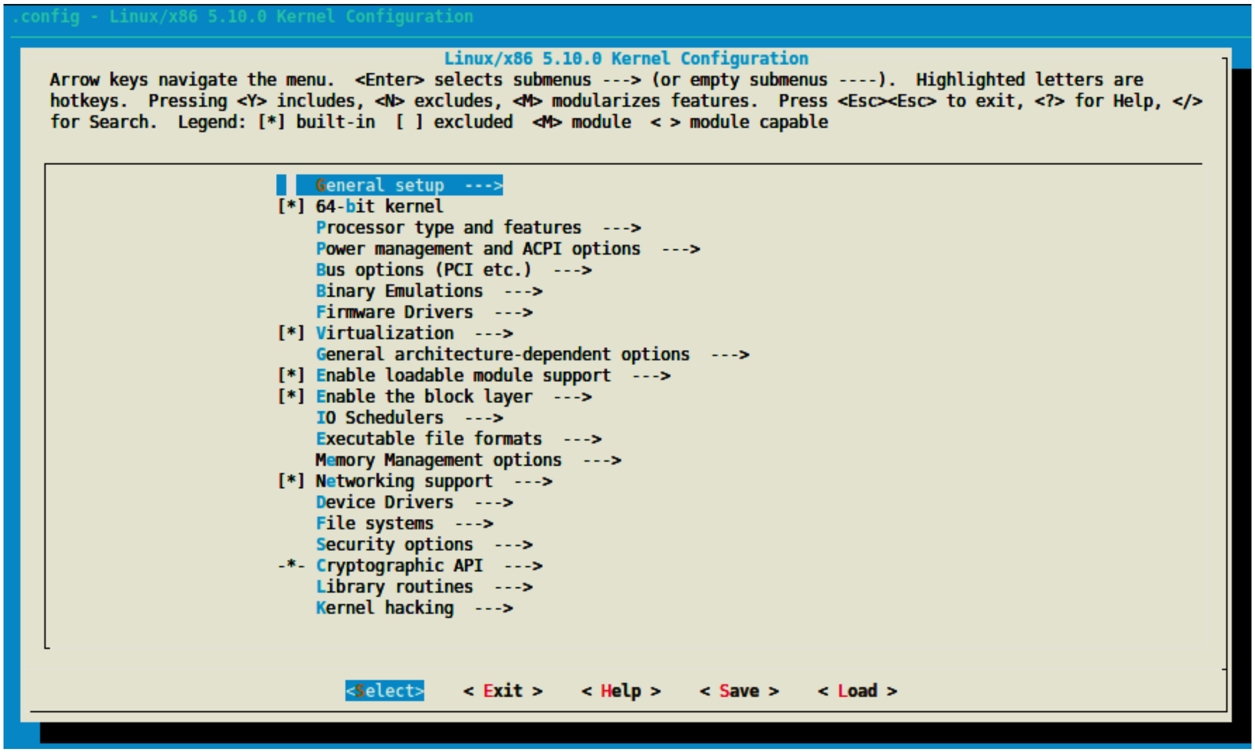
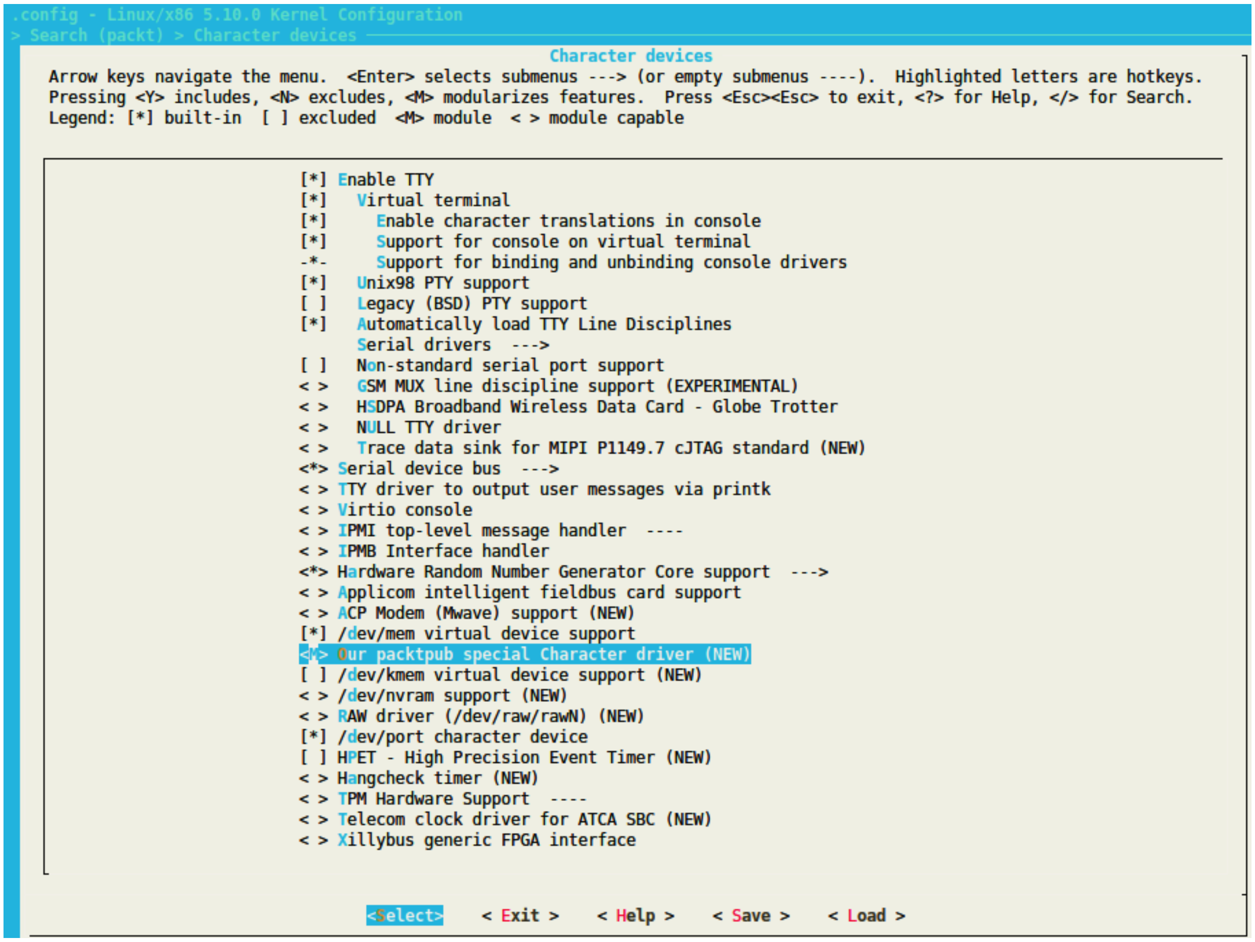
Chapter 1: Introduction to Kernel Development



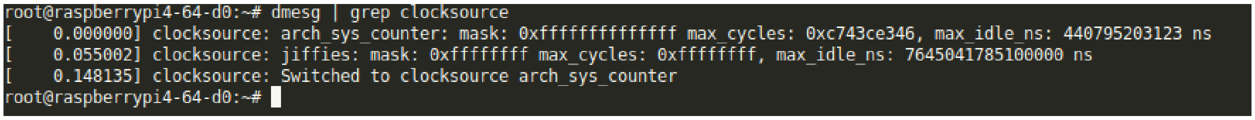
```
.config - Linux/x86 5.10.0 Kernel Configuration
                              Linux/x86 5.10.0 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                        General setup  --->
                    [*] 64-bit kernel
                        Processor type and features  --->
                        Power management and ACPI options  --->
                        Bus options (PCI etc.)  --->
                        Binary Emulations  --->
                        Firmware Drivers  --->
                    [*] Virtualization  --->
                        General architecture-dependent options  --->
                    [*] Enable loadable module support  --->
                    [*] Enable the block layer  --->
                        IO Schedulers  --->
                        Executable file formats  --->
                        Memory Management options  --->
                    [*] Networking support  --->
                        Device Drivers  --->
                        File systems  --->
                        Security options  --->
                    -*- Cryptographic API  --->
                        Library routines  --->
                        Kernel hacking  --->

                  <Select>    < Exit >    < Help >    < Save >    < Load >
```

Chapter 2: Understanding Linux Kernel Module Basic Concepts

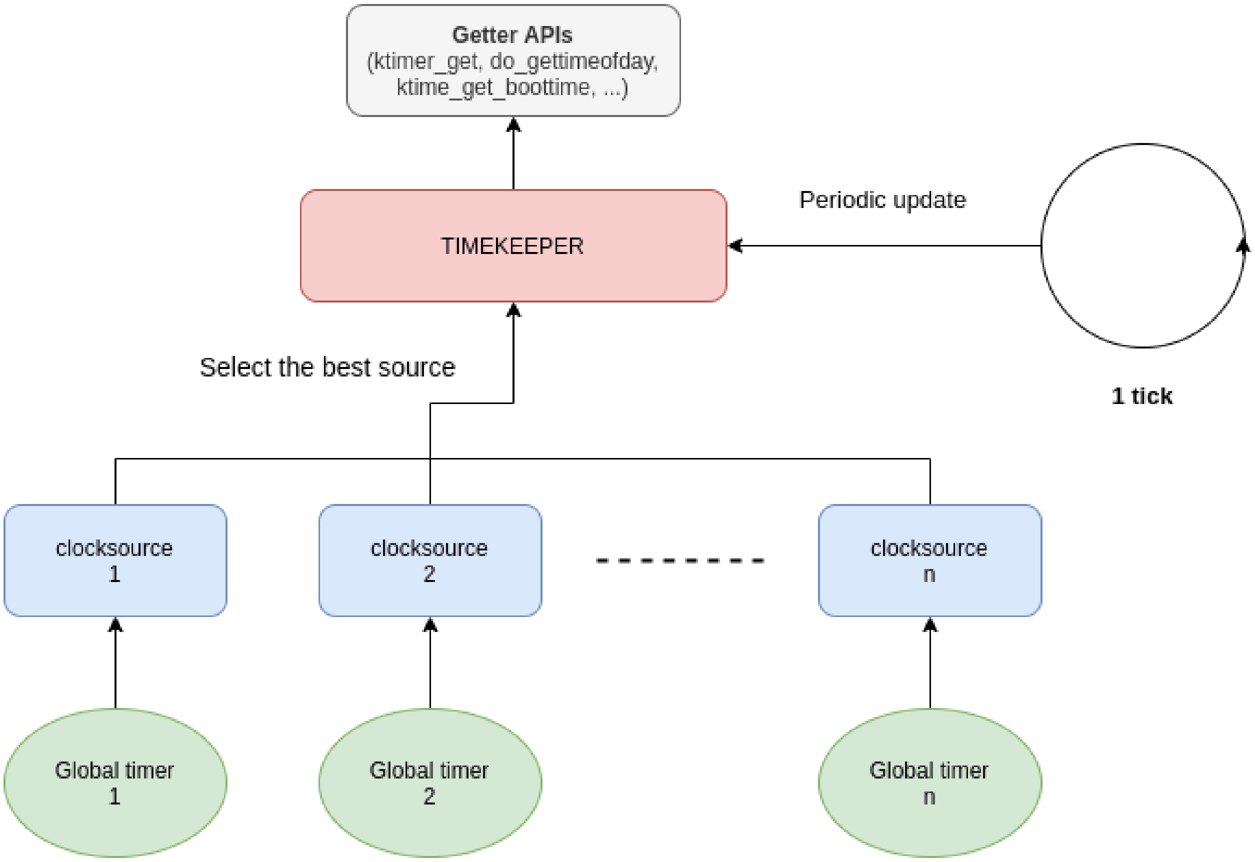


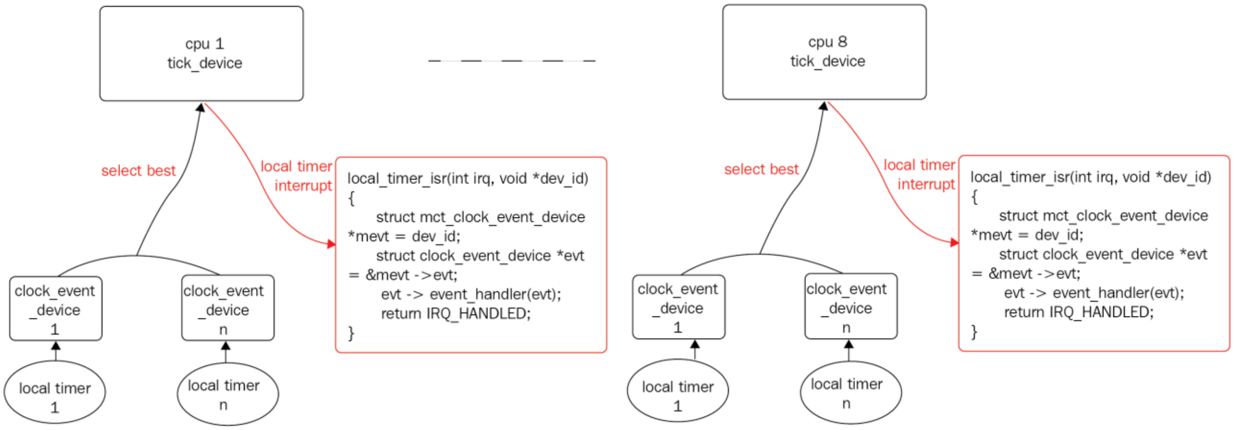
```
.config - Linux/x86 5.10.0 Kernel Configuration
> Search (packt) > Character devices ──────────────────────────────────────────
                                  Character devices
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are hotkeys.
  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                            [*] Enable TTY
                            [*]   Virtual terminal
                            [*]     Enable character translations in console
                            [*]     Support for console on virtual terminal
                            -*-     Support for binding and unbinding console drivers
                            [*]   Unix98 PTY support
                            [ ]   Legacy (BSD) PTY support
                            [*]   Automatically load TTY Line Disciplines
                                  Serial drivers  --->
                            [ ]   Non-standard serial port support
                            < >   GSM MUX line discipline support (EXPERIMENTAL)
                            < >   HSDPA Broadband Wireless Data Card - Globe Trotter
                            < >   NULL TTY driver
                            < >   Trace data sink for MIPI P1149.7 cJTAG standard (NEW)
                            <*> Serial device bus  --->
                            < > TTY driver to output user messages via printk
                            < > Virtio console
                            < > IPMI top-level message handler  ----
                            < > IPMB Interface handler
                            <*> Hardware Random Number Generator Core support  --->
                            < > Applicom intelligent fieldbus card support
                            < > ACP Modem (Mwave) support (NEW)
                            [*] /dev/mem virtual device support
                            <M> Our packtpub special Character driver (NEW)
                            [ ] /dev/kmem virtual device support (NEW)
                            < > /dev/nvram support (NEW)
                            < > RAW driver (/dev/raw/rawN) (NEW)
                            [*] /dev/port character device
                            [ ] HPET - High Precision Event Timer (NEW)
                            < > Hangcheck timer (NEW)
                            < > TPM Hardware Support  ----
                            < > Telecom clock driver for ATCA SBC (NEW)
                            < > Xillybus generic FPGA interface

                       <Select>    < Exit >    < Help >    < Save >    < Load >
```

Chapter 3: Dealing with Kernel Core Helpers



```
root@raspberrypi4-64-d0:~# dmesg | grep clocksource
[    0.000000] clocksource: arch_sys_counter: mask: 0xffffffffffffff max_cycles: 0xc743ce346, max_idle_ns: 440795203123 ns
[    0.055002] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 7645041785100000 ns
[    0.148135] clocksource: Switched to clocksource arch_sys_counter
root@raspberrypi4-64-d0:~#
```





```
local_timer_isr(int irq, void *dev_id)
{
      struct mct_clock_event_device
*mevt = dev_id;
      struct clock_event_device *evt
= &mevt ->evt;
      evt -> event_handler(evt);
      return IRQ_HANDLED;
}
```

local_timer_isr(int irq, void *dev_id)
{
struct mct_clock_event_device
*mevt = dev_id;
struct clock_event_device *evt
= &mevt ->evt;
evt -> event_handler(evt);
return IRQ_HANDLED;
}

Chapter 4: Writing Character Device Drivers

No Images…

Chapter 5: Understanding and Leveraging the Device Tree

No Images…

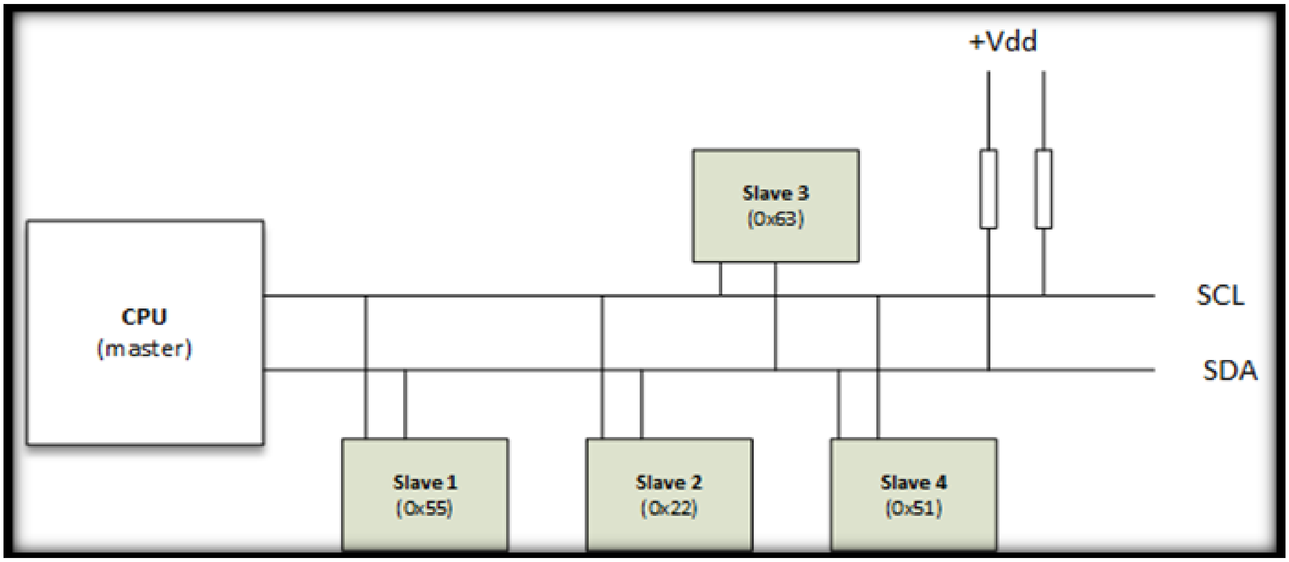
Chapter 6: Introduction to Devices, Drivers, and Platform Abstraction

No Images…

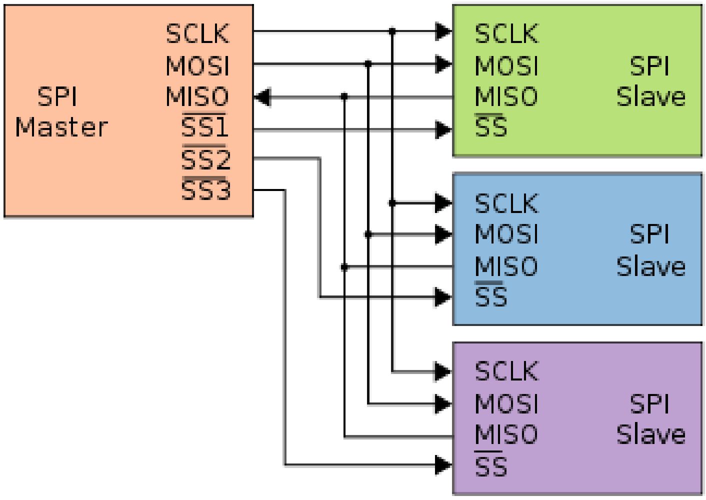
Chapter 7: Understanding the Concept of Platform Devices and Drivers

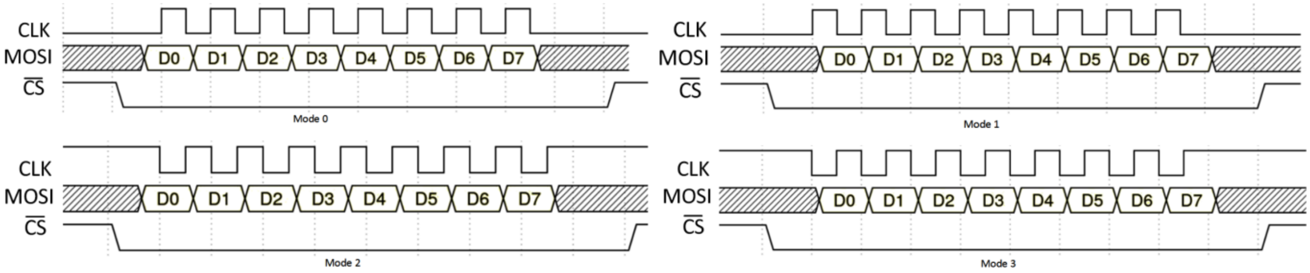
No Images…

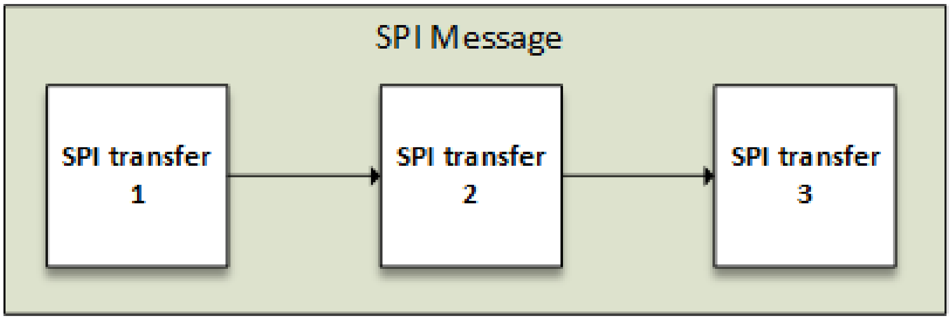
Chapter 8: Writing I2C Device Drivers



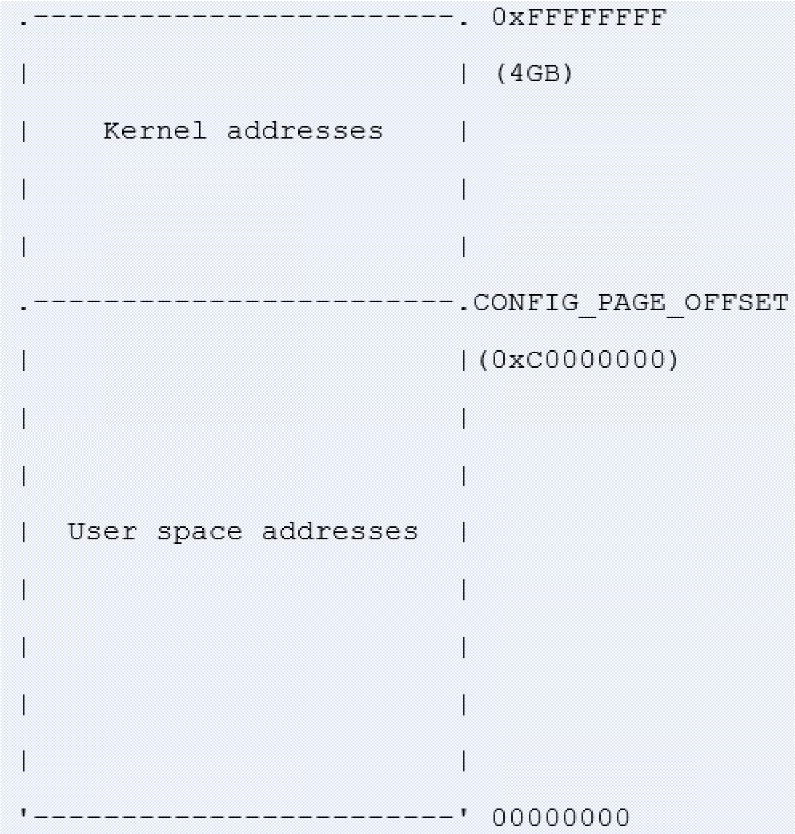
Chapter 9: Writing SPI Device Drivers



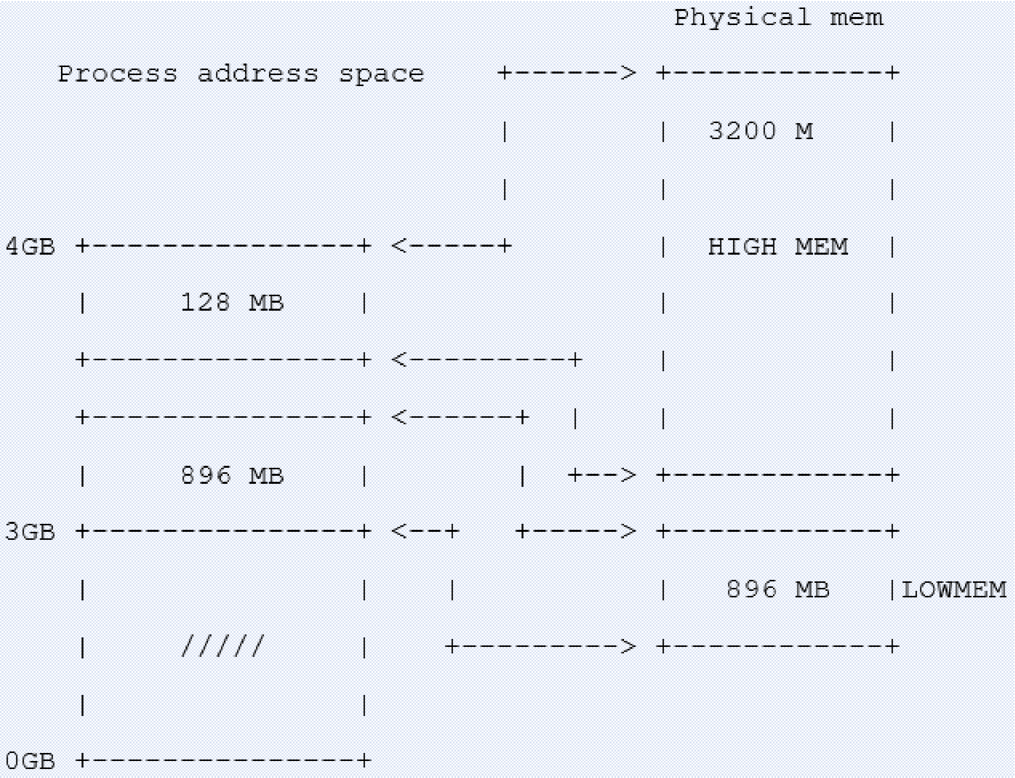




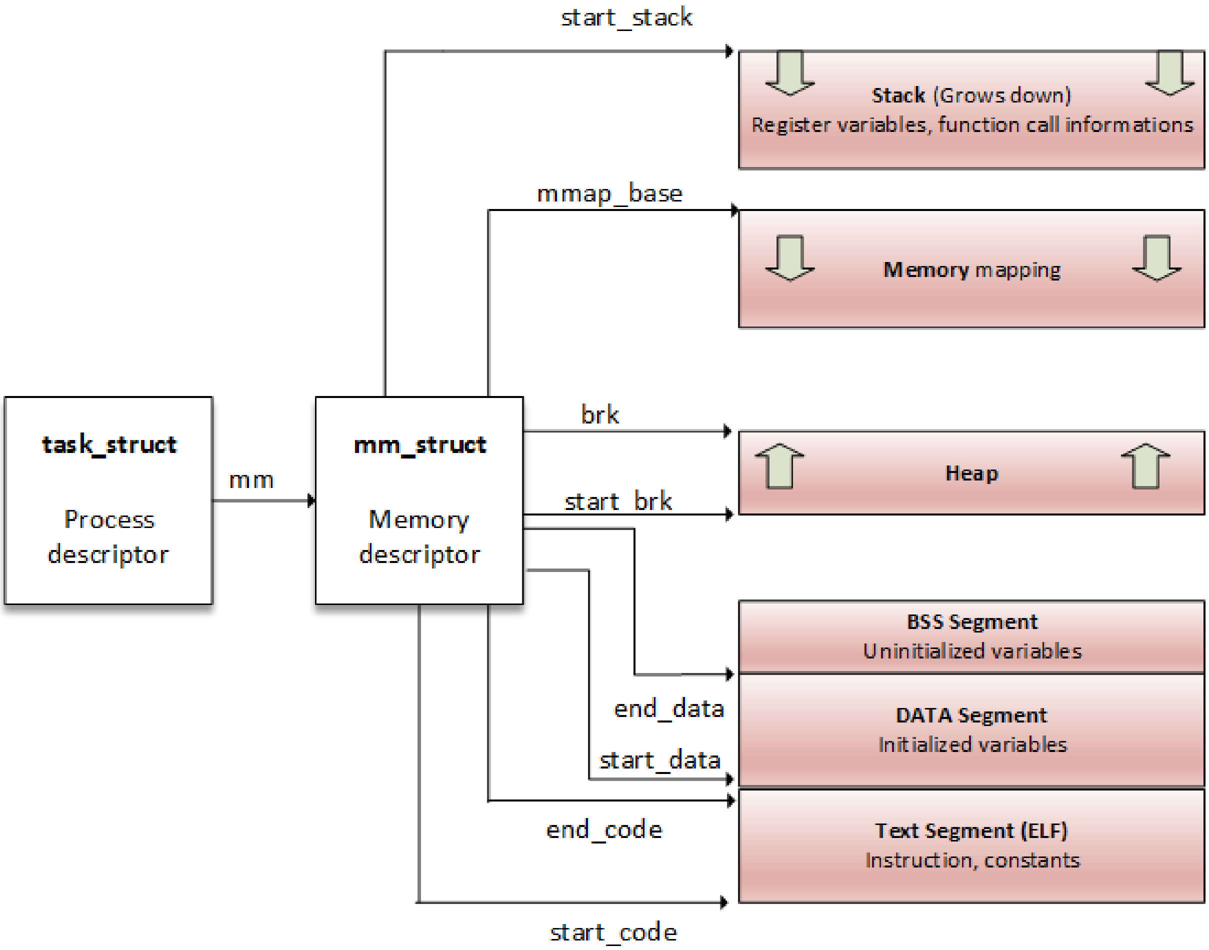
Chapter 10: Understanding the Linux Kernel Memory Allocation

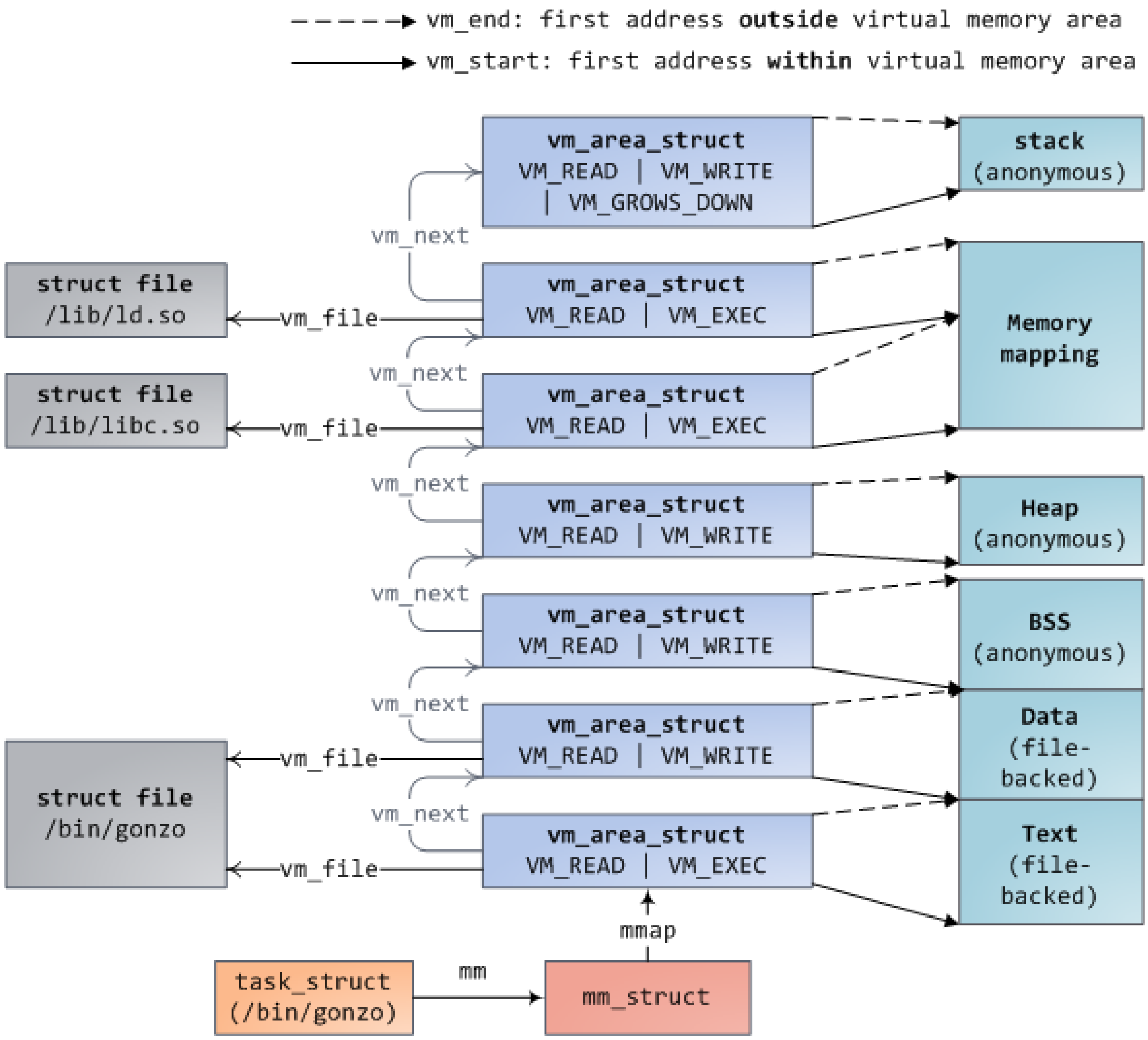


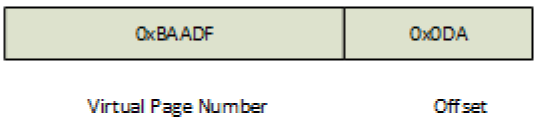
```
.------------------------. 0xFFFFFFFF
|                        |  (4GB)
|    Kernel addresses    |
|                        |
|                        |
.------------------------.CONFIG_PAGE_OFFSET
|                        |(0xC0000000)
|                        |
|                        |
|  User space addresses  |
|                        |
|                        |
|                        |
|                        |
'------------------------' 00000000
```

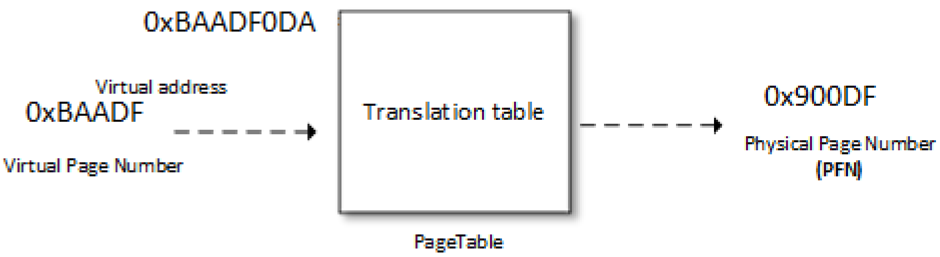


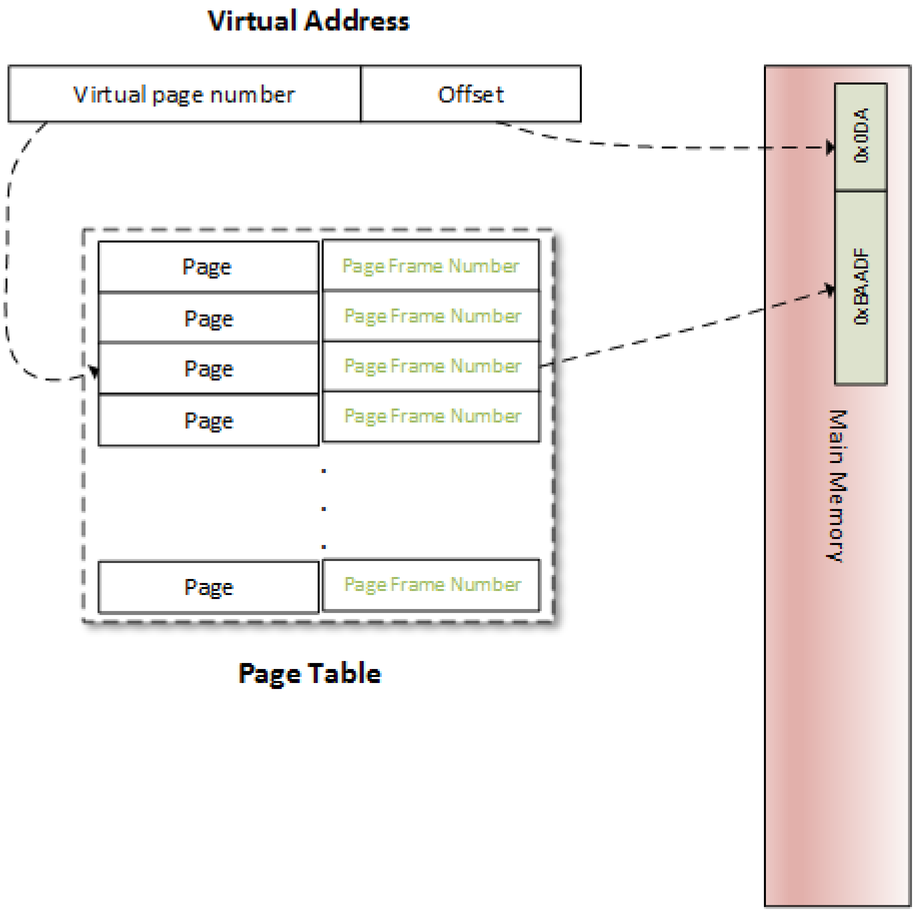
```
                                      Physical mem
   Process address space    +------> +------------+
                            |        |  3200 M    |
                            |        |            |
4GB +---------------+ <-----+        |  HIGH MEM  |
    |     128 MB    |                |            |
    +---------------+ <---------+    |            |
    +---------------+ <------+  |    |            |
    |     896 MB    |        |  +--> +------------+
3GB +---------------+ <--+   +-----> +------------+
    |               |    |           |   896 MB   |LOWMEM
    |     /////     |    +---------> +------------+
    |               |
0GB +---------------+
```

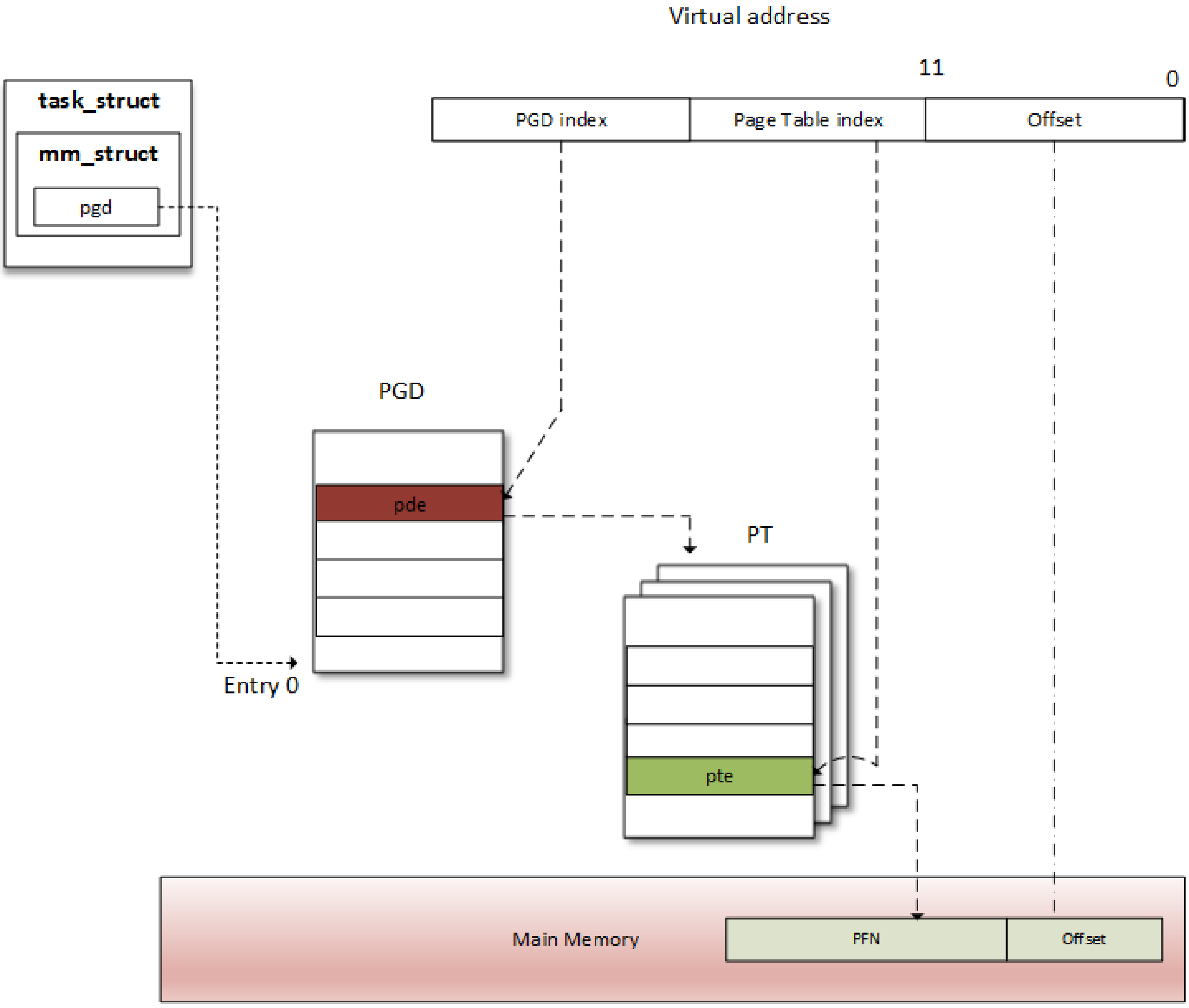

start_stack
Stack (Grows down)
Register variables, function call informations
mmap_base
Memory mapping
task_struct
Process descriptor
mm
mm_struct
Memory descriptor
brk
Heap
start_brk
BSS Segment
Uninitialized variables
end_data
DATA Segment
Initialized variables
start_data
end_code
Text Segment (ELF)
Instruction, constants
start_code

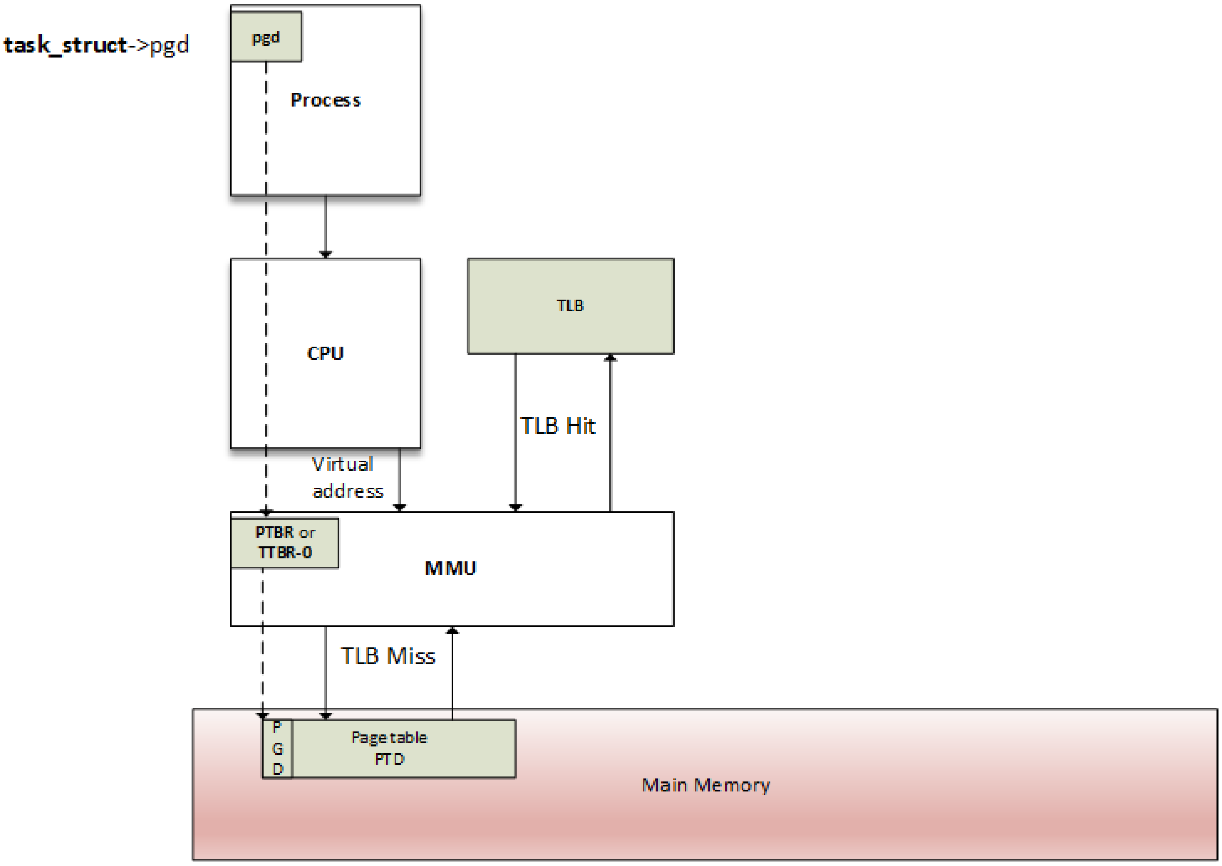

vm_end: first address **outside** virtual memory area
vm_start: first address **within** virtual memory area
vm_area_struct
VM_READ | VM_WRITE | VM_GROWS_DOWN
stack (anonymous)
vm_next
struct file /lib/ld.so
vm_file
vm_area_struct
VM_READ | VM_EXEC
Memory mapping
vm_next
struct file /lib/libc.so
vm_file
vm_area_struct
VM_READ | VM_EXEC
vm_next
vm_area_struct
VM_READ | VM_WRITE
Heap (anonymous)
vm_next
vm_area_struct
VM_READ | VM_WRITE
BSS (anonymous)
vm_next
vm_area_struct
VM_READ | VM_WRITE
Data (file-backed)
vm_next
struct file /bin/gonzo
vm_file
vm_file
vm_area_struct
VM_READ | VM_EXEC
Text (file-backed)
mmap
task_struct (/bin/gonzo)
mm
mm_struct

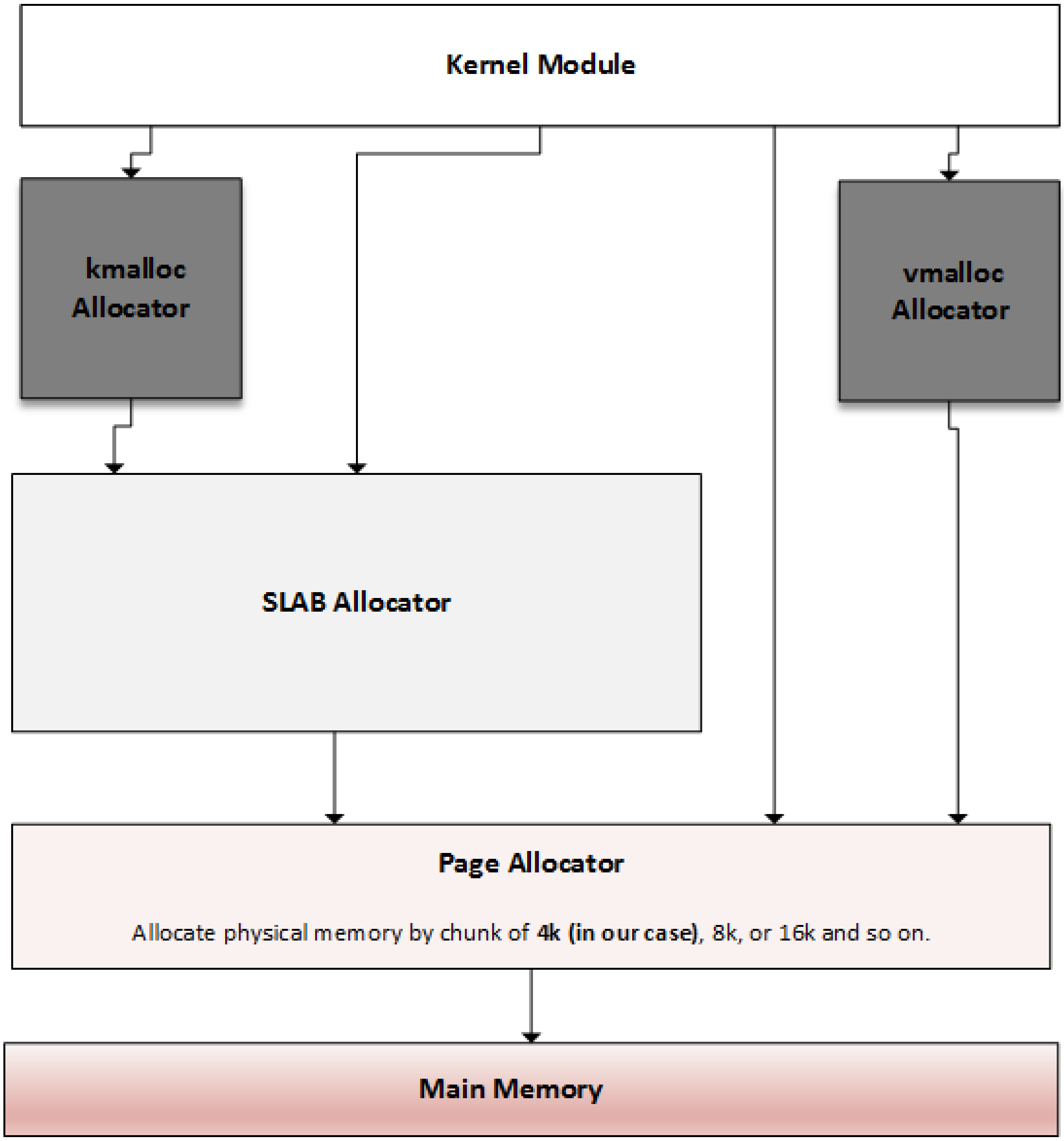

0xBAADF
0x0DA
Virtual Page Number
Offset

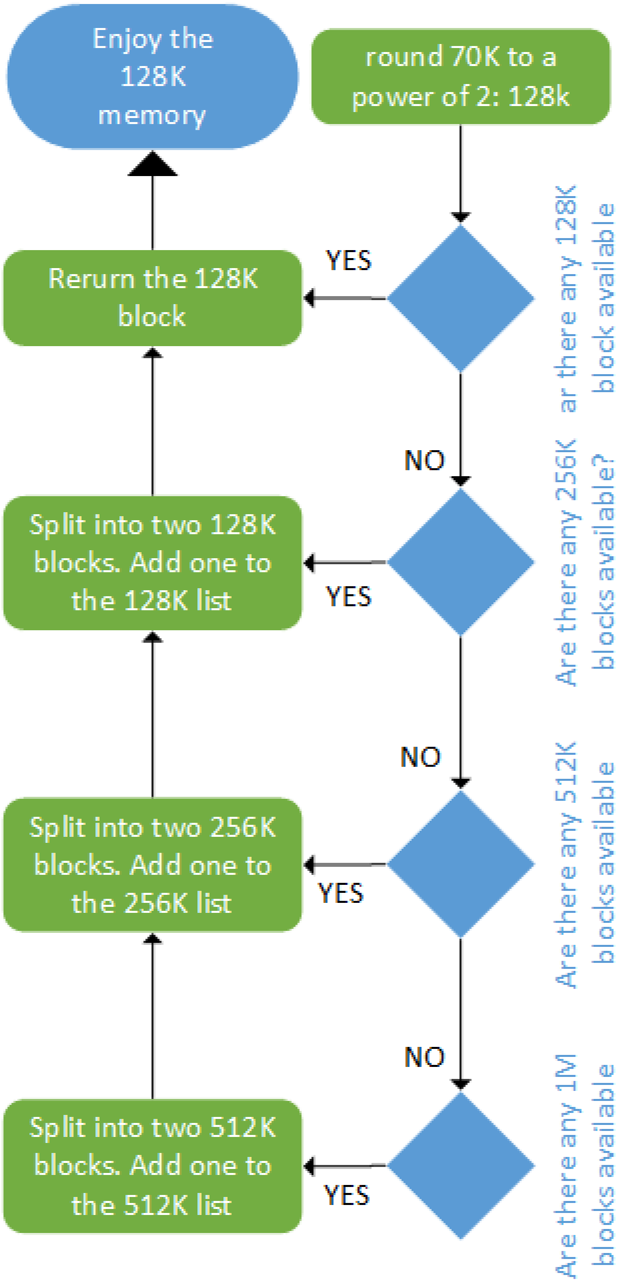

0xBAADF0DA
Virtual address
0xBAADF
Virtual Page Number
Translation table
PageTable
0x900DF
Physical Page Number
(PFN)

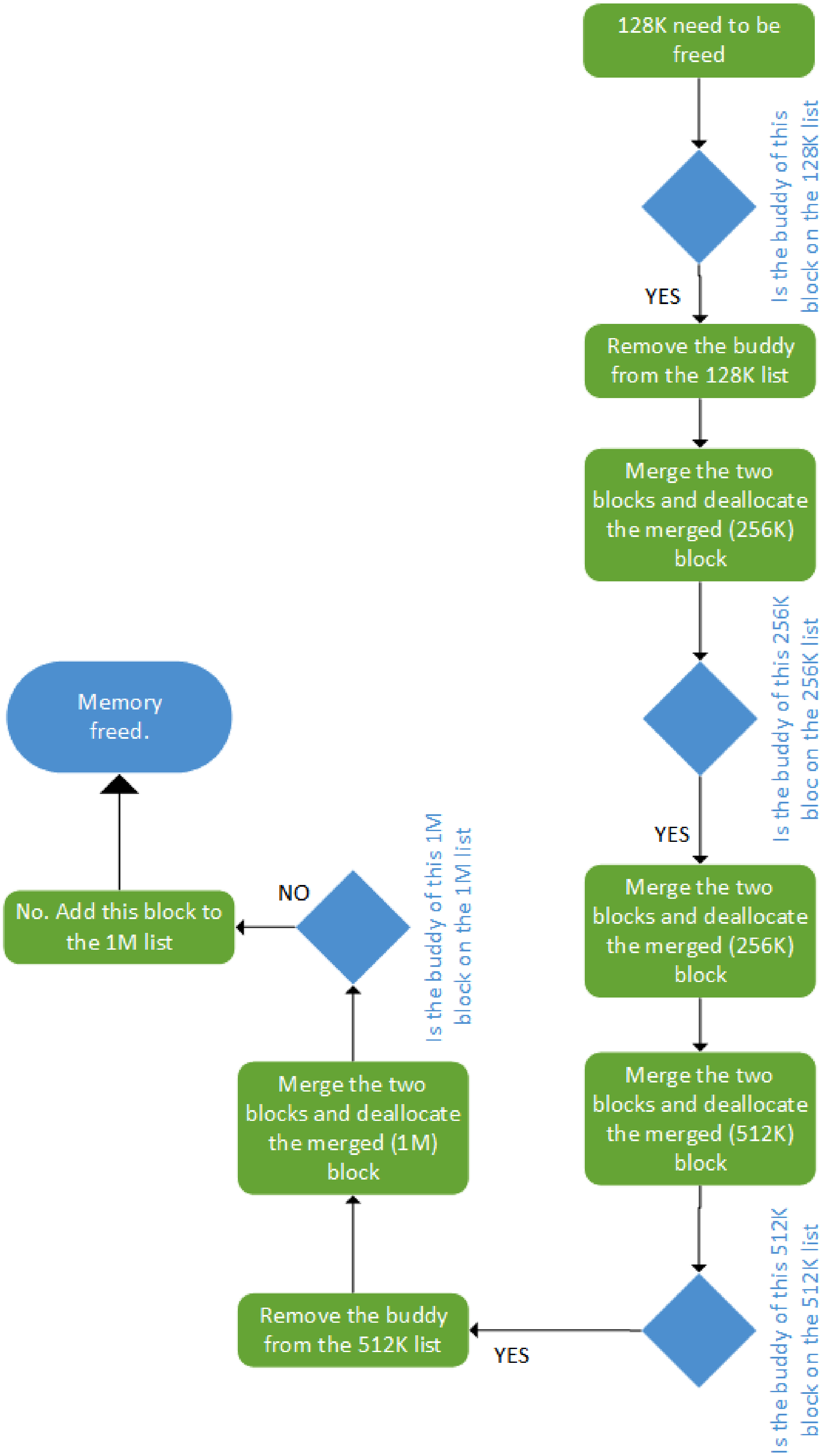

Virtual Address
Virtual page number
Offset
Page
Page Frame Number
Page
Page Frame Number
Page
Page Frame Number
Page
Page Frame Number
Page
Page Frame Number
Page Table
0x0DA
0xBAADF
Main Memory

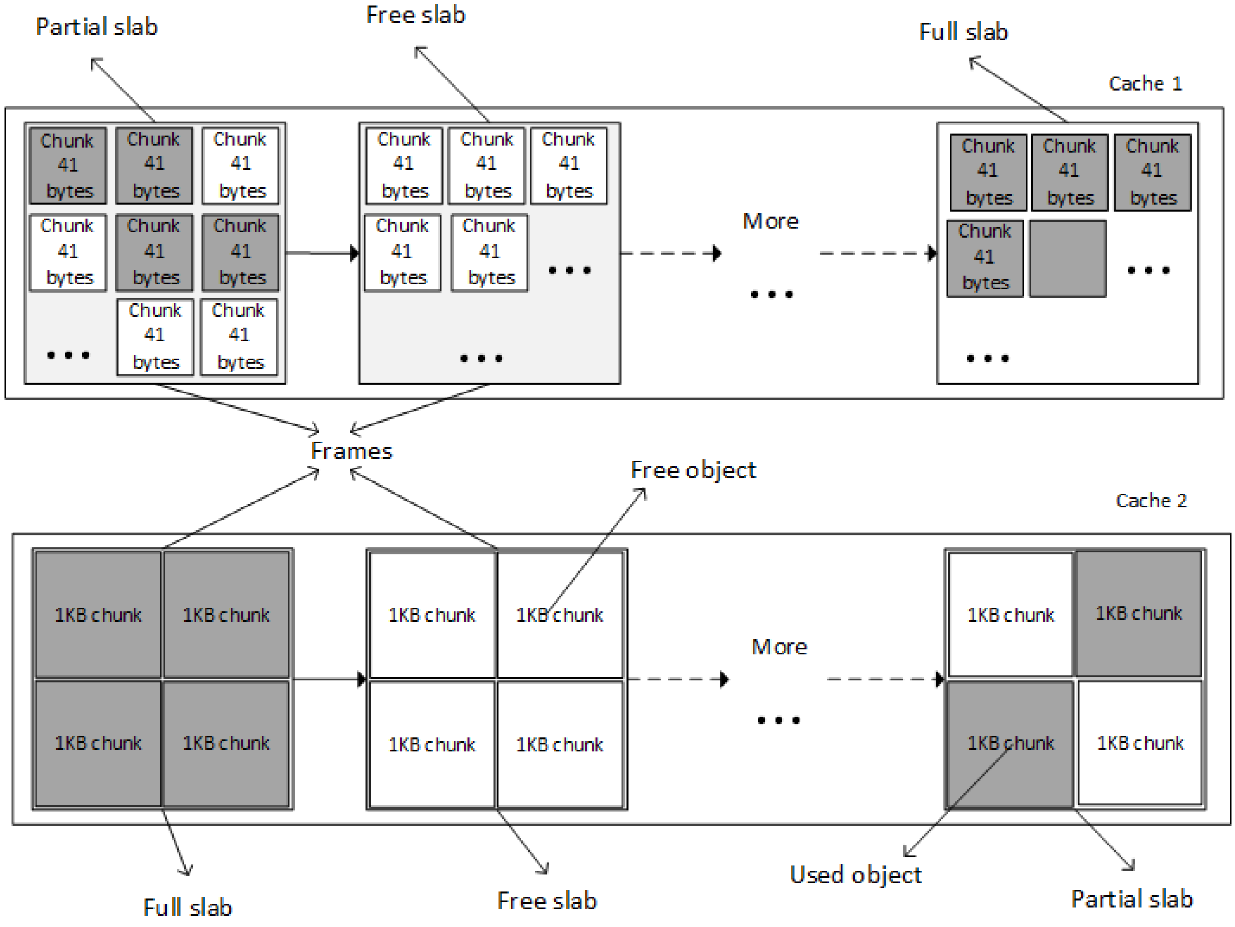

Virtual address
11
0
task_struct
mm_struct
pgd
PGD index
Page Table index
Offset
PGD
pde
Entry 0
PT
pte
Main Memory
PFN
Offset

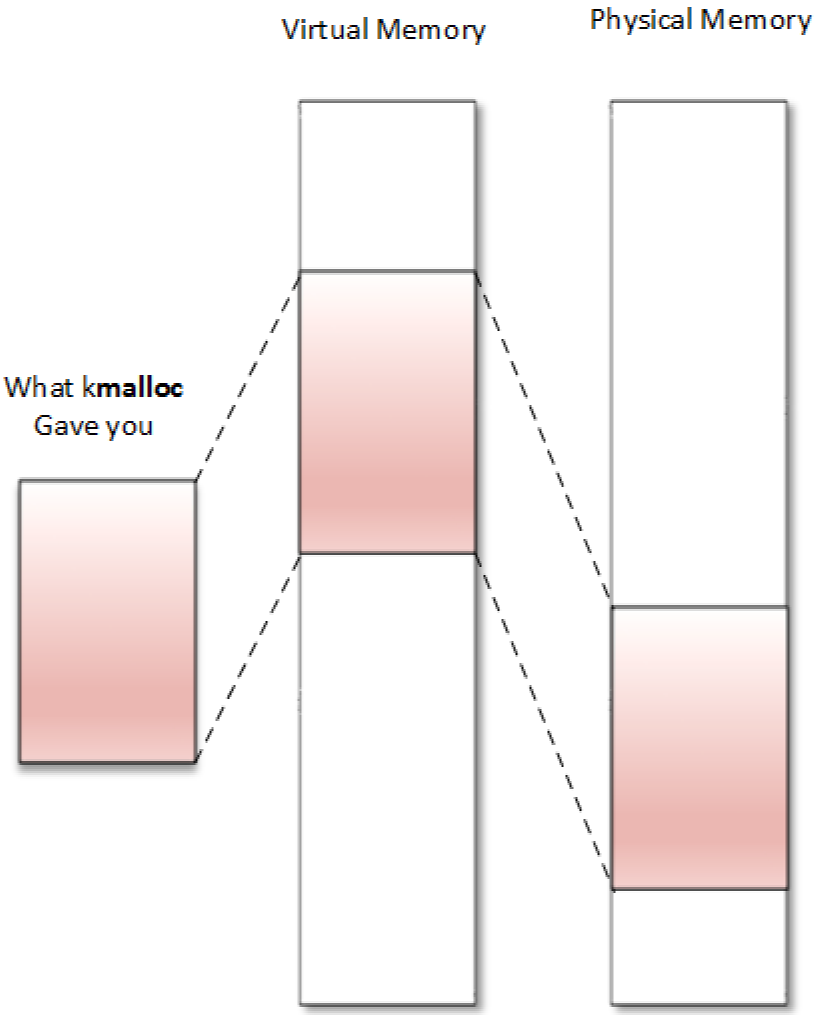

task_struct->pgd
pgd
Process
CPU
TLB
TLB Hit
Virtual address
PTBR or TTBR-0
MMU
TLB Miss
P G D
Page table PTD
Main Memory

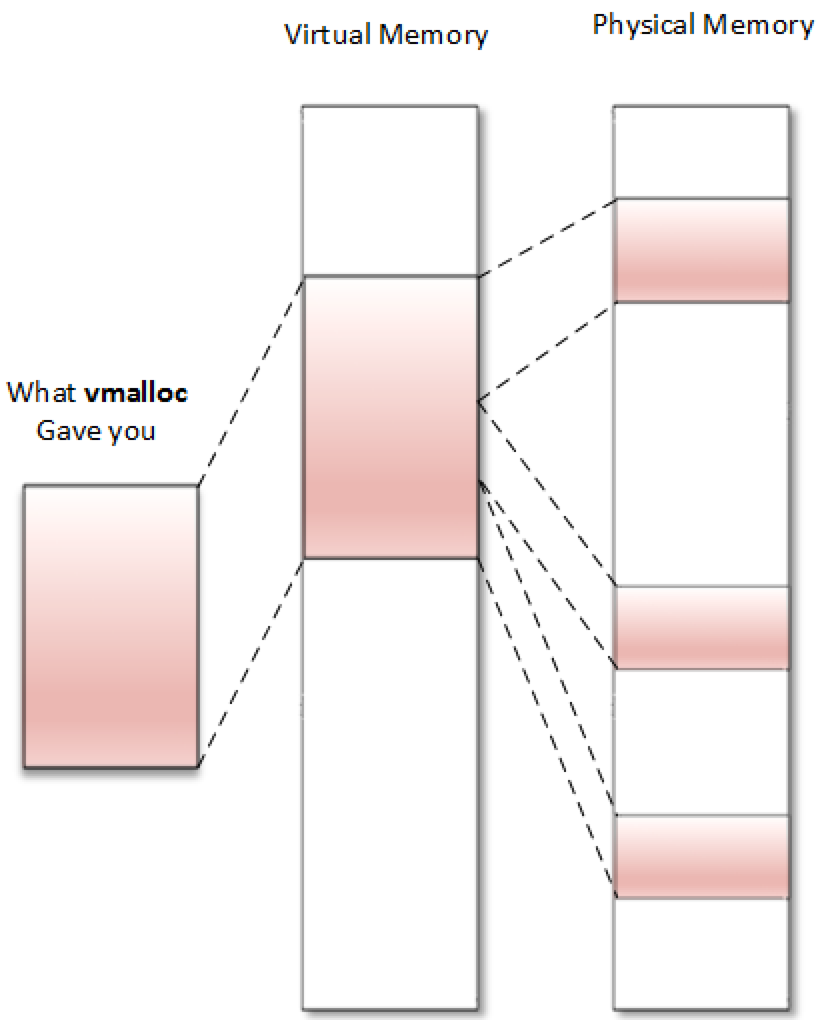

Kernel Module
kmalloc Allocator
vmalloc Allocator
SLAB Allocator
Page Allocator
Allocate physical memory by chunk of **4k (in our case)**, 8k, or 16k and so on.
Main Memory

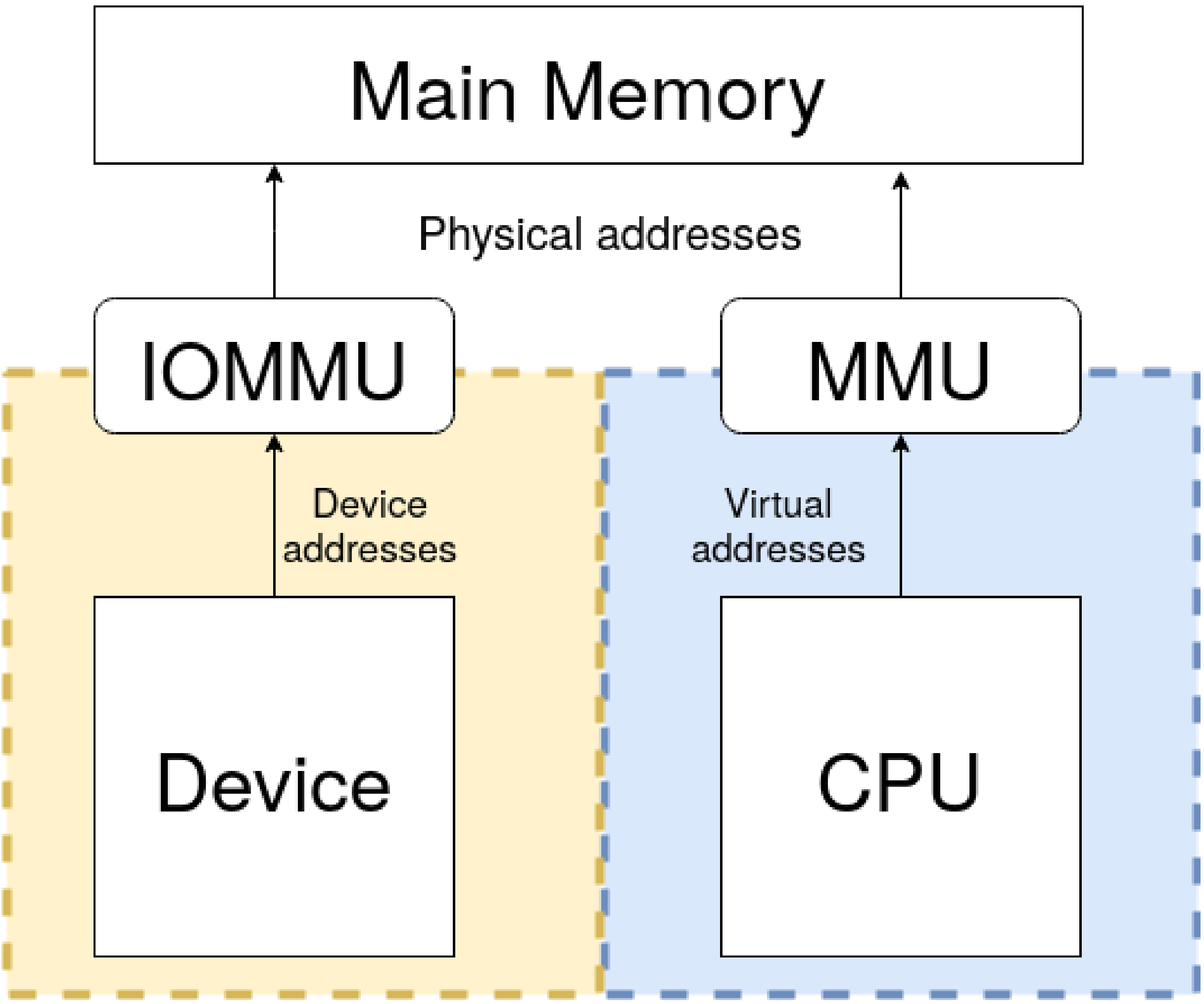

round 70K to a power of 2: 128k
ar there any 128K block available
YES
Rerurn the 128K block
Enjoy the 128K memory
NO
Are there any 256K blocks available?
YES
Split into two 128K blocks. Add one to the 128K list
NO
Are there any 512K blocks available
YES
Split into two 256K blocks. Add one to the 256K list
NO
Are there any 1M blocks available
YES
Split into two 512K blocks. Add one to the 512K list


128K need to be freed
Is the buddy of this block on the 128K list
YES
Remove the buddy from the 128K list
Merge the two blocks and deallocate the merged (256K) block
Is the buddy of this 256K bloc on the 256K list
YES
Merge the two blocks and deallocate the merged (256K) block
Merge the two blocks and deallocate the merged (512K) block
Is the buddy of this 512K block on the 512K list
YES
Remove the buddy from the 512K list
Merge the two blocks and deallocate the merged (1M) block
Is the buddy of this 1M block on the 1M list
NO
No. Add this block to the 1M list
Memory freed.

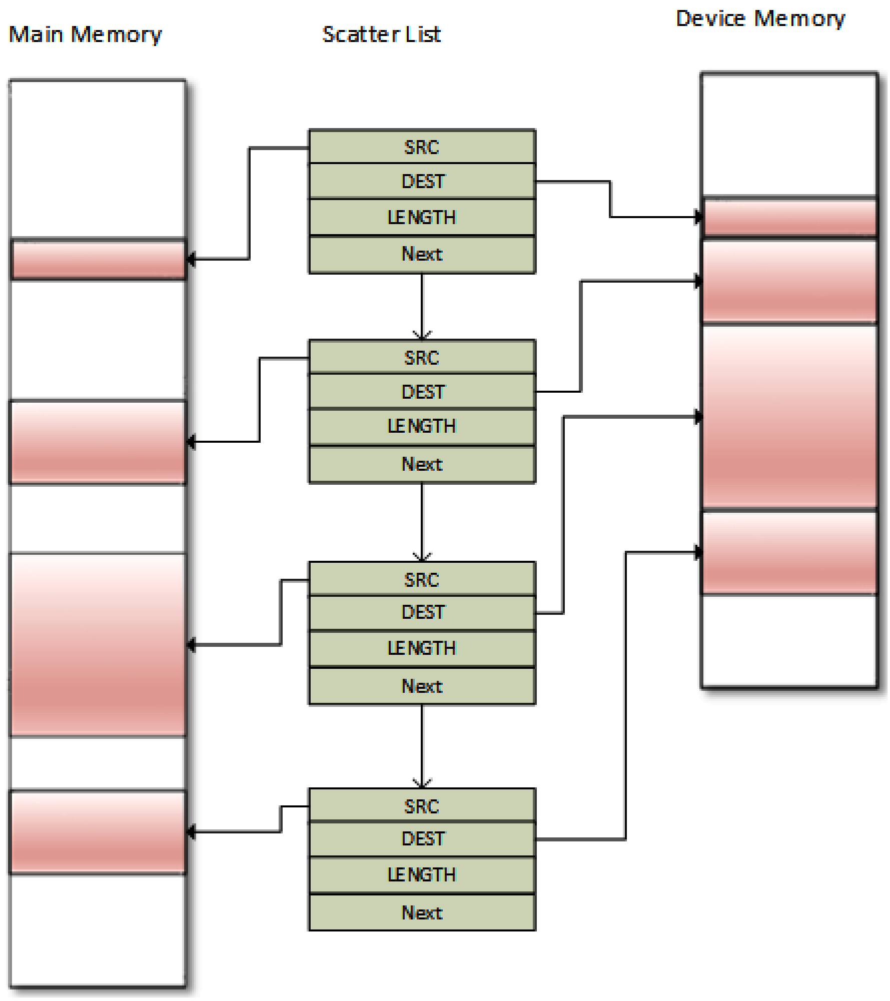

Partial slab
Free slab
Full slab
Cache 1
Chunk 41 bytes
More
Frames
Free object
Cache 2
1KB chunk
Full slab
Free slab
Used object
Partial slab

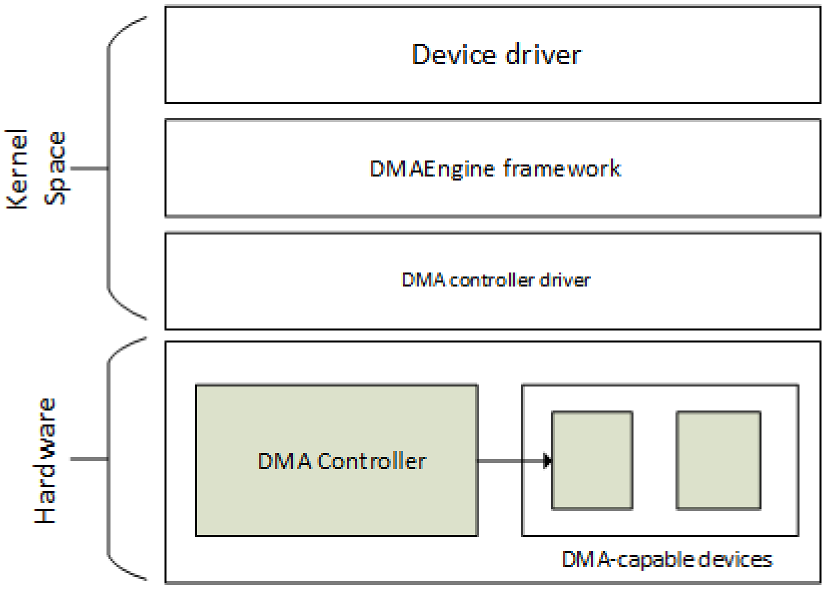

Virtual Memory
Physical Memory
What k**malloc**
Gave you

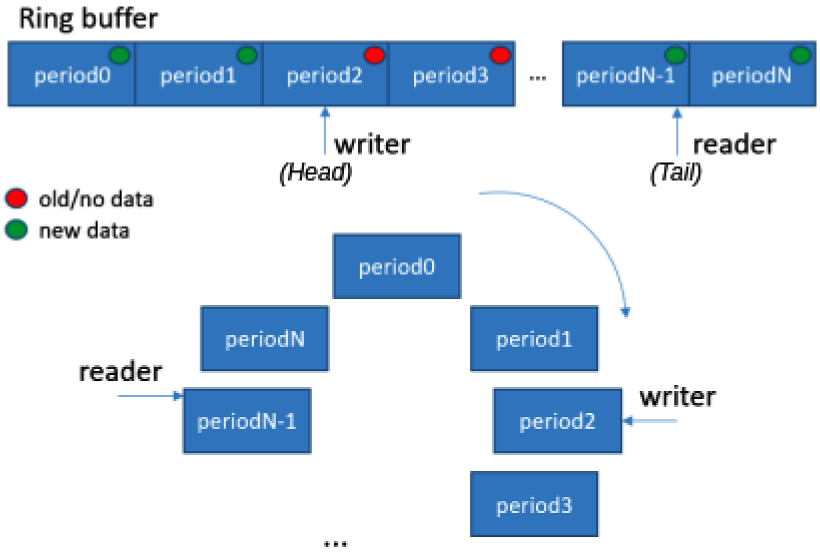

Virtual Memory
Physical Memory
What **vmalloc**
Gave you


Main Memory
Physical addresses
IOMMU
MMU
Device
addresses
Virtual
addresses
Device
CPU

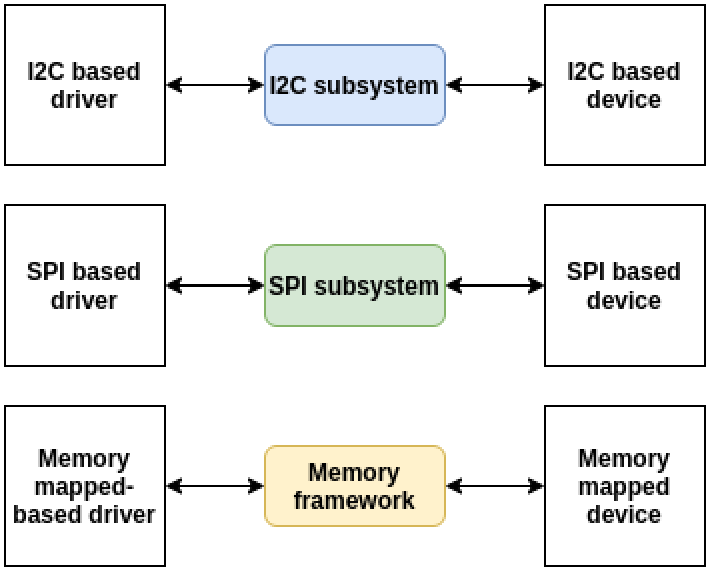
Chapter 11: Implementing Direct Memory Access (DMA) Support

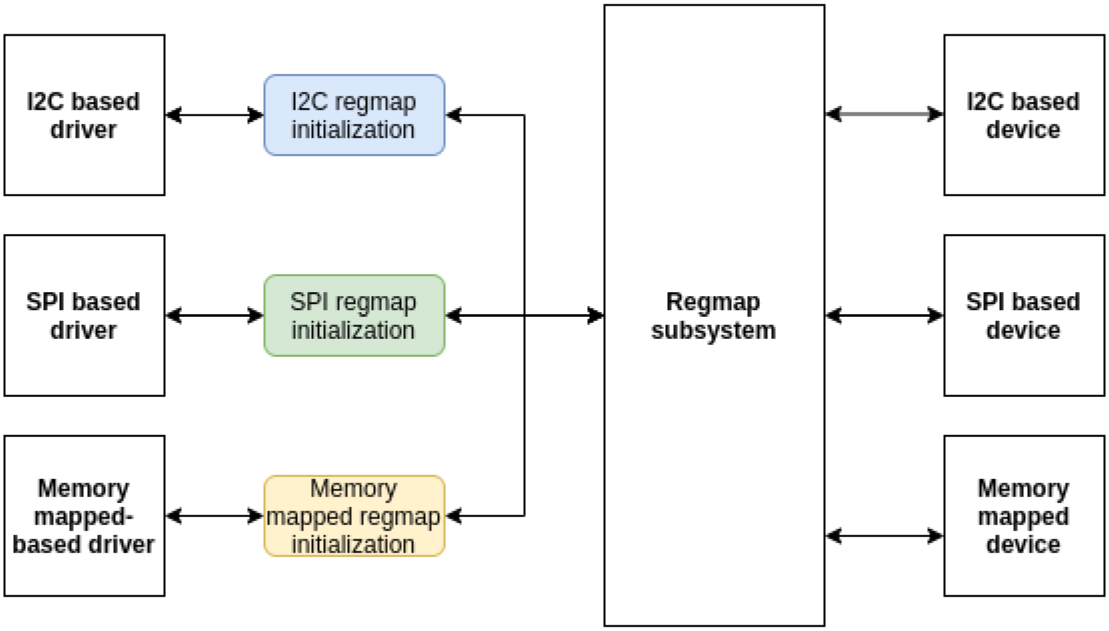



Kernel Space
Device driver
DMAEngine framework
DMA controller driver
Hardware
DMA Controller
DMA-capable devices

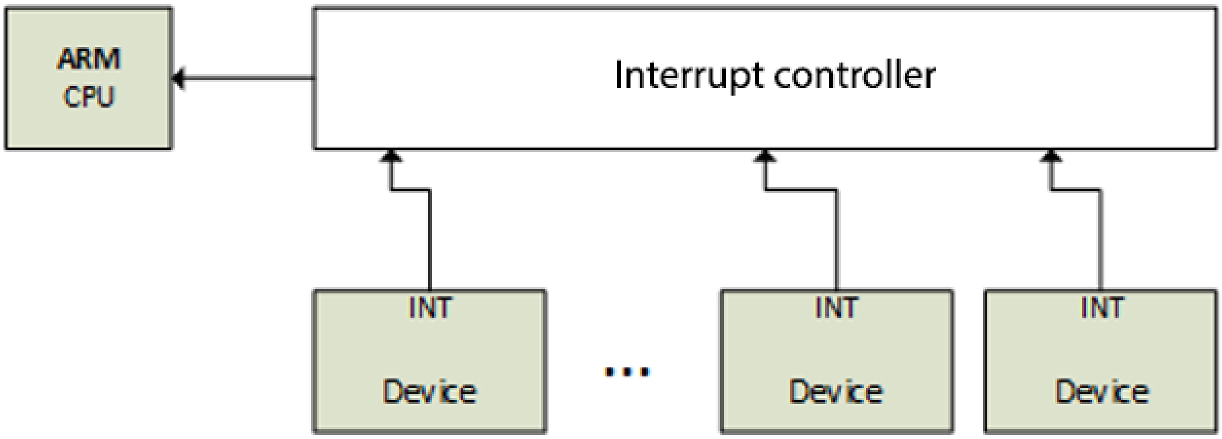

Ring buffer
period0
period1
period2
period3
...
periodN-1
periodN
writer
(Head)
reader
(Tail)
● old/no data
● new data
period0
periodN
period1
reader
periodN-1
period2
writer
period3
...

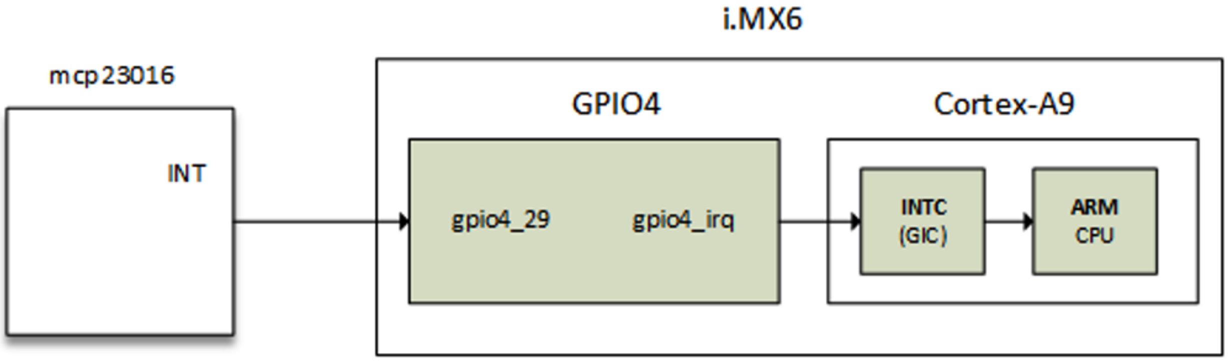
Chapter 12: Abstracting Memory Access – Introduction to the Regmap API: a Register Map Abstraction




I2C based driver
I2C regmap initialization
SPI based driver
SPI regmap initialization
Memory mapped-based driver
Memory mapped regmap initialization
Regmap subsystem
I2C based device
SPI based device
Memory mapped device

Chapter 13: Demystifying the Kernel IRQ Framework



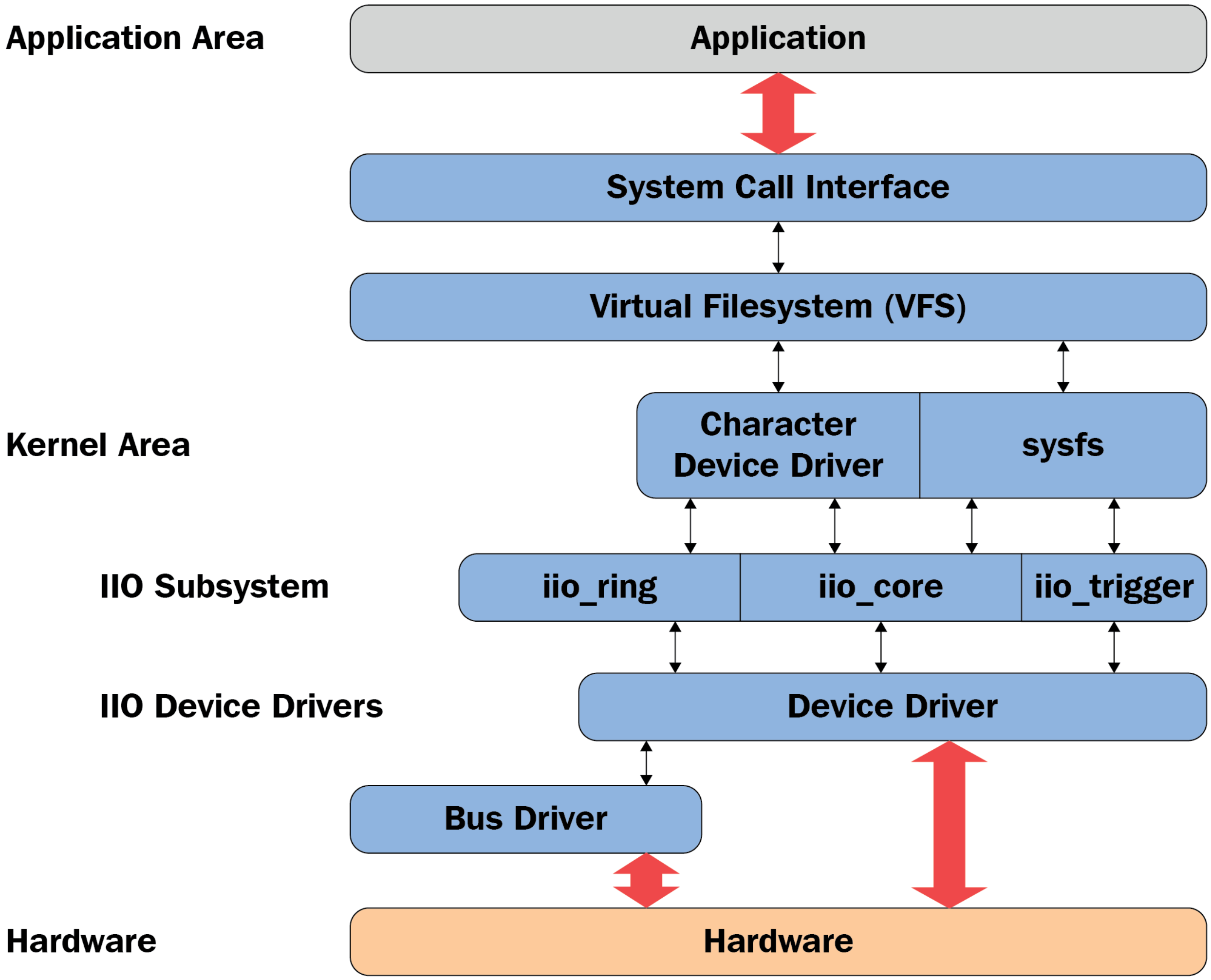


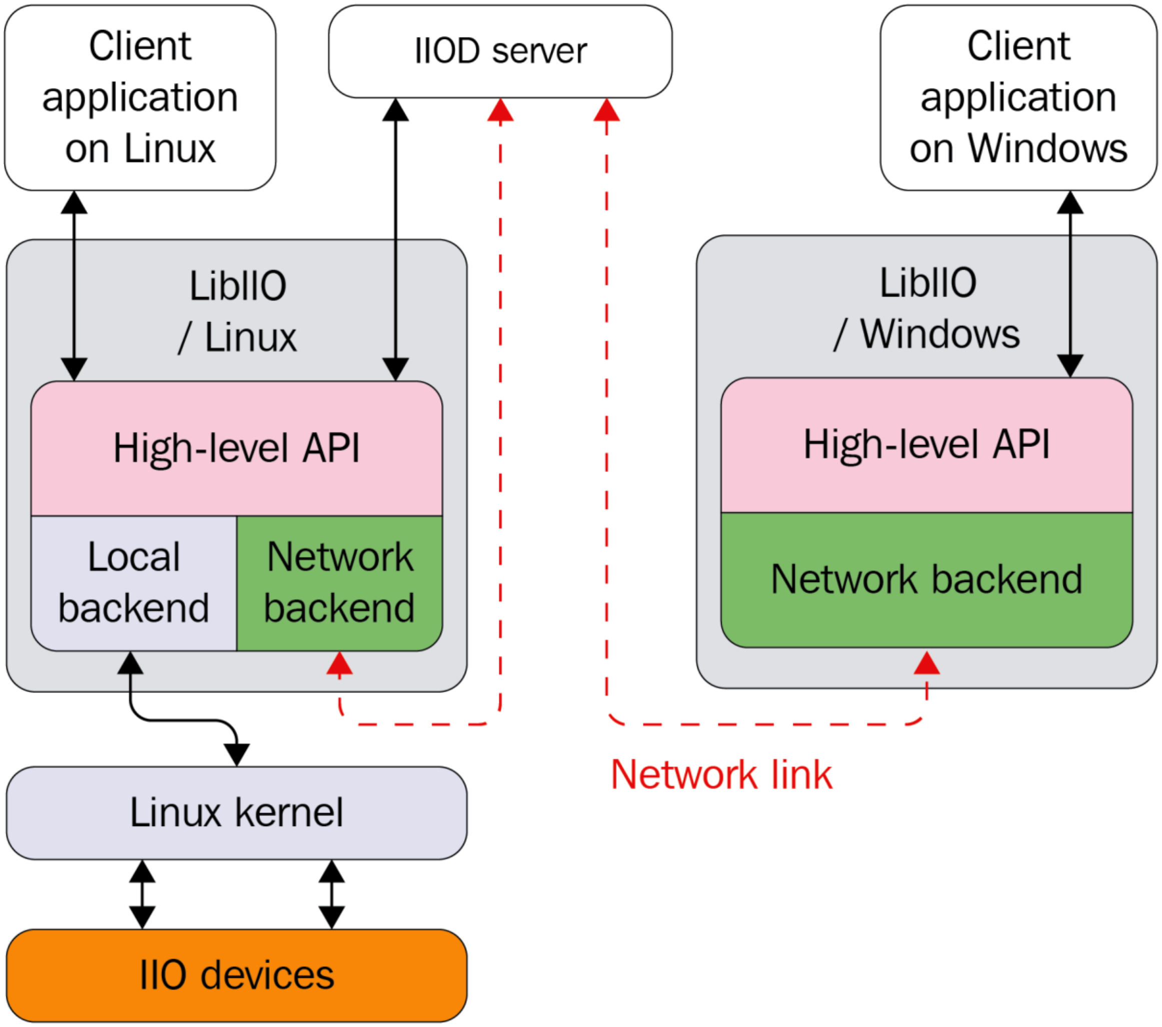
```
int irq = gpio_to_irq(125);
request_irq(irq, ...,mcp23016_irq_handler,
...);
```

Chapter 14: Introduction to the Linux Device Model

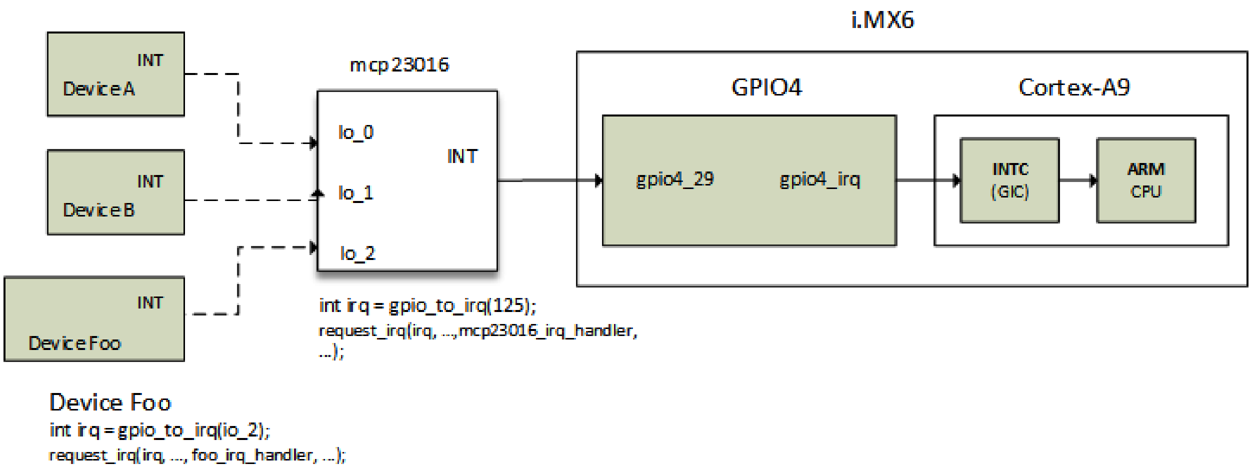
No Images…

Chapter 15: Digging into the IIO Framework




Client application on Linux
IIOD server
Client application on Windows
LibIIO / Linux
High-level API
Local backend
Network backend
LibIIO / Windows
High-level API
Network backend
Network link
Linux kernel
IIO devices

Chapter 16: Getting the Most Out of the Pin Controller and GPIO Subsystems



```
int irq = gpio_to_irq(125);
request_irq(irq, ...,mcp23016_irq_handler,
...);
```

Device Foo

```
int irq = gpio_to_irq(io_2);
request_irq(irq, ..., foo_irq_handler, ...);
```

Chapter 17: Leveraging the Linux Kernel Input Subsystem

No Images…