Circle x PegasOS Dependencies

Table of Contents

How to Use This Document

Preface

This document is, quite simply, a list of every single header file from Circle that PegasOS is using in its kernel. Each header file has its own page, with its own diagram and summary, showing what files that header depends on. This is to help us get an idea of where dependencies lie within the system, as well as to try and get a grasp on the system spaghetti that's being cooked here.

This document is, as you can tell, quite long as a result of all this. I'm doing my best to keep the flow of it somewhat sensical. I plan on adding hyperlinks to each page so that you can jump down the tree of dependencies.

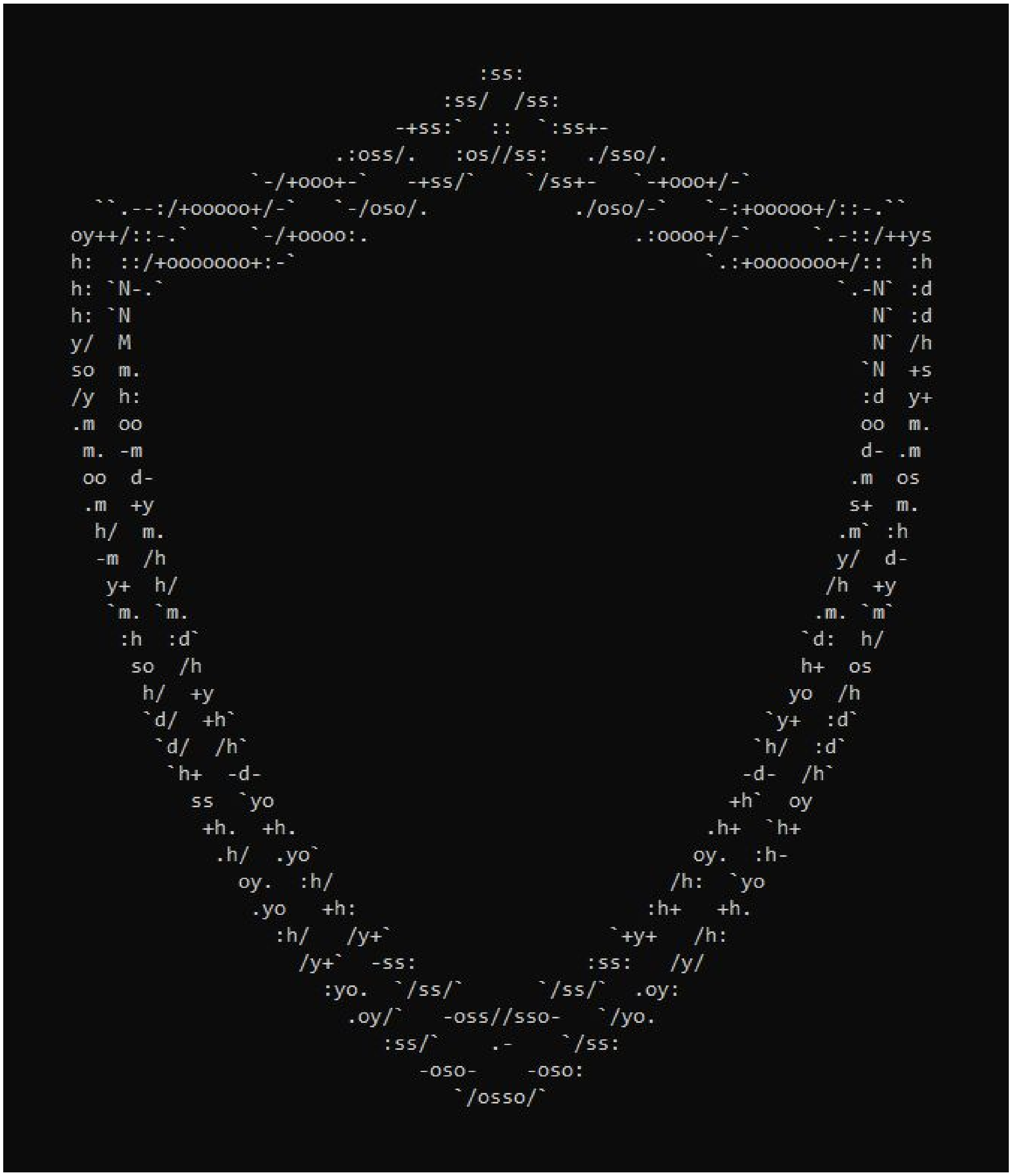
Dependency Graphs are shown as what I will call "dependencies down" or in other words, what header files *File A* needs to function. It will not show what I will call "dependencies up" or in other words, what header files rely on *File A* to function. In an effort to keep the document from being muddied down, this document will only contain "dependencies down".

The USB Dependencies will be finished at a later date.

How To Use

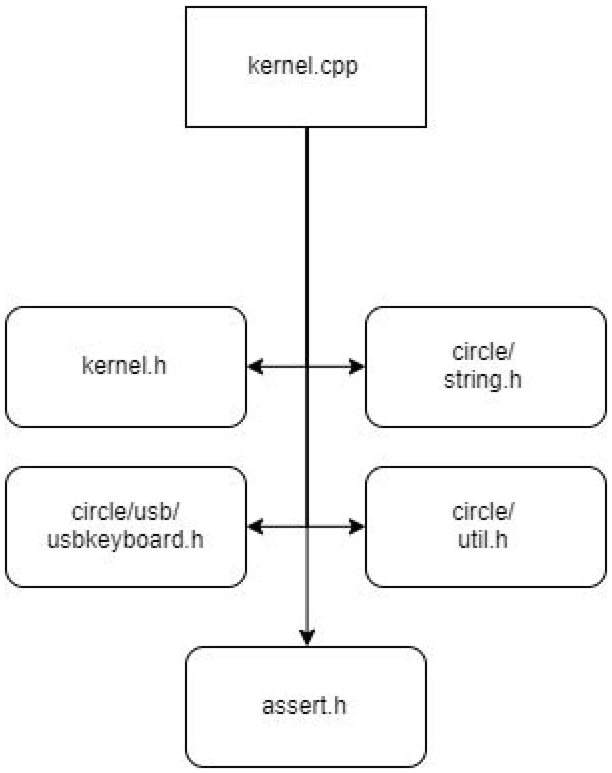
To use the document, locate the header file in question in the table of contents above. These will be hyperlinked to the appropriate header pages, and each page will contain a link back up to the table of contents.

Kernel Dependencies

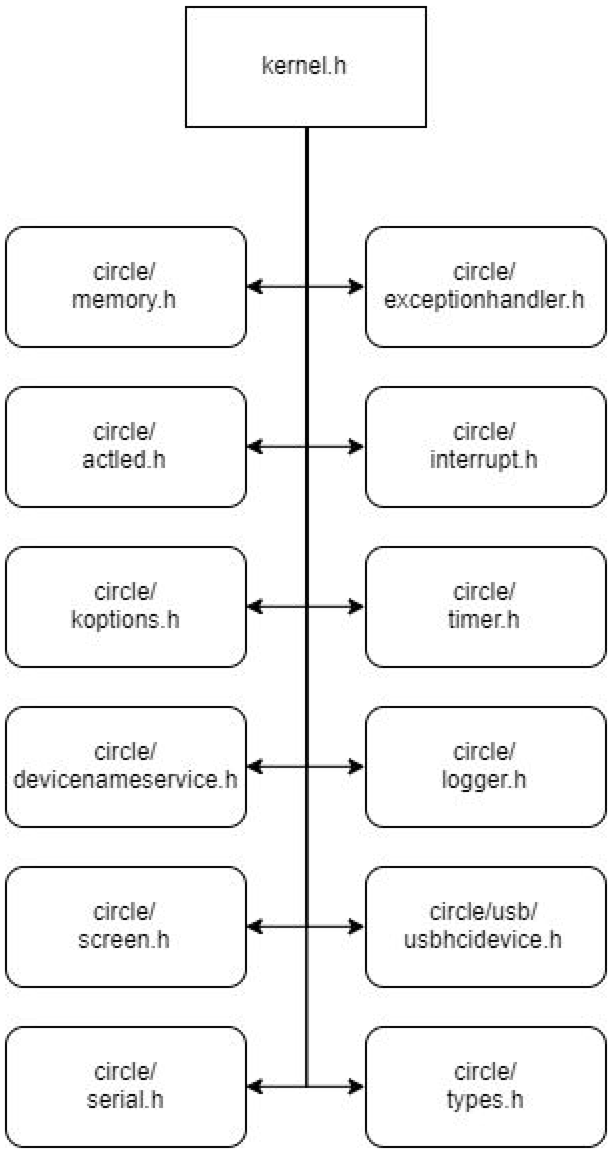


```
                                  :ss:
                               :ss/  /ss:
                           -+ss:`  ::  `:ss+-
                      .:oss/.   :os//ss:   ./sso/.
               `-/+ooo+-`   -+ss/`    `/ss+-   `-+ooo+/-`
  ``.--:/+ooooo+/-`   `-/oso/.            ./oso/-`   `-:+ooooo+/::-.``
oy++/::-.`     `-/+oooo:.                      .:oooo+/-`     `.-::/++ys
h:  ::/+ooooooo+:-`                                  `.:+ooooooo+/::  :h
h: `N-.`                                                        `.-N` :d
h: `N                                                              N` :d
y/  M                                                              N` /h
so  m.                                                            `N  +s
/y  h:                                                            :d  y+
.m  oo                                                            oo  m.
 m. -m                                                            d- .m
 oo  d-                                                          .m  os
 .m  +y                                                          s+  m.
  h/  m.                                                        .m` :h
  -m  /h                                                        y/  d-
   y+  h/                                                      /h  +y
   `m. `m.                                                    .m. `m`
    :h  :d`                                                  `d:  h/
     so  /h                                                  h+  os
      h/  +y                                                yo  /h
      `d/  +h`                                            `y+  :d`
       `d/  /h`                                          `h/  :d`
        `h+  -d-                                        -d-  /h`
          ss  `yo                                      +h`  oy
           +h.  +h.                                  .h+  `h+
            .h/  .yo`                               oy.  :h-
              oy.  :h/                            /h:  `yo
               .yo   +h:                        :h+   +h.
                 :h/   /y+`                  `+y+   /h:
                   /y+`  -ss:              :ss:   /y/
                     :yo.  `/ss/`      `/ss/`  .oy:
                       .oy/`   -oss//sso-   `/yo.
                          :ss/`    .-    `/ss:
                             -oso-    -oso:
                                `/osso/`
```

kernel.cpp

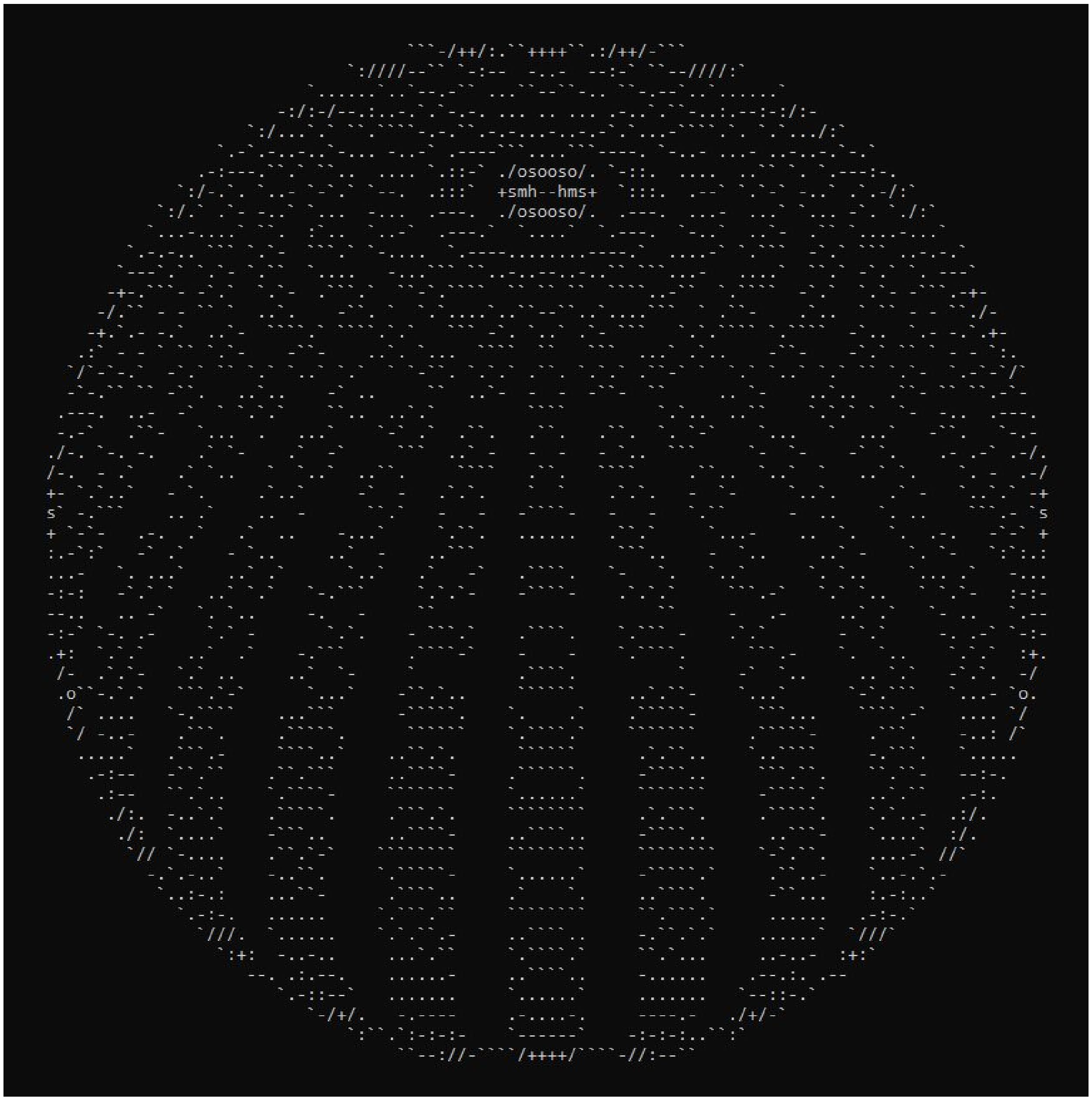

kernel.cpp
kernel.h
circle/ string.h
circle/usb/ usbkeyboard.h
circle/ util.h
assert.h

kernel.h

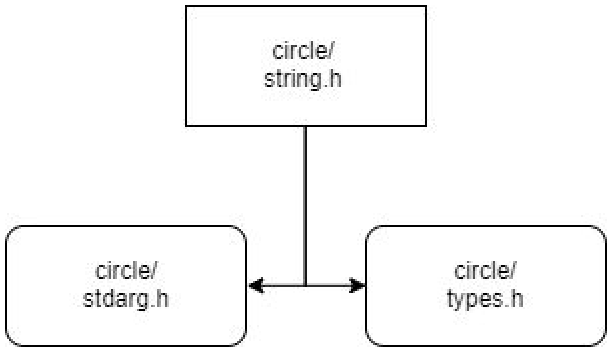


Summary:
Includes all of the core systems that the kernel needs to operate. The kernel image that PegasOS is based off of/extending is the USB keyboard example, which as is apparent in the above dependency graph (as well as the contents of this document) does not need the entire capabilities of the Circle library.

Circle Dependencies

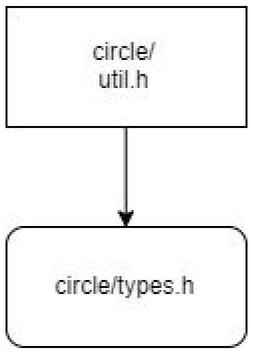


circle/string.h



Summary:
Defines a string class in Circle, called *CString*. It contains basic string functions, including: *Append*, *Compare*, *Find*, *Replace*, and *Format*. Also defines the buffer and size for the string in question.

circle/util.h

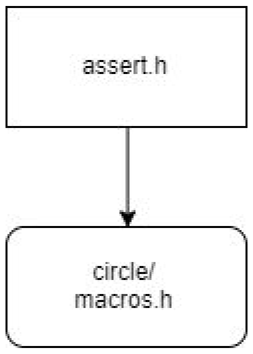


Summary:
Defines a number of ubiquitous low-level C functions (listed below). Also defines some bitswap keywords and functions.

Function List:

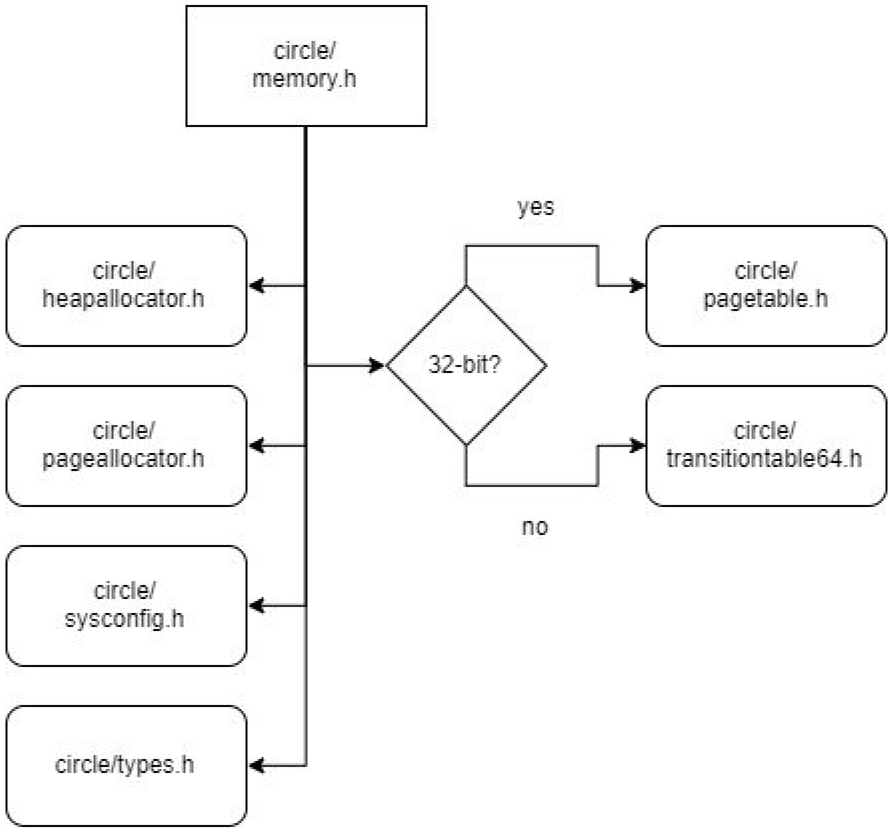
- memset()
- memcpy()
- memmove()
- memcmp()
- strlen()
- strcmp()
- strcasecmp()
- strncmp()
- strncasecmp()
- strcpy()
- strncpy()
- strcat()
- strchr()
- strstr()
- strtok_r()
- strtoul()
- strtoull()
- atoi()
- char2int()

assert.h



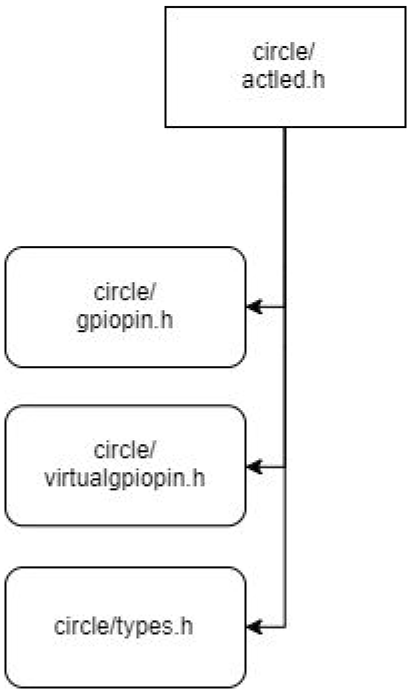
Summary:
Defines an *assert()* macro that appears to only be active when Circle is in C++ mode and when not in debugging mode.

circle/memory.h



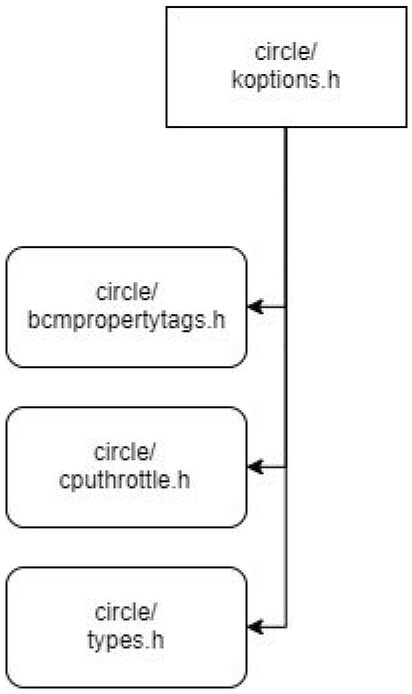
Summary:
Defines a memory management class *CMemorySystem*. Activates more functionality for bigger heap allocation when Circle is set to RASPPI >= 4 in the *Rules.mk* file.

circle/actled.h



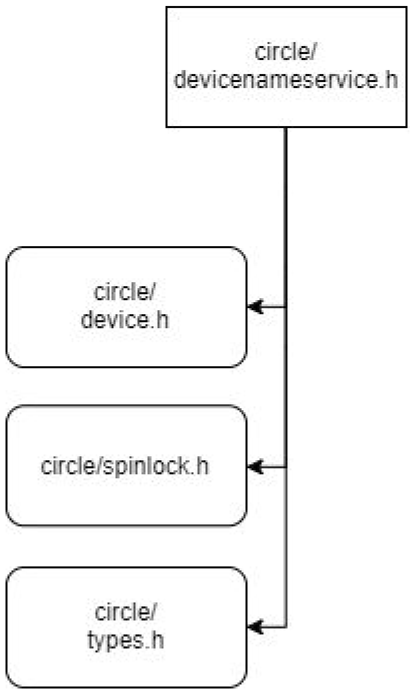
Summary:
Defines simple controls for the on-board LED. *On*, *Off*, and *Blink*. Can also return the current status of the LED.

circle/koptions.h



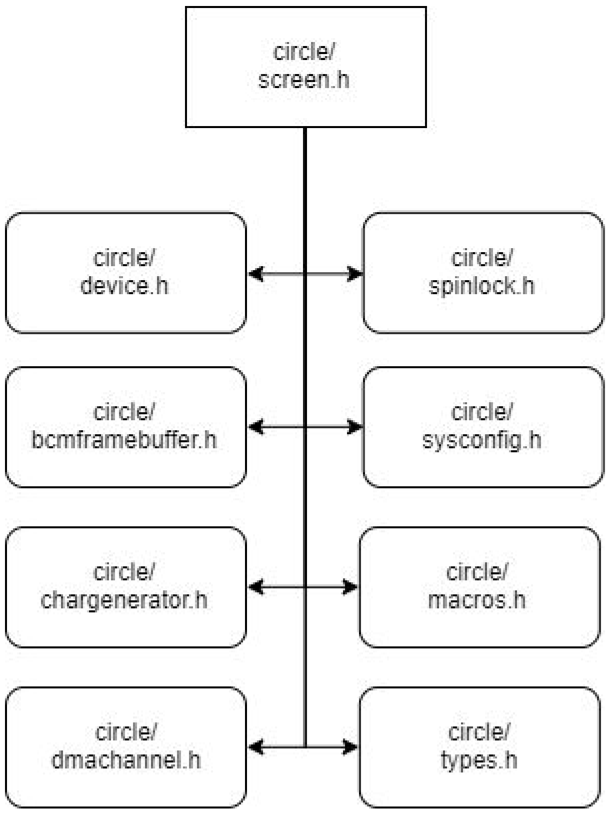
Summary:
Defines a number of options for the kernel in the *CKernelOptions* class. The kernel can be given parameters through a command line (somewhere?) which can directly affect the resolution, log device and level, keyboard map, usb power delay, usb speed, and sound devices and options. There are also settings that allow for an unlocked CPU clock and to allow the CPU to be unthrottled between 40℃ and 78℃ (the class will not let you set the temperature maximum beyond these values).

circle/devicenameservice.h



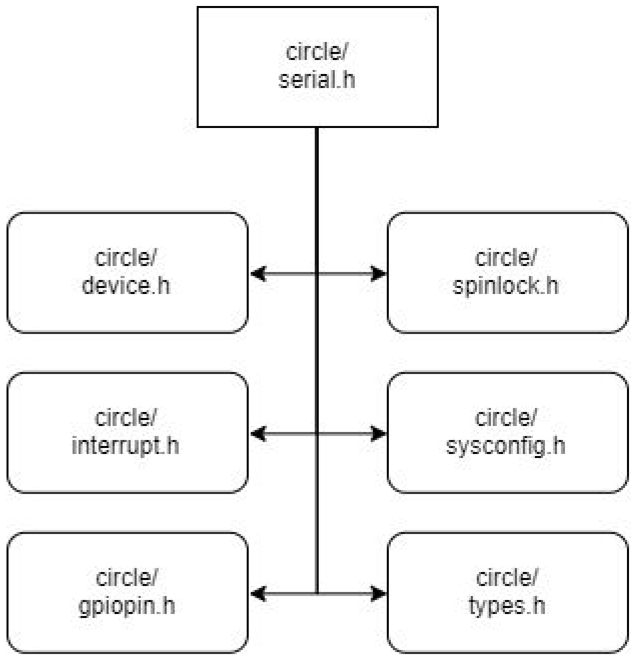
Summary:
Defines a class named *CDeviceNameService*. Can add/remove devices either by index or by pointer to said device. Can also return a pointer to the device by searching for a matching name or by searching the prefix with a device name index. Can also list devices the system currently sees.

circle/screen.h



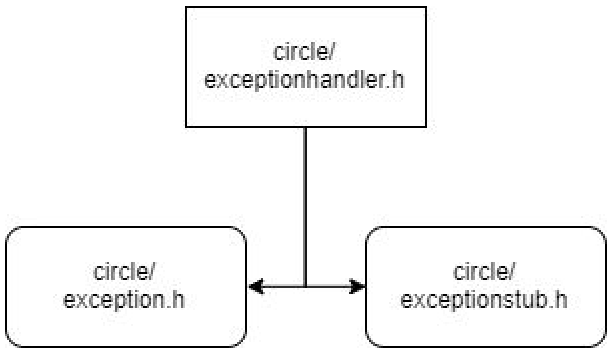
Summary:
Defines some options for displaying to the monitor/screen, as well as a class for the display called *CScreenDevice*. Defaults to a depth of 16 (allows for 8, 16, and 32). Defines *COLOR16* and *COLOR32*, and changes the screen's color mode based on the depth assigned before. Handles writing characters to the screen, as well as drawing to specific pixels on the screen/framebuffer.

circle/serial.h



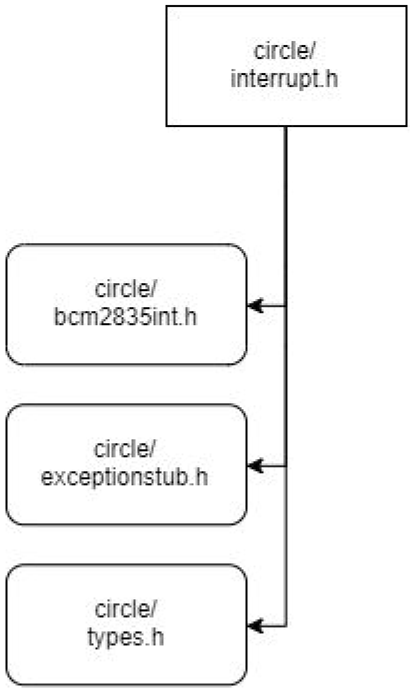
Summary:
Defines functionality for a serial addressing mode of the SoC. Can be initialized with a variable baud rate, and can the *CSerialDevice* class defined here can be configured with a *SetOptions()* function.

circle/exceptionhandler.h



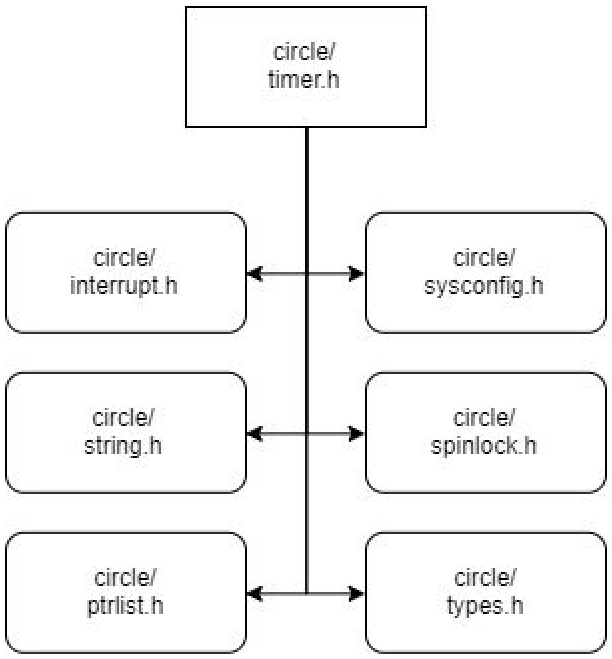
Summary:
Defines a class called *CExceptionHandler* that will *Throw()* an exception based on the given unsigned number.

circle/interrupt.h



Summary:
Defines a class called *CInterruptSystem*. Handles the standard interrupts (or normal priority) as IRQs and handles the fast interrupts (ARM-specific) as FIQs. Can connect and disconnect IRQs and FIQs to given handlers to service the requests. For RASPPI >= 4, can also access the Secure Monitor for interrupts.

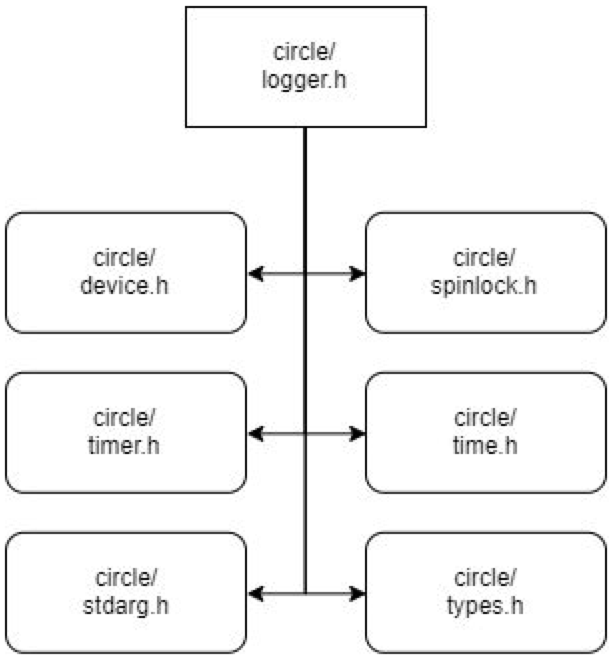
circle/timer.h



Summary:
“Manages the system clock, supports kernel timers and a calibrated delay loop.”

Defines a class called *CTimer*. Allows for timezone setting, standard computer time, local time, universal time, ticks since boot, starting and cancelling kernel timers. Also has some functions for getting the days in the month (?) and millisecond and microsecond delays.

circle/logger.h

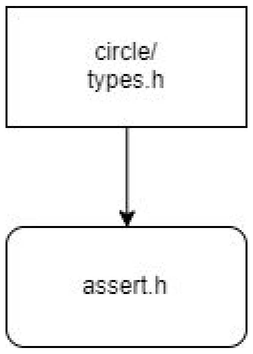


Summary:
Defines a class called *CLogger*, as well as the types of logs that can be submitted in Circle. Includes various methods for writing logs as well as reading logs. Can also write logs when the system is on/has low memory.

Log Types:

- LogPanic - System stops after processing message
- LogError - Severe error has occurred in component, system may be unstable
- LogWarning - Non-severe error has occurred in component, system is OK
- LogNotice - Informative message for the system user
- LogDebug - Informative message for debugging the component

circle/types.h



Summary:
Defines every ‘extended’ data type, those being the signed and unsigned chars, shorts, ints, and longs. Also defines booleans.

circle/stdarg.h

Summary:
Defines the black magic that is standard/variable arguments. Functions are: *va_start()*, *va_end()*, and *va_arg()*. Also defines the variable argument list, *va_list*.

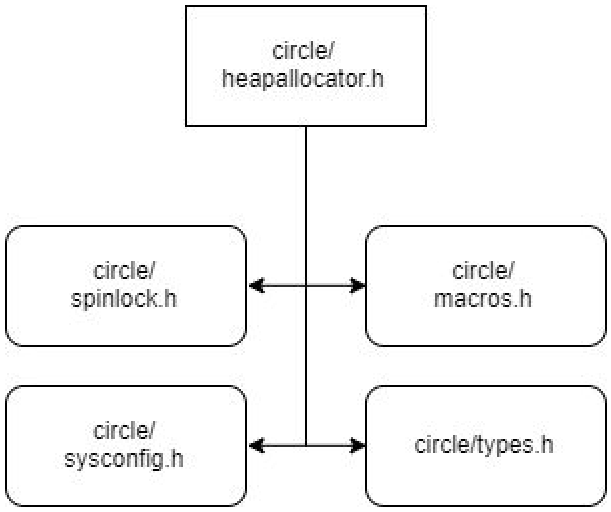
circle/macros.h

Summary:
Defines a number of macros, though their apparent use for some of them is lost on me.

Macros:

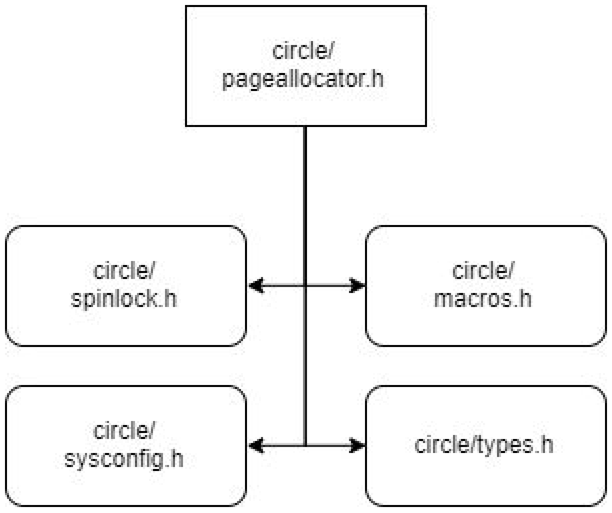
- PACKED - ?
- ALIGN(n) - ?
- NORETURN - ?
- NOOPT - ?
- MAXOPT - ?
- WEAK - ?
- likely(exp) - Returns true if the given expression is ‘likely’ to return true(?)
- unlikely(exp) - Returns true if the given expression is ‘unlikely’ to return true(?)
- BIT(n) - Gives you a bit, shifted n places to the left
- BE(value) - Big Endian mode for constants only

circle/heapallocator.h



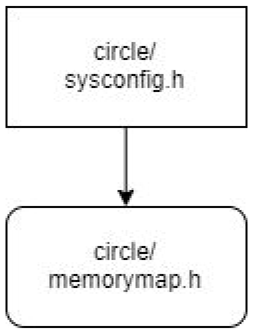
Summary:
Defines a class called *CHeapAllocator* that can *gasp* allocate blocks from the heap (which is a flat memory region). Can determine the size of said flat memory region with *Setup()*. Can allocate, reallocate, free, and return the amount of free space left in the heap. Talks about heap block magic which is pretty amusing.

circle/pageallocator.h



Summary:
Defines a class called *CPageAllocator* that will allocate an aligned page from a flat memory region. Can allocate and free pages, as well as return how much free space is left in the flat memory region. Can determine the size of the flat memory region with *Setup()*.

circle/sysconfig.h

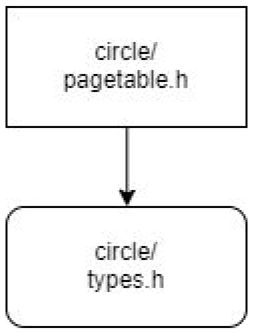


Summary:
Dictates a few system settings for the kernel, such as:

- The size of a megabyte in hexadecimal (0x100000)
- The maximum size of the kernel image
- The default heap types (new and malloc)
- The default heap block/bucket sizes
- GPU L2 cache (RASPPI = 1)
- RPI Stub (RASPPI >= 2)
- Multi-core use (RASPPI >= 2)
- Screen DMA burst length (default is 2, range is 0-15)
- Maximum system tasks (default is 20)
- Default task size (0x8000, assuming this is in bits)
- Default keyboard keymap (is usually DE because Circle is German, we have it set to US)
- Serial GPIO Pin selection (choice of three pairs, (14, 15), (32, 33), (36, 37))

There are a few more settings, but these are either commented out or otherwise not filled out.

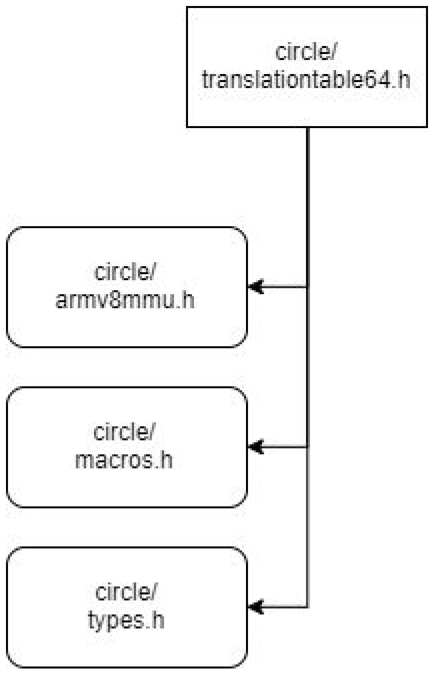
circle/pagetable.h



Summary:
Defines a class called *CPageTable*, though the definition depends on the given version of Raspberry Pi.

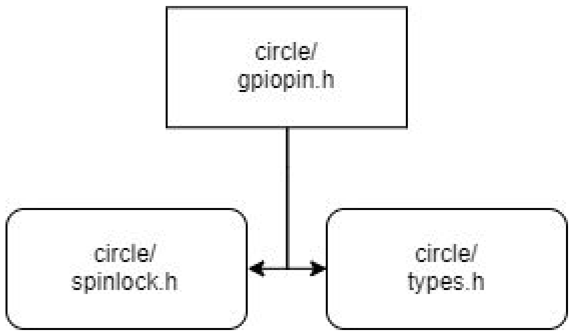
The main difference between the two is that the PI 4 page table has Level 2 capabilities and the actual pages are of *u64* type. The memory size is also stored in the class as a *u32* variable. Whereas for any PI older than the 3, the page table does not have Level 2 capabilities and is of *u32* type, and the size is not stored (as everything about it is 32-bit).

circle/transitiontable64.h



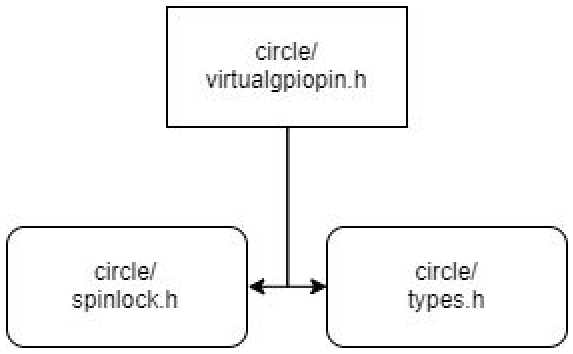
Summary:
Defiles a class called *CTranslationTable*. The class contains a Level 2 page table, with the ability to create a Level 3 page table if desired.

circle/gpiopin.h



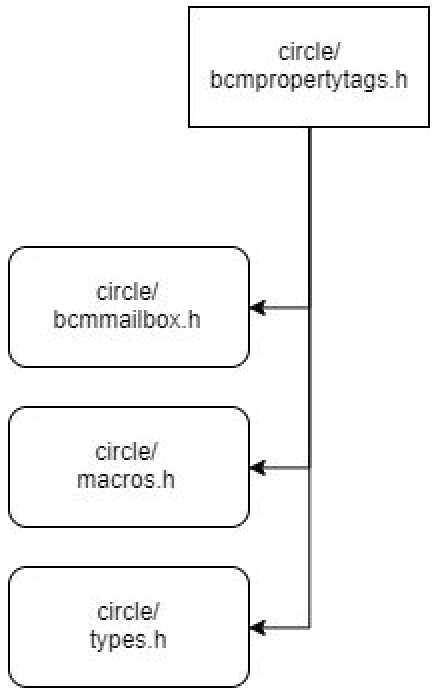
Summary:
Defines several enumerators for Virtual Pins, GPIO Modes, and GPIO Interrupts. Also defines a *CGPIOManager* and *CGPIOPin* class. The manager is defined, but with no functions or variables. The *CGPIOPin* class contains functionality to assign to pins, set the mode/pull mode, write and read to the pins, as well as enabling/disabling the interrupt system on the GPIO pins (of which it allows these interrupts to be used as a second source for system interrupts). Also contains a *CSpinlock*.

circle/virtualgpiopin.h



Summary:
Defines a class called *CVirtualGPIOPin*. The virtual pins can be written to, and inverted. Contains a *CSpinlock*.

circle/bcmpropertytags.h



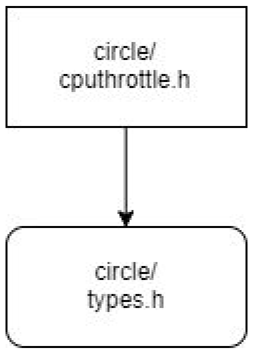
Summary:
Defines a number of hexadecimal (address) constants for a number of hardware features, such as clock rate, turbo, SDHost clock, MAC address, etc.

Also defines some structs:

TPropertyTag	TPropertyTagTurbo
TPropertyTagSimple	TPropertyTagGPIOState
TPropertyTagSetCursorInfo	TPropertyTagEDIDBlock
TPropertyTagSetCursorState	TPropertyTagSetClockRate
TPropertyTagMACAddress	TPropertyTagAllocateBuffer
TPropertyTagSerial	TPropertyTagDisplayDimensions
TPropertyTagMemory	TPropertyTagVirtualOffset
TPropertyTagPowerState	TPropertyTagSetPalette
TPropertyTagClockRate	TPropertyTagCommandLine
TPropertyTemperature	

As well as a class called *CBcmPropertyTags*.

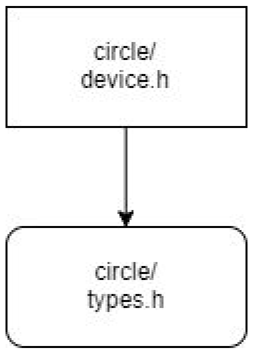
circle/cputhrottle.h



Summary:
Defines an enum *TCPUSpeed*, as well as a class called *CCPUThrottle*. The enum defines three speeds; *CPUSpeedLow*, *CPUSpeedMaximum*, *CPUSpeedUnknown*.

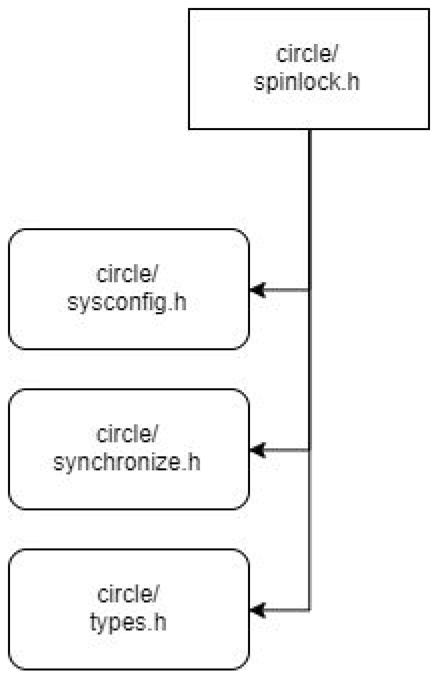
The class contains some functions for getting the current clock rate, as well as the minimum rate and maximum rate for the CPU. It can also get the current temperature of the CPU, set the speed of the CPU, and set the speed of the CPU based on the temperature of the SoC. The class can also do an infodump on the current state of the CPU.

circle/device.h



Summary:
Defines a class called *CDevice*. Can read and write a number of bytes to and from the given device respectively. Can also seek the device's offset, as well as remove the device from the system.

circle/spinlock.h

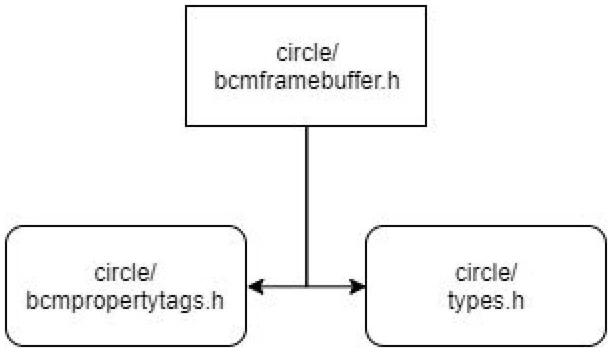


Summary:
Contains two different definitions for the *CSpinlock* class; one for multi-core usage, and one for single-core usage. In case you're not aware what a spinlock is, it's a mechanism that will 'lock' a core so that the locking process is the only process operating on a specific section of code. There are two ways to lock, a normal lock that still allows interrupts to take the core, or a irqsave lock that prevents both the scheduler and interrupts from taking the core until the lock is released.

Multi-Core:
Can be acquired, released, and enabled. Operates on a maximum execution level set on class instantiation.

Single-Core:
Can be acquired and released, though these are conditional. Operates on a maximum execution level set on the class instantiation. The given target level for single core usage must be greater than or equal to the level of the interrupt requests, which I take to mean that interrupts are still able to get through regardless in single-core mode.

circle/bcmframebuffer.h

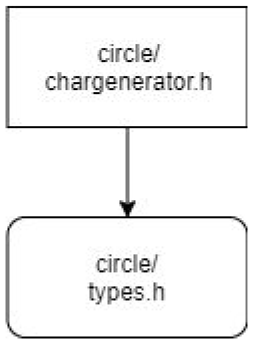


Summary:
Defines a structure called *TBcmFrameBufferInitTags*, as well as a class called *CBcmFrameBuffer*. The frame buffer allows for up to 256 palettes.

The struct contains the physical and virtual size of the display, as well as a virtual offset, depth, pitch, and allocation buffer.

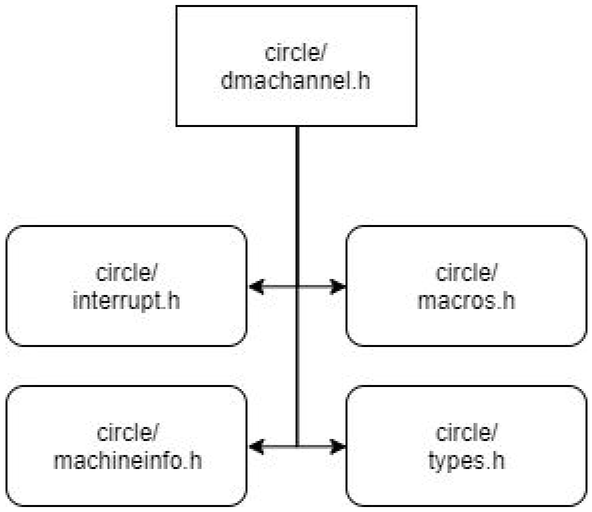
The class allows for setting the palettes by index (8-bit number, supports both 16-bit palette and 32-bit palette). The struct information can be retrieved from the class as single unsigned 32-bit numbers (i.e. one at a time). The virtual offset can be set from a function in the class, vsync can be toggled on and off, and the brightness of the display can be controlled (though it is noted to have only been tested on the official RPi touchscreen).

circle/chargenerator.h



Summary:
Defines a class called *CCharGenerator*. Gets the width and height of a given char, as well as the underline of a given char. You can also get a specific pixel inside of a char, as defined by an unsigned int x and y offset.

circle/dmachannel.h



Summary:
(Direct Memory Access channel)
Defines a struct *TDMAControlBlock*, an enum *TDREQ*, and a class called *CDMAChannel*.

The struct contains unsigned ints representing the transfer information, source address, destination address, length of the transfer, 2D mode stride, the next control block's address, and two unsigned ints representing reservation.

The enum defines a number of sources: None, PCMTX, PCMRX, PWM, SPITX, SPIRX, EMMC, UARTTX, and UARTRX.

The class contains functions to set up the MemCopy, IORead and IOWrite, MemCopy2D, and CompletionRoutine. Transfer beings(?) on call of Start() method, and Wait() can be called "for synchronous call without completion routine". The status of the transfer can also be retrieved. The transfers can also be interrupted(?) with the InterruptHandler() and InterruptStub() functions.

circle/exception.h

Summary:
Defines some constant (exception indices) depending on if the system is in 32-bit mode or 64-bit mode.

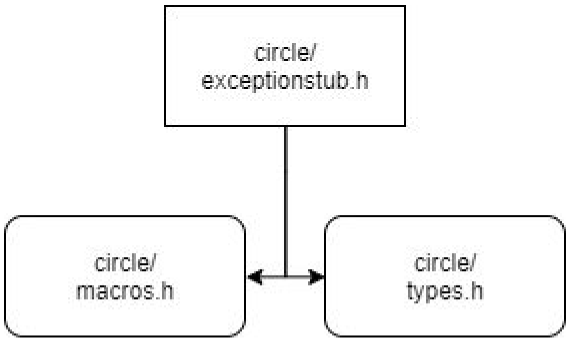
32-bit Mode

- EXCEPTION_DIVISION_BY_ZERO
- EXCEPTION_UNDEFINED_INSTRUCTION
- EXCEPTION_PREFETCH_ABORT
- EXCEPTION_DATA_ABORT
- EXCEPTION_UNKOWN

64-bit Mode

- EXCEPTION_UNEXPECTED
- EXCEPTION_SYNCHRONOUS
- EXCEPTION_SYSTEM_ERROR

circle/exceptionstub.h



Summary:
Defines different structures in 32-bit mode and in 64-bit mode. Also has additional definitions if RASPPI >= 4. Regardless of the mode, it will define a *TFIQData* struct that contains a pointer to the handler, FIQNumber and the list of parameters. Uses a TFIQHandler() function. Also contains an *extern* IRQReturnAddress.

32-bit Mode
Defines two structs called *TExceptionTable* and *TAbortFrame*. The table contains unsigned ints to an undefined instruction, supervisor call, prefetch abort, data abort, unused (not sure what this is for yet), IRQ (interrupt request), and FIQ (fast interrupt request). The abort frame contains register information for the exception. There are some additional functions that fetch the stubs for undefined instructions, prefetch, data, IRQ, FIQ, and SMC. Each handler can also be called - the exception handler, the interrupt handler, and the secure monitor handler.

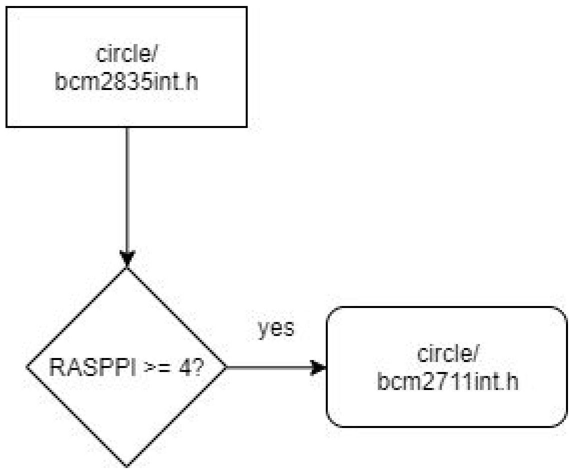
64-bit Mode
Two macros for 64-bit are defined: an opcode branch that operates on a distance, and the distance macro that returns the address(?) distance from the first address to the second address.

AARCH64_OPCODE_BRANCH(distance)
AARCH64_DISTANCE(from, to)

Two structs are defined as well: *TVectorTable*, *TAbortFrame*. The table contains another struct with a Branch and 31 Dummies, along with 16 Vectors. The frame contains some register information, though these are not the same registers as in the 32-bit mode.

RASPPI >= 4
Defines a ARMSTUB_FIQ magic number and magic address. Defines a SMCStub(), UnexpectedStub(), and SecureMontiorHandler().

circle/bcm2835int.h



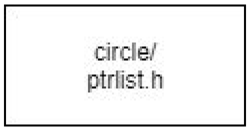
Summary:
Operates differently based on RASPPI's value. Assuming that REG in each #define refers to the Pi's registers.

In general, this header file dictates the IRQs and FIQs on the system, such as for timers, DMAs, and VideoScaler (see the actual file for the complete list of IRQs and FIQs supported on the system). Each index constant is based on the base values set in the beginning so *don't change these*. The maximum amount of ARM FIQs is 71.

RASPPI >= 4
Includes the bcm2711int.h header and skips the rest of the file's contents. Refer to the bcm2711int.h page for more information.
When not using a Pi 4 or later, allows for 32 IRQs per register, and 8 IRQs per basic register.

RASPPI >= 2
Allows for 12 IRQs per local register. Also includes additional IRQs to handle those local registers.
When not using a Pi 2 or later, allows for 0 IRQs per local register.

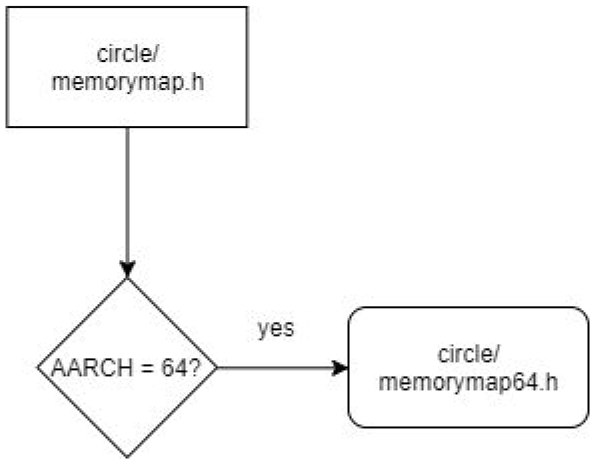
circle/ptrlist.h



Summary:
Defines a class called *CPtrList*. Appears to handle a singly-linked list of pointers, where the first pointer can be retrieved, the ‘next’ pointer can be retrieved, and the pointer for a specific element can be retrieved.

List elements can be inserted before or after a given element, and elements can be removed. A search function Find() is also included.

circle/memorymap.h



Summary:
If the system is in 64-bit, uses the memorymap64.h file instead. Go there for more information about the 64-bit memory map.

Defines some memory sizes for Megabyte and Gigabyte, then goes on to define the sizes of memory spaces for the system. The default size of memory is 256MB, the default size of GPU memory is 64MB (though this can be overridden in config.txt), and the ARM memory size is the difference between these two.

We have overwritten the page size to be a more traditional 4KB instead of the default 65536 bytes (64KB) pages that Circle uses.

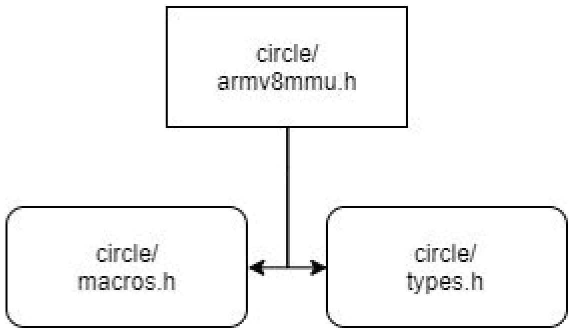
Defines the Kernel stack size, exception stack size, and page reserve (default is 16MB). The abort stack, IRQ stack, FIQ stack, and page table are defined. The size of these depends on if RASPPI == 1 or not.

The kernel code starts at address 0x8000, and takes up about 2MB of space on disk.

The heap start is defined if RASPPI <= 3, as the Pi 4 allows for High and Low heap allocation unlike the previous models.

High Memory and the PCIE memory range (inbound and outbound) is defined if RASPPI >= 4.

circle/armv8mmu.h



Summary:
Defines some structs and constants for operating the ARMv8 MMU. Including:

Structs
- TARMV8MMU_LEVEL2_TABLE_DESCRIPTOR
- TARMV8MMU_LEVEL2_BLOCK_DESCRIPTOR
- TARMV8MMU_LEVEL2_INVALID_DESCRIPTOR

- TARMV8MMU_LEVEL3_PAGE_DESCRIPTOR
- TARMV8MMU_LEVEL3_PAGE_DESCRIPTOR_4BIT
- TARMV8MMU_LEVEL3_INVALID_DESCRIPTOR

(Struct) Unions
- TARMV8MMU_LEVEL2_DESCRIPTOR
- TARMV8MMU_LEVEL3_DESCRIPTOR
- TARMV8MMU_LEVEL3_DESCRIPTOR_4BIT

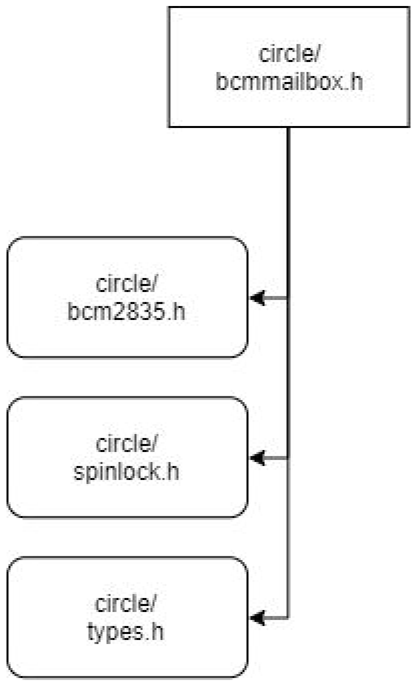
Two macros are defined to give an address to the Level 2 Tables as well as a pointer to those tables.

The Level 2 ARMv8 MMU block size is 512MB. Two macros are defined to access the block addresses and pointers to those blocks.

The Level 3 ARMv8 MMU page size has a 64KB definition and a separate 4KB definition. Two macros are defined to access the page addresses and pointers to those pages.

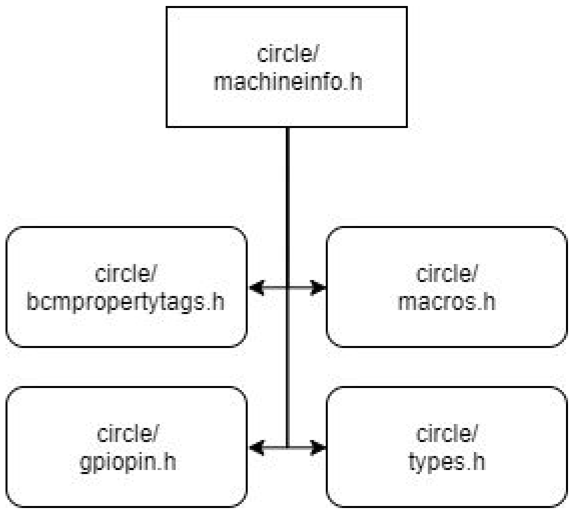
Some system registers are also defined at the end.

circle/bcmmailbox.h



Summary:
Defines a class called *CBcmMailbox*. It contains functionality to WriteRead() unsigned data, and internally can Flush, Read, and Write. Is initialized onto a channel, and can be toggled to have early use.

circle/machineinfo.h



Summary:
Contains two enums *TMachineModel* and *TSoCType* that represent the model of Pi and the processor powering the board. A third 'enum' determines whether the DeviceID is *DeviceI2CMaster* or *DeviceUnkown*.

A class *CMachineInfo* is defined, which can get the hardware's model, name, major, revision, SoC Type, RAM size, SoC Name, Raw Revision Info.

Also defines functionality to determine the state of the LED and system clock, clock rate, GPIO pin, and other devices.

Can also determine if the PWM Audio channels have been swapped.

Specific definitions are included for the DMA Channel for RASPPI <= 3.

circle/bcm2711int.h

Summary:
Defines all of the IRQs and FIQs for RASPPI >= 4. The list is smaller so I will include it in its entirety.

Device Tree Source

GIC_PPI(n) -> 16+n
GIC_SPI(n) -> 32+n

IRQs

ARM_IRQ_ARM_DOORBELL_0
ARM_IRQ_TIMER1
ARM_IRQ_DMA0
ARM_IRQ_DMA1
ARM_IRQ_DMA2
ARM_IRQ_DMA3
ARM_IRQ_DMA4
ARM_IRQ_DMA5
ARM_IRQ_DMA6
ARM_IRQ_DMA7
ARM_IRQ_DMA8
ARM_IRQ_DMA9
ARM_IRQ_DMA10
ARM_IRQ_DMA11
ARM_IRQ_DMA12
ARM_IRQ_DMA13
ARM_IRQ_DMA14
ARM_IRQ_GPIO0
ARM_IRQ_GPIO1
ARM_IRQ_GPIO2
ARM_IRQ_GPIO3
ARM_IRQ_UART
ARM_IRQ_ARASANSDIO
ARM_IRQ_PCIE_HOST_MSI
ARM_IRQ_BCM54213_0
ARM_IRQ_BCM54213_1

IRQ_LINES -> 256

FIQ

ARM_FIQ_TIMER1
ARM_FIQ_GPIO3
ARM_FIQ_UART

ARM_MAX_FIQ -> IRQ_LINES

circle/memorymap64.h

Summary:
Like memorymap.h, it defines the sizes of Megabyte and Gigabyte. It defines the number of cores (4), the default memory size (512MB), default GPU memory (64MB, can be changed in config.txt), and the ARM memory size is the difference between those two. IMPORTANT: DO NOT INCLUDE THIS HEADER FILE BY ITSELF. INCLUDE memorymap.h INSTEAD.

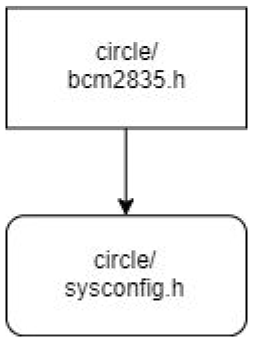
As before, the default Circle page size is 64KB and PegasOS's page size is 4KB. The kernel stack size is the same at ~128K (must be a multiple of 16K) and so is the exception stack size at ~32K. However the page reserve is larger at 16MB as opposed to 4MB.

The kernel code starts at 0x80000 instead, as the 64-bit kernel uses more space for memory.

The heap definitions from 32-bit mode are here for RASPPI <= 3

The PCIE outbound and inbound ranges are defined for RASPPI >= 4. The high heap for Pi 4 starts at 1GB and is about 2GB long.

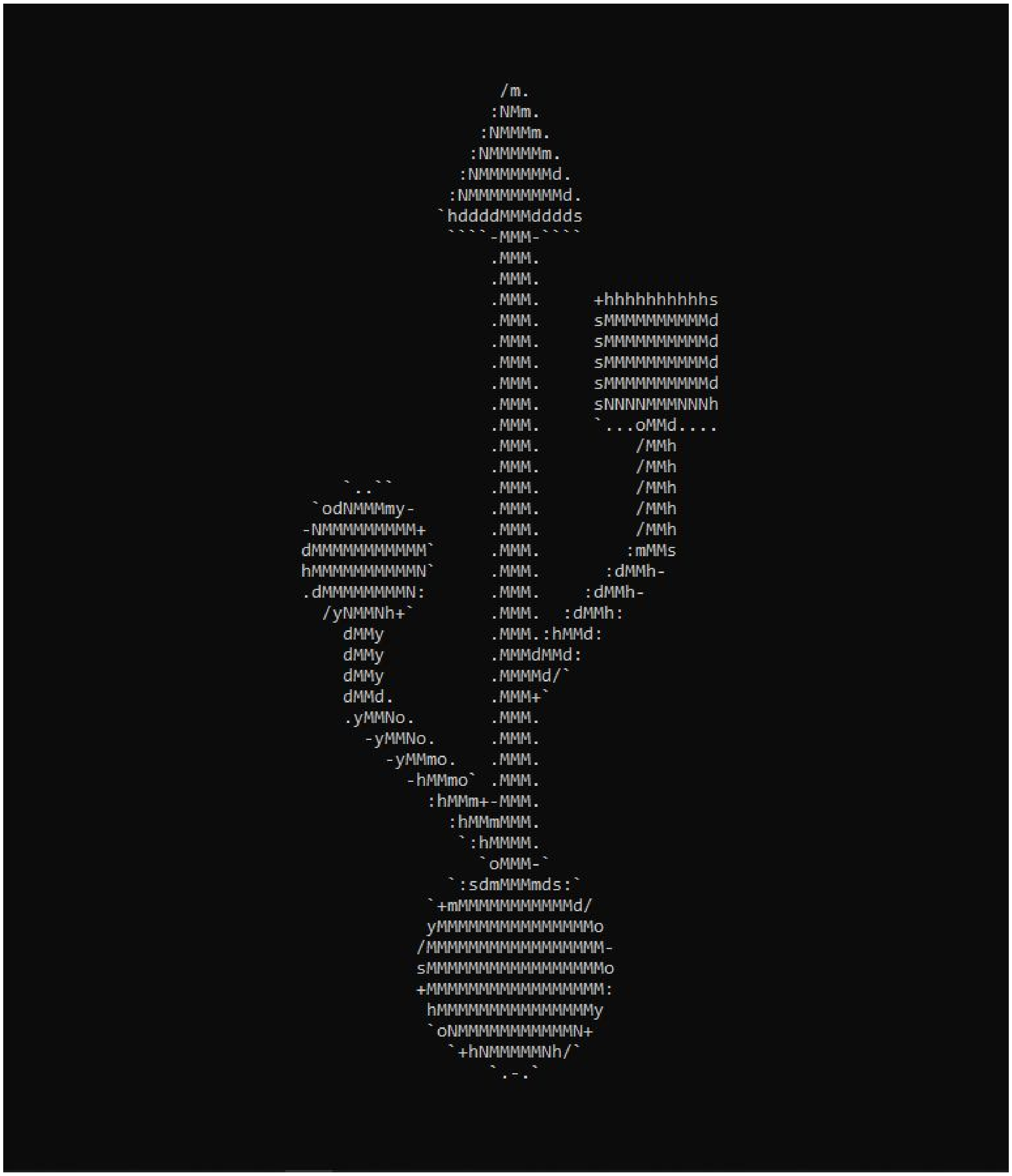
circle/bcm2835.h



Summary:
Defines a number of addresses for the SoC. Including the GPIO pins, ARM IO, UART0, ARM System Timers, DMA base, ARM IC, ARM Timers, Mailbox, PWM (Pulse Width Modulator), Clock Manager, USB Host Controller, External Mass-Media Controller, SDHost Controller, Power Manager, BSC Master, SPI0 Master, BSC/SPI Slave, Auxiliary Peripherals, Hardware RNG, PCM/I2S Audio Module, and VC4 VCHIQ.

The GPU cached and uncached base addresses are defined, as well as the base ARM IO addresses and GPU Memory base address.

Circle’s USB Dependencies



~~~
                   /m.
                  :NMm.
                 :NMMMm.
                :NMMMMMm.
               :NMMMMMMMd.
              :NMMMMMMMMMd.
             `hddddMMMdddds
              ````-MMM-````
                  .MMM.
                  .MMM.
                  .MMM.     +hhhhhhhhhhs
                  .MMM.     sMMMMMMMMMMd
                  .MMM.     sMMMMMMMMMMd
                  .MMM.     sMMMMMMMMMMd
                  .MMM.     sMMMMMMMMMMd
                  .MMM.     sNNNNMMMNNNh
                  .MMM.     `...oMMd....
                  .MMM.         /MMh
                  .MMM.         /MMh
    `..``         .MMM.         /MMh
 `odNMMMmy-       .MMM.         /MMh
-NMMMMMMMMM+      .MMM.         /MMh
dMMMMMMMMMMM`     .MMM.        :mMMs
hMMMMMMMMMMN`     .MMM.      :dMMh-
.dMMMMMMMMN:      .MMM.    :dMMh-
  /yNMMNh+`       .MMM.  :dMMh:
    dMMy          .MMM.:hMMd:
    dMMy          .MMMdMMd:
    dMMy          .MMMMd/`
    dMMd.         .MMM+`
    .yMMNo.       .MMM.
      -yMMNo.     .MMM.
        -yMMmo.   .MMM.
          -hMMmo` .MMM.
            :hMMm+-MMM.
              :hMMmMMM.
               `:hMMMM.
                 `oMMM-`
              `:sdmMMMmds:`
            `+mMMMMMMMMMMMd/
            yMMMMMMMMMMMMMMMo
           /MMMMMMMMMMMMMMMMM-
           sMMMMMMMMMMMMMMMMMo
           +MMMMMMMMMMMMMMMMM:
            hMMMMMMMMMMMMMMMy
            `oNMMMMMMMMMMMN+
              `+hNMMMMMNh/`
                  `.-.`
~~~

circle/usb/usbkeyboard.h

<image goes here>

circle/usb/usbhcidevice.h

<image goes here>