用 Raspberry Pi 學 Linux 驅動程式

台灣樹莓派 <sosorry@raspberrypi.com.tw>

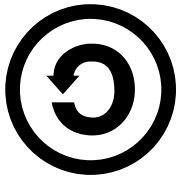
Jun 25, 2014/Raspberry Pi #5

CC (Creative Commons)

姓名標示 — 非商業性 — 相同方式分享

姓名標示 — 你必須給予 適當表彰、提供指向本授權條款的連結，以及 指出（本作品的原始版本）是否已被變更。你可以任何合理方式為前述表彰，但不得以任何方式暗示授權人為你或你的使用方式背書。

非商業性 — 你不得將本素材進行商業目的之使用。



相同方式分享 — 若你重混、轉換本素材，或依本素材建立新素材，你必須依本素材的授權條款來散布你的貢獻物。

關於台灣樹莓派

- Element14 指定台灣地區 Raspberry Pi 獨家經銷商
- 專注於 Raspberry Pi 應用與推廣
- 舉辦 Raspberry Pi 社群聚會和工作坊

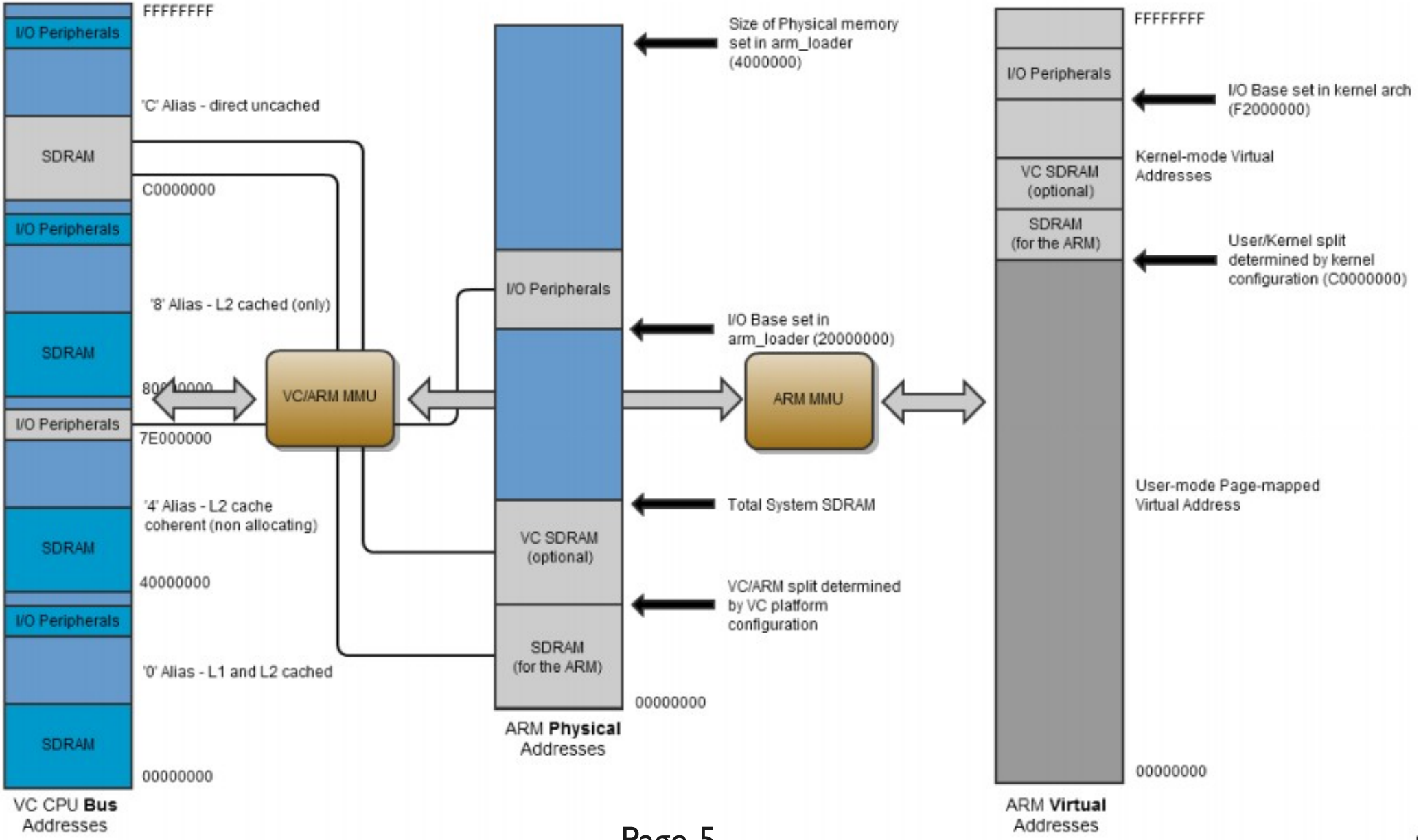
相關議程

- 深入淺出 Raspberry Pi GPIO
- 用 Raspberry Pi 體驗嵌入式系統開發

前情提要

- 如何控制 Raspberry Pi 的 GPIO
- 控制硬體 = 修改 register 的值
  1. 看 datasheet
  2. 查 register
  3. 填對應的值

BCM2835 ARM Peripherals



6.1 Register View

The GPIO has 41 registers. All accesses are assumed to be 32-bit.

Address	Field Name	Description	Size	Read/ Write
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0004	GPFSEL1	GPIO Function Select 1	32	R/W
0x 7E20 0008	GPFSEL2	GPIO Function Select 2	32	R/W
0x 7E20 000C	GPFSEL3	GPIO Function Select 3	32	R/W
0x 7E20 0010	GPFSEL4	GPIO Function Select 4	32	R/W
0x 7E20 0014	GPFSEL5	GPIO Function Select 5	32	R/W
0x 7E20 0018	-	Reserved	-	-
0x 7E20 001C	GPSET0	GPIO Pin Output Set 0	32	W
0x 7E20 0020	GPSET1	GPIO Pin Output Set 1	32	W
0x 7E20 0024	-	Reserved	-	-

找到對應的 GPFSEL table

6.1 Register View

The GPIO has 41 registers. All accesses are assumed to be 32-bit.

Address	Field Name	Description
0x 7E20 0000	GPFSEL0	GPIO Function Select 0
0x 7E20 0000	GPFSEL0	GPIO Function Select 0
0x 7E20 0004	GPFSEL1	GPIO Function Select 1
0x 7E20 0008	GPFSEL2	GPIO Function Select 2
0x 7E20 000C	GPFSEL3	GPIO Function Select 3
0x 7E20 0010	GPFSEL4	GPIO Function Select 4
0x 7E20 0014	GPFSEL5	GPIO Function Select 5
0x 7E20 0018	-	Reserved
0x 7E20 001C	GPSET0	GPIO Pin Output Set 0
0x 7E20 0020	GPSET1	GPIO Pin Output Set 1
0x 7E20 0024	-	Reserved

GPIO Register Assignment

Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0
29-27	FSEL9	FSEL9 - Function Select 9 000 = GPIO Pin 9 is an input 001 = GPIO Pin 9 is an output 100 = GPIO Pin 9 takes alternate function 0 101 = GPIO Pin 9 takes alternate function 1 110 = GPIO Pin 9 takes alternate function 2 111 = GPIO Pin 9 takes alternate function 3 011 = GPIO Pin 9 takes alternate function 4 010 = GPIO Pin 9 takes alternate function 5	R/W	0
26-24	FSEL8	FSEL8 - Function Select 8	R/W	0
23-21	FSEL7	FSEL7 - Function Select 7	R/W	0
20-18	FSEL6	FSEL6 - Function Select 6	R/W	0
17-15	FSEL5	FSEL5 - Function Select 5	R/W	0
14-12	FSEL4	FSEL4 - Function Select 4	R/W	0
11-9	FSEL3	FSEL3 - Function Select 3	R/W	0
8-6	FSEL2	FSEL2 - Function Select 2	R/W	0
5-3	FSEL1	FSEL1 - Function Select 1	R/W	0
2-0	FSEL0	FSEL0 - Function Select 0	R/W	0

GPIO Alternate function select register 0

讓 LED 一明一滅的程式流程

- map 虛擬記憶體到實體記憶體
- 初始化 PIN 為 OUTPUT
- 跑一個無窮迴圈 while

  {

    SET 該 PIN 為 HIGH

    休息一秒

    CLEAR 該 PIN

    休息一秒

  }

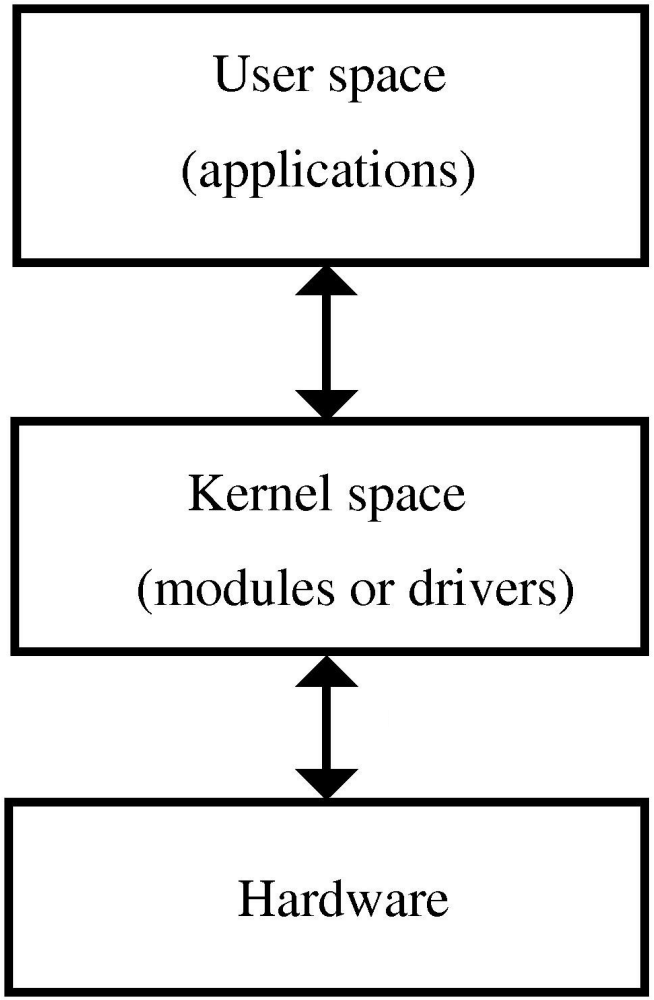
```
#define INP_GPIO(g) (*(gpio.addr + ((g)/10)) &= ~(7<<(((g)
%10)*3)))
#define OUT_GPIO(g) (*(gpio.addr + ((g)/10)) |= (1<<(((g)
%10)*3)))
#define GPIO_SET(g) (*(gpio.addr + 7))  = 1 << g
#define GPIO_CLR(g) (*(gpio.addr + 10)) = 1 << g

if (map_peripheral(&gpio) == -1)return -1;
OUT_GPIO(4);
while (1)
{
        GPIO_SET(4);
        sleep(1);

        GPIO_CLR(4);
        sleep(1);
}
```

以上為 user space application

透過 driver 進行操作
是今天的主題

什麼是 device driver ?

- 一種軟體，能讓 kernel 認識某種硬體裝置，並且讓應用程式能夠利用這個硬體
- 提供統一的存取介面
  (ex: read, write, ioctl...)



http://www.freesoftwaremagazine.com/articles/drivers_linux

寫 driver 必要的準備

被证明有用的方案
http://www.eevblog.com/forum/reviews/howto-get-the-raspian-kernel-installed-with-headers/

- driver source, Makerfile, compiler
- kernel-header or kernel source tree
  - 取得方法：從 github 下載 or 套件 (pkg)

sudo apt-get install linux-headers-3.18.0-trunk-rpi

- Module.symvers
  - 已經 exported 的 kernel 和 module 資訊
  - 取得方法：自己編譯 or 下載現成的
    - https://github.com/raspberrypi/firmware/commits/master/extra/Module.symvers

核心版本和 kernel-header 要匹配

- 從 uname -r 看目前核心版本
- 是否存在 /lib/modules/`uname -r`/build
- 懶人作法：
  - $ sudo rpi-update 更新 firmware 和 kernel
  - $ rpi-source 更新對應的 kernel-header

https://github.com/notro/rpi-source/wiki

device driver 和核心的連結方式

- 動態連結 (kernel module)
  - 以模組方式在需要時載入到核心
  - 一個 driver 對應一個 ko 檔
  - 可降低 kernel 的體積

- 靜態連結 (built-in)
  - 驅動程式將直接連接到 kernel
  - 不會產生獨立的 ko 檔

Hello World Kernel Module

hello.c

```c
#include <linux/module.h>
#include <linux/kernel.h>
int hello_init(void)
{
    printk("hello world!\n");
    return 0;
}
void hello_exit(void)
{
    printk("goodbye world!\n");
}
MODULE_LICENSE("GPL");
module_init(hello_init);
module_exit(hello_exit);
```

Makefile

```
obj-m += hello.o

all:
	make -C  /lib/modules/$(shell uname -r)/build M=$(PWD) modules


clean:
	make -C  /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

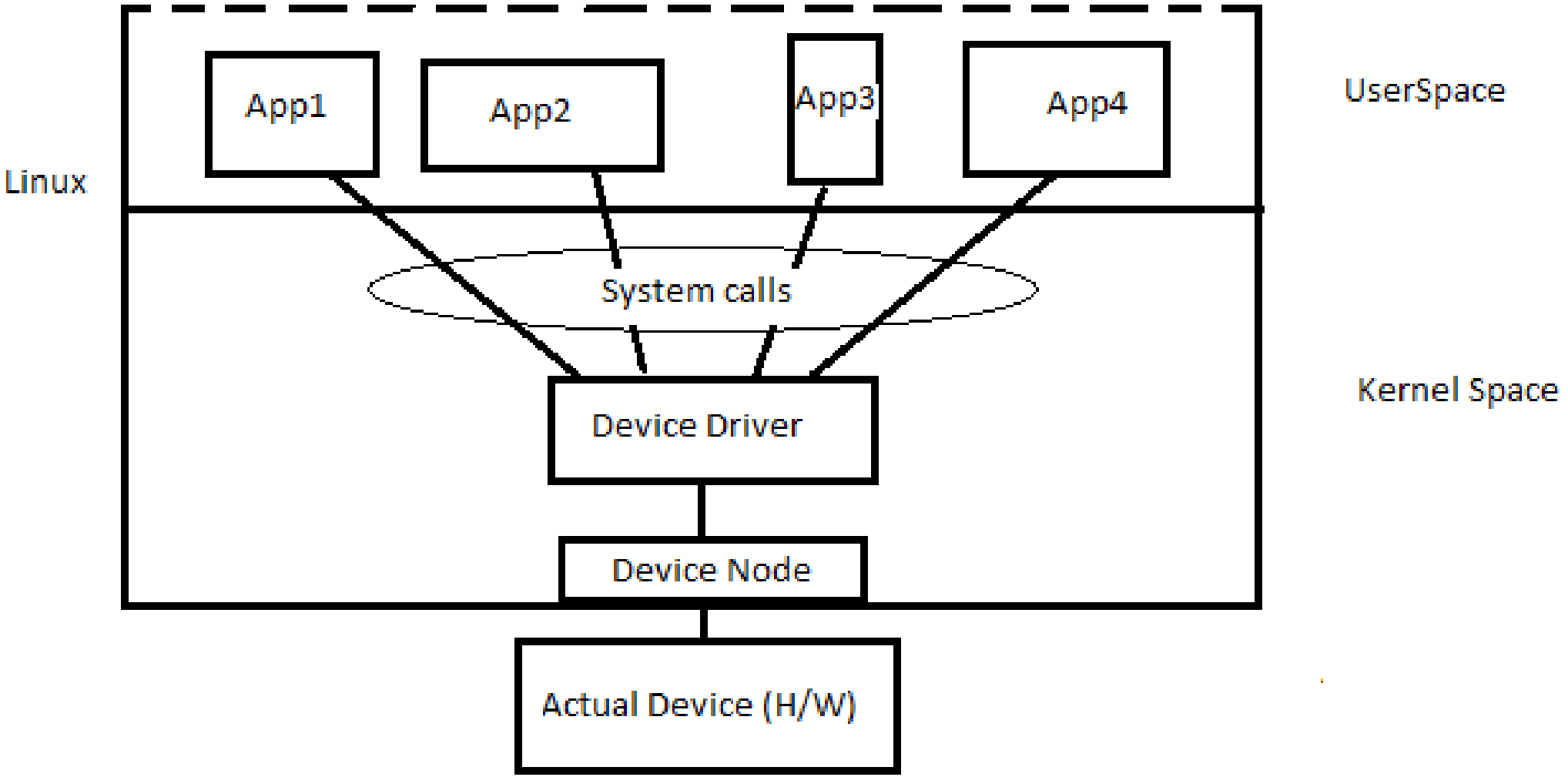
從 modinfo 看 kernel module

```
pi@raspberrypi ~ $ modinfo hello.ko
filename:       /home/pi/hello.ko
srcversion:     64AA8A235ECE3BD4EC04769
depends:
vermagic:       3.12.22+ preempt mod_unload modversions
ARMv6
```

載入並查看 kernel module

```
pi@raspberrypi ~ $ sudo insmod ./hello.ko
pi@raspberrypi ~ $ dmesg | tail
[ 805.268569] hello world!

pi@raspberrypi ~ $ cat /proc/modules | grep hello
hello 792 0 - Live 0xbf0d2000 (O)

pi@raspberrypi ~ $ sudo rmmod hello
pi@raspberrypi ~ $ dmesg | tail
[ 840.639390] goodbye world!
```

應用程式和驅動程式的關係



http://mylinuxbook.com/linux-device-files/

字元裝置驅動程式

```
pi@raspberrypi ~ $ ls -l /dev
total 0
crw------- 1 root root     10, 235 Jan  1  1970 autofs
drwxr-xr-x 2 root root         580 Jan  1  1970 block
crw------T 1 root root     10, 234 Jan  1  1970 btrfs-
control
lrwxrwxrwx 1 root root          13 Jan  1  1970 fd ->
/proc/self/fd
crw-r--r-- 1 root root      1,  11 Jan  1  1970 kmsg
srw-rw-rw- 1 root root           0 Jun 24 01:30 log
brw-rw---T 1 root disk      7,   1 Jan  1  1970 loop1
brw-rw---T 1 root disk      7,   2 Jan  1  1970 loop2
```

Everything is a file.

建立裝置節點

- user process 和 device driver 透過裝置節點交換資料
- 使用 mknod 建立裝置節點
  - 裝置名稱
  - 裝置型態 (character, block)
  - 主裝置號 (major number)
  - 次裝置號 (minor number)
- Ex: sudo mknod /dev/dht11 c 80 0

- 查詢裝置號： documentation/devices.txt

註冊與釋放裝置號

- 靜態註冊
  - register_chrdev()
- 動態註冊
  - alloc_chrdev_region()
- 釋放
  - unregister_chrdev_region()

提供系統呼叫介面

```
struct file_operations {
        struct module *owner;
        (*read) (struct file *, char __user *, size_t, loff_t
*);
        (*write) (struct file *, const char __user *, size_t,
loff_t *);
        (*poll) (struct file *, struct poll_table_struct *);
        (*ioctl) (struct inode *, struct file *, unsigned
int, unsigned long);
        (*mmap) (struct file *, struct vm_area_struct *);
        (*open) (struct inode *, struct file *);
        (*release) (struct inode *, struct file *);
};
```

實做對應的硬體操作功能

```
static ssize_t read(struct file *filp, char
*buffer, size_t length,  loff_t * offset)
{
   ...
   if (size > 8)
      copy_to_user(buf, ...);
   else
      put_user(..., buf);
   ...
}
```

驅動程式與應用程式的資料交換

- Kernel 讀寫 user space 的記憶體位置前，判斷
  - 是不是合法的位置
  - 有沒有被 swap out

- 透過核心提供的專用函數
  - copy_to_user()
  - copy_from_user()
  - put_user()
  - get_user()

這還是一個 Hello World 的模組

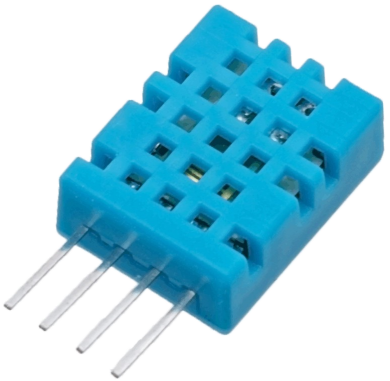
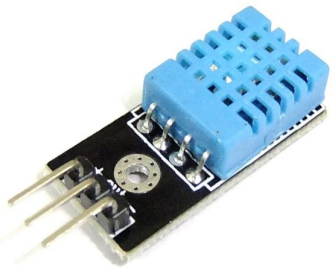
如何用 Raspberry Pi 學 Linux Driver

寫驅動程式需要和硬體打交道

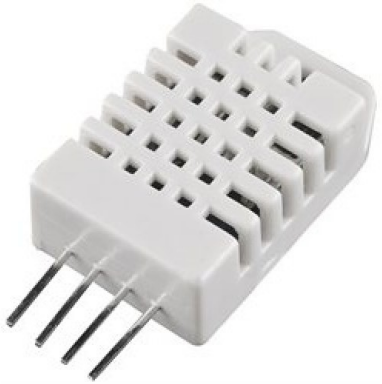
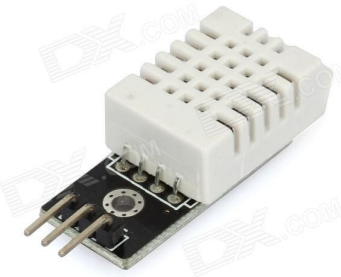
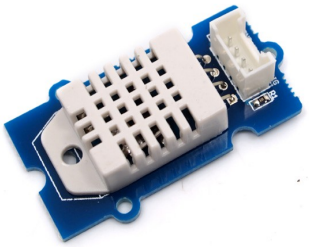
讀硬體的規格與使用的通訊協定

DHTxx 溫濕度感測器系列

- DHT11

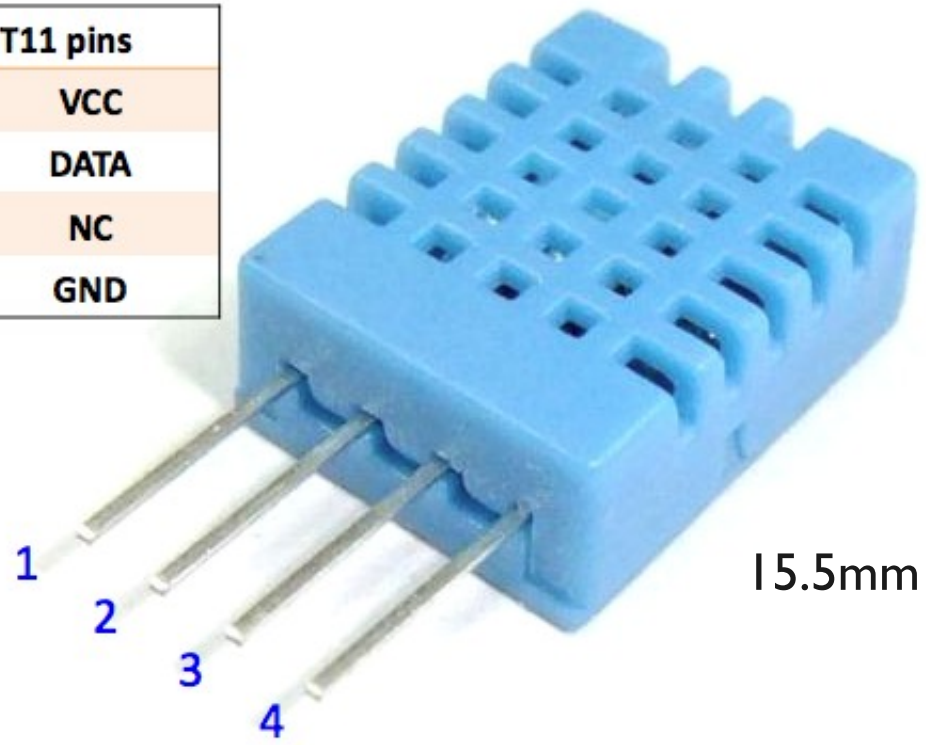
- DHT22


https://www.google.com.tw/search?q=dht11&tbm=isch
https://www.google.com.tw/search?q=dht22&tbm=isch

外觀

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



15.5mm x 12mm x 5.5mm

http://www.tortosaforum.com/raspberrypi/dht11driver.htm

規格

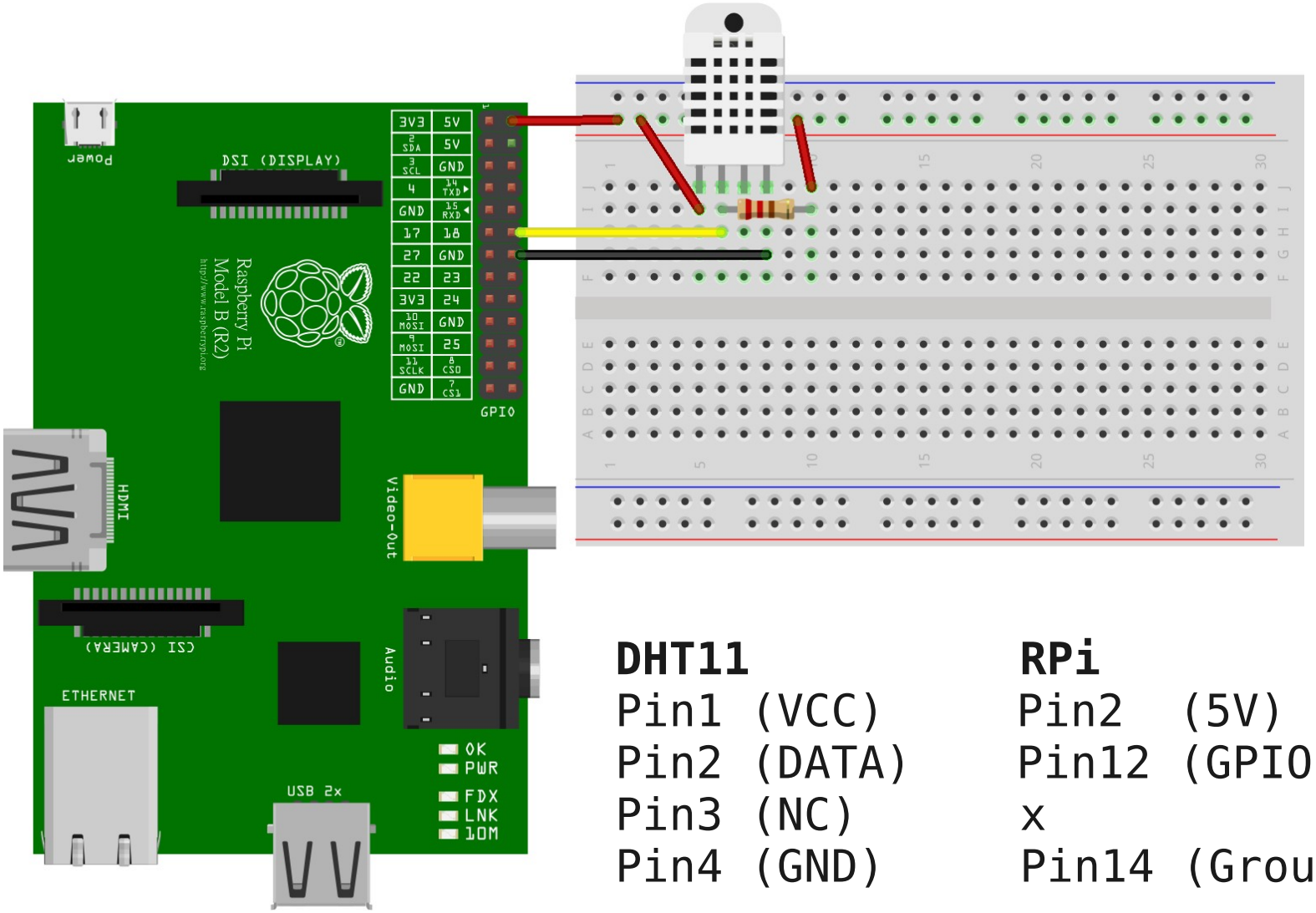
- 3 - 5.5V 工作電壓
- 2.5mA 最高工作電流
- 20 - 90% 相對濕度 (RH) 量測範圍 , ± 5%RH 誤差
- 0 - 50°C 溫度量測範圍 , ± 2°C 誤差
- 取樣頻率為 1 Hz

http://www.adafruit.com/datasheets/DHT11-chinese.pdf

可應用範圍

- 冷暖空調
- 汽車
- 氣象站
- 除濕機
- 測試及檢測設備
- 自動控制
- 醫療

線路圖

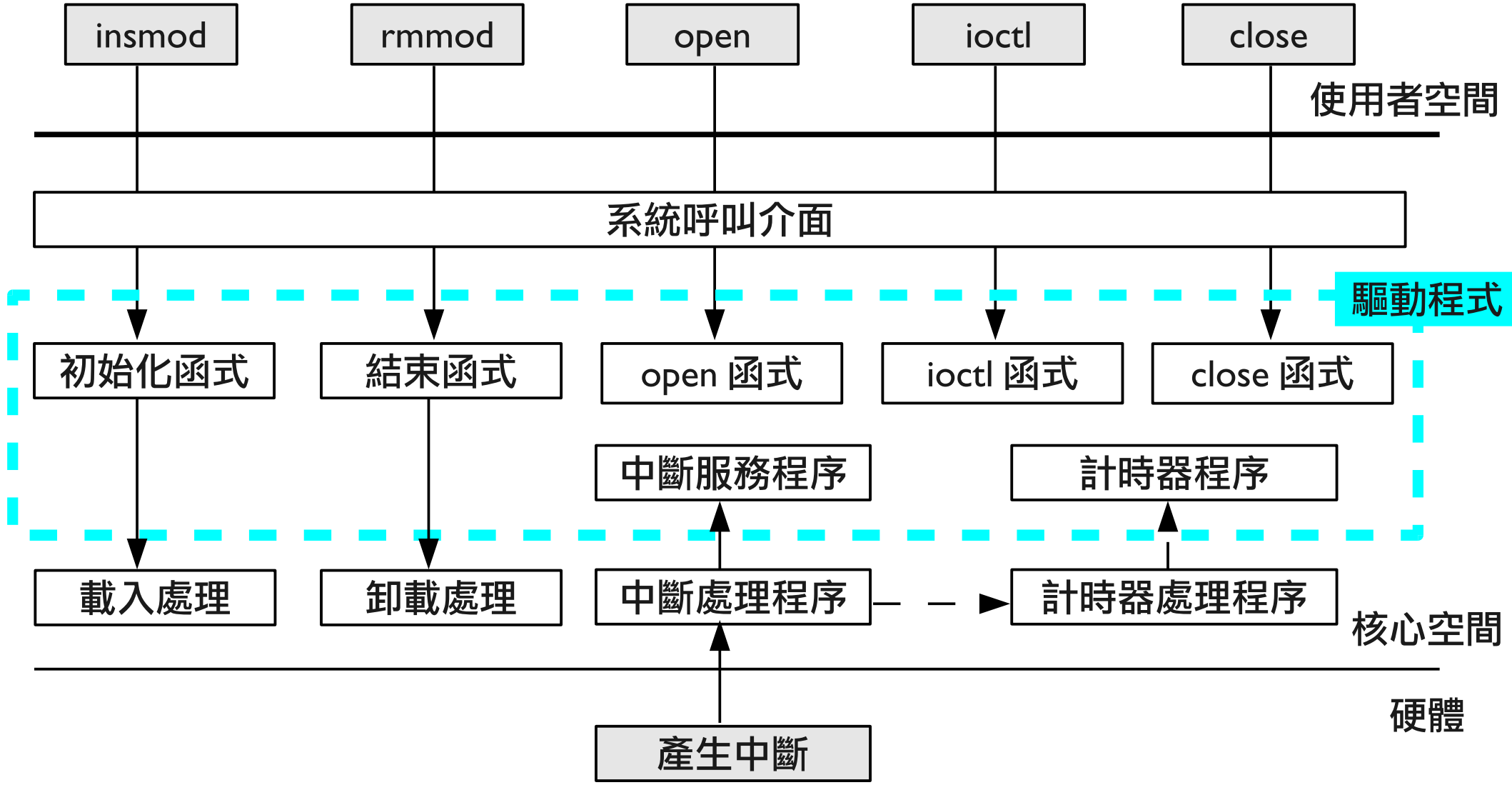


DHT11	**RPi**
Pin1 (VCC)	Pin2 (5V)
Pin2 (DATA)	Pin12 (GPIO1)
Pin3 (NC)	x
Pin4 (GND)	Pin14 (Ground)

開始寫 DHTxx 的驅動程式

從另外一個角度來看

多種進入點


insmod
rmmod
open
ioctl
close
使用者空間
系統呼叫介面
驅動程式
初始化函式
結束函式
open 函式
ioctl 函式
close 函式
中斷服務程序
計時器程序
載入處理
卸載處理
中斷處理程序
計時器處理程序
核心空間
硬體
產生中斷

為每一個進入點寫處理函式

Events	User functions	Kernel functions
load module	insmod	module_init()
open device	open()	file_operation: open()
close device	close()	file_operation: close()
read device	read()	file_operation: read()
write device	write()	file_operation: write()
ioctl device	ioctl()	file_operation: ioctl()
remove module	rmmod	module_exit()
interrupt		irq_handler()

Linux Device Driver Programming 驅動程式設計 (平田豐)

```
struct file_operations fops = {
    .open = open_dev,
    .read  = read_dev,
    .close = close_dev,
    .ioctl  = ioctl_dev
};
open_dev( );
read_dev( );
close_dev( );
ioctl_dev( );
irq_handler( );
__init( );
__exit( );
```

__init() 實做

/* 靜態註冊 major number */

- register_chrdev()

/* 保留 memory mapped i/o 位置 */

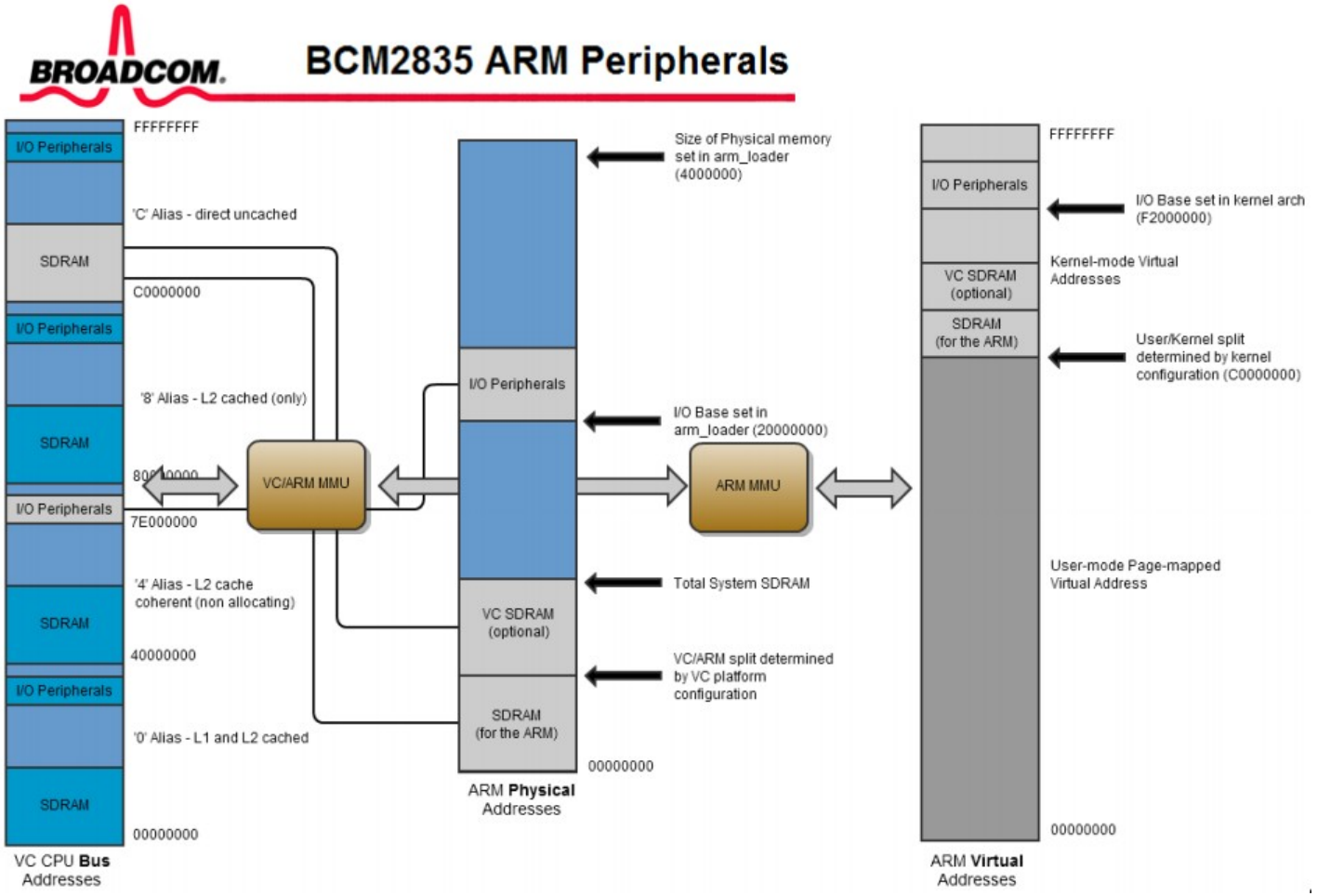
- request_mem_region()

/* 讀寫 memory mapped i/o */

- ioremap_nocache()

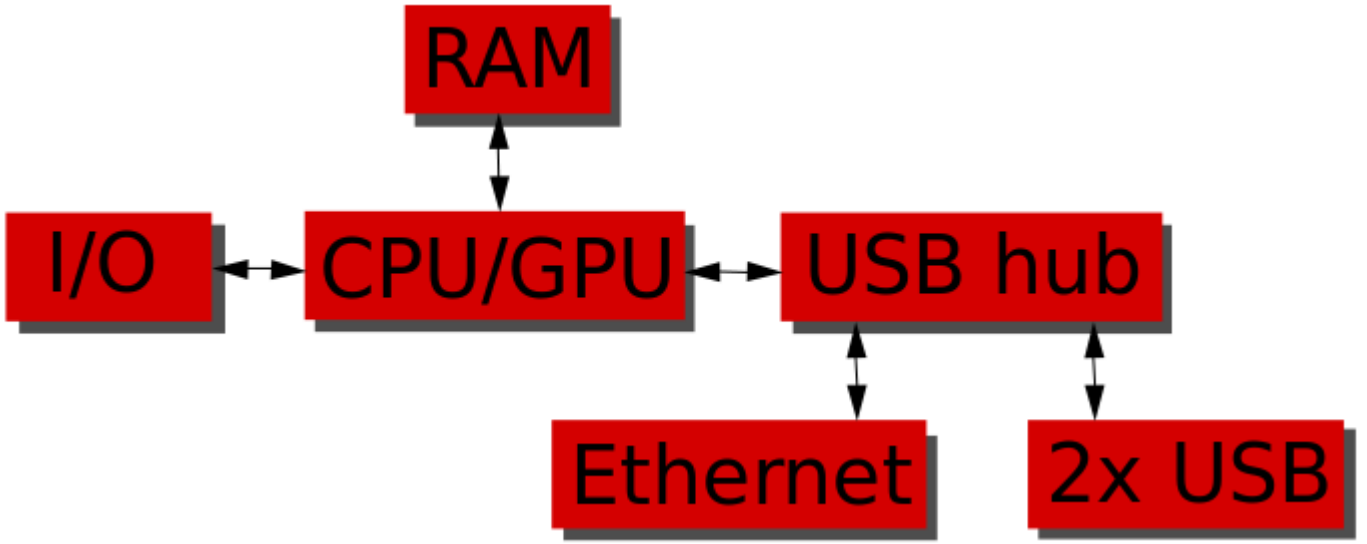
ioremap_nocache()

- 根據 datasheet，將實體記憶體位址映射到 kernel 的虛擬記憶體空間



Raspberry Pi Block Function

- 將 I/O memory mapping 到 CPU



http://en.wikipedia.org/wiki/Raspberry_Pi

__exit() 實做

/* 取消記憶體映射 */

- iounmap()

/* 取消記憶體保留 */

- release_mem_region()

/* 釋放裝置號 */

- unregister_chrdev()

open_dev() 實做

/ * setup gpio */

- GPIO_DIR_OUTPUT()
- GPIO_CLEAR_PIN()
- GPIO_SET_PIN()
- GPIO_DIR_INPUT()

/* setup interrupt */

- request_irq()
- GPIO_INT_RISING()
- GPIO_INT_FALLING()
- GPIO_INT_CLEAR()

read_dev() 實做

/ * 將資料送到 user space */

- put_user()

如何讀取 DHTxx 的資料？

從通訊協定了解起

1-Wire Protocol

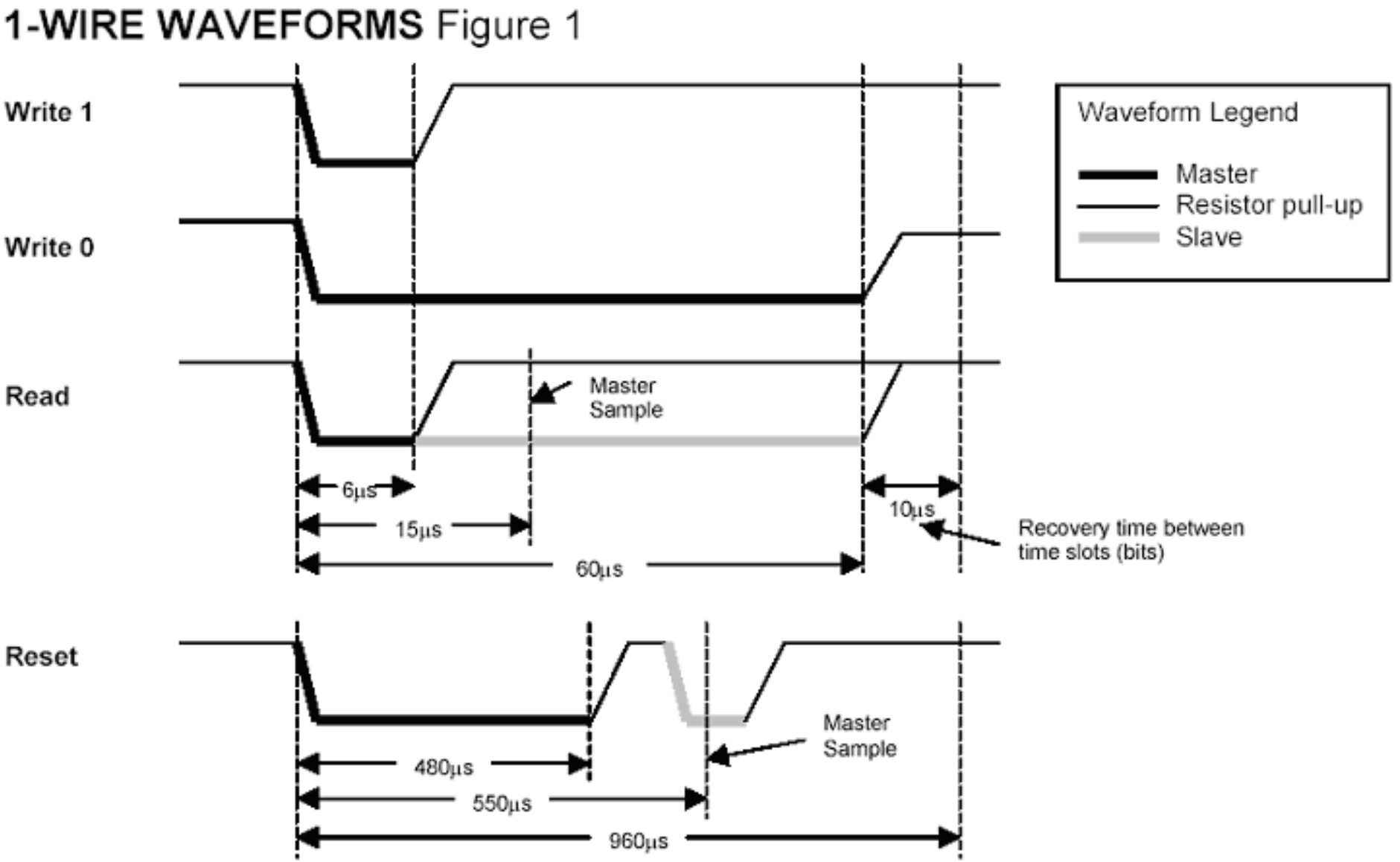
1-Wire Operation Table (範例)

1-WIRE OPERATIONS Table 1

Operation	Description	Implementation
Write 1 bit	Send a '1' bit to the 1-Wire slaves (Write 1 time slot)	Drive bus low, delay 6µs Release bus, delay 64µs
Write 0 bit	send a '0' bit to the 1-Wire slaves (Write 0 time slot)	Drive bus low, delay 60µs Release bus, delay 10µs
Read bit	Read a bit from the 1-Wire slaves (Read time slot)	Drive bus low, delay 6µs Release bus, delay 9µs Sample bus to read bit from slave Delay 55µs
Reset	Reset the 1-Wire bus slave devices and ready them for a command	Drive bus low, delay 480µs Release bus, delay 70µs Sample bus, 0 = device(s) present, 1 = no device present Delay 410µs

http://coecsl.ece.illinois.edu/ge423/sensorprojects/1-wire_full.doc

1-Wire Waveform



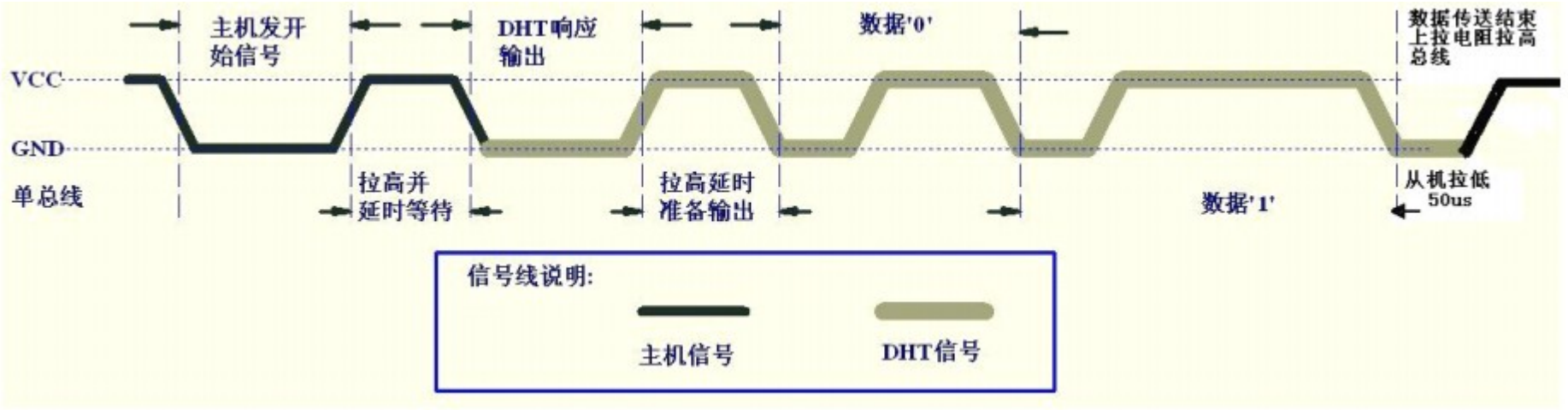
http://coecsl.ece.illinois.edu/ge423/sensorprojects/1-wire_full.doc

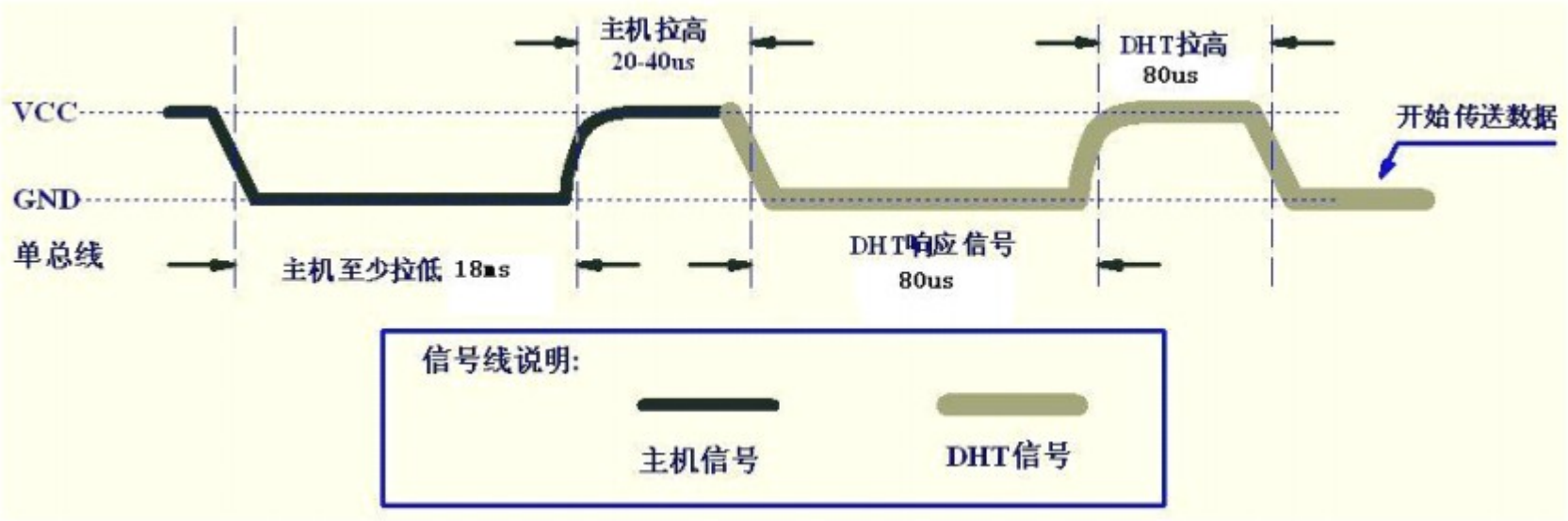
DHTxx Communication Protocol

- 1-wire protocol
- 每次通訊時間： 4ms
- 數據格式： 40bit, MSB
  - 8bit 濕度整數 +8bit 濕度小數
  - 8bit 溫度整數 +8bit 溫度小數
  - 8bit CRC

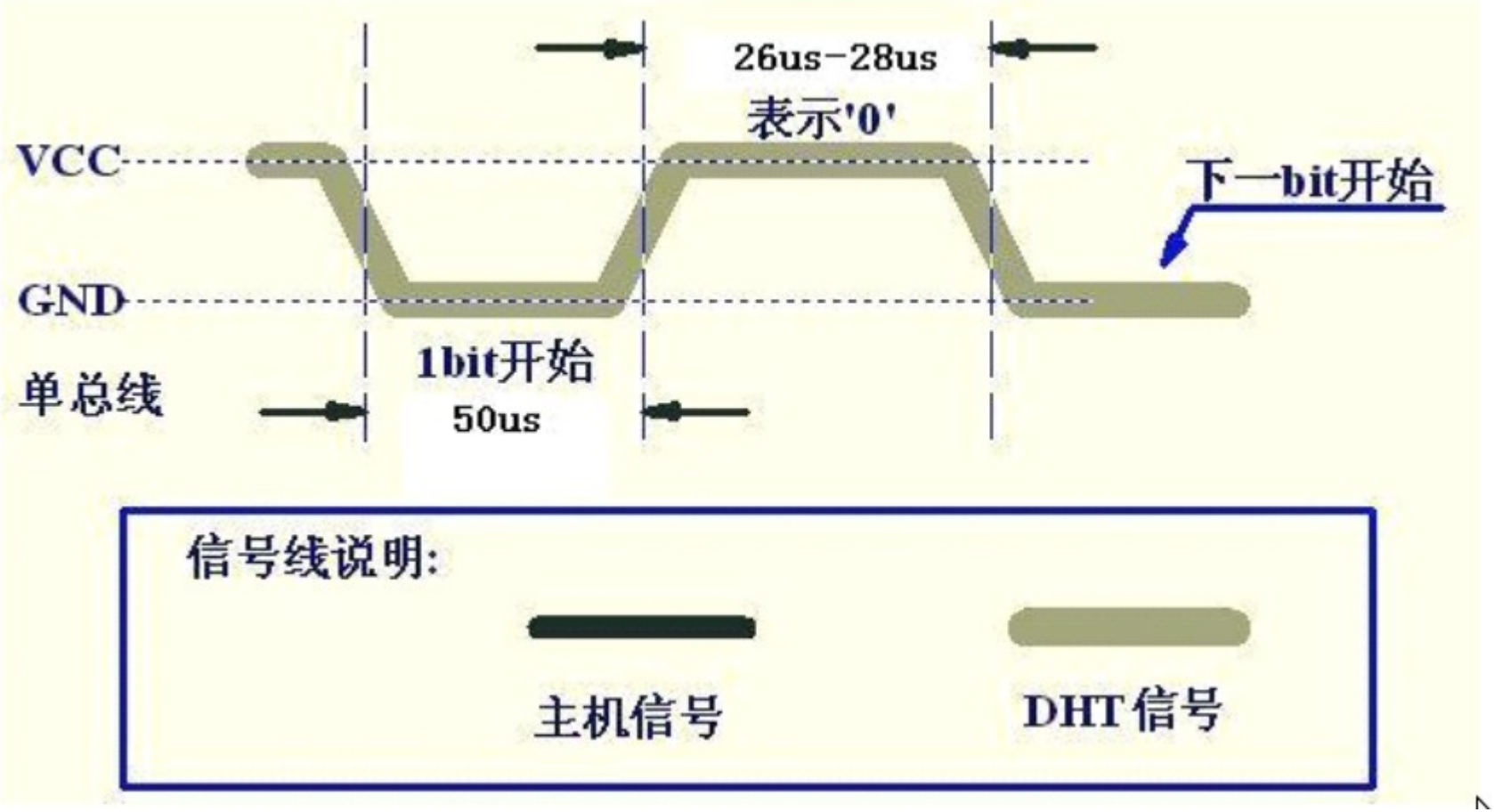
http://www.adafruit.com/datasheets/DHT11-chinese.pdf

DHTxx Communication Protocol

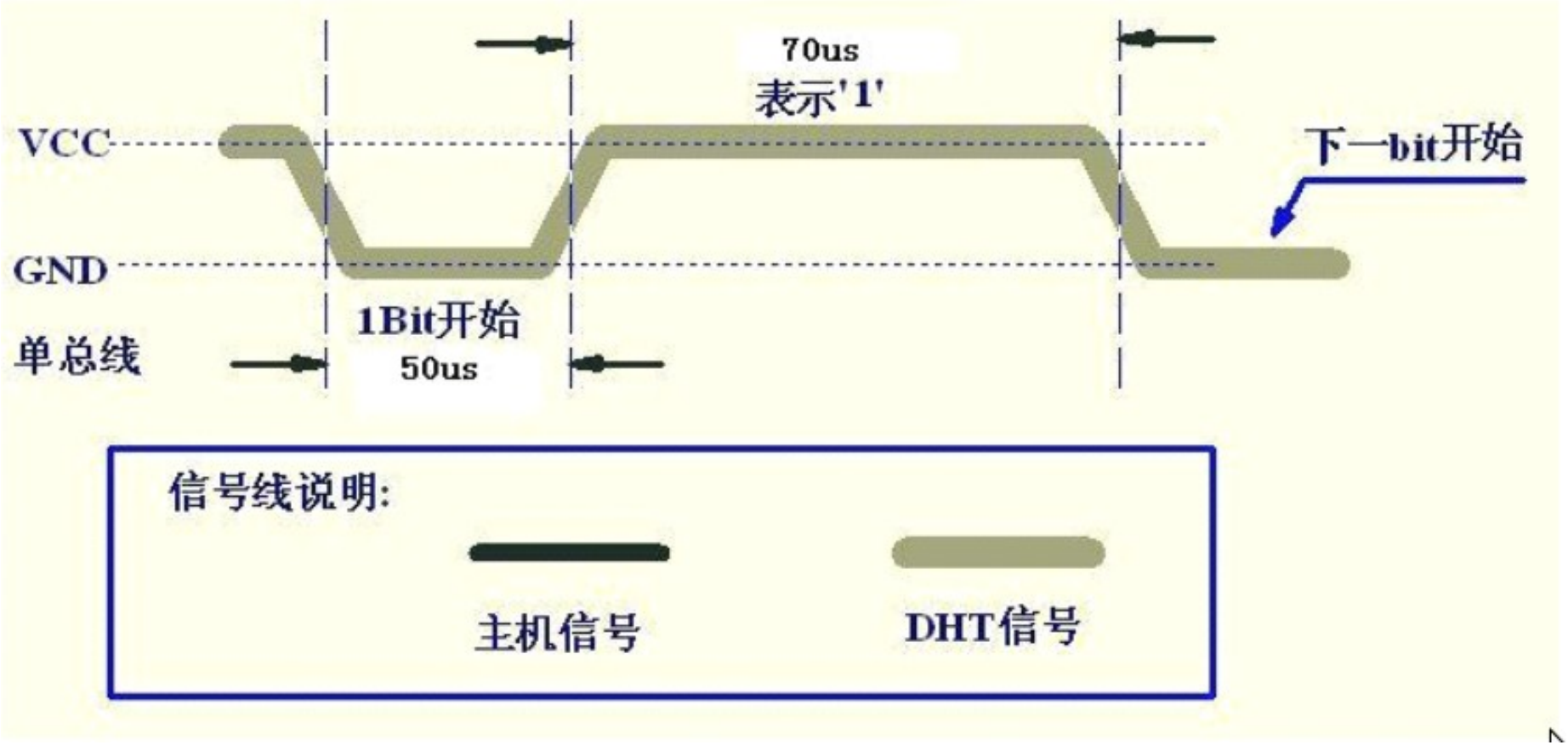




Send 0

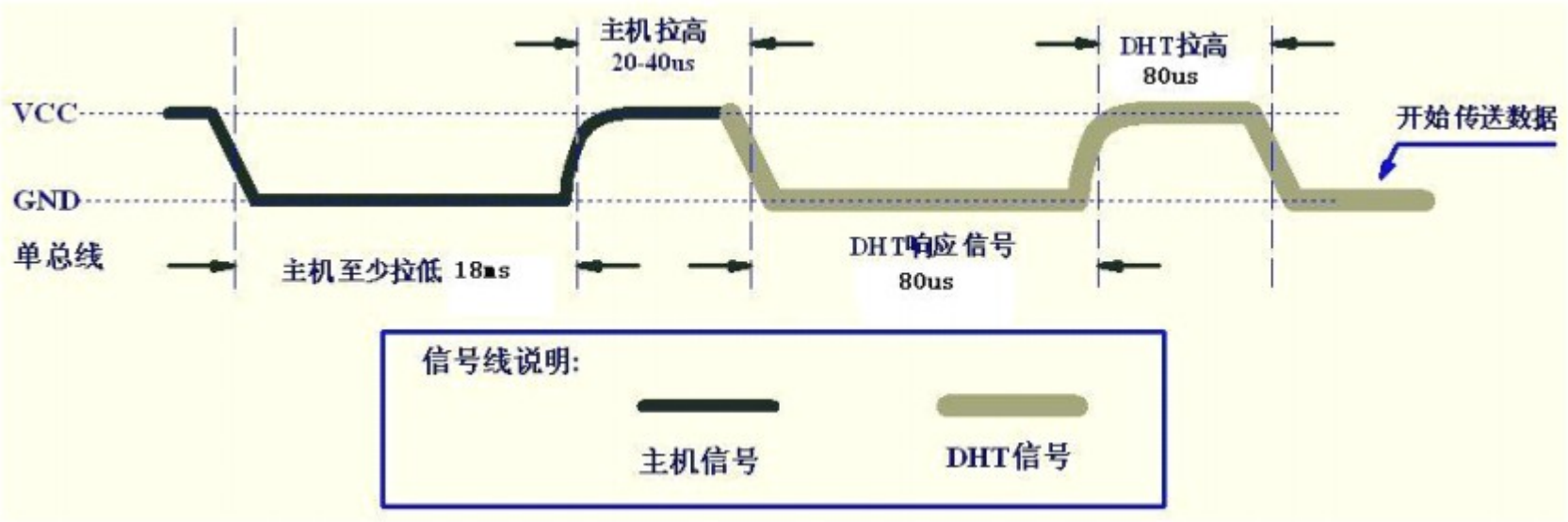

26us-28us
表示'0'
VCC
下一bit开始
GND
1bit开始
50us
单总线
信号线说明:
主机信号
DHT信号

Send 1



實做 protocol

- 通訊初始化
- Pi 開始讀取信號



- 通訊初始化步驟
  - 設定 Pin 為 output
  - Pin 拉 low
  - delay 18ms
  - Pin 拉 high
  - delay 40us
- 開始讀取信號

開始讀取信號是要一直問嗎？

- 輪詢 (polling)
  - SoC 每隔一段時間檢查週邊硬體的資料

- 中斷 (interrupt)
  - 當週邊硬體的狀態改變時，通知 SoC

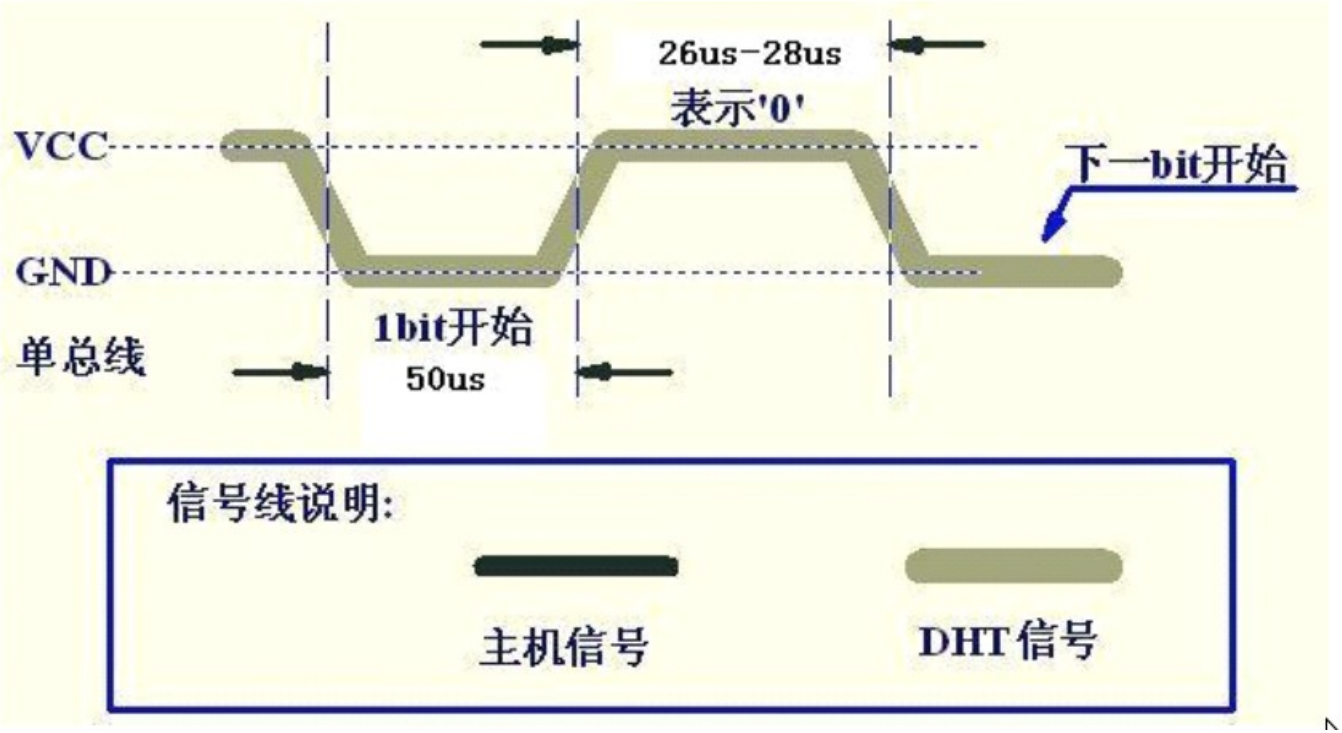
程式碼片段

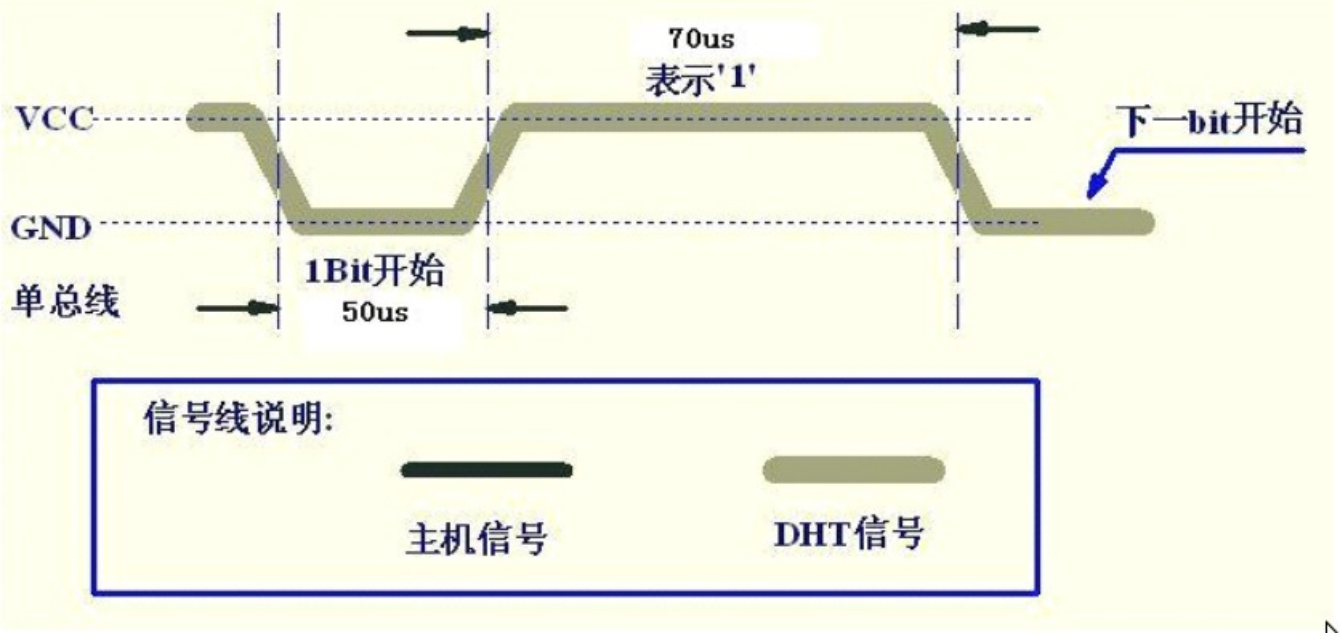
```
int open_dht11(...)
{
    ...
    GPIO_DIR_OUTPUT(gpio_pin);
    GPIO_CLEAR_PIN(gpio_pin);
    mdelay(18);
    GPIO_SET_PIN(gpio_pin);
    udelay(40);
    GPIO_DIR_INPUT(gpio_pin);
    setup_interrupt();
    ...
}
```

通訊初始化

所有電位高低的改變都會產生中斷

```
int setup_interrupts()
{
    unsigned long flags;
    request_irq(INTERRUPT_GPIO0, (irq_handler_t)
    irq_handler, 0, DHT11_DRIVER_NAME, (void*) gpio);
    GPIO_INT_RISING(gpio_pin, 1);
    GPIO_INT_FALLING(gpio_pin, 1);
    GPIO_INT_CLEAR(gpio_pin);
    ...
}
```


26us-28us
表示'0'
VCC
下一bit开始
GND
1bit开始
50us
单总线
信号线说明:
主机信号
DHT信号


70us
表示'1'
VCC
下一bit开始
GND
1Bit开始
50us
单总线
信号线说明:
主机信号
DHT信号

分辨 DHT 傳送的訊號

- DHT 拉 low，持續 50us，為開始信號
- 若 DHT 拉 high，持續 26-28us，為傳送 0
- 若 DHT 拉 high，持續 70us，為傳送 1
- 其他表示雜訊，或讀取錯誤等

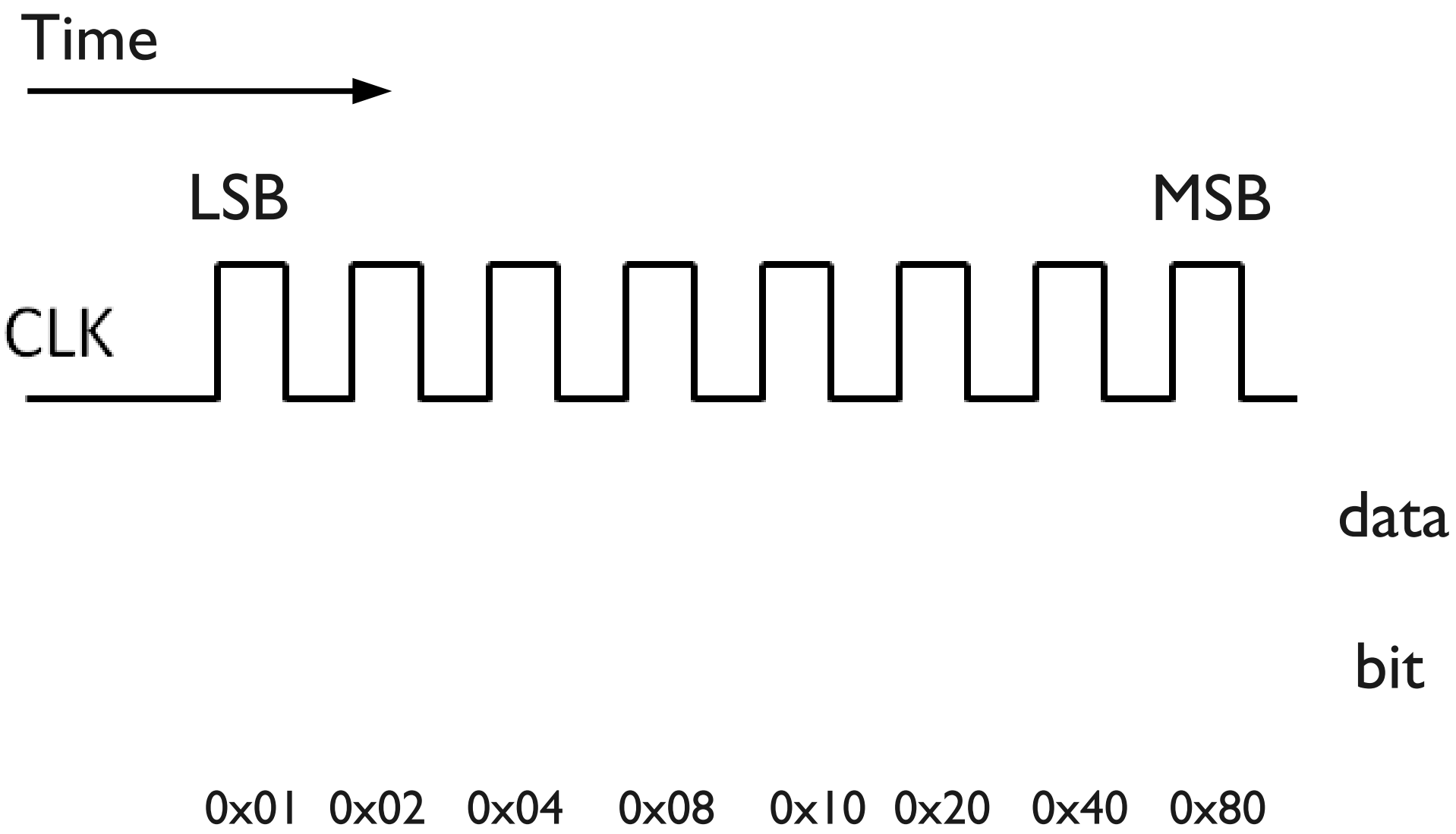
程式碼片段

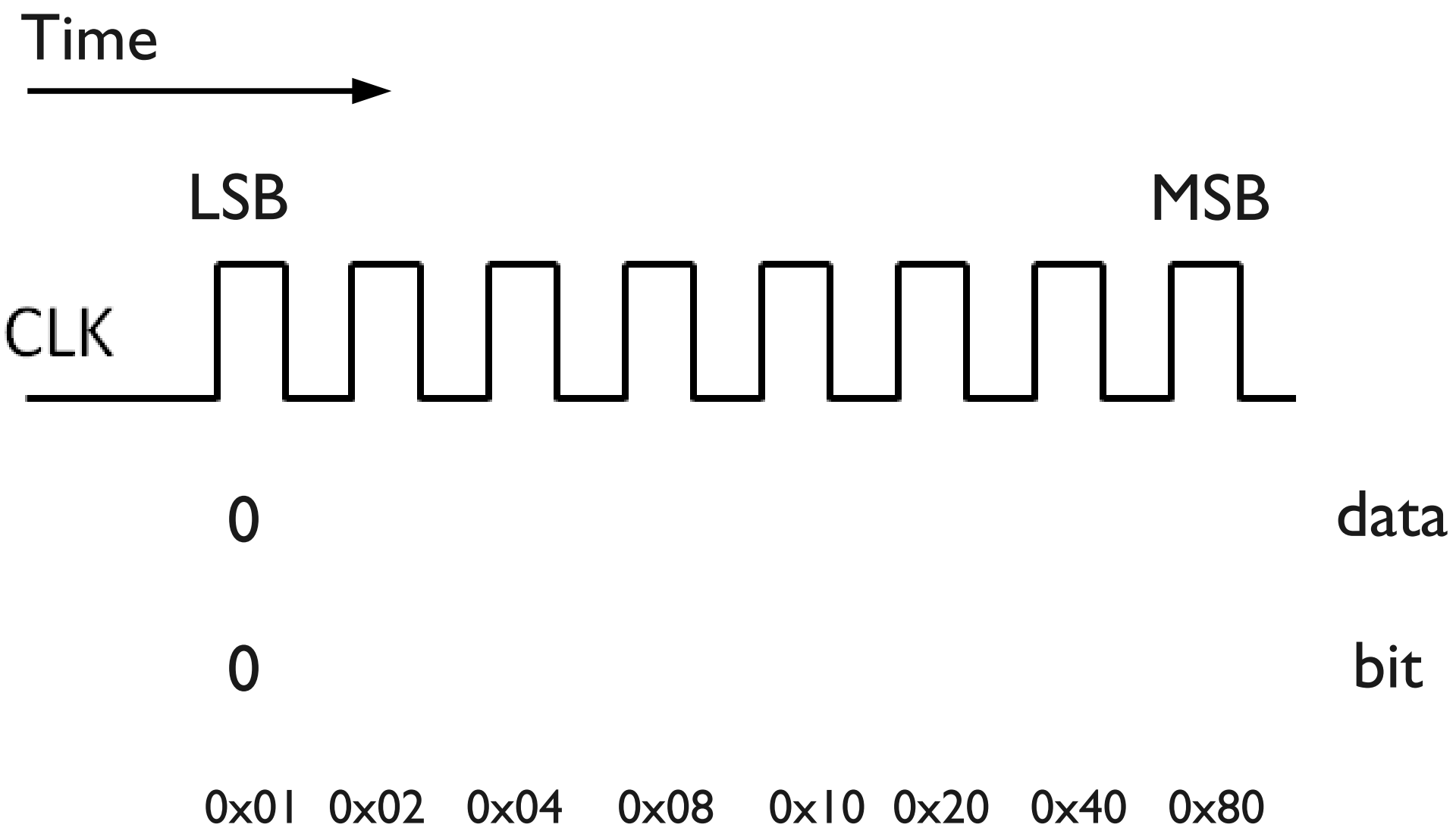
```
irqreturn_t irq_handler(...)
{
   signal = GPIO_READ(gpio_pin);
   GPIO_INT_CLEAR(gpio_pin);
   if ((signal == 1) && (elapse > 50)) {
      started = 1;
      return IRQ_HANDLED;
   }
   if ((signal == 0) && (started == 1)) {
      if (elapse > 70) return IRQ_HANDLED;
      if (elapse < 26) return IRQ_HANDLED;
      if (elapse > 28)
      /* send 1 */
   }
}
```

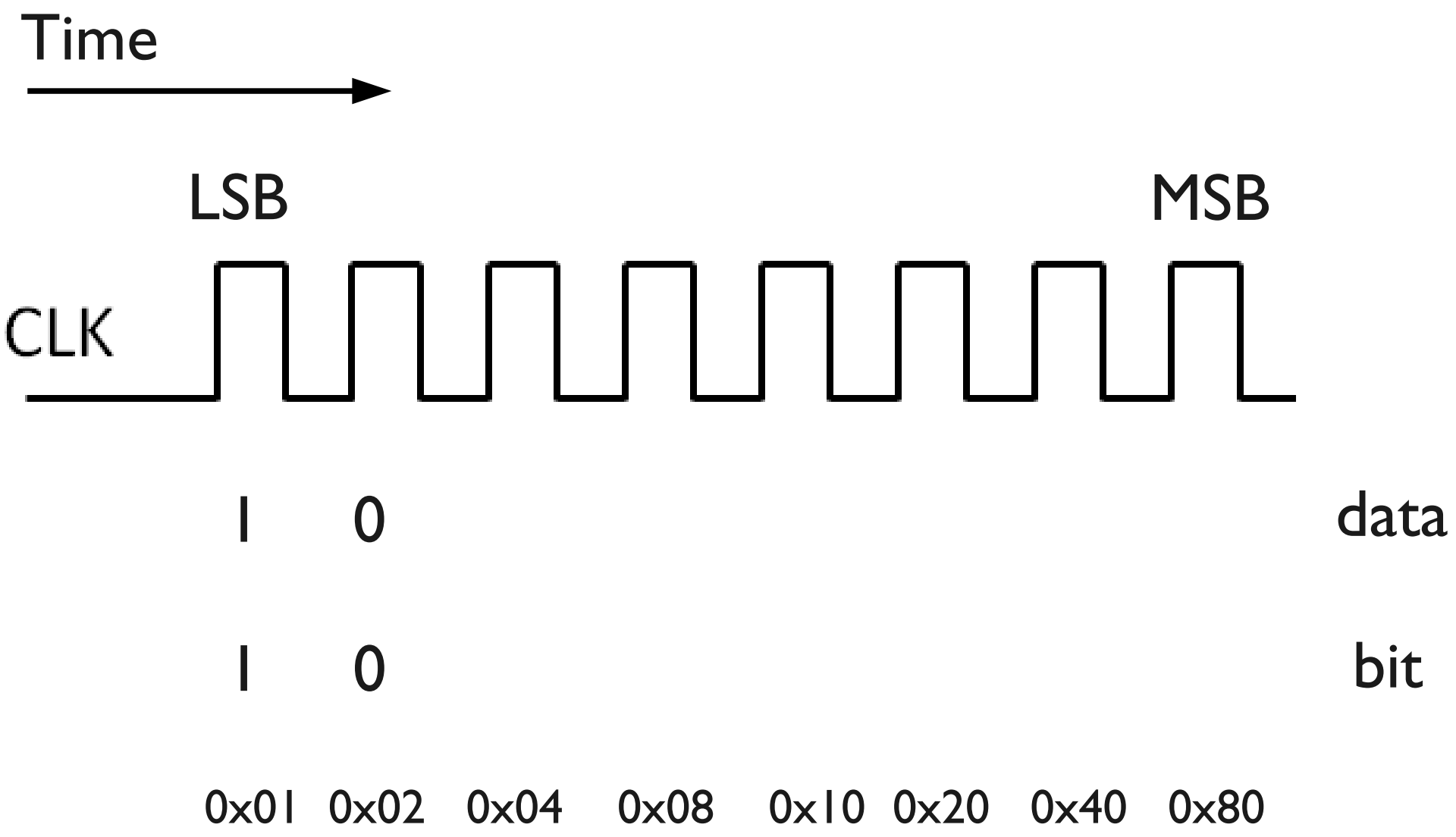
如何讀 0x65, MSB 先出 ？

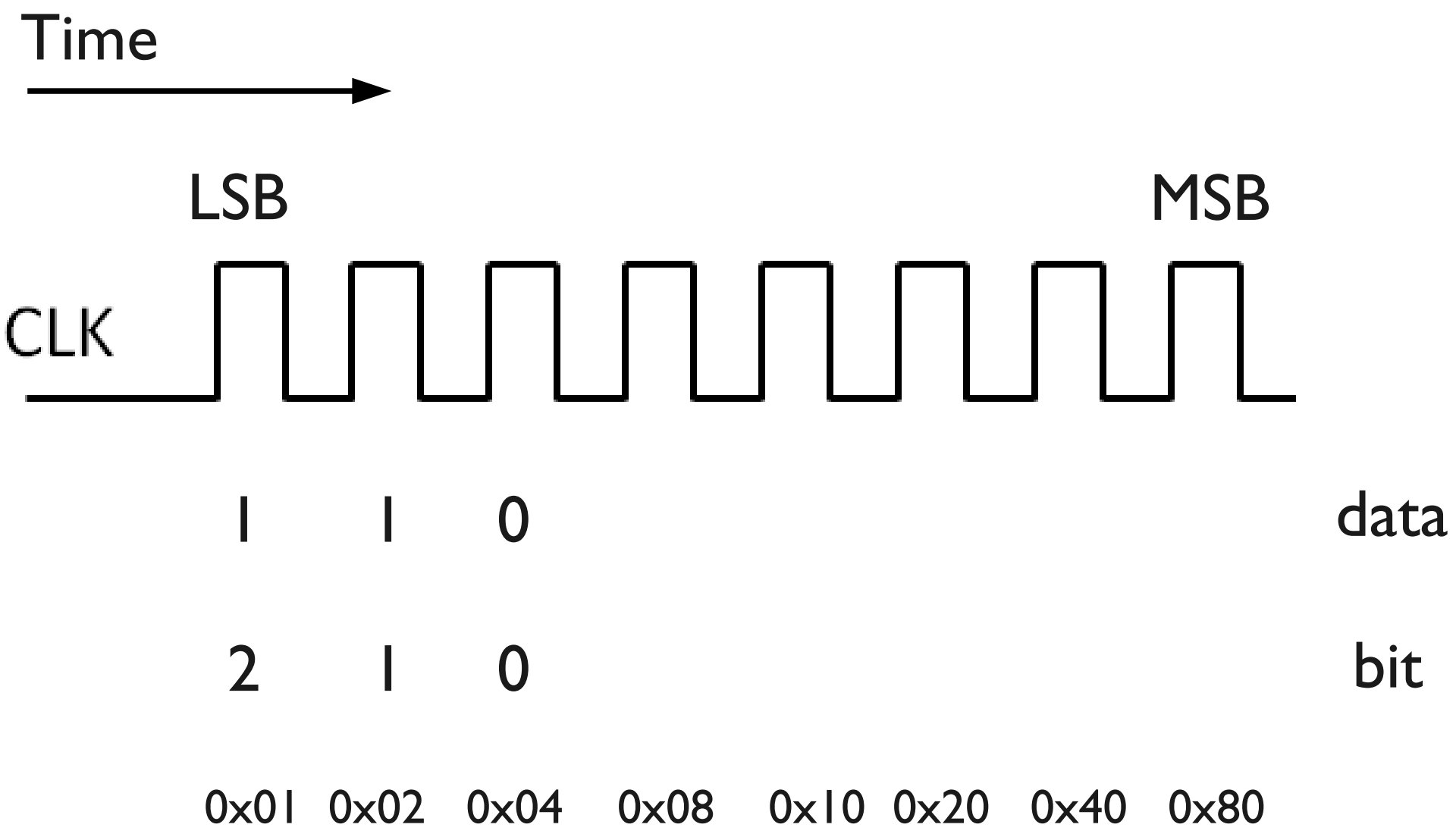
0x65 = 0110 0101

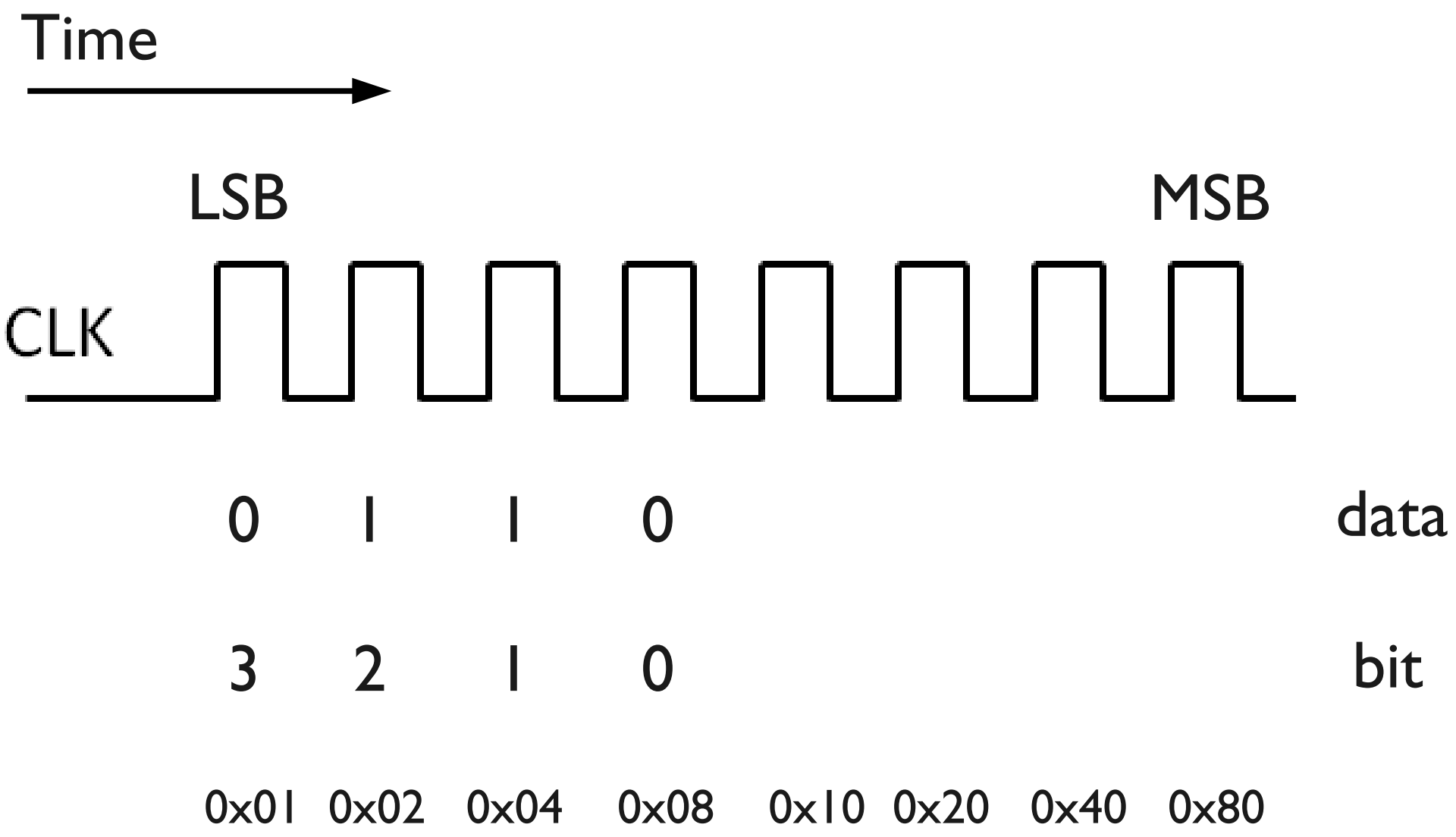

Time

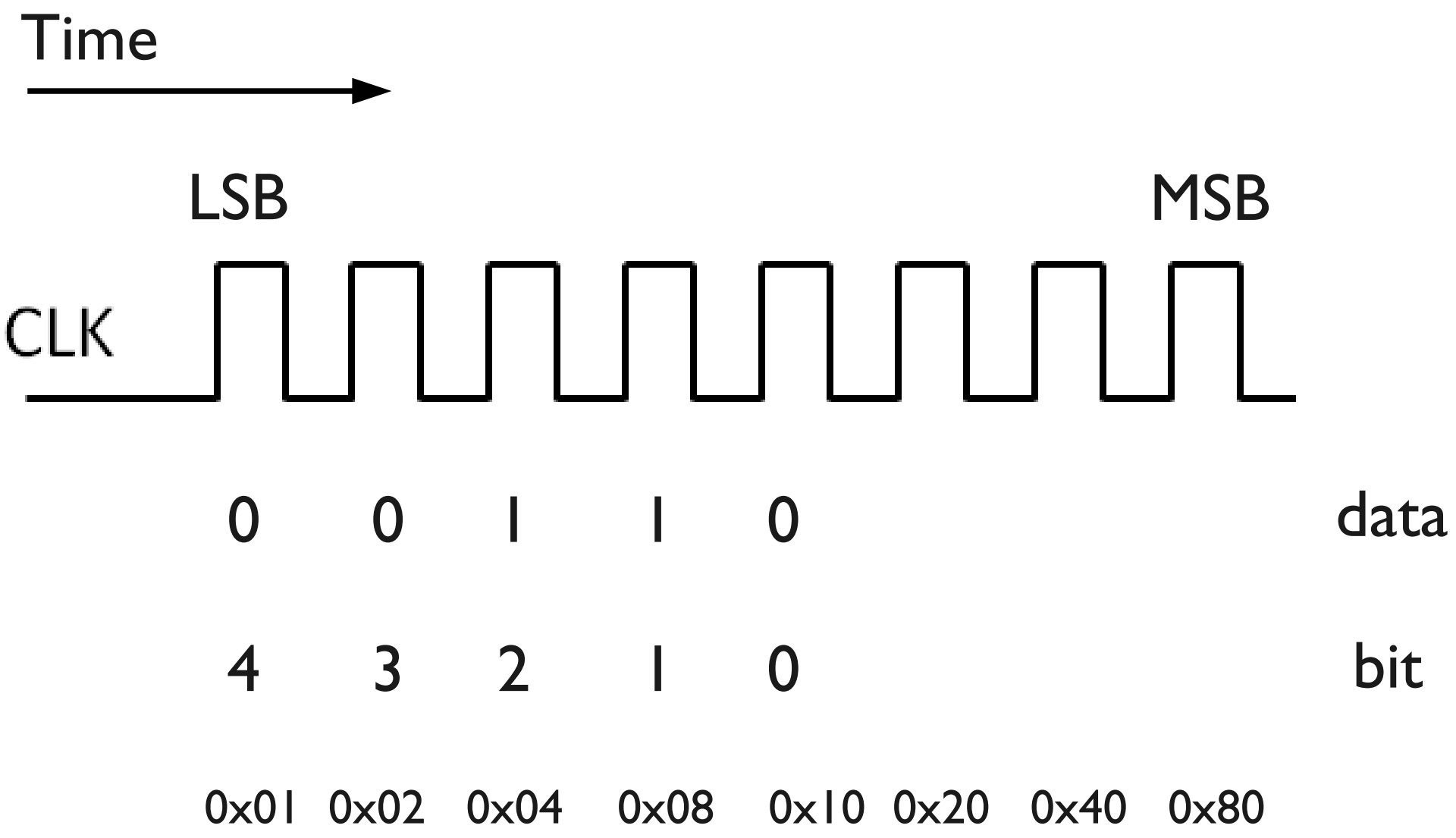

Time
LSB
MSB
CLK
data
bit
0x01 0x02 0x04 0x08 0x10 0x20 0x40 0x80

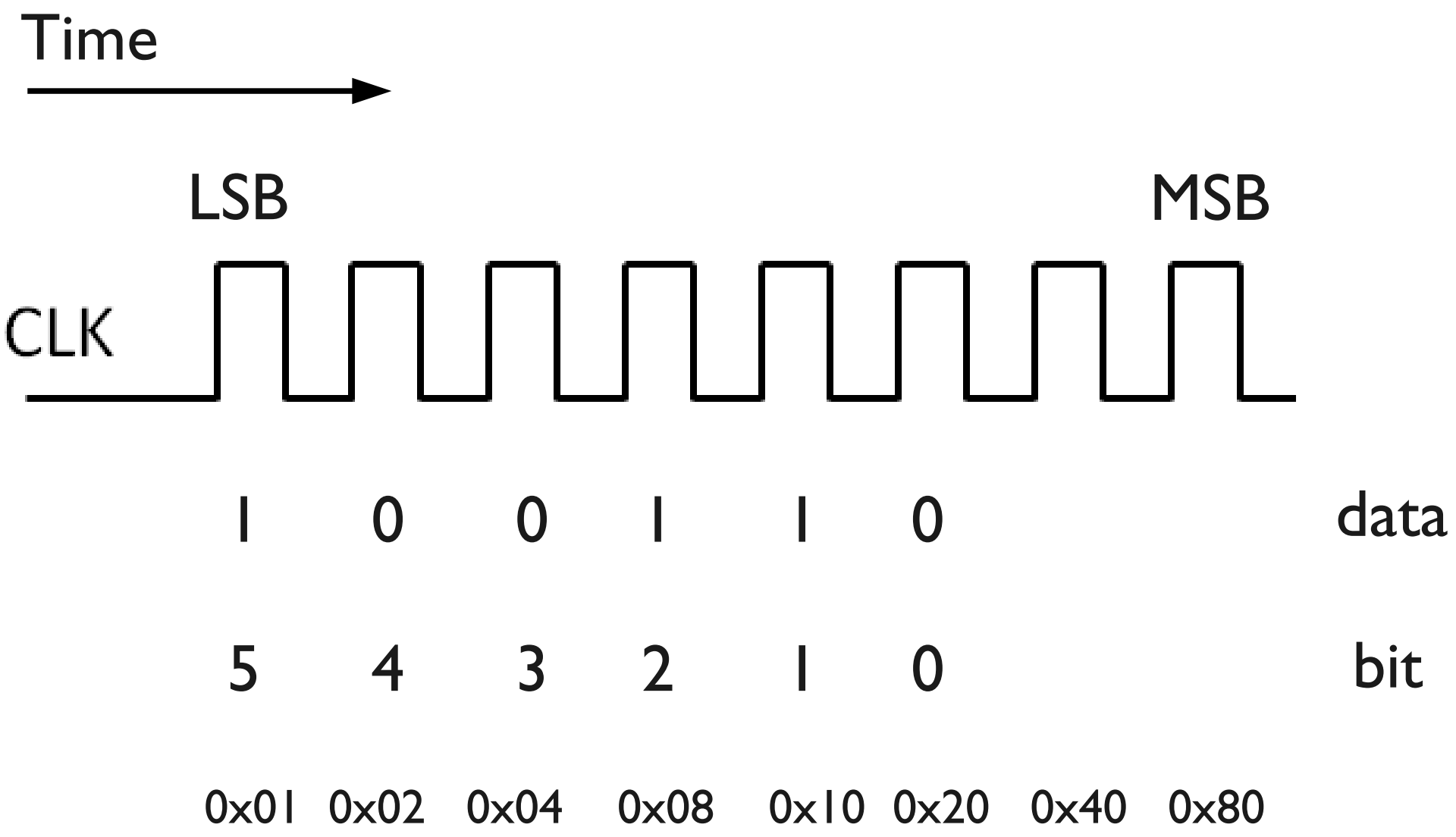

Time
LSB
MSB
CLK
0
data
0
bit
0x01 0x02 0x04 0x08 0x10 0x20 0x40 0x80

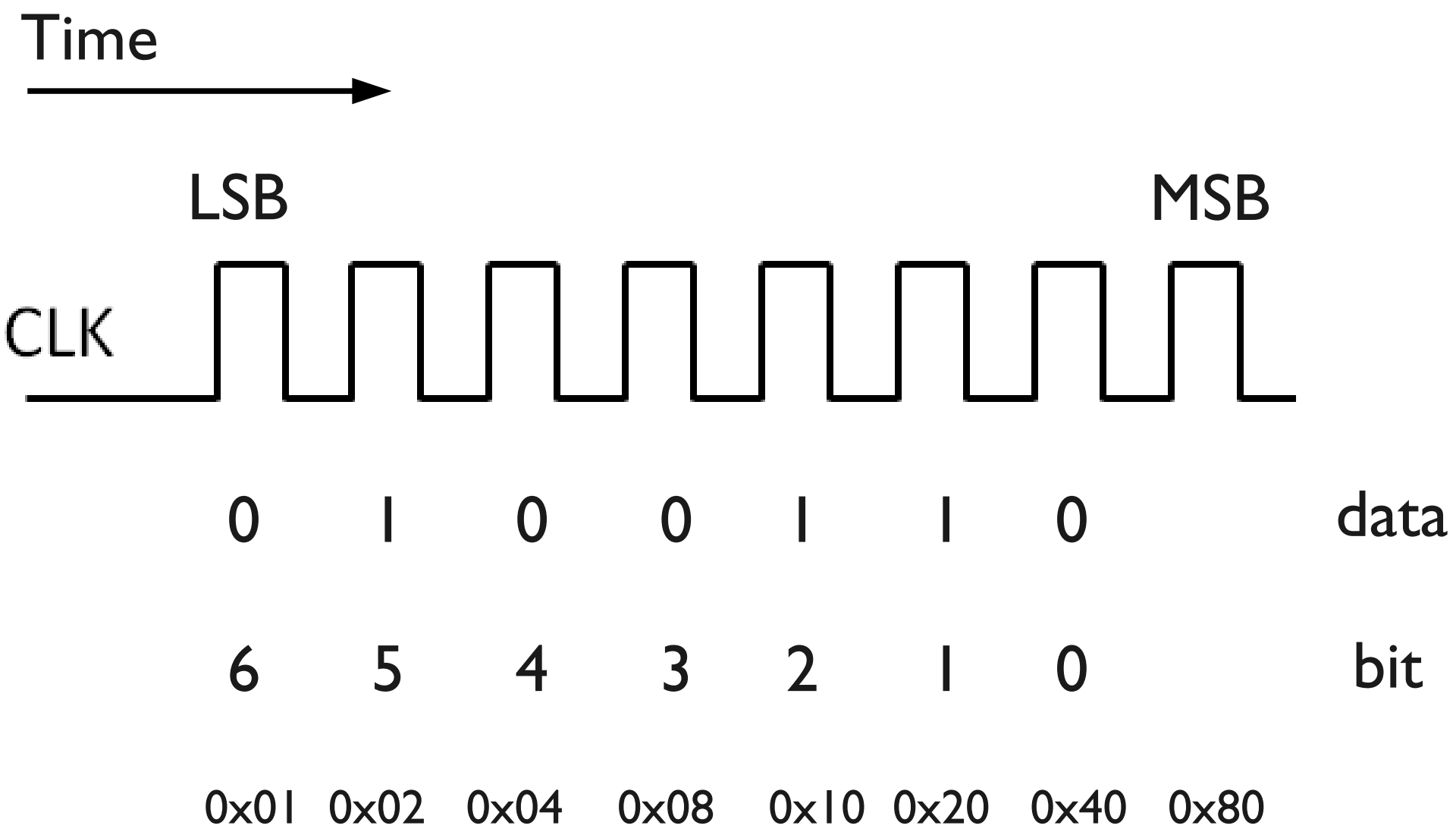

Time
LSB
MSB
CLK
1
0
data
1
0
bit
0x01
0x02
0x04
0x08
0x10
0x20
0x40
0x80

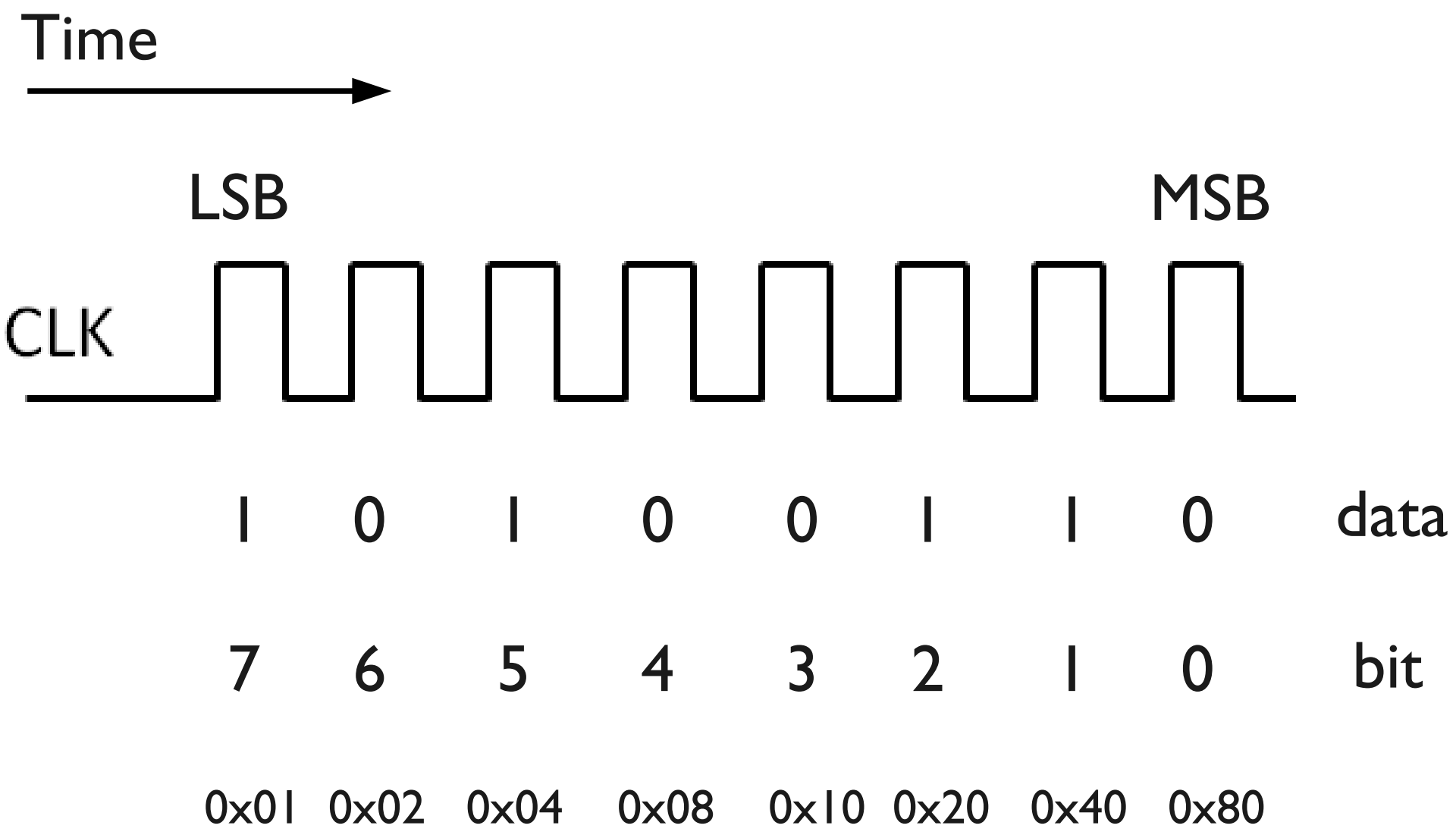

Time
LSB
MSB
CLK
1 1 0
data
2 1 0
bit
0x01 0x02 0x04 0x08 0x10 0x20 0x40 0x80

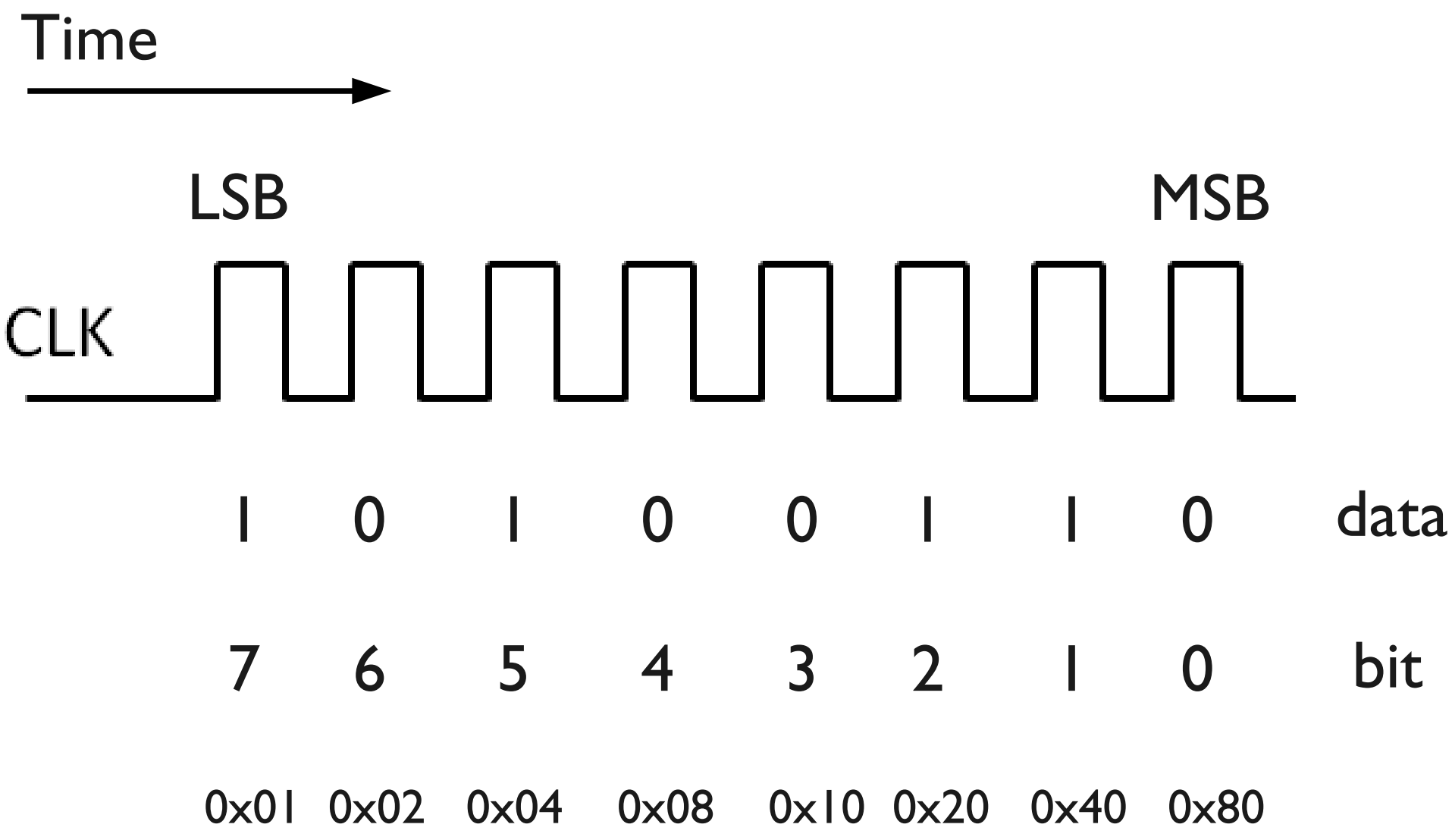

Time
LSB
MSB
CLK
0 1 1 0
data
3 2 1 0
bit
0x01 0x02 0x04 0x08 0x10 0x20 0x40 0x80


Time
LSB
MSB
CLK
0 0 1 1 0
data
4 3 2 1 0
bit
0x01 0x02 0x04 0x08 0x10 0x20 0x40 0x80

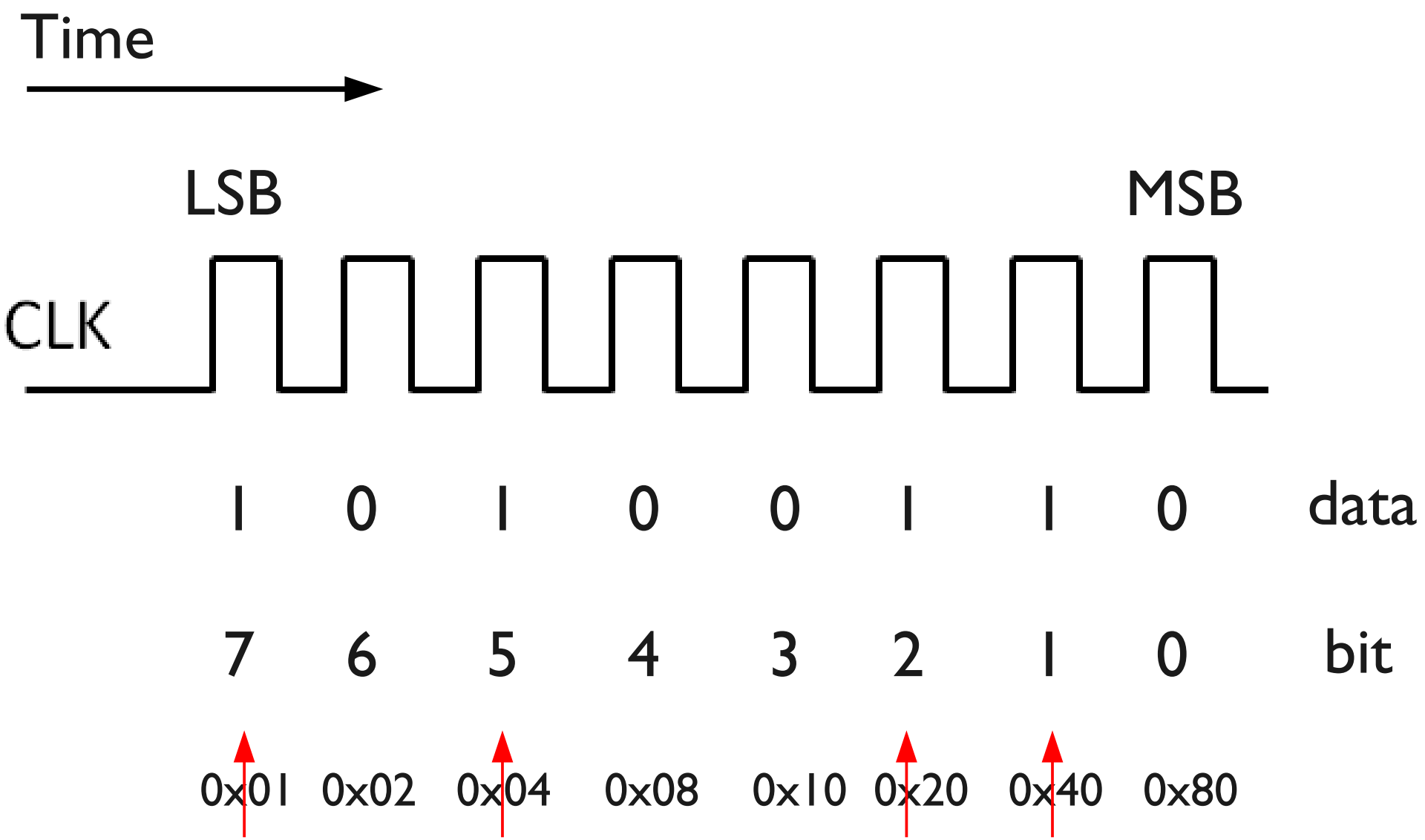

Time
LSB
MSB
CLK
1 0 0 1 1 0
data
5 4 3 2 1 0
bit
0x01 0x02 0x04 0x08 0x10 0x20 0x40 0x80


Time
LSB
MSB
CLK
0 1 0 0 1 1 0
data
6 5 4 3 2 1 0
bit
0x01 0x02 0x04 0x08 0x10 0x20 0x40 0x80


Time
LSB
MSB
CLK
1 0 1 0 0 1 1 0 data
7 6 5 4 3 2 1 0 bit
0x01 0x02 0x04 0x08 0x10 0x20 0x40 0x80



0x65 = 0110 0101



0x80 >> n 的意義是將第 1, 2, 5, 7 個 bit 寫為 1

用 bitwise operation 設值

```
unsigned int bitcnt = 0;
unsigned int bytecnt = 0;
unsigned char dht[5];
...
irq_handler()
{
  /* send 1 if elapse time > 70us */
  if ( elapse > 60 )
     dht[ bytecnt ] = dht[bytecnt] | ( 0x80 >> bitcnt);
}
```

整理一下

```
file_operations fops =
    .read    = read_dht11
    .open    = open_dht11
irqreturn_t irq_handler()
    - signal = GPIO_READ_PIN(gpio_pin);
    - /* read */
setup_interrupts(..., irq_handler)
    - request_irq()
__init dht11_init_module()
    - register_chrdev()
    - request_mem_region()
    - gpio = ioremap_nocache()
open_dht11()
    - setup_interrupts();
read_dht11()
    - put_user();
```

Test

```
$ make
$ sudo mknod /dev/dht11 c 80 0
$ sudo insmod ./dht11km.ko gpio_pin=18 format=3
$ cat /dev/dht11
Humidity: 73%
Temperature: 29%
Result:OK
```

DEMO

注意

- 程式裡有關時間的參數不一定和規格書的定義相同
  - 實際參數需要使用示波器或邏輯分析儀取得
- 投影片和範例程式(詳參考資料)的函式名稱不完全相同
- 鎖定機制在範例程式(詳參考資料)不是必要的

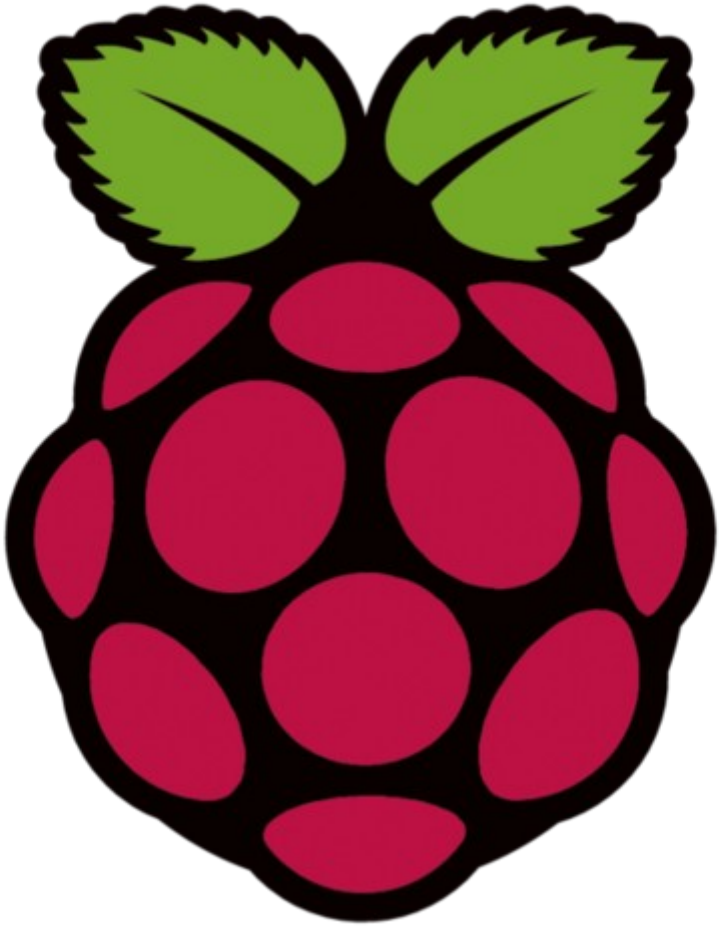
至於 用邏輯分析儀讀取正確的數值

那又是另外一個故事了

參考資料

- 原始 Source
  - http://www.tortosaforum.com/raspberrypi/dht11km.tar
- RaspberryPi DHT11 temperature and humidity sensor driver
  - http://www.tortosaforum.com/raspberrypi/dht11driver.htm
- Linux Device Driver Programming 驅動程式設計 (平田豊)
- 王者歸來 Linux 驅動程式開發權威指南

Raspberry Pi Rocks the World



Thanks