           printf("\x1b[32m""\n The Blocked Queue is empty \n""\x1b[0m");
712       }
713       else    {
714           int class, check, state, prior, status;
715           char name[20];
716           char block[] = "\x1B[34m""Blocked Queue: \n""\x1b[0m";
717           char cname[] = "Name: ";
718           char cclass[] = "Class: ";
719           char cstate[] = "State: ";
720           char cstatus[] = "Status: ";
721           char cprior[] = "Priority: ";
722           char line[] = "\n";
723           check = 15;
724
725           sys_req(WRITE, COM1, block, &check );
726
727           PCB* pcb = getBlocked()->head;
728
729           if(pcb->next == NULL) {
730             class = pcb->Process_Class;
731                 strcpy(name,pcb->Process_Name);
732                 state = pcb->ReadyState;
733                 status = pcb->SuspendedState;
734                 prior = pcb->Priority;
735
736                 printf(cname);
737                 printf(name);
738                 printf(line);
739
740                 printf(cclass);
741                 if(pcb->Process_Class == 0)  {
742                   printf("0");
743                 }
744                 else  {
745                   sys_req(WRITE, COM1, itoa(class), &check);
746                 }
747                 printf(line);
748
749                 printf(cstate);
750                 if(pcb->ReadyState == 0)  {
751                   printf("0");
752                 }
753                 else  {
754                   sys_req(WRITE, COM1, itoa(state), &check);
755                 }
756                 printf(line);
757
758                 printf(cstatus);
759                 if(pcb->SuspendedState == 0)  {
760                   printf("0");
761                 }
762                 else  {
763                   sys_req(WRITE, COM1, itoa(status), &check);
764                 }
765                 printf(line);
766
767                 printf(cprior);
768                 if(pcb->Priority == 0)  {
769                   printf("0");
770                   printf("\n\n");
771                 }
772                 else  {
773                   sys_req(WRITE, COM1, itoa(prior), &check);
774                   printf("\n\n");
775                 }
776           }
777           else  {
778             while(pcb != NULL)  {
779                 class = pcb->Process_Class;
780                     strcpy(name,pcb->Process_Name);
781                     state = pcb->ReadyState;
782                     status = pcb->SuspendedState;
783                     prior = pcb->Priority;
784
785                     printf(cname);
786                     printf(name);
787                     printf(line);
788
789                     printf(cclass);
790                     if(pcb->Process_Class == 0)  {
791                       printf("0");
792                     }
793                     else  {
794                       sys_req(WRITE, COM1, itoa(class), &check);
795                     }
796                     printf(line);
797
```

```
798                     printf(cstate);
799                     if(pcb->ReadyState == 0)  {
800                       printf("0");
801                     }
802                     else  {
803                       sys_req(WRITE, COM1, itoa(state), &check);
804                     }
805                     printf(line);
806 
807                     printf(cstatus);
808                     if(pcb->SuspendedState == 0)  {
809                       printf("0");
810                     }
811                     else  {
812                       sys_req(WRITE, COM1, itoa(status), &check);
813                     }
814                     printf(line);
815 
816                     printf(cprior);
817                     if(pcb->Priority == 0)  {
818                       printf("0");
819                       printf("\n\n");
820                     }
821                     else  {
822                       sys_req(WRITE, COM1, itoa(prior), &check);
823                       printf("\n\n");
824                     }
825                 pcb = pcb->next;
826             }
827         }
828     }
829 }
```

4.21.1.17 Show_PCB()

```
void Show_PCB (
             char * ProcessName )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of a PCB.

Description: Can except a string as a pointer that is the Process Name. The process name, claas, state, suspend status, and priority of a PCB are displayed. An error check for a valid name occurs.

Parameters

Process_Name	Character pointer that matches the name of process

Definition at line 497 of file userFunctions.c.

```
497                                         {
498     if (FindPCB(ProcessName) == NULL)    {
499         printf("\x1b[31m""\nERROR: PCB does not exist \n""\x1b[0m");
500     }
501     else    {
502         int check = 5;
503         char name[10];
504         char cname[] = "Name: ";
505         char cclass[] = "Class: ";
506         char cstate[] = "State: ";
507         char cstatus[] = "Status: ";
508         char cprior[] = "Priority: ";
509         char line[] = "\n";
510         PCB* pcb = FindPCB(ProcessName);
511         strcpy(name,pcb->Process_Name);
512         int class = pcb->Process_Class;
513         int state = pcb->ReadyState;
514         int status = pcb->SuspendedState;
515         int prior = pcb->Priority;
516 
517         if(name == NULL){
518             printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
519         }
```

```
520         else    {
521             printf(cname);
522             printf(ProcessName);
523             printf(line);
524             printf(cclass);
525             if(pcb->Process_Class == 0)  {
526                 printf("0");
527             }
528             else  {
529                 sys_req(WRITE, COM1, itoa(class), &check);
530             }
531             printf(line);
532             printf(cstate);
533             if(pcb->ReadyState == 0)  {
534                 printf("0");
535             }
536             else  {
537                 sys_req(WRITE, COM1, itoa(state), &check);
538             }
539             printf(line);
540             printf(cstatus);
541             if(pcb->SuspendedState == 0)  {
542                 printf("0");
543             }
544             else  {
545                 sys_req(WRITE, COM1, itoa(status), &check);
546             }
547             printf(line);
548             printf(cprior);
549             if(pcb->Priority == 0)  {
550                 printf("0");
551                 printf("\n\n");
552             }
553             else  {
554                 sys_req(WRITE, COM1, itoa(prior), &check);
555                 printf("\n\n");
556             }
557         }
558     }
559 }
```

4.21.1.18 Show_Ready()

```
void Show_Ready ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the ready queue.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the ready queue.

Definition at line 580 of file userFunctions.c.

```
580                     {
581     if(getReady()->head == NULL)    {
582          printf("\x1b[32m""\n The Ready Queue is empty \n""\x1b[0m");
583     }
584     else    {
585         int class, check, state, prior, status;
586       char name[10];
587       char ready[] = "\x1B[34m""\nReady Queue:\n""\x1B[0m";
588       char cname[] = "Name: ";
589       char cclass[] = "Class: ";
590       char cstate[] = "State: ";
591       char cstatus[] = "Status: ";
592       char cprior[] = "Priority: ";
593       char line[] = "\n";
594       check = 5;
595
596       sys_req(WRITE, COM1, ready, &check );
597
598       PCB* pcb = getReady()->head;
599
600         if(pcb->next == NULL)   {
601             class = pcb->Process_Class;
602             strcpy(name,pcb->Process_Name);
603             state = pcb->ReadyState;
604             status = pcb->SuspendedState;
```

```
605             prior = pcb->Priority;
606
607             printf(cname);
608             printf(name);
609             printf(line);
610
611             printf(cclass);
612             if(pcb->Process_Class == 0)  {
613               printf("0");
614             }
615             else  {
616               sys_req(WRITE, COM1, itoa(class), &check);
617             }
618             printf(line);
619
620             printf(cstate);
621             if(pcb->ReadyState == 0)  {
622               printf("0");
623             }
624             else  {
625               sys_req(WRITE, COM1, itoa(state), &check);
626             }
627             printf(line);
628
629             printf(cstatus);
630             if(pcb->SuspendedState == 0)  {
631               printf("0");
632             }
633             else  {
634               sys_req(WRITE, COM1, itoa(status), &check);
635             }
636             printf(line);
637
638             printf(cprior);
639             if(pcb->Priority == 0)  {
640               printf("0");
641               printf("\n\n");
642             }
643             else  {
644               sys_req(WRITE, COM1, itoa(prior), &check);
645               printf("\n\n");
646             }
647       }
648       else  {
649         while(pcb != NULL)  {
650                 class = pcb->Process_Class;
651                 strcpy(name,pcb->Process_Name);
652                 state = pcb->ReadyState;
653                 status = pcb->SuspendedState;
654                 prior = pcb->Priority;
655
656                 printf(cname);
657                 printf(name);
658                 printf(line);
659
660                 printf(cclass);
661                 if(pcb->Process_Class == 0)  {
662                   printf("0");
663                 }
664                 else  {
665                   sys_req(WRITE, COM1, itoa(class), &check);
666                 }
667                 printf(line);
668
669                 printf(cstate);
670                 if(pcb->ReadyState == 0)  {
671                   printf("0");
672                 }
673                 else  {
674                   sys_req(WRITE, COM1, itoa(state), &check);
675                 }
676                 printf(line);
677
678                 printf(cstatus);
679                 if(pcb->SuspendedState == 0)  {
680                   printf("0");
681                 }
682                 else  {
683                   sys_req(WRITE, COM1, itoa(status), &check);
684                 }
685                 printf(line);
686
687                 printf(cprior);
688                 if(pcb->Priority == 0)  {
689                   printf("0");
690                   printf("\n\n");
691                 }
```

```
692                 else  {
693                   sys_req(WRITE, COM1, itoa(prior), &check);
694                   printf("\n\n");
695                 }
696                 pcb = pcb->next;
697             }
698         }
699     }
700 }
```

4.21.1.19 Suspend()

```
void Suspend (
              char * ProcessName )
```

Brief Description: Places a PCD in the suspended state and reinserts it into the appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Places a PCB in the suspended state and reinserts it into the appropriate queue. An error check for valid Process Name.

Parameters

Process_Name	Character pointer that matches the name of process.

Definition at line 419 of file userFunctions.c.

```
419                                 {
420     PCB* pcb = FindPCB(ProcessName);
421     if (pcb == NULL)    {
422       printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
423     }
424     else {
425         if(pcb->SuspendedState == YES)  {
426             printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
427         }
428         else if(pcb -> Process_Class == APPLICATION)    {
429             pcb->SuspendedState = YES;
430         }
431         else
432             printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
433     }
434 }
```

4.21.1.20 toLowercase()

```
char toLowercase (
              char c )
```

Description: If a letter is uppercase, it changes it to lowercase. (char)

Parameters

c	Character that is to be changed to its lowercase equivalent

Definition at line 301 of file userFunctions.c.

```
301                             {
302     if((c >= 65) && (c <= 90))  {
```

```
303         c = c + 32;
304     }
305     return c;
306 }
```

4.21.1.21 Unblock()

```
void Unblock (
             char * ProcessName )
```

Brief Description: Places a PCD in the unblocked state and reinserts it into the correct queue.

Description: Can except a string as a pointer that is the Process Name. The specified PCB will be places in an unblocked state and reinserted into the appropriate queue. An error check for a valid name occurs.

Parameters

Process_Name	Character pointer that matches the name of process.

Definition at line 925 of file userFunctions.c.

```
925                                 {
926   PCB* pcb = FindPCB(ProcessName);
927   if (pcb == NULL)  {
928     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
929   }
930   else {
931     if(pcb->ReadyState == READY)    {
932         printf("\x1b[32m""\nThis Process is already in the READY state \n""\x1b[0m");
933     }
934     else    {
935       RemovePCB(pcb);
936       pcb->ReadyState = READY;
937       InsertPCB(pcb);
938     }
939   }
940 }
```

4.21.1.22 Version()

```
void Version ( )
```

Description: Simply returns a char containing "Version: R(module).(the iteration that module is currently on).

No parameters.

Definition at line 294 of file userFunctions.c.

```
294                 {
295     printf("Version: R4.6 \n");
296 }
```

4.21.2 Variable Documentation

4.21.2.1 AlarmList

```
List AlarmList
```

Initial value:

```
={
  .head = NULL,
  .tail = NULL
}
```

Definition at line 987 of file userFunctions.c.

4.22 modules/R1/userFunctions.h File Reference

Classes

- struct Alarm
- struct List

Macros

- #define **RED** "\x1B[31m"
- #define **GRN** "\x1B[32m"
- #define **YEL** "\x1B[33m"
- #define **BLU** "\x1B[34m"
- #define **MAG** "\x1B[35m"
- #define **CYN** "\x1B[36m"
- #define **WHT** "\x1B[37m"
- #define **RESET** "\x1B[0m"

Typedefs

- typedef struct Alarm **Alarm**
- typedef struct List **List**

Functions

- void SetTime (int hours, int minutes, int seconds)
- void GetTime ()
- int DectoBCD (int Decimal)
- void **clear** ()
- char * itoa (int num)
- void SetDate (int day, int month, int millennium, int year)
- int BCDtoDec (int BCD)
- void GetDate ()
- void Version ()
- void Help (char *request)
- void **printf** (char msg[ ])
- int EdgeCase (char *pointer)
- char toLowercase (char c)
- void Suspend (char *ProcessName)

- void Resume (char *ProcessName)
- void Set_Priority (char *ProcessName, int Priority)
- void Show_PCB (char *ProcessName)
- void Show_All ()
- void Show_Ready ()
- void Show_Blocked ()
- void Create_PCB (char *ProcessName, int Priority, int Class)
- void Delete_PCB (char *ProcessName)
- void Block (char *ProcessName)
- void Unblock (char *ProcessName)
- void **loader** ()
- void **loadr3** (char *name, u32int func)
- void **yield** ()
- void **loaderinfinite** ()
- List * **getList** ()
- void **loaderalarm** ()

4.22.1 Function Documentation

4.22.1.1 BCDtoDec()

```
int BCDtoDec (
            int BCD )
```

Description: Changes binary number to decimal numbers.

Parameters

value	Binary number to be changed to decimal

Definition at line 79 of file userFunctions.c.

```
79                          {
80     return (((BCD»4)*10) + (BCD & 0xF));
81 }
```

4.22.1.2 Block()

```
void Block (
            char * ProcessName )
```

Brief Description: Places a PCD in the blocked state and reinserts it into the correct queue.

Description: Can except a string as a pointer that is the Process Name. The specified PCB will be places in a blocked state and reinserted into the appropriate queue. An error check for a valid name occurs.

Parameters

Process_Name	Character pointer that matches the name of process.

Definition at line 899 of file userFunctions.c.

```
899                                 {
900   PCB* pcb = FindPCB(ProcessName);
901   if (pcb == NULL)  {
902     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
903   }
904   else {
905     if(pcb->ReadyState == BLOCKED)  {
906         printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
907     }
908     else    {
909       RemovePCB(pcb);
910       pcb->ReadyState = BLOCKED;
911       InsertPCB(pcb);
912     }
913   }
914 }
```

4.22.1.3 Create_PCB()

```
void Create_PCB (
             char * ProcessName,
             int Priority,
             int Class )
```

Brief Description: Calls SetupPCB() and inserts PCB into appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Can accept two integers, Priority and Class. SetupPCB() will be called and the PCB will be inserted into the appropriate queue. An error check for unique and valid Process Name, an error check for valid process class, and an error check for process priority.

Parameters

Process_Name	Character pointer that matches the name of process.
Priority	integer that matches the priority number.
Class	integer that matches the class number.

Definition at line 842 of file userFunctions.c.

```
842                                                                 {
843   if (FindPCB(ProcessName) == NULL) {
844     if(Priority >= 0 && Priority < 10){
845       if(Class == 0 || Class == 1){
846         PCB* pcb = SetupPCB(ProcessName, Class, Priority);
847         InsertPCB(pcb);
848       } else{
849         printf("\x1b[31m""\nERROR: Not a valid Class \n""\x1b[0m");
850       }
851     } else{
852       printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
853     }
854   } else{
855     printf("\x1b[31m""\nERROR: This Process Name already exists \n""\x1b[0m");
856   }
857 }
```

4.22.1.4 DectoBCD()

```
int DectoBCD (
              int Decimal )
```

Description: Changes decimal numbers to binary numbers.

Parameters

Decimal	Decimal number to be changed to binary

Definition at line 86 of file userFunctions.c.

```
86                              {
87     return (((Decimal/10) « 4) | (Decimal % 10));
88 }
```

4.22.1.5 Delete_PCB()

```
void Delete_PCB (
              char * ProcessName )
```

Brief Description: Removes PCB from appropriate queue and frees all associated memory.

Description: Can except a string as a pointer that is the Process Name. Removes PCB from the appropriate queue and then frees all associated memory. An error check to make sure process name is valid.

Parameters

Process_Name	Character pointer that matches the name of process.

Definition at line 868 of file userFunctions.c.

```
868                                     {
869   PCB* pcb = FindPCB(ProcessName);
870   if (pcb == NULL)  {
871     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
872   }
873   else if(strcmp(pcb->Process_Name,"InfProc") == 0)   {
874         if(pcb->SuspendedState == YES)  {
875             RemovePCB(pcb);
876             FreePCB(pcb);
877         }
878         else
879             printf("\x1b[31m""\nERROR:This process cannot be deleted unless it is in the suspended
       state\n""\x1b[0m");
880   }
881   else if(pcb -> Process_Class == SYSTEM)   {
882       printf("\x1b[31m""\nERROR: System Processes cannot be deleted from the system. \n""\x1b[0m");
883   }
884     else {
885         RemovePCB(pcb);
886         FreePCB(pcb);
887     }
888 }
```

4.22.1.6 EdgeCase()

```
int EdgeCase (
              char * pointer )
```

Description: Compares pointer char to validate if it is a number or not.

Parameters

Compares	pointer char to validate if it is a number or not.

Definition at line 107 of file userFunctions.c.

```
107                                 {
108   int valid = 0;
109   if (strcmp(pointer, "00") == 0) {
110     valid = 1;
111     return valid;
112   }
113   else if (strcmp(pointer, "0") == 0) {
114     valid = 1;
115     return valid;
116   }
117   else  {
118     int j;
119     valid = 0;
120     for(j = 0; j <= 99; j++)  {
121       if(strcmp(pointer,itoa(j)) == 0)
122         valid = 1;
123     }
124     if(valid == 0)  {
125       return valid;
126     }
127   }
128   return valid;
129 }
```

4.22.1.7 GetDate()

```
void GetDate ( )
```

Description: Returns the full date back to the user in decimal form.

No parameters.

Definition at line 264 of file userFunctions.c.

```
264                     {
265   int check = 2;
266   outb(0x70,0x07);
267     unsigned char day = BCDtoDec(inb(0x71));
268     outb(0x70,0x08);
269     unsigned char month = BCDtoDec(inb(0x71));
270     outb(0x70,0x32);
271     unsigned char millennium = BCDtoDec(inb(0x71));
272     char msg[2] = "-";
273     char msg3[10] = "Date: ";
274     printf(msg3);
275     sys_req(WRITE, COM1, itoa(day), &check);
276     printf(msg);
277     sys_req(WRITE, COM1, itoa(month), &check);
278     printf(msg);
279     sys_req(WRITE, COM1, itoa(millennium), &check);
280   outb(0x70,0x09);
281   if(BCDtoDec(inb(0x71)) == 0){
282     sys_req(WRITE, COM1, "00", &check);
283   }
284   else {
285         unsigned char year = BCDtoDec(inb(0x71));
286         sys_req(WRITE, COM1, itoa(year), &check);
287   }
288     printf("\n");
289 }
```

4.22.1.8 GetTime()

```
void GetTime ( )
```

Description: retrieve and return the time values for hours, minutes, and seconds form the clock register using inb(Port,address).

No parameters.

Definition at line 186 of file userFunctions.c.

```
186                 {
187     int check = 2;
188     int hour;
189     int minute;
190     int second;
191     outb(0x70,0x04);
192     unsigned char hours = inb(0x71);
193     outb(0x70,0x02);
194     unsigned char minutes = inb(0x71);
195     outb(0x70,0x00);
196     unsigned char seconds = inb(0x71);
197     char msg1[2] = ":";
198     char msg2[10] = "Time: ";
199     printf(msg2);
200     hour = BCDtoDec(hours);
201     sys_req(WRITE, COM1, itoa(hour), &check);
202     printf(msg1);
203     minute = BCDtoDec(minutes);
204     sys_req(WRITE, COM1, itoa(minute), &check);
205     printf(msg1);
206     second = BCDtoDec(seconds);
207     sys_req(WRITE, COM1, itoa(second), &check);
208   printf("\n");
209 }
```

4.22.1.9 Help()

```
void Help (
              char * request )
```

Brief Description: Gives helpful information for one of the functions

Description: Can except a string as a pointer, if the pointer is null then the function will print a complete list of avaliable commands to the console. If the pointer is a avaliable commands then instructions on how to use the command will be printed. If the command does not exist then a message explaining that it is not a valid command will be displayed.

Parameters

request	Character pointer that matches the name of the function that you need help with.

Definition at line 318 of file userFunctions.c.

```
318                             {
319     if (request[0] == '\0') {
320         //removed for R3/R4 from active command list
321         //\n createPCB \n block \n unblock
322         printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
323         printf("\n getDate \n setDate \n getTime \n setTime \n version \n suspend \n resume \n
      setPriority \n showPCB \n showAll \n showReady \n showBlocked  \n deletePCB  \n shutdown \n alarm \n
      clear \n loadr3 \n infinte \n\n");
324     }
325     else if (strcmp(request, "GetDate") == 0) {
326         printf("\n getDate returns the current date that is loaded onto the operating system.\n");
327     }
```

```
328     else if (strcmp(request, "SetDate") == 0) {
329         printf("\n setDate allows the user to reset the correct date into the system, as follows
      setDate-"BLU"day"RESET"-"BLU"month"RESET"-"BLU"year"RESET".\n Time must be inputed as a two digit
      number, Example 02 or 00");
330     }
331     else if (strcmp(request, "GetTime") == 0) {
332         printf("\n getTime returns the current time as hours, minutes, seconds that is loaded onto the
      operating system.\n");
333     }
334     else if (strcmp(request, "SetTime") == 0) {
335         printf("\n setTime allows the user to reset the correct time into the system, as follows
      setTime-"BLU"hour"RESET"-"BLU"minute"RESET"-"BLU"second"RESET".\n Time must be inputed as a two digit
      number, Example 02 or 00");
336     }
337     else if (strcmp(request, "Version") == 0) {
338         printf("\n version returns the current operating software version that the system is
      running.\n");
339     }
340     else if (strcmp(request, "infinte") == 0) {
341         printf("\n infinite Loads the infinite process into the ready queue.\n");
342     }
343     else if (strcmp(request, "loadr3") == 0)  {
344         printf("\n loadr3 Loads in all five of the R3 test processes in a suspended state into the
      queue.\n");
345     }
346     else if (strcmp(request, "alarm") == 0)       {
347         printf("\n alarm creates a user specified alarm with a user set message and time
      alarm-MSG-hour-minute-second.\n");
348     }
349     else if (strcmp(request, "clear") == 0)       {
350         printf("\n clear erases the console of all typed commands and refreshes it with just the command
      list.\n");
351     }
352
353   else if(strcmp(request, "shutdown") == 0)   {
354     printf("\n shutdown shuts down the system.\n");
355   }
356
357
358
      /*****************************************************************************************************
359             R2 Commands
360
      *****************************************************************************************************/
361   else if(strcmp(request,"suspend") == 0) {
362         printf("\n Suspend takes in the name of a PCB (suspend-NAME) then places it into the suspended
      state and reinserts it into the correct queue.\n");
363   }
364   else if(strcmp(request,"resume") == 0) {
365         printf("\n Resume takes in the name of a PCB (resume-NAME) then removes it from the suspended
      state and adds it to the correct queue.\n");
366   }
367   else if(strcmp(request,"setPriority") == 0) {
368         printf("\n SetPriority takes in the name of a PCB and the priority (setPrioriry-NAME-PRIORITY)
      it needs to be set to then reinstates the specified PCB into a new location by priority.\n");
369   }
370   else if(strcmp(request,"showPCB") == 0) {
371         printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
      user.\n");
372   }
373   else if(strcmp(request,"showAll") == 0) {
374         printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
      queues.\n");
375   }
376   else if(strcmp(request,"showReady") == 0) {
377         printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
      currently are in the ready state.\n");
378   }
379   else if(strcmp(request,"showBlocked") == 0) {
380         printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
      currently are in the blocked state.\n");
381   }
382
383 /******************************** R2 Temp Commands
      *******************************************************/
384   else if(strcmp(request,"deletePCB") == 0) {
385         printf("\n DeletePCB takes in the process_name (deletePCB-NAME) then deletes it from the queue
      and free's all the memory that was previously allocated to the specified PCB.\n");
386   }
387   //removed for R3/R4 from active command list
388   /*
389   else if(strcmp(request,"createPCB") == 0) {
390         printf("\n CreatePCB takes in the process_name, process_class, and
      process_priority.(createPCB-NAME-PRIORITY-CLASS) Then assigns this new process into the correct
      queue.\n");
391   }
392   else if(strcmp(request,"block") == 0) {
```

```
393         printf("\n Block takes in the process_name (block-NAME) then sets it's state to blocked and
       reinserts it back into the correct queue.\n");
394    }
395    else if(strcmp(request,"unblock") == 0) {
396         printf("\n Unblock takes in the process_name (unblock-NAME) then sets it's state to ready and
       reinserts it back into the correct queue.\n");
397    }
398    */
399    else  {
400         printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
       full list of commands.\n""\x1b[0m");
401    }
402 }
```

4.22.1.10 itoa()

```
char* itoa (
             int num )
```

Description: An integer is taken and seperated into individual chars and then all placed into a character array. Adapted from geeksforgeeks.org.

Parameters

num	integer to be put into array Title: itoa Author: Neha Mahajan Date: 29 May, 2017 Availability: https://www.geeksforgeeks.org/implement-itoa/

Definition at line 49 of file userFunctions.c.

```
49                    {
50     int i,j,k,count;
51     i = num;
52     j = 0;
53     count = 0;
54     while(i){ // count number of digits
55         count++;
56         i /= 10;
57     }
58
59     char* arr1;
60     char arr2[count];
61     arr1 = (char*)sys_alloc_mem(count); //memory allocation
62
63     while(num){ // seperate last digit from number and add ASCII
64         arr2[++j] = num%10 + '0';
65         num /= 10;
66     }
67
68     for(k = 0; k < j; k++){ // reverse array results
69         arr1[k] = arr2[j-k];
70     }
71     arr1[k] = '\0';
72
73     return(char*)arr1;
74 }
```

4.22.1.11 Resume()

```
void Resume (
             char * ProcessName )
```

Brief Description: Places a PCD in the not suspended state and reinserts it into the appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Places a PCB in the not suspended state and reinserts it into the appropriate queue. An error check for valid Process Name.

Parameters

Process_Name	Character pointer that matches the name of process.

Definition at line 445 of file userFunctions.c.

```
445                                 {
446     PCB* pcb = FindPCB(ProcessName);
447     if (pcb == NULL)    {
448       printf(RED"\nERROR: Not a valid process name \n"RESET);
449     }
450     else {
451         if(pcb->SuspendedState == NO)   {
452             printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
453         }
454         else if(pcb -> Process_Class == APPLICATION)    {
455             pcb->SuspendedState = NO;
456         }
457         else
458             printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
459     }
460 }
```

4.22.1.12 Set_Priority()

```
void Set_Priority (
              char * ProcessName,
              int Priority )
```

Brief Description: Sets PCB priority and reinserts the process into the correct place in the correct queue.

Description: Can except a string as a pointer that is the Process Name. Can accept and integer than is the Priority. Sets a PCB's priority and reinserts the process into the correct place in the correct queue. An error check for valid Process Name and an error check for a valid priority 1 - 9.

Parameters

Process_Name	Character pointer that matches the name of process.
Priority	integer that matches the priority number.

Definition at line 472 of file userFunctions.c.

```
472                                                 {
473     PCB* pcb = FindPCB(ProcessName);
474     if (pcb == NULL)    {
475         printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
476     }
477     else if(Priority >= 10){
478         printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
479     }
480     else if(pcb -> Process_Class == APPLICATION) {
481         RemovePCB(pcb);
482         pcb->Priority = Priority;
483         InsertPCB(pcb);
484     }
485     else
486         printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
487 }
```

4.22.1.13 SetDate()

```
void SetDate (
              int day,
```

```
            int month,
            int millennium,
            int year )
```

Description: Sets the date register to the new values that the user inputed, all values must be inputed as SetDime(day, month, millenial, year).

Parameters

day	Integer to be set in the Day position
month	Integer to be set in the Month position
millenial	Integer to be set in the Millenial position
year	Integer to be set in the Year position

Definition at line 217 of file userFunctions.c.

```
217                                                                 {
218   outb(0x70,0x07);
219   int tempDay = BCDtoDec(inb(0x71));
220   outb(0x70,0x08);
221   int tempMonth = BCDtoDec(inb(0x71));
222   outb(0x70,0x32);
223   int tempMillennium = BCDtoDec(inb(0x71));
224   outb(0x70,0x09);
225   int tempYear = BCDtoDec(inb(0x71));
226   cli();
227     outb(0x70,0x07);
228     outb(0x71,DectoBCD (day));
229     outb(0x70,0x08);
230     outb(0x71,DectoBCD (month));
231     outb(0x70,0x32);
232     outb(0x71,DectoBCD (millennium));
233     outb(0x70,0x09);
234     outb(0x71,DectoBCD (year));
235     sti();
236   outb(0x70,0x07);
237   unsigned char newDay = BCDtoDec(inb(0x71));
238   outb(0x70,0x08);
239   unsigned char newMonth = BCDtoDec(inb(0x71));
240   outb(0x70,0x32);
241   unsigned char newMillennium = BCDtoDec(inb(0x71));
242   outb(0x70,0x09);
243   unsigned char newYear = BCDtoDec(inb(0x71));
244   if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
245     printf("Your input was invalid\n");
246     cli();
247         outb(0x70,0x07);
248         outb(0x71,DectoBCD (tempDay));
249         outb(0x70,0x08);
250         outb(0x71,DectoBCD (tempMonth));
251         outb(0x70,0x32);
252         outb(0x71,DectoBCD (tempMillennium));
253         outb(0x70,0x09);
254         outb(0x71,DectoBCD (tempYear));
255         sti();
256   }
257   else
258     printf("Date Set\n");
259 }
```

4.22.1.14 SetTime()

```
void SetTime (
            int hours,
            int minutes,
            int seconds )
```

Description: sets the time register to the new values that the user inputed, all values must be inputed as SetTime(Hours, Minutes, Seconds).

Parameters

hours	Integer to be set in the Hour position
minutes	Integer to be set in the Minutes position
seconds	Integer to be set in the Seconds position

Definition at line 147 of file userFunctions.c.

```
147                                                 {
148   outb(0x70,0x04);
149   unsigned char tempHours = BCDtoDec(inb(0x71));
150   outb(0x70,0x02);
151   unsigned char tempMinutes = BCDtoDec(inb(0x71));
152   outb(0x70,0x00);
153   unsigned char tempSeconds = BCDtoDec(inb(0x71));
154     cli(); //outb(device + 1, 0x00); //disable interrupts
155     outb(0x70,0x04);
156     outb(0x71, DectoBCD(hours));// change to bcd
157     outb(0x70,0x02);
158     outb(0x71, DectoBCD(minutes));
159     outb(0x70,0x00);
160     outb(0x71, DectoBCD(seconds));
161     sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
162   outb(0x70,0x04);
163   unsigned char newHours = BCDtoDec(inb(0x71));
164   outb(0x70,0x02);
165   unsigned char newMinutes = BCDtoDec(inb(0x71));
166   outb(0x70,0x00);
167   unsigned char newSeconds = BCDtoDec(inb(0x71));
168   if(newHours != hours || newMinutes != minutes || newSeconds != seconds){
169     printf("Your input was invalid\n");
170     cli(); //outb(device + 1, 0x00); //disable interrupts
171         outb(0x70,0x04);
172         outb(0x71, DectoBCD(tempHours));// change to bcd
173         outb(0x70,0x02);
174         outb(0x71, DectoBCD(tempMinutes));
175         outb(0x70,0x00);
176         outb(0x71, DectoBCD(tempSeconds));
177         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
178   }
179   else
180     printf("Time Set\n");
181 }
```

4.22.1.15 Show_All()

```
void Show_All ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the ready and blocked queues.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the ready and blocked queues.

Definition at line 568 of file userFunctions.c.

```
568                 {
569     Show_Ready();
570     Show_Blocked();
571 }
```

4.22.1.16 Show_Blocked()

```
void Show_Blocked ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the blocked queue.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the blocked queue.

Definition at line 709 of file userFunctions.c.

```
709                     {
710     if(getBlocked()->head == NULL)  {
711         printf("\x1b[32m""\n The Blocked Queue is empty \n""\x1b[0m");
712     }
713     else    {
714          int class, check, state, prior, status;
715          char name[20];
716          char block[] = "\x1B[34m""Blocked Queue: \n""\x1b[0m";
717          char cname[] = "Name: ";
718          char cclass[] = "Class: ";
719          char cstate[] = "State: ";
720          char cstatus[] = "Status: ";
721          char cprior[] = "Priority: ";
722          char line[] = "\n";
723          check = 15;
724
725          sys_req(WRITE, COM1, block, &check );
726
727          PCB* pcb = getBlocked()->head;
728
729          if(pcb->next == NULL) {
730            class = pcb->Process_Class;
731                strcpy(name,pcb->Process_Name);
732                state = pcb->ReadyState;
733                status = pcb->SuspendedState;
734                prior = pcb->Priority;
735
736                printf(cname);
737                printf(name);
738                printf(line);
739
740                printf(cclass);
741                if(pcb->Process_Class == 0)  {
742                  printf("0");
743                }
744                else  {
745                  sys_req(WRITE, COM1, itoa(class), &check);
746                }
747                printf(line);
748
749                printf(cstate);
750                if(pcb->ReadyState == 0)  {
751                  printf("0");
752                }
753                else  {
754                  sys_req(WRITE, COM1, itoa(state), &check);
755                }
756                printf(line);
757
758                printf(cstatus);
759                if(pcb->SuspendedState == 0)  {
760                  printf("0");
761                }
762                else  {
763                  sys_req(WRITE, COM1, itoa(status), &check);
764                }
765                printf(line);
766
767                printf(cprior);
768                if(pcb->Priority == 0)  {
769                  printf("0");
770                  printf("\n\n");
771                }
772                else  {
773                  sys_req(WRITE, COM1, itoa(prior), &check);
774                  printf("\n\n");
775                }
776          }
777          else  {
778            while(pcb != NULL)  {
779                class = pcb->Process_Class;
780                    strcpy(name,pcb->Process_Name);
```

```
781                     state = pcb->ReadyState;
782                     status = pcb->SuspendedState;
783                     prior = pcb->Priority;
784 
785                     printf(cname);
786                     printf(name);
787                     printf(line);
788 
789                     printf(cclass);
790                     if(pcb->Process_Class == 0)  {
791                       printf("0");
792                     }
793                     else  {
794                       sys_req(WRITE, COM1, itoa(class), &check);
795                     }
796                     printf(line);
797 
798                     printf(cstate);
799                     if(pcb->ReadyState == 0)  {
800                       printf("0");
801                     }
802                     else  {
803                       sys_req(WRITE, COM1, itoa(state), &check);
804                     }
805                     printf(line);
806 
807                     printf(cstatus);
808                     if(pcb->SuspendedState == 0)  {
809                       printf("0");
810                     }
811                     else  {
812                       sys_req(WRITE, COM1, itoa(status), &check);
813                     }
814                     printf(line);
815 
816                     printf(cprior);
817                     if(pcb->Priority == 0)  {
818                       printf("0");
819                       printf("\n\n");
820                     }
821                     else  {
822                       sys_req(WRITE, COM1, itoa(prior), &check);
823                       printf("\n\n");
824                     }
825                 pcb = pcb->next;
826             }
827         }
828     }
829 }
```

4.22.1.17 Show_PCB()

```
void Show_PCB (
             char * ProcessName )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of a PCB.

Description: Can except a string as a pointer that is the Process Name. The process name, claas, state, suspend status, and priority of a PCB are displayed. An error check for a valid name occurs.

Parameters

Process_Name	Character pointer that matches the name of process

Definition at line 497 of file userFunctions.c.

```
497                                     {
498     if (FindPCB(ProcessName) == NULL)   {
499         printf("\x1b[31m""\nERROR: PCB does not exist \n""\x1b[0m");
500     }
501     else    {
502         int check = 5;
```

```
503         char name[10];
504         char cname[] = "Name: ";
505         char cclass[] = "Class: ";
506         char cstate[] = "State: ";
507         char cstatus[] = "Status: ";
508         char cprior[] = "Priority: ";
509         char line[] = "\n";
510         PCB* pcb = FindPCB(ProcessName);
511         strcpy(name,pcb->Process_Name);
512         int class = pcb->Process_Class;
513         int state = pcb->ReadyState;
514         int status = pcb->SuspendedState;
515         int prior = pcb->Priority;
516 
517         if(name == NULL){
518             printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
519         }
520         else    {
521             printf(cname);
522             printf(ProcessName);
523             printf(line);
524             printf(cclass);
525             if(pcb->Process_Class == 0)  {
526                 printf("0");
527             }
528             else  {
529                 sys_req(WRITE, COM1, itoa(class), &check);
530             }
531             printf(line);
532             printf(cstate);
533             if(pcb->ReadyState == 0)  {
534                 printf("0");
535             }
536             else  {
537                 sys_req(WRITE, COM1, itoa(state), &check);
538             }
539             printf(line);
540             printf(cstatus);
541             if(pcb->SuspendedState == 0)  {
542                 printf("0");
543             }
544             else  {
545                 sys_req(WRITE, COM1, itoa(status), &check);
546             }
547             printf(line);
548             printf(cprior);
549             if(pcb->Priority == 0)  {
550                 printf("0");
551                 printf("\n\n");
552             }
553             else  {
554                 sys_req(WRITE, COM1, itoa(prior), &check);
555                 printf("\n\n");
556             }
557         }
558     }
559 }
```

4.22.1.18 Show_Ready()

```
void Show_Ready ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the ready queue.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the ready queue.

Definition at line 580 of file userFunctions.c.

```
580                     {
581     if(getReady()->head == NULL)    {
582          printf("\x1b[32m""\n The Ready Queue is empty \n""\x1b[0m");
583     }
584     else    {
585         int class, check, state, prior, status;
586       char name[10];
587       char ready[] = "\x1B[34m""\nReady Queue:\n""\x1B[0m";
```

```
588       char cname[] = "Name: ";
589       char cclass[] = "Class: ";
590       char cstate[] = "State: ";
591       char cstatus[] = "Status: ";
592       char cprior[] = "Priority: ";
593       char line[] = "\n";
594       check = 5;
595
596       sys_req(WRITE, COM1, ready, &check );
597
598       PCB* pcb = getReady()->head;
599
600         if(pcb->next == NULL)   {
601             class = pcb->Process_Class;
602             strcpy(name,pcb->Process_Name);
603             state = pcb->ReadyState;
604             status = pcb->SuspendedState;
605             prior = pcb->Priority;
606
607             printf(cname);
608             printf(name);
609             printf(line);
610
611             printf(cclass);
612             if(pcb->Process_Class == 0)  {
613               printf("0");
614             }
615             else  {
616               sys_req(WRITE, COM1, itoa(class), &check);
617             }
618             printf(line);
619
620             printf(cstate);
621             if(pcb->ReadyState == 0)  {
622               printf("0");
623             }
624             else  {
625               sys_req(WRITE, COM1, itoa(state), &check);
626             }
627             printf(line);
628
629             printf(cstatus);
630             if(pcb->SuspendedState == 0)  {
631               printf("0");
632             }
633             else  {
634               sys_req(WRITE, COM1, itoa(status), &check);
635             }
636             printf(line);
637
638             printf(cprior);
639             if(pcb->Priority == 0)  {
640               printf("0");
641               printf("\n\n");
642             }
643             else  {
644               sys_req(WRITE, COM1, itoa(prior), &check);
645               printf("\n\n");
646             }
647       }
648       else  {
649         while(pcb != NULL)  {
650                 class = pcb->Process_Class;
651                 strcpy(name,pcb->Process_Name);
652                 state = pcb->ReadyState;
653                 status = pcb->SuspendedState;
654                 prior = pcb->Priority;
655
656                 printf(cname);
657                 printf(name);
658                 printf(line);
659
660                 printf(cclass);
661                 if(pcb->Process_Class == 0)  {
662                   printf("0");
663                 }
664                 else  {
665                   sys_req(WRITE, COM1, itoa(class), &check);
666                 }
667                 printf(line);
668
669                 printf(cstate);
670                 if(pcb->ReadyState == 0)  {
671                   printf("0");
672                 }
673                 else  {
674                   sys_req(WRITE, COM1, itoa(state), &check);
```

```
675                 }
676                 printf(line);
677
678                 printf(cstatus);
679                 if(pcb->SuspendedState == 0)  {
680                   printf("0");
681                 }
682                 else  {
683                   sys_req(WRITE, COM1, itoa(status), &check);
684                 }
685                 printf(line);
686
687                 printf(cprior);
688                 if(pcb->Priority == 0)  {
689                   printf("0");
690                   printf("\n\n");
691                 }
692                 else  {
693                   sys_req(WRITE, COM1, itoa(prior), &check);
694                   printf("\n\n");
695                 }
696                 pcb = pcb->next;
697             }
698         }
699     }
700 }
```

4.22.1.19 Suspend()

```
void Suspend (
              char * ProcessName )
```

Brief Description: Places a PCD in the suspended state and reinserts it into the appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Places a PCB in the suspended state and reinserts it into the appropriate queue. An error check for valid Process Name.

Parameters

Process_Name	Character pointer that matches the name of process.

Definition at line 419 of file userFunctions.c.

```
419                                 {
420     PCB* pcb = FindPCB(ProcessName);
421     if (pcb == NULL)    {
422       printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
423     }
424     else {
425         if(pcb->SuspendedState == YES)  {
426             printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
427         }
428         else if(pcb -> Process_Class == APPLICATION)    {
429             pcb->SuspendedState = YES;
430         }
431         else
432             printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
433     }
434 }
```

4.22.1.20 toLowercase()

```
char toLowercase (
              char c )
```

Description: If a letter is uppercase, it changes it to lowercase. (char)

Parameters

c	Character that is to be changed to its lowercase equivalent

Definition at line 301 of file userFunctions.c.

```
301                             {
302     if((c >= 65) && (c <= 90))  {
303         c = c + 32;
304     }
305     return c;
306 }
```

4.22.1.21 Unblock()

```
void Unblock (
            char * ProcessName )
```

Brief Description: Places a PCD in the unblocked state and reinserts it into the correct queue.

Description: Can except a string as a pointer that is the Process Name. The specified PCB will be places in an unblocked state and reinserted into the appropriate queue. An error check for a valid name occurs.

Parameters

Process_Name	Character pointer that matches the name of process.

Definition at line 925 of file userFunctions.c.

```
925                                 {
926   PCB* pcb = FindPCB(ProcessName);
927   if (pcb == NULL)  {
928     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
929   }
930   else {
931     if(pcb->ReadyState == READY)    {
932         printf("\x1b[32m""\nThis Process is already in the READY state \n""\x1b[0m");
933     }
934     else    {
935       RemovePCB(pcb);
936       pcb->ReadyState = READY;
937       InsertPCB(pcb);
938     }
939   }
940 }
```

4.22.1.22 Version()

```
void Version ( )
```

Description: Simply returns a char containing "Version: R(module).(the iteration that module is currently on).

No parameters.

Definition at line 294 of file userFunctions.c.

```
294                 {
295     printf("Version: R4.6 \n");
296 }
```

4.23 modules/sys_proc_loader.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include <core/io.h>
#include "mpx_supt.h"
#include "R1/userFunctions.h"
#include "procsr3.h"
#include "R1/comHand.h"
#include "sys_proc_loader.h"
```

Functions

- void **sysLoader** ()
- void **loadSysProc** (char *name, u32int func, int priority)
- void **InfiniteProc** ()
- void **AlarmProc** ()

4.24 modules/sys_proc_loader.h File Reference

Functions

- void **sysLoader** ()
- void **loadSysProc** (char *name, u32int func, int priority)
- void **InfiniteProc** ()
- void **AlarmProc** ()

Index