Runtime Terror

Generated by Doxygen 1.9.1

Chapter 1

CS_450_RunTime_Terror

R1 Implementation

Bonus assignments included are -Variable Text Color -itoa funciton

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 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**

4.1.1 Detailed Description

Definition at line 32 of file system.h.

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

- mpx_core/include/system.h

4.2 footer Struct Reference

Public Attributes

- header **head**

4.2.1 Detailed Description

Definition at line 18 of file heap.h.

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

- mpx_core/include/mem/heap.h

4.3 gdt_descriptor_struct Struct Reference

Public Attributes

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

4.3.1 Detailed Description

Definition at line 25 of file tables.h.

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

- mpx_core/include/core/tables.h

4.4 gdt_entry_struct Struct Reference

Public Attributes

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

4.4.1 Detailed Description

Definition at line 32 of file tables.h.

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

- mpx_core/include/core/tables.h

4.5 header Struct Reference

Public Attributes

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

4.5.1 Detailed Description

Definition at line 13 of file heap.h.

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

- mpx_core/include/mem/heap.h

4.6 heap Struct Reference

Public Attributes

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

4.6.1 Detailed Description

Definition at line 35 of file heap.h.

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

- mpx_core/include/mem/heap.h

4.7 idt_entry_struct Struct Reference

Public Attributes

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

4.7.1 Detailed Description

Definition at line 8 of file tables.h.

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

- mpx_core/include/core/tables.h

4.8 idt_struct Struct Reference

Public Attributes

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

4.8.1 Detailed Description

Definition at line 18 of file tables.h.

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

- mpx_core/include/core/tables.h

4.9 index_entry Struct Reference

Public Attributes

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

4.9.1 Detailed Description

Definition at line 22 of file heap.h.

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

- mpx_core/include/mem/heap.h

4.10 index_table Struct Reference

Public Attributes

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

4.10.1 Detailed Description

Definition at line 29 of file heap.h.

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

- mpx_core/include/mem/heap.h

4.11 page_dir Struct Reference

Public Attributes

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

4.11.1 Detailed Description

Definition at line 36 of file paging.h.

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

- mpx_core/include/mem/paging.h

4.12 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

4.12.1 Detailed Description

Definition at line 14 of file paging.h.

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

- mpx_core/include/mem/paging.h

4.13 page_table Struct Reference

Public Attributes

- page_entry **pages** [1024]

4.13.1 Detailed Description

Definition at line 28 of file paging.h.

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

- mpx_core/include/mem/paging.h

4.14 param Struct Reference

Public Attributes

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

4.14.1 Detailed Description

Definition at line 33 of file mpx_supt.h.

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

- mpx_core/modules/mpx_supt.h

4.15 PCB Struct Reference

Public Attributes

- unsigned char **stack** [1KMEM]
- unsigned char * **stackTop**
- struct PCB * **prev**
- struct PCB * **next**
- char **Process_Name** [10]
- int **Process_Class**
- int **Priority**
- int **ReadyState**
- int **SuspendedState**

4.15.1 Detailed Description

Definition at line 27 of file PCB.c.

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

- mpx_core/modules/R2/PCB.c

4.16 Queue Struct Reference

Public Attributes

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

4.16.1 Detailed Description

Definition at line 7 of file PCB.c.

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

- mpx_core/modules/R2/PCB.c

4.17 struct Struct Reference

Public Attributes

- int **count**

4.17.1 Detailed Description

Definition at line 9 of file PCB.h.

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

- mpx_core/modules/R2/PCB.h

Chapter 5

File Documentation

5.1 mpx_core/include/core/asm.h File Reference

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

5.2 mpx_core/include/core/interrupts.h File Reference

Functions

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

5.3 mpx_core/include/core/io.h File Reference

Macros

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

5.3.1 Macro Definition Documentation

5.3.1.1 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.

5.4 mpx_core/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)

5.5 mpx_core/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**

5.6 mpx_core/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)

5.7 mpx_core/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)

5.8 mpx_core/include/string.h File Reference

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

Functions

- int isspace (const char *c)

  Description: Determine if a character is whitespace.
- void * memset (void *s, int c, size_t n)

  Description: Set a region of memory.
- char * strcpy (char *s1, const char *s2)

  Description: Copy one string to another.
- char * strcat (char *s1, const char *s2)

  Description: Concatenate the contents of one string onto another.
- int strlen (const char *s)

  Description: Returns the length of a string.
- int strcmp (const char *s1, const char *s2)

  Description: String comparison.
- char * strtok (char *s1, const char *s2)

  Description: Split string into tokens.
- int atoi (const char *s)

  Description: Convert an ASCII string to an integer.

5.8.1 Function Documentation

5.8.1.1 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 }
```

5.8.1.2 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 }
```

5.8.1.3 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 }
```

5.8.1.4 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 }
```

5.8.1.5 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 }
```

5.8.1.6 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 }
```

5.8.1.7 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 }
```

5.8.1.8 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 }
```

5.9 mpx_core/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)

5.10 mpx_core/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 **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]

5.11 mpx_core/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"
```

Functions

- void **kmain** (void)

5.12 mpx_core/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

5.13 mpx_core/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)

5.14 mpx_core/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]

5.15 mpx_core/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

5.16 mpx_core/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**

5.17 mpx_core/lib/string.c File Reference

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

Functions

- int strlen (const char *s)

  Description: Returns the length of a string.
- char * strcpy (char *s1, const char *s2)

  Description: Copy one string to another.
- int atoi (const char *s)

  Description: Convert an ASCII string to an integer.
- int strcmp (const char *s1, const char *s2)

  Description: String comparison.
- char * strcat (char *s1, const char *s2)

  Description: Concatenate the contents of one string onto another.
- int isspace (const char *c)

  Description: Determine if a character is whitespace.
- void * memset (void *s, int c, size_t n)

  Description: Set a region of memory.
- char * strtok (char *s1, const char *s2)

  Description: Split string into tokens.

5.17.1 Function Documentation

5.17.1.1 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 }
```

5.17.1.2 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 }
```

5.17.1.3 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 }
```

5.17.1.4 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 }
```

5.17.1.5 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 }
```

5.17.1.6 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 }
```

5.17.1.7 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 }
```

5.17.1.8 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 }
```

5.18 mpx_core/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** ()

Variables

- param **params**
- int **current_module** = -1
- u32int(* **student_malloc**)(u32int)
- int(* **student_free**)(void *)

5.19 mpx_core/modules/mpx_supt.h File Reference

```
#include <system.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** ()

5.20 mpx_core/modules/R1/comHand.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"
```

Functions

- int comHand ()

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

5.21 mpx_core/modules/R1/comHand.h File Reference

Functions

- int comHand ()

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

5.22 mpx_core/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"
```

Functions

- char * itoa (int num)

  Description: An integer is taken and seperated into individual chars and then all placed into a character array.
- int BCDtoDec (int BCD)

  Description: Changes binary number to decimal numbers.
- int DectoBCD (int Decimal)

  Description: Changes decimal numbers to binary numbers.
- void **printf** (char msg[ ])
- int EdgeCase (char *pointer)

  Description: Compares pointer char to validate if it is a number or not.
- 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).
- void GetTime ()

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

  Description: Returns the full date back to the user in decimal form.
- void Version ()

  Description: Simply returns a char containing "Version: R(module).
- char toLowercase (char c)

  Description: If a letter is uppercase, it changes it to lowercase.
- void Help (char *request)

  Brief Description: Gives helpful information for one of the functions.
- void Suspend (Char *Process_Name)

  Brief Description: Places a PCD in the suspended state and reinserts it into the appropriate queue.
- void Resume (Char *Process_Name)

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

- void Set_Priority (Char *Process_Name, int Priority)

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

- void Show_PCB (char *Process_Name)

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

- void Show_All ()

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

- void Show_Ready ()

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

- void Show_Blocked ()

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

5.22.1 Function Documentation

5.22.1.1 BCDtoDec()

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

Description: Changes binary number to decimal numbers.

Parameters

value	Binary number to be changed to decimal

Definition at line 64 of file userFunctions.c.

```
64                                  {
65          return (((BCD»4)*10) + (BCD & 0xF));
66      }
```

5.22.1.2 DectoBCD()

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

Description: Changes decimal numbers to binary numbers.

Parameters

Decimal	Decimal number to be changed to binary

Definition at line 71 of file userFunctions.c.

```
71                                  {
72          return (((Decimal/10) << 4) | (Decimal % 10));
73      }
```

5.22.1.3 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 83 of file userFunctions.c.

```
83                                      {
84      int valid = 0;
85      if (strcmp(pointer, "00") == 0){
86        valid = 1;
87        return valid;
88      }
89      int i, j;
90      for (i = 0; i < strlen(pointer); i++){
91          valid = 0;
92          for(j = 0; j <= 99; j++){
93              if(strcmp(pointer,itoa(j)) == 0)
94                  valid = 1;
95          }
96          if(valid == 0){
97            return valid;
98          }
99      }
100      return valid;
101    }
```

5.22.1.4 GetDate()

```
void GetDate ( )
```

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

No parameters.

Definition at line 225 of file userFunctions.c.

```
225                         {
226        int check = 2;
227        outb(0x70,0x07);
228          unsigned char day = BCDtoDec(inb(0x71));
229          outb(0x70,0x08);
230          unsigned char month = BCDtoDec(inb(0x71));
231          outb(0x70,0x32);
232          unsigned char millennium = BCDtoDec(inb(0x71));
233          char msg[2] = "-";
234          char msg3[10] = "Date: ";
235          printf(msg3);
236          sys_req(WRITE, COM1, itoa(day), &check);
237          printf(msg);
238          sys_req(WRITE, COM1, itoa(month), &check);
239          printf(msg);
240          sys_req(WRITE, COM1, itoa(millennium), &check);
241      outb(0x70,0x09);
```

```
242     if(BCDtoDec(inb(0x71)) == 0){
243       sys_req(WRITE, COM1, "00", &check);
244     }
245     else {
246         unsigned char year = BCDtoDec(inb(0x71));
247         sys_req(WRITE, COM1, itoa(year), &check);
248     }
249         printf("\n");
250     }
```

5.22.1.5 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 147 of file userFunctions.c.

```
147                     {
148     int check = 2;
149     int hour;
150     int minute;
151     int second;
152         outb(0x70,0x04);
153         unsigned char hours = inb(0x71);
154         outb(0x70,0x02);
155         unsigned char minutes = inb(0x71);
156         outb(0x70,0x00);
157         unsigned char seconds = inb(0x71);
158         char msg1[2] = ":";
159         char msg2[10] = "Time: ";
160         printf(msg2);
161         hour = BCDtoDec(hours);
162         sys_req(WRITE, COM1, itoa(hour), &check);
163         printf(msg1);
164         minute = BCDtoDec(minutes);
165         sys_req(WRITE, COM1, itoa(minute), &check);
166         printf(msg1);
167         second = BCDtoDec(seconds);
168         sys_req(WRITE, COM1, itoa(second), &check);
169       printf("\n");
170     }
```

5.22.1.6 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 274 of file userFunctions.c.

```
274                             {
275     int check = 1;
276         if (request[0] == '\0') {
277                 printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
278             printf("\n getDate \n setDate \n getTime \n setTime \n version \n shutdown \n\n");
279         }
280         else if (strcmp(request, "GetDate") == 0) {
281                 printf("\n getDate returns the current date that is loaded onto the operating
    system.\n");
282         }
283         else if (strcmp(request, "SetDate") == 0) {
284                 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");
285         }
286         else if (strcmp(request, "GetTime") == 0) {
287                 printf("\n getTime returns the current time as hours, minutes, seconds that is loaded
    onto the operating system.\n");
288         }
289         else if (strcmp(request, "SetTime") == 0) {
290                 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");
291              }
292         else if (strcmp(request, "Version") == 0) {
293                 printf("\n version returns the current operating software version that the system is
    running.\n");
294         }
295      else if(strcmp(request, "shutdown") == 0) {
296        printf("\n shutdown shuts down the system.\n");
297      }
298
299 /************************R2 Commands*******************************************************/
300      else if(strcmp(request,"suspend") == 0) {
301      printf("\n Suspend takes in the name of a PCB then places it into the suspended state and reinserts
    it into the correct queue.\n");
302      }
303      else if(strcmp(FirstToken,"resume") == 0) {
304      printf("\n Resume takes in the name of a PCB then removes it from the suspended state and adds it to
    the correct queue.\n");
305      }
306      else if(strcmp(FirstToken,"setPriority") == 0) {
307      printf("\n SetPriority takes in the name of a PCB and the priority it needs to be set to then
    reinstates the specified PCB into a new location by priority.\n");
308      }
309      else if(strcmp(FirstToken,"showPCB") == 0) {
310      printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
    user.\n");
311      }
312      else if(strcmp(FirstToken,"showAll") == 0) {
313      printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
    queues.\n");
314      }
315      else if(strcmp(FirstToken,"showReady") == 0) {
316      printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
    currently are in the ready state.\n");
317      }
318      else if(strcmp(FirstToken,"showBlocked") == 0) {
319      printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
    currently are in the blocked state.\n");
320      }
321
322 /******************************** R2 Temp Commands
    *******************************************************/
323      else if(strcmp(FirstToken,"createPCB") == 0) {
324      printf("\n  CreatePCB takes in the process_name, process_class, and process_priority. Then assigns
    this new process into the correct queue.\n");
325      }
326      else if(strcmp(FirstToken,"deletePCB") == 0) {
327      printf("\n DeletePCB takes in the process_name then deletes it from the queue and free's all the
    memory that was previously allocated to the specified PCB.\n");
328      }
329      else if(strcmp(FirstToken,"block") == 0) {
330      printf("\n Block takes in the process_name then sets it's state to blocked and reinserts it back
    into the correct queue.\n");
331      }
332      else if(strcmp(FirstToken,"unblock") == 0) {
333      printf("\n Unblock takes in the process_name then sets it's state to ready and reinserts it back
    into the correct queue.\n");
334      }
335      else    {
336       printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
    full list of commands.\n""\x1b[0m");
337      }
338 }
```

5.22.1.7 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 33 of file userFunctions.c.

```
34     {
35             int i,j,k,count;
36             i = num;
37             j = 0;
38             count = 0;
39         while(i){ // count number of digits
40             count++;
41             i /= 10;
42         }
43
44         char* arr1;
45         char arr2[count];
46         arr1 = (char*)sys_alloc_mem(count); //memory allocation
47
48         while(num){ // seperate last digit from number and add ASCII
49             arr2[++j] = num%10 + '0';
50             num /= 10;
51         }
52
53         for(k = 0; k < j; k++){ // reverse array results
54             arr1[k] = arr2[j-k];
55         }
56         arr1[k] = '\0';
57
58         return(char*)arr1;
59     }
```

5.22.1.8 Resume()

```
void Resume (
            Char * Process_Name )
```

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 371 of file userFunctions.c.

```
371                                 {
372
373
374   // Name Error check
375   // Error check (Valid Name)
376   //if (Process_Name != valid name){
377   //  printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
378   //}
379
380 }
```

5.22.1.9 Set_Priority()

```
void Set_Priority (
             Char * Process_Name,
             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 388 of file userFunctions.c.

```
388                                                     {
389   int i;
390
391   // Name Error check
392   // Error check (Valid Name)
393   //if (Process_Name != valid name){
394   //  printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
395   //}
396   // Priority error check
397   for(i = 0; i < 9; i++){
398     if(Priority == i){
399       break;
400     }
401     else{
402       printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m")
403     }
404   }
405
406 }
```

5.22.1.10 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 178 of file userFunctions.c.

```
178                                                                 {
179     outb(0x70,0x07);
180     int tempDay = BCDtoDec(inb(0x71));
181     outb(0x70,0x08);
182     int tempMonth = BCDtoDec(inb(0x71));
183     outb(0x70,0x32);
184     int tempMillennium = BCDtoDec(inb(0x71));
185     outb(0x70,0x09);
186     int tempYear = BCDtoDec(inb(0x71));
187     cli();
188         outb(0x70,0x07);
189         outb(0x71,DectoBCD (day));
190         outb(0x70,0x08);
191         outb(0x71,DectoBCD (month));
192         outb(0x70,0x32);
193         outb(0x71,DectoBCD (millennium));
194         outb(0x70,0x09);
195         outb(0x71,DectoBCD (year));
196         sti();
197     outb(0x70,0x07);
198     unsigned char newDay = BCDtoDec(inb(0x71));
199     outb(0x70,0x08);
200     unsigned char newMonth = BCDtoDec(inb(0x71));
201     outb(0x70,0x32);
202     unsigned char newMillennium = BCDtoDec(inb(0x71));
203     outb(0x70,0x09);
204     unsigned char newYear = BCDtoDec(inb(0x71));
205     if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
206       printf("Your input was invalid\n");
207       cli();
208         outb(0x70,0x07);
209         outb(0x71,DectoBCD (tempDay));
210         outb(0x70,0x08);
211         outb(0x71,DectoBCD (tempMonth));
212         outb(0x70,0x32);
213         outb(0x71,DectoBCD (tempMillennium));
214         outb(0x70,0x09);
215         outb(0x71,DectoBCD (tempYear));
216         sti();
217     }
218     else
219       printf("Date Set\n");
220     }
```

5.22.1.11 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 108 of file userFunctions.c.

```
108                                                     {
109     outb(0x70,0x04);
110     unsigned char tempHours = BCDtoDec(inb(0x71));
111     outb(0x70,0x02);
112     unsigned char tempMinutes = BCDtoDec(inb(0x71));
113     outb(0x70,0x00);
114     unsigned char tempSeconds = BCDtoDec(inb(0x71));
115         cli(); //outb(device + 1, 0x00); //disable interrupts
116         outb(0x70,0x04);
117         outb(0x71, DectoBCD(hours));// change to bcd
118         outb(0x70,0x02);
119         outb(0x71, DectoBCD(minutes));
120         outb(0x70,0x00);
121         outb(0x71, DectoBCD(seconds));
122         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
123     outb(0x70,0x04);
124     unsigned char newHours = BCDtoDec(inb(0x71));
125     outb(0x70,0x02);
126     unsigned char newMinutes = BCDtoDec(inb(0x71));
127     outb(0x70,0x00);
128     unsigned char newSeconds = BCDtoDec(inb(0x71));
129     if(newHours != hours || newMinutes != minutes || newSeconds != seconds){
130       printf("Your input was invalid\n");
131       cli(); //outb(device + 1, 0x00); //disable interrupts
132         outb(0x70,0x04);
133         outb(0x71, DectoBCD(tempHours));// change to bcd
134         outb(0x70,0x02);
135         outb(0x71, DectoBCD(tempMinutes));
136         outb(0x70,0x00);
137         outb(0x71, DectoBCD(tempSeconds));
138         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
139     }
140     else
141       printf("Time Set\n");
142     }
```

5.22.1.12 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 439 of file userFunctions.c.

```
439                 {
440   int check = 20;
441   int i;
442   int j;
443   for(i = 0; i < sizeof(ready queue);i++)   {
444     char rProcess_Name = ready queue [i] Process_Name;
445     int rClass =  ready queue [i] class;
446     char rState = ready queue[i] state;
447     char rStatus = ready queue[i] status;
448     int rPriority = ready queue[i] priority;
449     sys_req(WRITE, COM1, rProcess_Name, &check);
450     sys_req(WRITE, COM1, itoa(rClass), &check);
451     sys_req(WRITE, COM1, rState, &check);
452     sys_req(WRITE, COM1, rStatus, &check);
453     sys_req(WRITE, COM1, itoa(rPriority), &check);
454   }
455   for(j = 0; j < sizeof(blocked queue); j++){
456     char bProcess_Name = blocked queue [j] Process_Name;
457     int bClass =  blocked queue [j] class;
458     char bState = blocked queue[j] state;
459     char bStatus = blocked queue[j] status;
460     int bPriority = blocked queue[j] priority;
461     sys_req(WRITE, COM1, bProcess_Name, &check);
462     sys_req(WRITE, COM1, itoa(bClass), &check);
463     sys_req(WRITE, COM1, bState, &check);
464     sys_req(WRITE, COM1, bStatus, &check);
465     sys_req(WRITE, COM1, itoa(bPriority), &check);
466   }
467 }
```

5.22.1.13 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.
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.

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.

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.

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 492 of file userFunctions.c.

5.22.1.14 Show_PCB()

```
void Show_PCB (
            char * Process_Name )
```

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 413 of file userFunctions.c.

```
413                                     {
414   int class, check, state, prior;
415   char[] name;
416   check = 10;
417   PCB* pcb = FindPCB(Process_Name);
418   class = pcb->Process_Class;
419   name = pcb->Process_Name;
420   state = pcb->ReadyState;
421   status = pcb->SuspendedState;
422   prior = pcb->Priority;
423
424   if(name == NULL){
425     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
426   } else{
427     sys_req(WRITE, COM1, name, &check);
428     sys_req(WRITE, COM1, itoa(class), &check);
429     sys_req(WRITE, COM1, itoa(state), &check);
430     sys_req(WRITE, COM1, itoa(status), &check);
431     sys_req(WRITE, COM1, itoa(priot), &check);
432 }
433 }
```

5.22.1.15 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 472 of file userFunctions.c.

```
472                       {
473   int check = 20;
474   int i;
475   for(i = 0; i < sizeof(ready queue);i++)   {
476     char Process_Name = ready queue [i] Process_Name;
477     char Class =  ready queue [i] class;
478     char State = ready queue[i] state;
479     char Status = ready queue[i] status;
480     char Priority = ready queue[i] priority;
481     sys_req(WRITE, COM1, Process_Name, &check);
482     sys_req(WRITE, COM1, Class, &check);
483     sys_req(WRITE, COM1, State, &check);
484     sys_req(WRITE, COM1, Status, &check);
485     sys_req(WRITE, COM1, Priority, &check);
486   }
487 }
```

5.22.1.16 Suspend()

```
void Suspend (
            Char * Process_Name )
```

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 356 of file userFunctions.c.

```
356                                     {
357
358   // Name Error check
359   // Error check (Valid Name)
360   //if (Process_Name != valid name){
361   //  printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
362   //}
363
364 }
```

5.22.1.17 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 262 of file userFunctions.c.

```
262                                     {
263         if((c >= 65) && (c <= 90))  {
264                 c = c + 32;
265             }
266         return c;
267         }
```

5.22.1.18 Version()

```
void Version ( )
```

Description: Simply returns a char containing "Version: R(module).

(the iteration that module is currently on).

No parameters.

Definition at line 255 of file userFunctions.c.

```
255                     {
256         printf("Version: R2.0 \n");
257     }
```

5.23 mpx_core/modules/R1/userFunctions.h File Reference

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"

Functions

- 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).
- void GetTime ()

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

  Description: Changes decimal numbers to binary numbers.
- char ∗ itoa (int num)

  Description: An integer is taken and seperated into individual chars and then all placed into a character array.
- 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).
- int BCDtoDec (int BCD)

  Description: Changes binary number to decimal numbers.
- void GetDate ()

  Description: Returns the full date back to the user in decimal form.
- void Version ()

  Description: Simply returns a char containing "Version: R(module).
- void Help (char ∗request)

  Brief Description: Gives helpful information for one of the functions.
- void **printf** (char msg[ ])
- int EdgeCase (char ∗pointer)

  Description: Compares pointer char to validate if it is a number or not.
- char toLowercase (char c)

  Description: If a letter is uppercase, it changes it to lowercase.
- void Suspend (Char ∗Process_Name)

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

- void Resume (Char *Process_Name)

  Brief Description: Places a PCD in the not suspended state and reinserts it into the appropriate queue.
- void Set_Priority (Char *Process_Name, int Priority)

  Brief Description: Sets PCB priority and reinserts the process into the correct place in the correct queue.
- void **Show_PCB** (Char *Process_Name)
- void Show_All ()

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

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

  Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the blocked queue.
- void **Create_PCB** (char *Process_Name, int Priority, int Class)
- void **Delete_PCB** (Char *Process_Name)
- void **Block** (Char *Process_Name)
- void **Unblock** (Char *Process_Name)

5.23.1 Function Documentation

5.23.1.1 BCDtoDec()

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

Description: Changes binary number to decimal numbers.

Parameters

value	Binary number to be changed to decimal

Definition at line 64 of file userFunctions.c.

```
64                              {
65         return (((BCD»4)*10) + (BCD & 0xF));
66     }
```

5.23.1.2 DectoBCD()

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

Description: Changes decimal numbers to binary numbers.

Parameters

Decimal	Decimal number to be changed to binary

Definition at line 71 of file userFunctions.c.

```
71                                  {
72          return (((Decimal/10) « 4) | (Decimal % 10));
73      }
```

5.23.1.3 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 83 of file userFunctions.c.

```
83                                      {
84      int valid = 0;
85      if (strcmp(pointer, "00") == 0){
86        valid = 1;
87        return valid;
88      }
89      int i, j;
90      for (i = 0; i < strlen(pointer); i++){
91          valid = 0;
92          for(j = 0; j <= 99; j++){
93              if(strcmp(pointer,itoa(j)) == 0)
94                  valid = 1;
95          }
96          if(valid == 0){
97            return valid;
98          }
99      }
100      return valid;
101    }
```

5.23.1.4 GetDate()

```
void GetDate ( )
```

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

No parameters.

Definition at line 225 of file userFunctions.c.

```
225                         {
226        int check = 2;
227        outb(0x70,0x07);
228          unsigned char day = BCDtoDec(inb(0x71));
229          outb(0x70,0x08);
230          unsigned char month = BCDtoDec(inb(0x71));
231          outb(0x70,0x32);
232          unsigned char millennium = BCDtoDec(inb(0x71));
```

```
233         char msg[2] = "-";
234         char msg3[10] = "Date: ";
235         printf(msg3);
236         sys_req(WRITE, COM1, itoa(day), &check);
237         printf(msg);
238         sys_req(WRITE, COM1, itoa(month), &check);
239         printf(msg);
240         sys_req(WRITE, COM1, itoa(millennium), &check);
241     outb(0x70,0x09);
242     if(BCDtoDec(inb(0x71)) == 0){
243       sys_req(WRITE, COM1, "00", &check);
244     }
245     else {
246         unsigned char year = BCDtoDec(inb(0x71));
247         sys_req(WRITE, COM1, itoa(year), &check);
248     }
249         printf("\n");
250     }
```

5.23.1.5 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 147 of file userFunctions.c.

```
147                     {
148     int check = 2;
149     int hour;
150     int minute;
151     int second;
152         outb(0x70,0x04);
153         unsigned char hours = inb(0x71);
154         outb(0x70,0x02);
155         unsigned char minutes = inb(0x71);
156         outb(0x70,0x00);
157         unsigned char seconds = inb(0x71);
158         char msg1[2] = ":";
159         char msg2[10] = "Time: ";
160         printf(msg2);
161         hour = BCDtoDec(hours);
162         sys_req(WRITE, COM1, itoa(hour), &check);
163         printf(msg1);
164         minute = BCDtoDec(minutes);
165         sys_req(WRITE, COM1, itoa(minute), &check);
166         printf(msg1);
167         second = BCDtoDec(seconds);
168         sys_req(WRITE, COM1, itoa(second), &check);
169       printf("\n");
170     }
```

5.23.1.6 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 274 of file userFunctions.c.

```
274                                {
275     int check = 1;
276         if (request[0] == '\0') {
277                 printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
278             printf("\n getDate \n setDate \n getTime \n setTime \n version \n shutdown \n\n");
279         }
280         else if (strcmp(request, "GetDate") == 0) {
281                 printf("\n getDate returns the current date that is loaded onto the operating
    system.\n");
282         }
283         else if (strcmp(request, "SetDate") == 0) {
284                 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");
285         }
286         else if (strcmp(request, "GetTime") == 0) {
287                 printf("\n getTime returns the current time as hours, minutes, seconds that is loaded
    onto the operating system.\n");
288         }
289         else if (strcmp(request, "SetTime") == 0) {
290                 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");
291             }
292         else if (strcmp(request, "Version") == 0) {
293                 printf("\n version returns the current operating software version that the system is
    running.\n");
294         }
295      else if(strcmp(request, "shutdown") == 0) {
296        printf("\n shutdown shuts down the system.\n");
297      }
298
299 /************************R2 Commands*******************************************************/
300     else if(strcmp(request,"suspend") == 0) {
301     printf("\n Suspend takes in the name of a PCB then places it into the suspended state and reinserts
    it into the correct queue.\n");
302     }
303     else if(strcmp(FirstToken,"resume") == 0) {
304     printf("\n Resume takes in the name of a PCB then removes it from the suspended state and adds it to
    the correct queue.\n");
305     }
306     else if(strcmp(FirstToken,"setPriority") == 0) {
307     printf("\n SetPriority takes in the name of a PCB and the priority it needs to be set to then
    reinstates the specified PCB into a new location by priority.\n");
308     }
309     else if(strcmp(FirstToken,"showPCB") == 0) {
310     printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
    user.\n");
311     }
312     else if(strcmp(FirstToken,"showAll") == 0) {
313     printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
    queues.\n");
314     }
315     else if(strcmp(FirstToken,"showReady") == 0) {
316     printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
    currently are in the ready state.\n");
317     }
318     else if(strcmp(FirstToken,"showBlocked") == 0) {
319     printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
    currently are in the blocked state.\n");
320     }
321
322 /********************************** R2 Temp Commands
    *********************************************************/
323     else if(strcmp(FirstToken,"createPCB") == 0) {
324     printf("\n  CreatePCB takes in the process_name, process_class, and process_priority. Then assigns
    this new process into the correct queue.\n");
325     }
326     else if(strcmp(FirstToken,"deletePCB") == 0) {
327     printf("\n DeletePCB takes in the process_name then deletes it from the queue and free's all the
    memory that was previously allocated to the specified PCB.\n");
328     }
329     else if(strcmp(FirstToken,"block") == 0) {
330     printf("\n Block takes in the process_name then sets it's state to blocked and reinserts it back
    into the correct queue.\n");
331     }
332     else if(strcmp(FirstToken,"unblock") == 0) {
333     printf("\n Unblock takes in the process_name then sets it's state to ready and reinserts it back
    into the correct queue.\n");
```

```
334     }
335     else    {
336      printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
       full list of commands.\n""\x1b[0m");
337     }
338 }
```

5.23.1.7 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 33 of file userFunctions.c.

```
34     {
35             int i,j,k,count;
36             i = num;
37             j = 0;
38             count = 0;
39         while(i){ // count number of digits
40             count++;
41             i /= 10;
42         }
43
44         char* arr1;
45         char arr2[count];
46         arr1 = (char*)sys_alloc_mem(count); //memory allocation
47
48         while(num){ // seperate last digit from number and add ASCII
49             arr2[++j] = num%10 + '0';
50             num /= 10;
51         }
52
53         for(k = 0; k < j; k++){ // reverse array results
54             arr1[k] = arr2[j-k];
55         }
56         arr1[k] = '\0';
57
58         return(char*)arr1;
59     }
```

5.23.1.8 Resume()

```
void Resume (
              Char * Process_Name )
```

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 371 of file userFunctions.c.

```
371                                 {
372
373
374   // Name Error check
375   // Error check (Valid Name)
376   //if (Process_Name != valid name){
377   //  printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
378   //}
379
380 }
```

5.23.1.9 Set_Priority()

```
void Set_Priority (
            Char * Process_Name,
            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 388 of file userFunctions.c.

```
388                                                 {
389   int i;
390
391   // Name Error check
392   // Error check (Valid Name)
393   //if (Process_Name != valid name){
394   //  printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
395   //}
396   // Priority error check
397   for(i = 0; i < 9; i++){
398     if(Priority == i){
399       break;
400     }
401     else{
402       printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m")
403     }
404   }
405
406 }
```

5.23.1.10 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 178 of file userFunctions.c.

```
178                                                                 {
179     outb(0x70,0x07);
180     int tempDay = BCDtoDec(inb(0x71));
181     outb(0x70,0x08);
182     int tempMonth = BCDtoDec(inb(0x71));
183     outb(0x70,0x32);
184     int tempMillennium = BCDtoDec(inb(0x71));
185     outb(0x70,0x09);
186     int tempYear = BCDtoDec(inb(0x71));
187     cli();
188         outb(0x70,0x07);
189         outb(0x71,DectoBCD (day));
190         outb(0x70,0x08);
191         outb(0x71,DectoBCD (month));
192         outb(0x70,0x32);
193         outb(0x71,DectoBCD (millennium));
194         outb(0x70,0x09);
195         outb(0x71,DectoBCD (year));
196         sti();
197     outb(0x70,0x07);
198     unsigned char newDay = BCDtoDec(inb(0x71));
199     outb(0x70,0x08);
200     unsigned char newMonth = BCDtoDec(inb(0x71));
201     outb(0x70,0x32);
202     unsigned char newMillennium = BCDtoDec(inb(0x71));
203     outb(0x70,0x09);
204     unsigned char newYear = BCDtoDec(inb(0x71));
205     if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
206       printf("Your input was invalid\n");
207       cli();
208         outb(0x70,0x07);
209         outb(0x71,DectoBCD (tempDay));
210         outb(0x70,0x08);
211         outb(0x71,DectoBCD (tempMonth));
212         outb(0x70,0x32);
213         outb(0x71,DectoBCD (tempMillennium));
214         outb(0x70,0x09);
215         outb(0x71,DectoBCD (tempYear));
216         sti();
217     }
218     else
219       printf("Date Set\n");
220     }
```

5.23.1.11 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 108 of file userFunctions.c.

```
108                                                 {
109     outb(0x70,0x04);
110     unsigned char tempHours = BCDtoDec(inb(0x71));
111     outb(0x70,0x02);
112     unsigned char tempMinutes = BCDtoDec(inb(0x71));
113     outb(0x70,0x00);
114     unsigned char tempSeconds = BCDtoDec(inb(0x71));
115         cli(); //outb(device + 1, 0x00); //disable interrupts
116         outb(0x70,0x04);
117         outb(0x71, DectoBCD(hours));// change to bcd
118         outb(0x70,0x02);
119         outb(0x71, DectoBCD(minutes));
120         outb(0x70,0x00);
121         outb(0x71, DectoBCD(seconds));
122         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
123     outb(0x70,0x04);
124     unsigned char newHours = BCDtoDec(inb(0x71));
125     outb(0x70,0x02);
126     unsigned char newMinutes = BCDtoDec(inb(0x71));
127     outb(0x70,0x00);
128     unsigned char newSeconds = BCDtoDec(inb(0x71));
129     if(newHours != hours || newMinutes != minutes || newSeconds != seconds){
130       printf("Your input was invalid\n");
131       cli(); //outb(device + 1, 0x00); //disable interrupts
132         outb(0x70,0x04);
133         outb(0x71, DectoBCD(tempHours));// change to bcd
134         outb(0x70,0x02);
135         outb(0x71, DectoBCD(tempMinutes));
136         outb(0x70,0x00);
137         outb(0x71, DectoBCD(tempSeconds));
138         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
139     }
140     else
141       printf("Time Set\n");
142     }
```

5.23.1.12 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 439 of file userFunctions.c.

```
439                 {
440   int check = 20;
441   int i;
442   int j;
443   for(i = 0; i < sizeof(ready queue);i++)    {
444     char rProcess_Name = ready queue [i] Process_Name;
445     int rClass =  ready queue [i] class;
446     char rState = ready queue[i] state;
447     char rStatus = ready queue[i] status;
448     int rPriority = ready queue[i] priority;
449     sys_req(WRITE, COM1, rProcess_Name, &check);
450     sys_req(WRITE, COM1, itoa(rClass), &check);
451     sys_req(WRITE, COM1, rState, &check);
452     sys_req(WRITE, COM1, rStatus, &check);
453     sys_req(WRITE, COM1, itoa(rPriority), &check);
454   }
```

```
455   for(j = 0; j < sizeof(blocked queue); j++){
456     char bProcess_Name = blocked queue [j] Process_Name;
457     int bClass =  blocked queue [j] class;
458     char bState = blocked queue[j] state;
459     char bStatus = blocked queue[j] status;
460     int bPriority = blocked queue[j] priority;
461     sys_req(WRITE, COM1, bProcess_Name, &check);
462     sys_req(WRITE, COM1, itoa(bClass), &check);
463     sys_req(WRITE, COM1, bState, &check);
464     sys_req(WRITE, COM1, bStatus, &check);
465     sys_req(WRITE, COM1, itoa(bPriority), &check);
466   }
467 }
```

5.23.1.13 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.
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.

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.

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.

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 492 of file userFunctions.c.

5.23.1.14 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 472 of file userFunctions.c.

```
472                     {
473   int check = 20;
474   int i;
475   for(i = 0; i < sizeof(ready queue);i++)   {
476     char Process_Name = ready queue [i] Process_Name;
477     char Class =  ready queue [i] class;
478     char State = ready queue[i] state;
479     char Status = ready queue[i] status;
480     char Priority = ready queue[i] priority;
481     sys_req(WRITE, COM1, Process_Name, &check);
482     sys_req(WRITE, COM1, Class, &check);
483     sys_req(WRITE, COM1, State, &check);
484     sys_req(WRITE, COM1, Status, &check);
485     sys_req(WRITE, COM1, Priority, &check);
486   }
487 }
```

5.23.1.15 Suspend()

```
void Suspend (
             Char * Process_Name )
```

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 356 of file userFunctions.c.

```
356                                     {
357
```

```
358   // Name Error check
359   // Error check (Valid Name)
360   //if (Process_Name != valid name){
361   //  printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
362   //}
363
364 }
```

5.23.1.16 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 262 of file userFunctions.c.

```
262                                 {
263         if((c >= 65) && (c <= 90))  {
264                 c = c + 32;
265             }
266         return c;
267         }
```

5.23.1.17 Version()

```
void Version ( )
```

Description: Simply returns a char containing "Version: R(module).

(the iteration that module is currently on).

No parameters.

Definition at line 255 of file userFunctions.c.

```
255                     {
256         printf("Version: R2.0 \n");
257     }
```

Index