深入淺出 Raspberry Pi GPIO

台灣樹莓派 <sosorry@raspberrypi.com.tw>

Dec 08, 2013/Raspberry Pi #2

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
- Maker Faire 2013, PyCon 2013, 2013 科學玩意節
- 舉辦台灣第一次 Raspberry Pi 社群聚會

相關議程

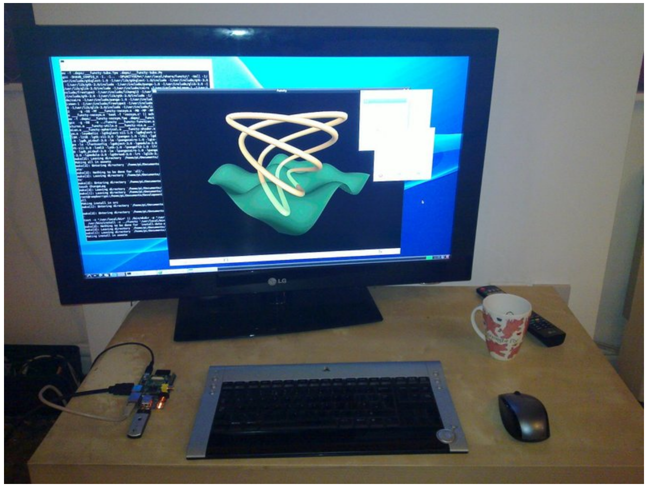
- Raspberry Pi 好好玩
- 用 Raspberry Pi 體驗嵌入式系統開發

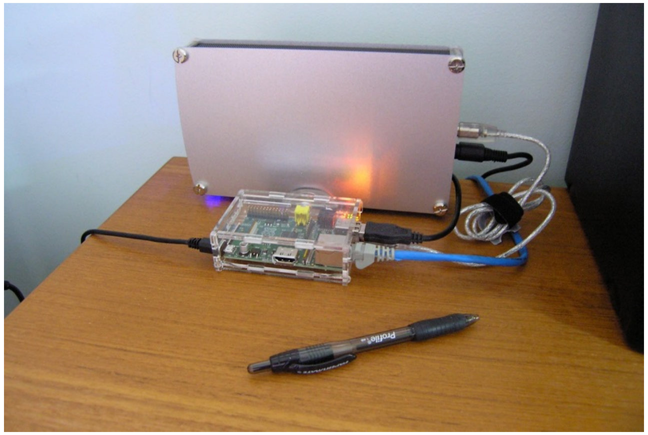
Raspberry Pi 是什麼？

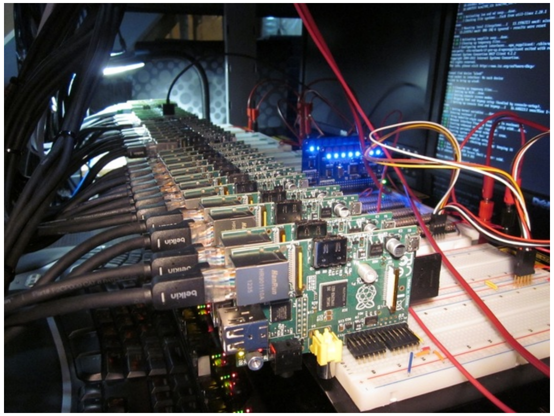
- 信用卡大小般的電腦

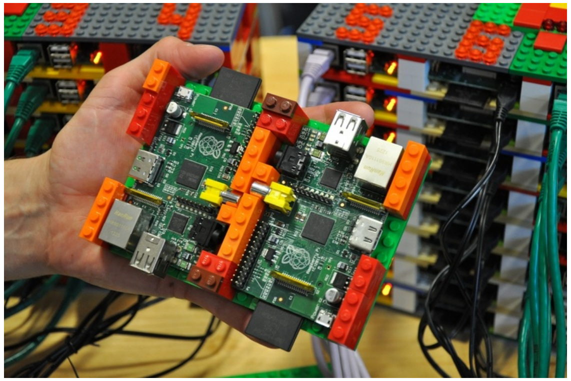
http://www.flickr.com/photos/fotero/7697063016/

Raspberry Pi 怎麼玩？









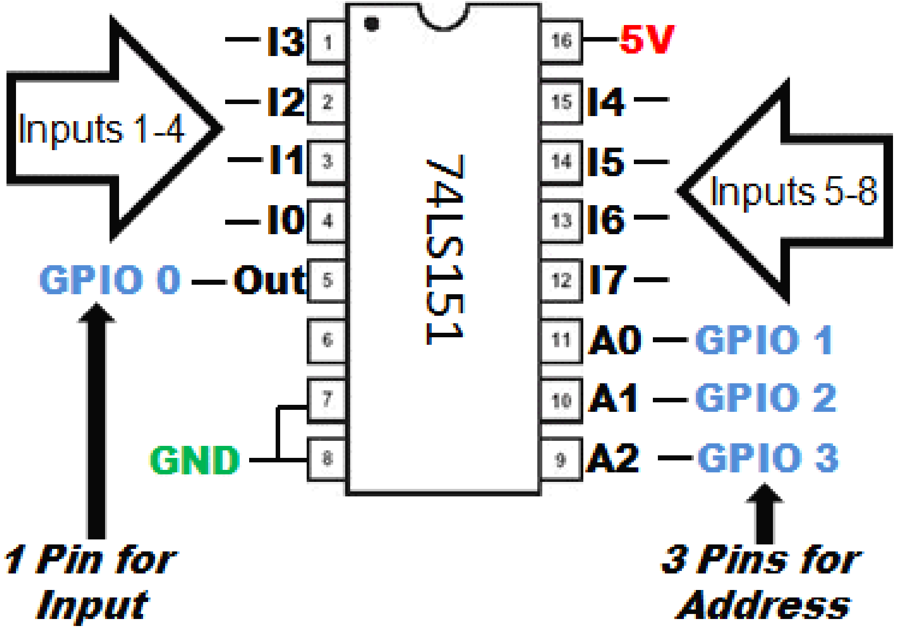
http://www.slideshare.net/raspberrypi-tw/introduction-toraspberrypi

Raspberry Pi 還可以怎麼玩？

Raspberry Pi 還可以怎麼玩？
玩他的 GPIO

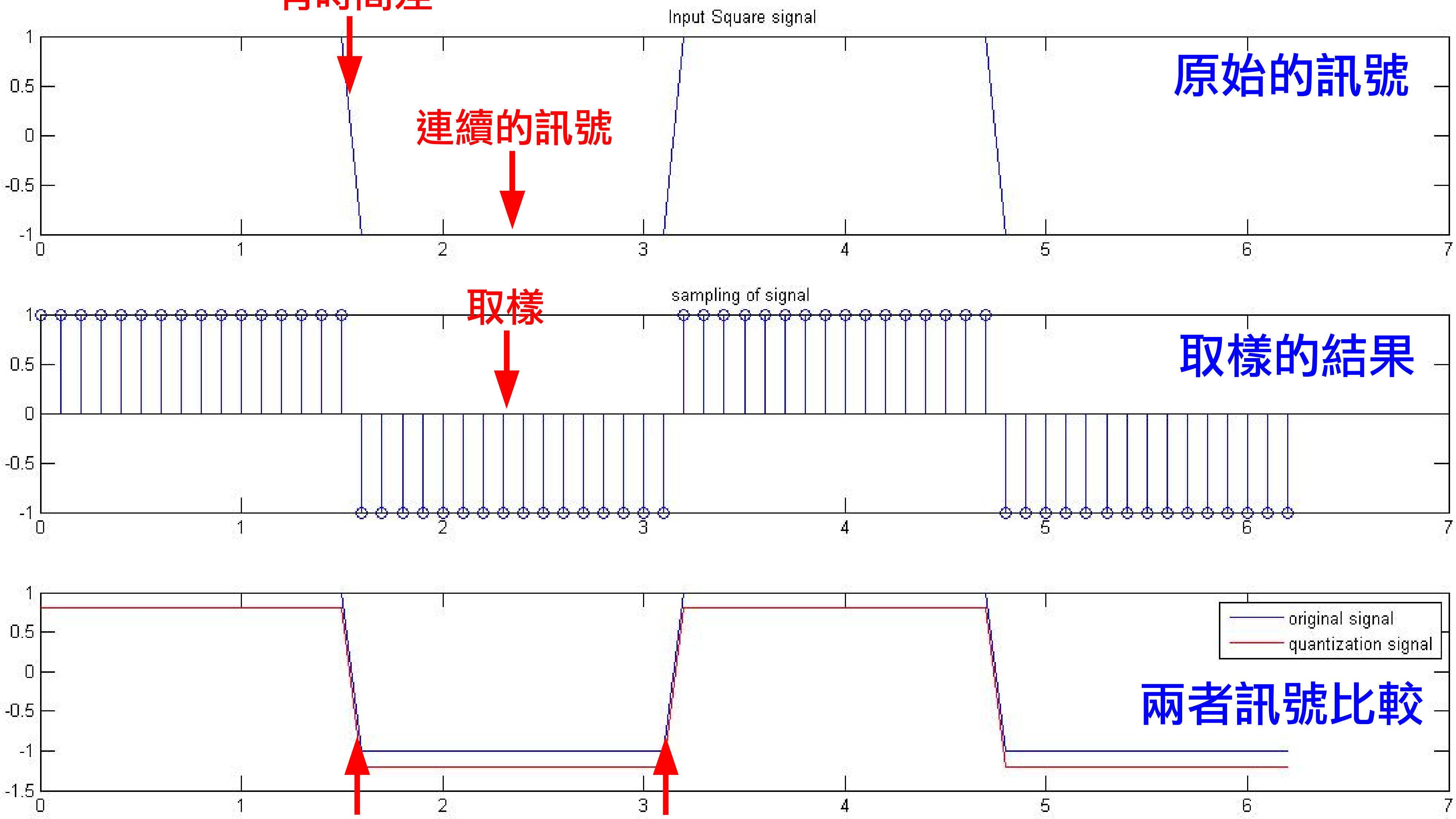
General Purpose Input Output(GPIO)

- A generic pin on an IC



http://raspberrypihobbyist.blogspot.tw/2012/09/so-many-inputs-so-few-gpio-pins.html

真實的電流輸入



有時間差

連續的訊號

原始的訊號

取樣

取樣的結果

兩者訊號比較

負緣觸發

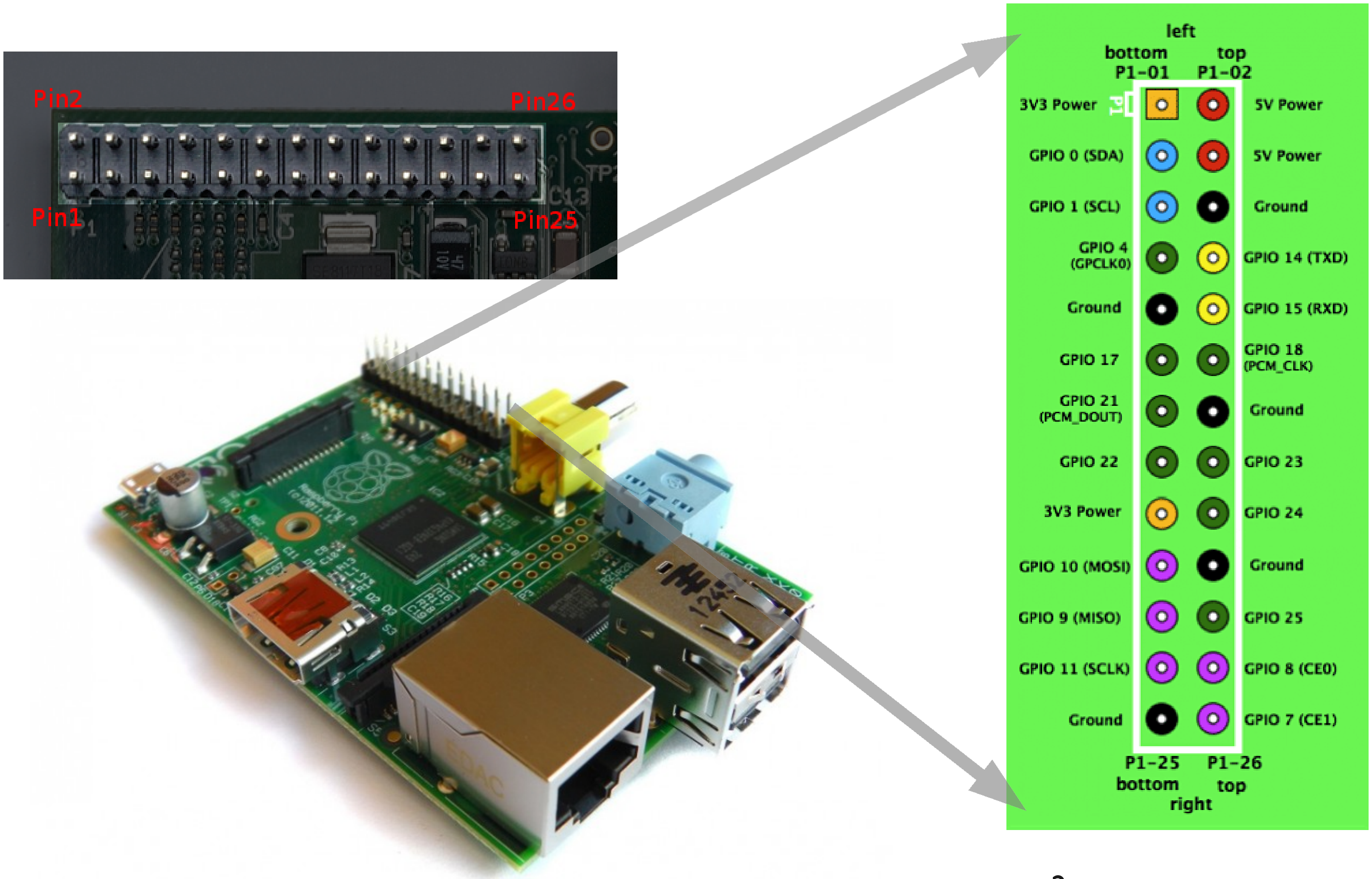
正緣觸發

http://goo.gl/IzwE0K

那軟體做什麼？

- 開啟或關閉 GPIO
- 決定是 0 激活還是 1 要激活
- 決定是輸入還是輸出
- 寫值到某根腳位
- 從某根腳位讀值
- 決定是正緣觸發還是負緣觸發
- 等待中斷 (interrupt) 的發生

Raspberry Pi 的 GPIO



left bottom P1-01			top P1-02
3V3 Power	P1		5V Power
GPIO 0 (SDA)			5V Power
GPIO 1 (SCL)			Ground
GPIO 4 (GPCLK0)			GPIO 14 (TXD)
Ground			GPIO 15 (RXD)
GPIO 17			GPIO 18 (PCM_CLK)
GPIO 21 (PCM_DOUT)			Ground
GPIO 22			GPIO 23
3V3 Power			GPIO 24
GPIO 10 (MOSI)			Ground
GPIO 9 (MISO)			GPIO 25
GPIO 11 (SCLK)			GPIO 8 (CE0)
Ground			GPIO 7 (CE1)
P1-25 bottom right			P1-26 top

SPI / I²C / UART / PWM

http://elinux.org/RPi_Low-level_peripherals

深入淺出 GPIO

- 深入
  - 用 C 控制 GPIO
- 淺出
  - 用 Python 控制 GPIO

控制硬體的方法

- 直接修改 register 的值
- 透過 driver 進行操作

用 C 直接修改 register 的值？

先來看 code 吧

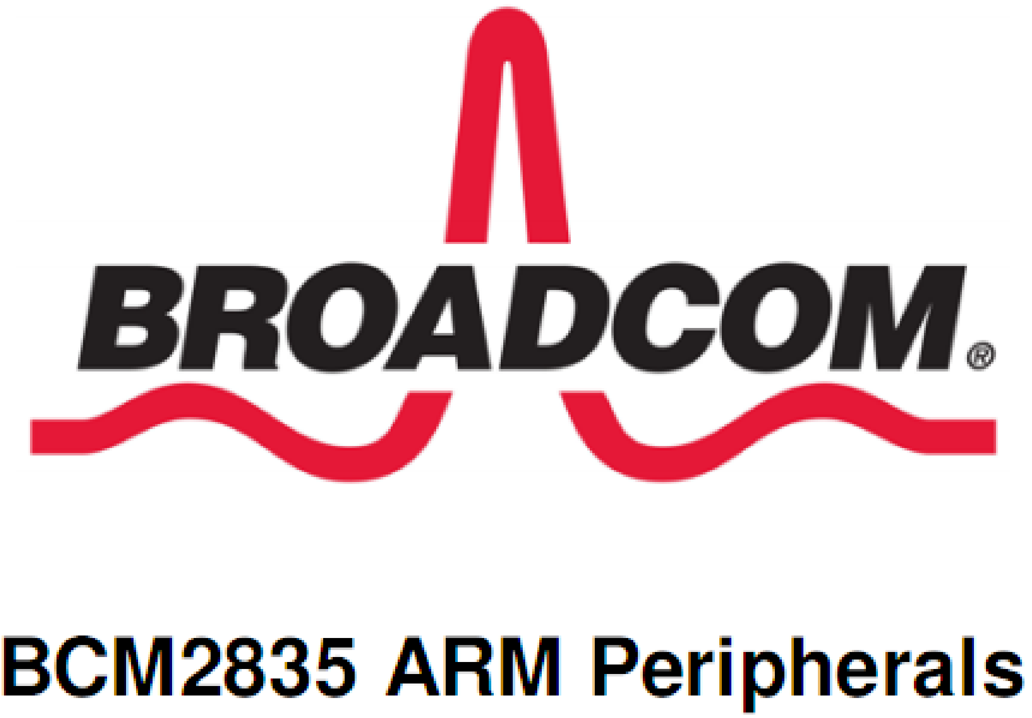
https://github.com/raspberrypi-tw/tutorial/tree/master/gpio/led/c

三言以蔽之

1. 看 datasheet
2. 查 register
3. 填對應的值

看 datasheet

BCM2835 ARM Peripherals



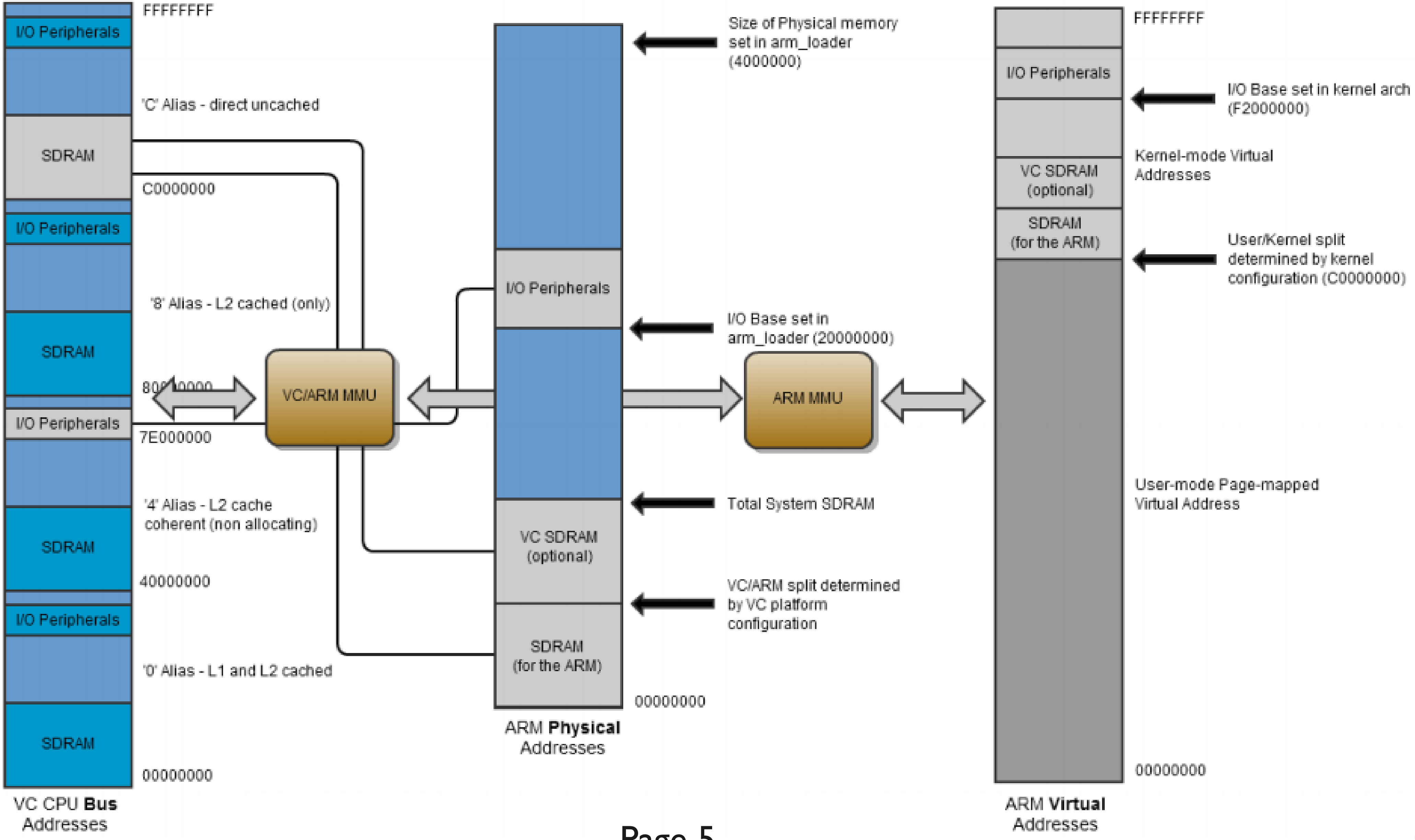
BCM2835 ARM Peripherals

http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf

共 205 頁

查 register

BCM2835 ARM Peripherals


FFFFFFFF
I/O Peripherals
'C' Alias - direct uncached
SDRAM
C0000000
I/O Peripherals
'8' Alias - L2 cached (only)
SDRAM
80000000
I/O Peripherals
7E000000
'4' Alias - L2 cache coherent (non allocating)
SDRAM
40000000
I/O Peripherals
'0' Alias - L1 and L2 cached
SDRAM
00000000
VC CPU Bus Addresses
VC/ARM MMU
Size of Physical memory set in arm_loader (4000000)
I/O Peripherals
I/O Base set in arm_loader (20000000)
Total System SDRAM
VC SDRAM (optional)
VC/ARM split determined by VC platform configuration
SDRAM (for the ARM)
00000000
ARM Physical Addresses
ARM MMU
FFFFFFFF
I/O Peripherals
I/O Base set in kernel arch (F2000000)
Kernel-mode Virtual Addresses
VC SDRAM (optional)
SDRAM (for the ARM)
User/Kernel split determined by kernel configuration (C0000000)
User-mode Page-mapped Virtual Address
00000000
ARM Virtual Addresses

Address Translation (Page 6)

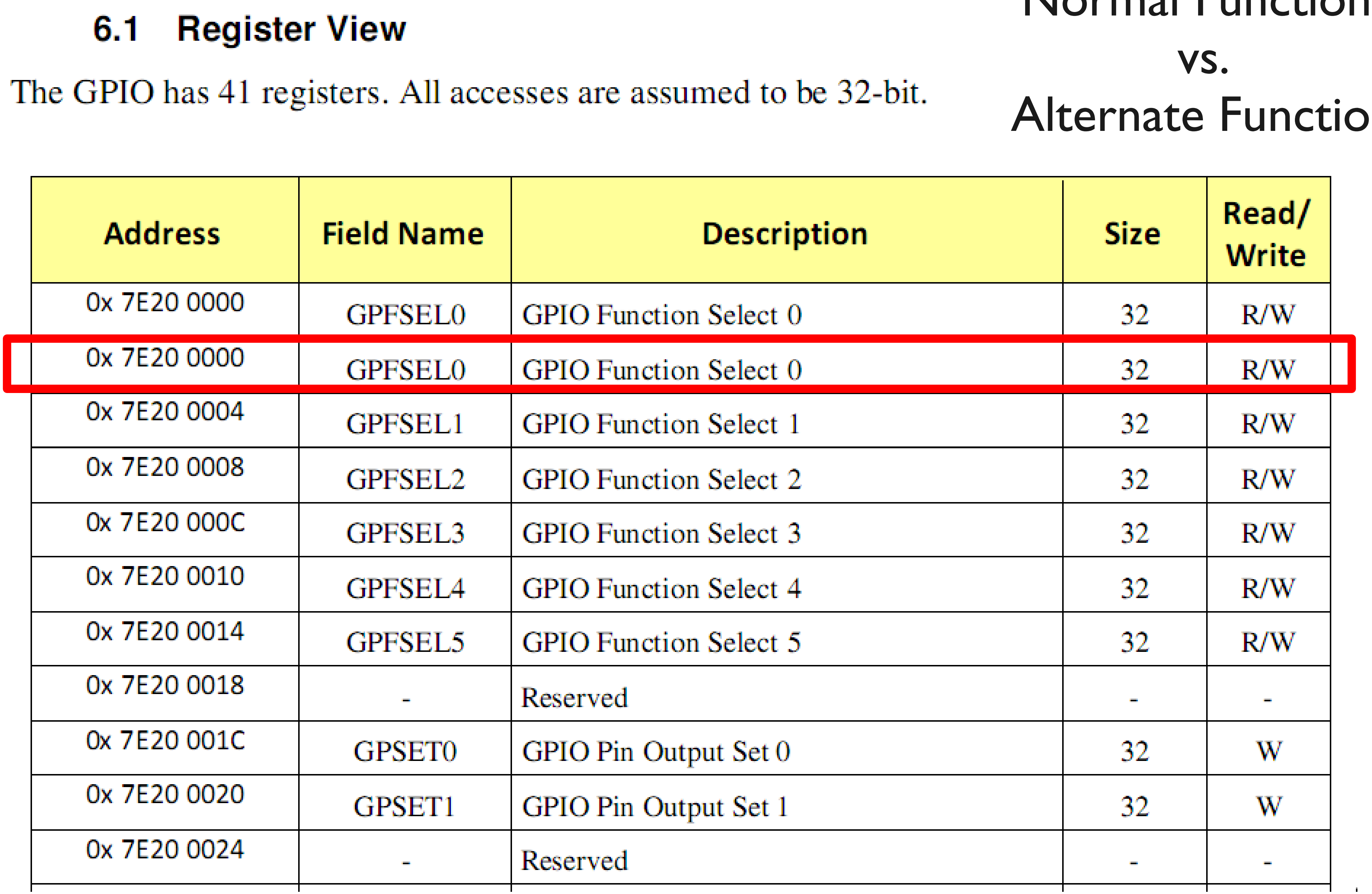
- Address 映射過程
  - virtual address → physical address → bus address

- Peripheral address 起始位址
  - Physical addresses: 0x20000000 - 0x20FFFFFF
  - Bus address: 0x7E000000 -

- 實際位址是多少？ 查表可得知

Page 90

Normal Function
vs.
Alternate Function



6.1 Register View

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

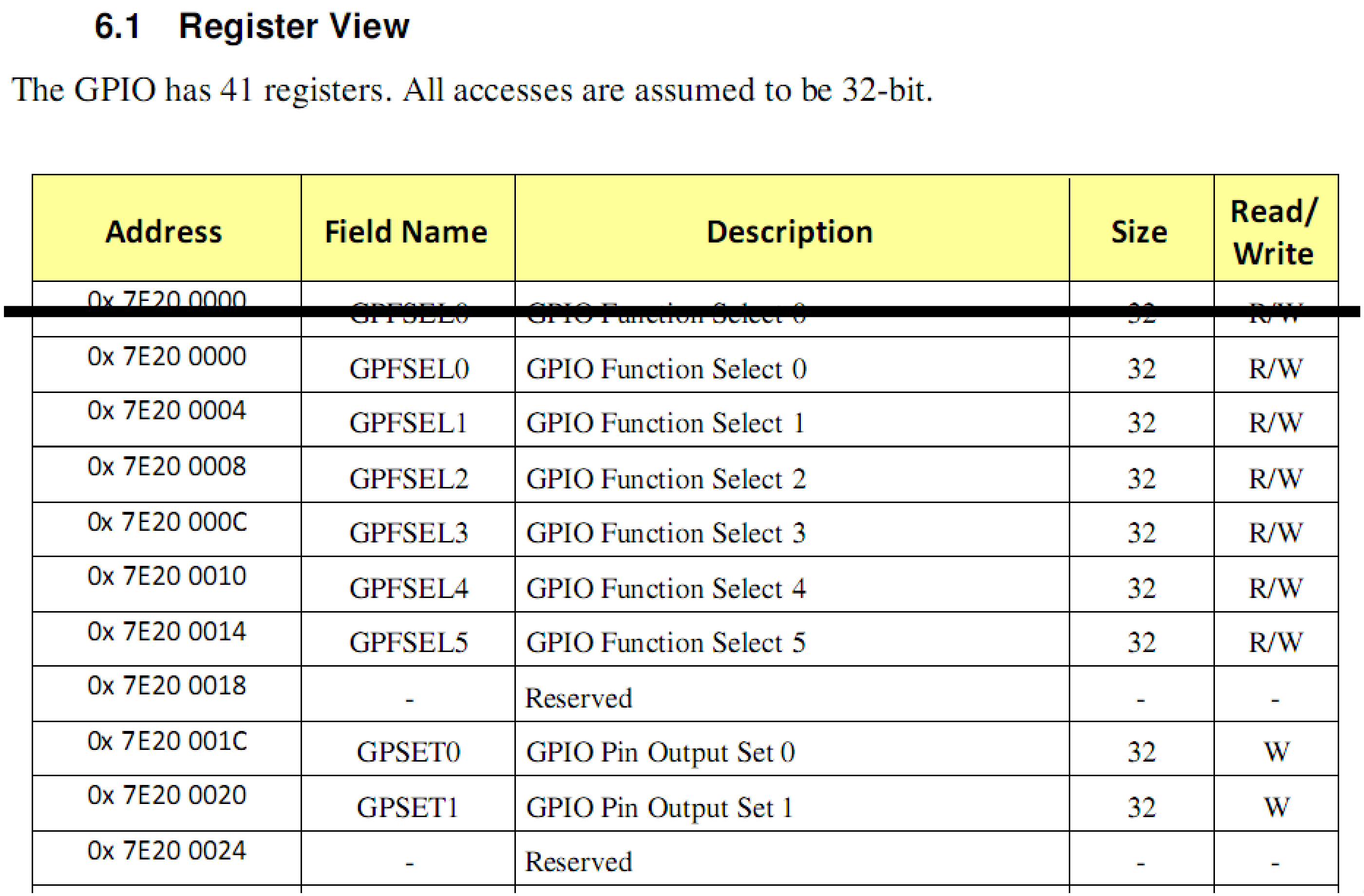
Address	Field Name	Description	Size	Read/ Write
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0004	GPFSEL1	GPIO Function Select 1	32	R/W
0x 7E20 0008	GPFSEL2	GPIO Function Select 2	32	R/W
0x 7E20 000C	GPFSEL3	GPIO Function Select 3	32	R/W
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	-	-

重點

- 41 個 register, 每個 register 是 32bit
- 起始位址 : 0x7E200000
- 表畫錯了
- 勘誤可見

↓

http://goo.gl/msNCRO



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	-	-

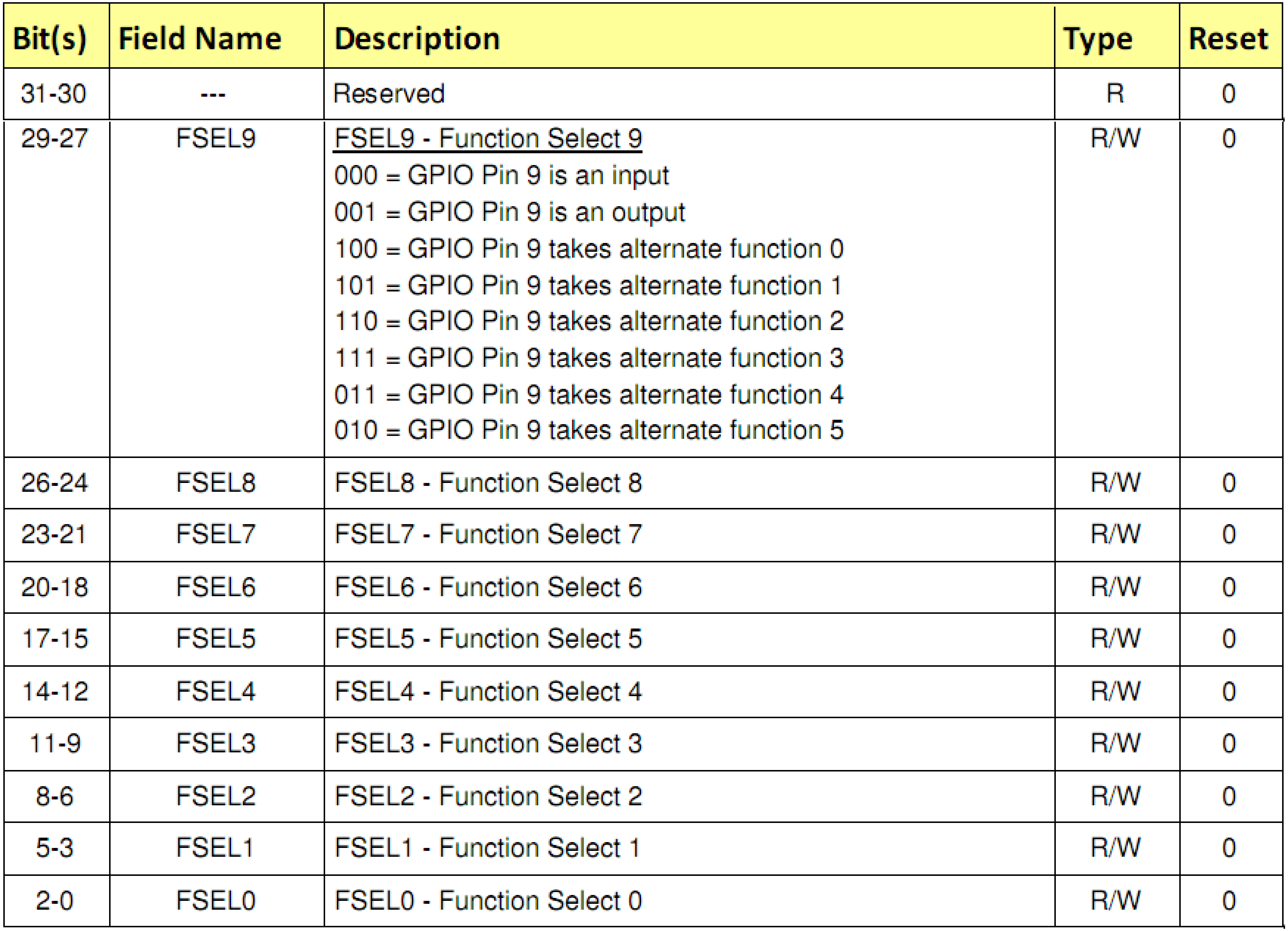
Address 映射結果

```
// RPI.h
#define BCM2708_PERI_BASE        0x20000000
#define GPIO_BASE                \
                (BCM2708_PERI_BASE + 0x200000)
```

填對應的值

每一個 GPIO Function Select 會對應到一個 32-bit 的表

Page 91 & Page 92



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

範例 1 ：
將某根 PIN 腳 (g=4) 設成 INPUT

註 :BCM2835 的 4 號腳位對應到實體腳位 7

如何做？

將記憶體位置依 datasheet 寫入值

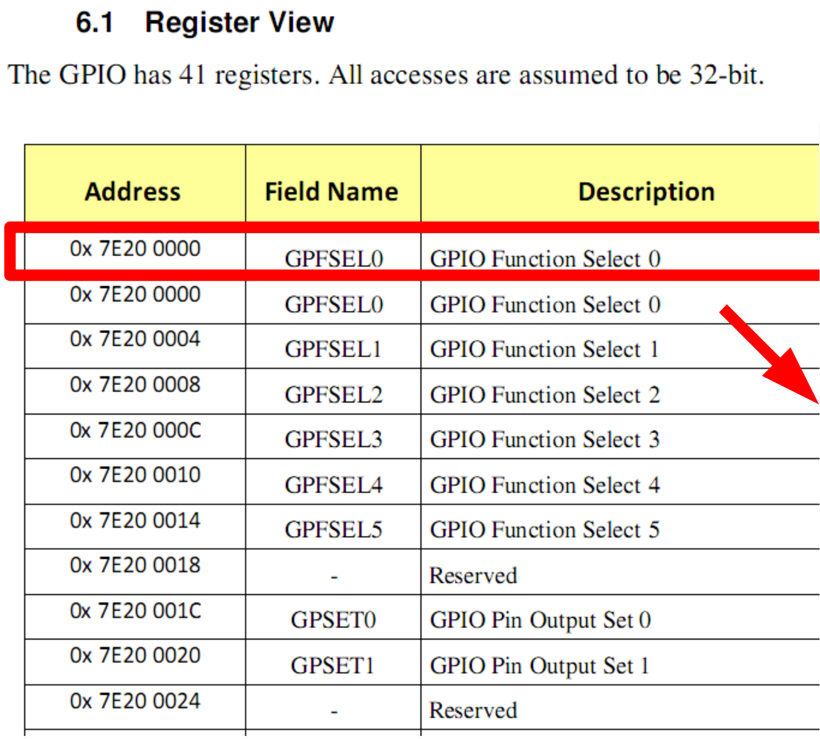
寫一個 macro 吧

```
// RPI.h
#define INP_GPIO(g)                    \
            (*(gpio.addr + ((g)/10)) &= ~(7<<(((g)%10)*3)))
```

處理步驟

1. 根據 g 找到對應的 GPFSEL table
2. 根據 g 取得對應到的 FSEL 起始位置
3. 查表決定 FSEL 的 bit 值設定

1. 根據 g 找到對應的 GPFSEL table

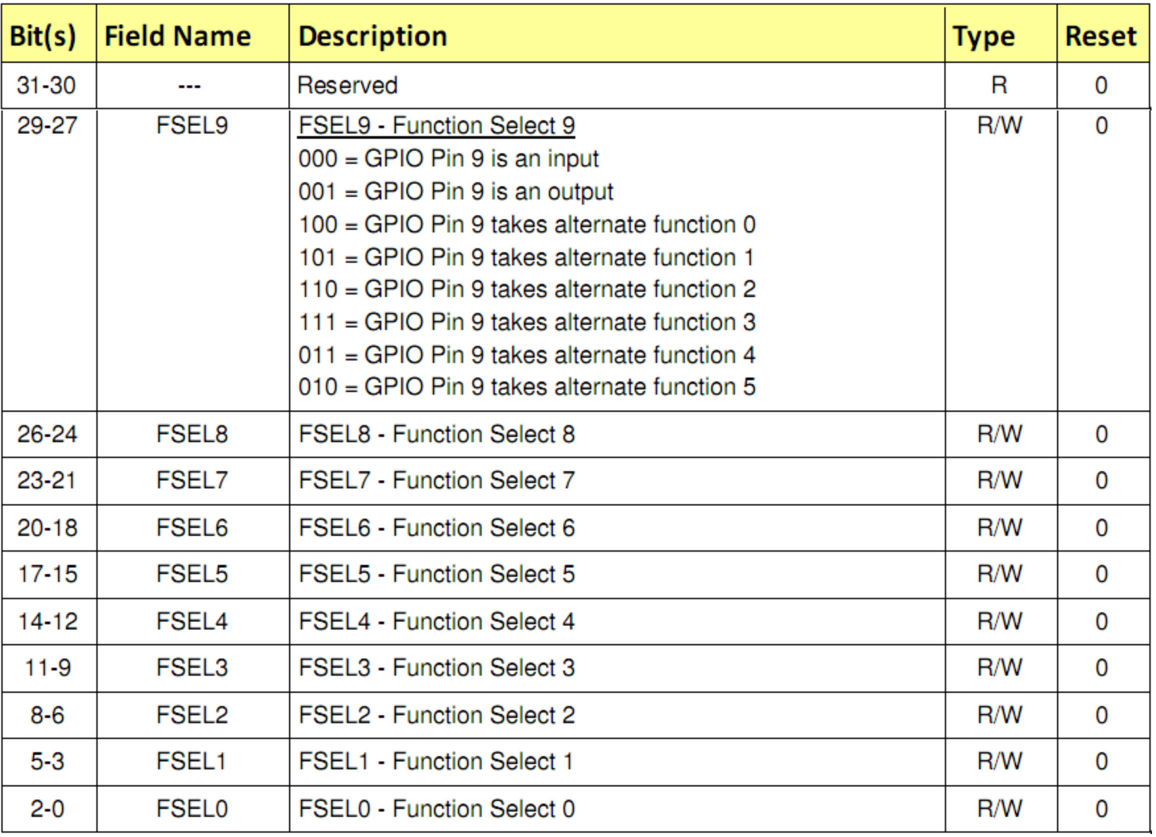


6.1 Register View

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



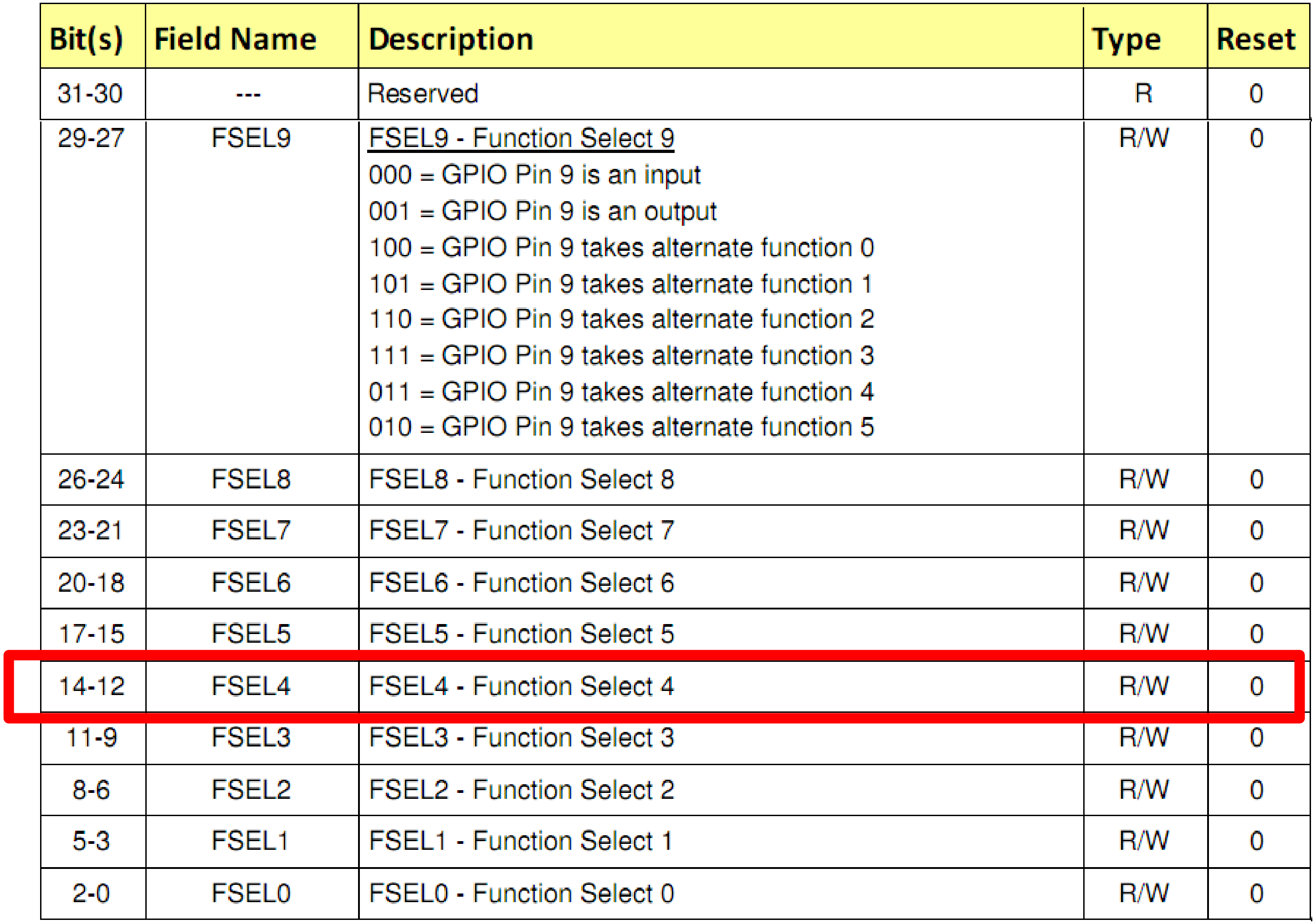
Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0
29-27	FSEL9	FSEL9 - Function Select 9 000 = GPIO Pin 9 is an input 001 = GPIO Pin 9 is an output 100 = GPIO Pin 9 takes alternate function 0 101 = GPIO Pin 9 takes alternate function 1 110 = GPIO Pin 9 takes alternate function 2 111 = GPIO 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

每十個 Function Select 為一張表

(g)%10：取得第 1 張 GPFSEL table

2. 根據 g 取得對應到的 FSEL 起始位置

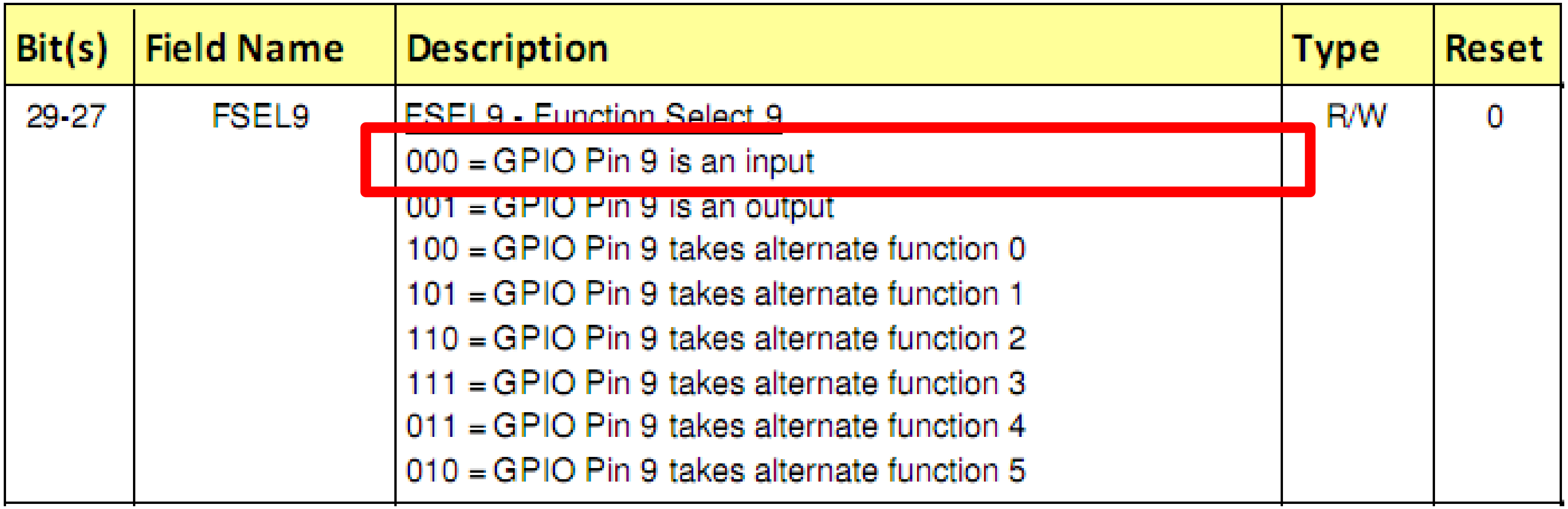


Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0
29-27	FSEL9	FSEL9 - Function Select 9 000 = GPIO Pin 9 is an input 001 = GPIO Pin 9 is an output 100 = GPIO Pin 9 takes alternate function 0 101 = GPIO Pin 9 takes alternate function 1 110 = GPIO Pin 9 takes alternate function 2 111 = GPIO 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

((g)%10)*3：取得第 4 個 FSEL 起始位置

3. 查表決定 FSEL 的 bit 值設定



Bit(s)	Field Name	Description	Type	Reset
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

000 表示 INPUT

- 將 000 寫到原 register 中第 12-14 個 bit

Bitwise 運算 , g=4

原 : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (32-bit)

&= ~(7<<(((g)%10)*3)))

Bitwise 運算 , g=4

原 : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (32-bit)

&= ~(7<<(((g)%10)*3)))

(4%10)*3 : 找第 12 個 bit

7 << 12 : 將 111 左移 12 位

Bitwise 運算 , g=4

原 : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (32-bit)

&= ~(7<<(((g)%10)*3)))

(4%10)*3 : 找第 12 個 bit

7 << 12 : 將 111 左移 12 位

00000000000000000111000000000000 (NOT 運算)

Bitwise 運算 , g=4

原 : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (32-bit)

```
&= ~(7<<(((g)%10)*3)))
```

(4%10)*3 : 找第 12 個 bit

7 << 12 : 將 111 左移 12 位

00000000000000000111000000000000 (NOT 運算)

11111111111111111000111111111111 (運算結果)

Bitwise 運算 , g=4

原 : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (32-bit)

&= ~(7<<(((g)%10)*3)))

(4%10)*3 : 找第 12 個 bit

7 << 12 : 將 111 左移 12 位

00000000000000000111000000000000 (NOT 運算)

11111111111111111000111111111111 (運算結果)

原 : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bitwise 運算 , g=4

原 : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (32-bit)

&= ~(7<<(((g)%10)*3)))

(4%10)*3 : 找第 12 個 bit

7 << 12 : 將 111 左移 12 位

00000000000000000111000000000000 (NOT 運算)

11111111111111111000111111111111 (運算結果)

原 : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

寫 : 11111111111111111000111111111111 (AND 運算)

Bitwise 運算 , g=4

原　: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (32-bit)

```
&= ~(7<<(((g)%10)*3)))
```

(4%10)*3 : 找第 12 個 bit

7 << 12　: 將 111 左移 12 位

00000000000000000111000000000000 (NOT 運算)

11111111111111111000111111111111 (運算結果)

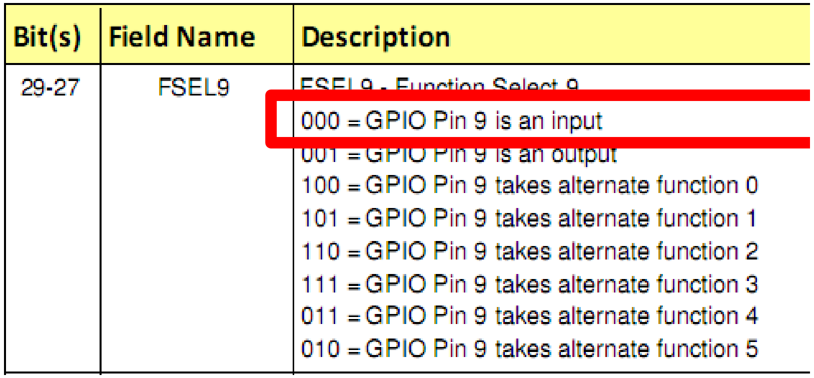
原　: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

寫　: 11111111111111111000111111111111 (AND 運算)
新　: xxxxxxxxxxxxxxxxx000xxxxxxxxxxxx (運算結果)

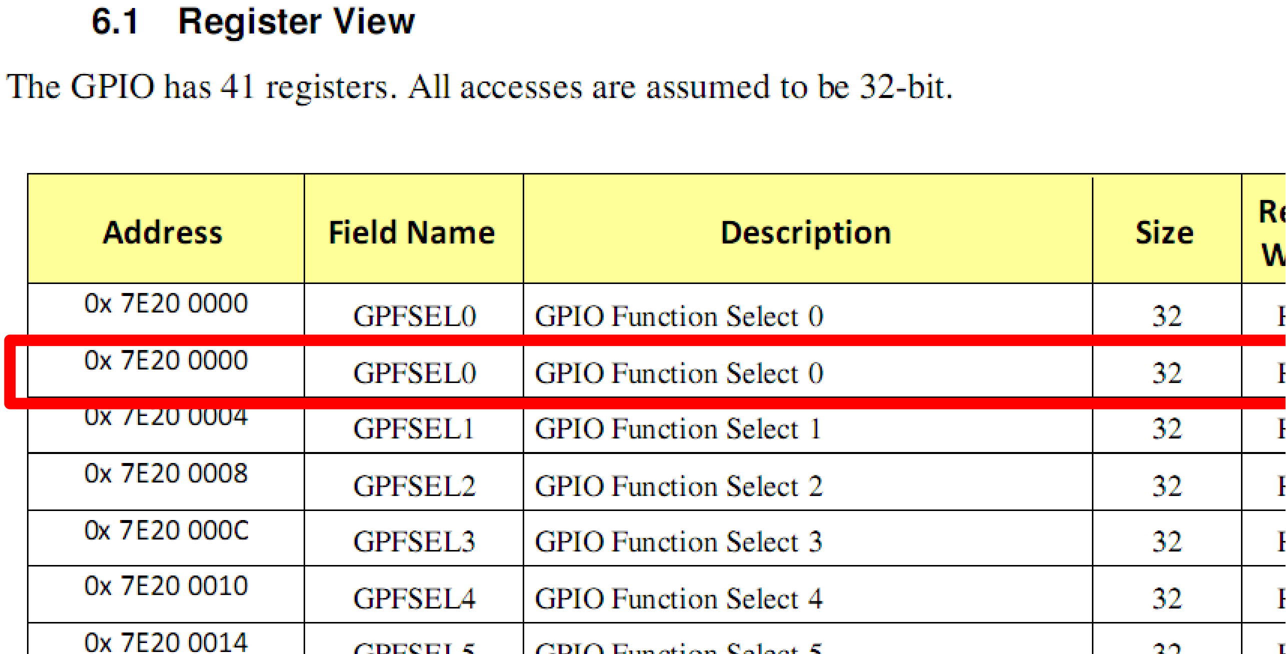
將某根 PIN 腳 (g=4) 設成 INPUT

```
#define INP_GPIO(g)                \
        (*(gpio.addr + ((g)/10)) &= ~(7<<(((g)%10)*3)))
```

xxxxxxxxxxxxxxxxxx000xxxxxxxxxxxxx 寫到 0x7E200000



Bit(s)	Field Name	Description
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



6.1 Register View

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

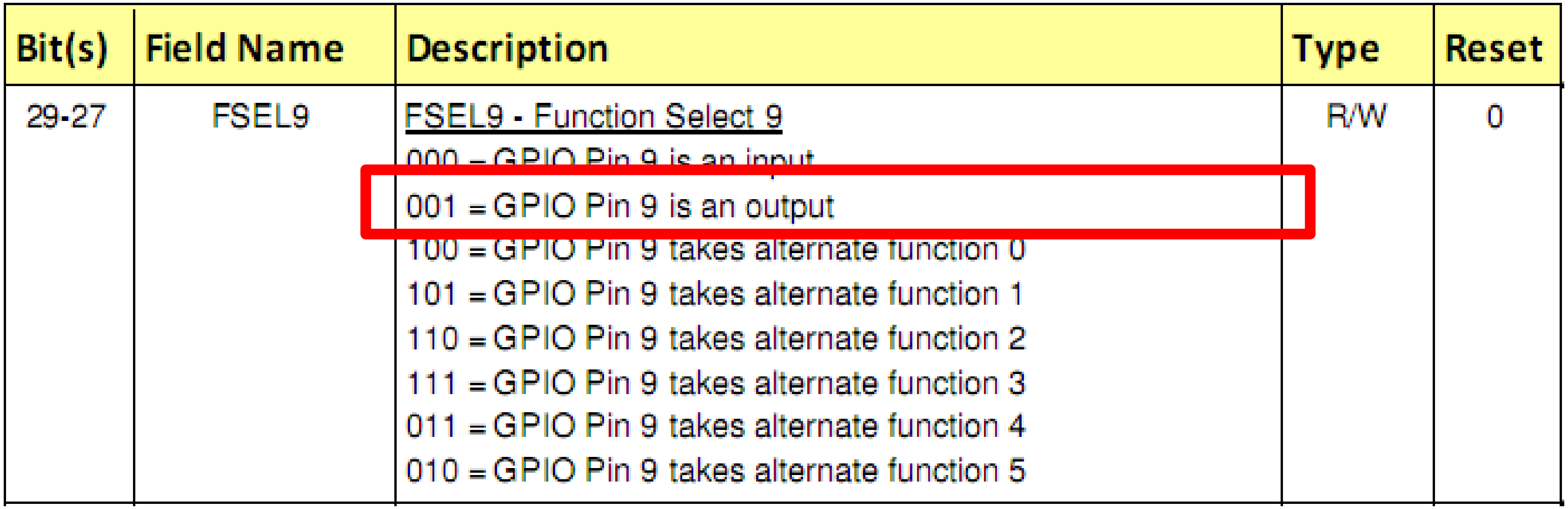
Address	Field Name	Description	Size	Re W
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R
0x 7E20 0004	GPFSEL1	GPIO Function Select 1	32	R
0x 7E20 0008	GPFSEL2	GPIO Function Select 2	32	R
0x 7E20 000C	GPFSEL3	GPIO Function Select 3	32	R
0x 7E20 0010	GPFSEL4	GPIO Function Select 4	32	R
0x 7E20 0014	GPFSEL5	GPIO Function Select 5	32	R

範例 2 ：
將某根 PIN 腳 (g=4) 設成 OUTPUT

依樣畫葫蘆

```
// RPI.h
#define OUT_GPIO(g)                      \
            (*(gpio.addr + ((g)/10)) |= (1<<(((g)%10)*3)))
```

查表決定 FSEL 的 bit 值設定



Bit(s)	Field Name	Description	Type	Reset
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

001 表示 INPUT

- 將 001 寫到原 register 中第 12-14 個 bit

Bitwise 運算 , g=4

原 : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (32-bit)

```
|= (1<<(((g)%10)*3)))
```

(4%10)*3 : 找第 12 個 bit

1 << 12 : 將 001 左移 12 位

原 : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

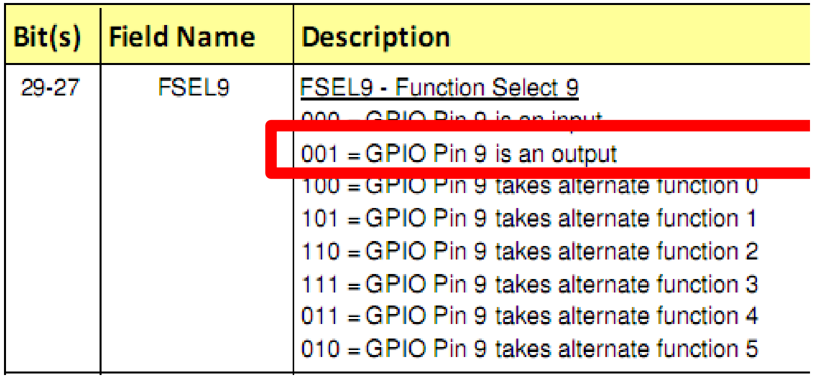
寫 : 0000000000000000000100000000000 (OR 運算)

新 : xxxxxxxxxxxxxxxxxxx|xxxxxxxxxxxx (結果)

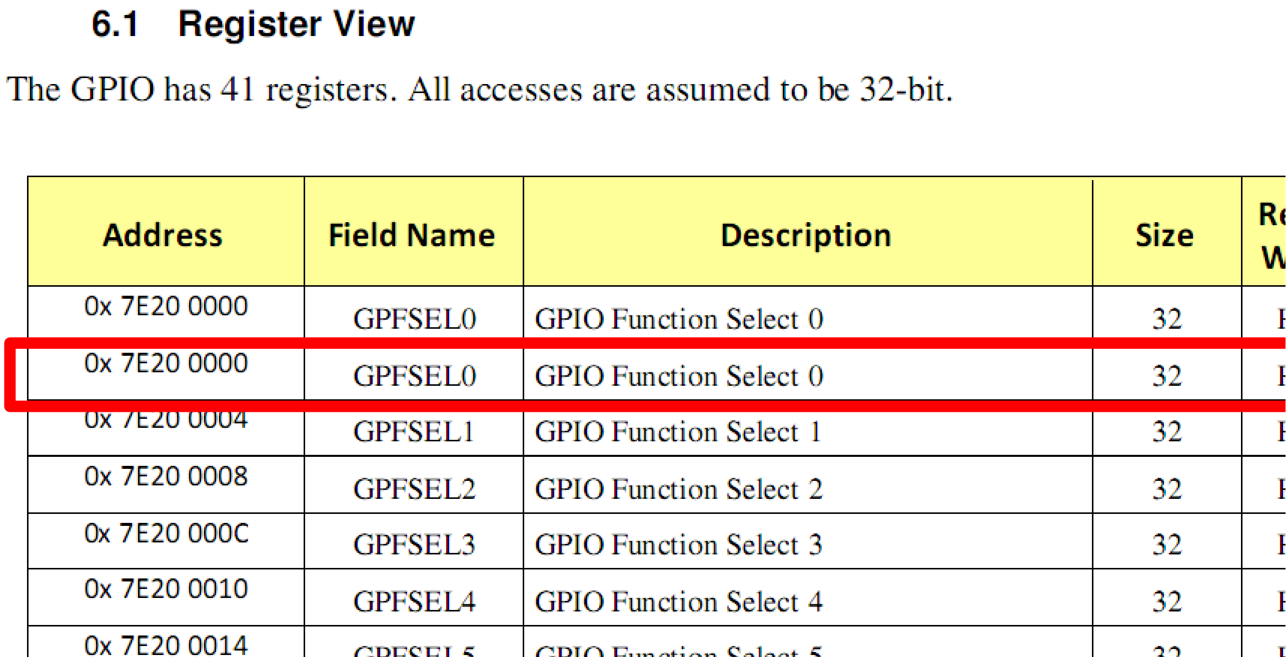
將某根 PIN 腳 (g=4) 設成 OUTPUT

```
#define INP_GPIO(g)                \
            (*(gpio.addr + ((g)/10)) |= (1<<(((g)%10)*3)))
```

xxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxxxx 寫到 0x7E200000



Bit(s)	Field Name	Description
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



6.1 Register View

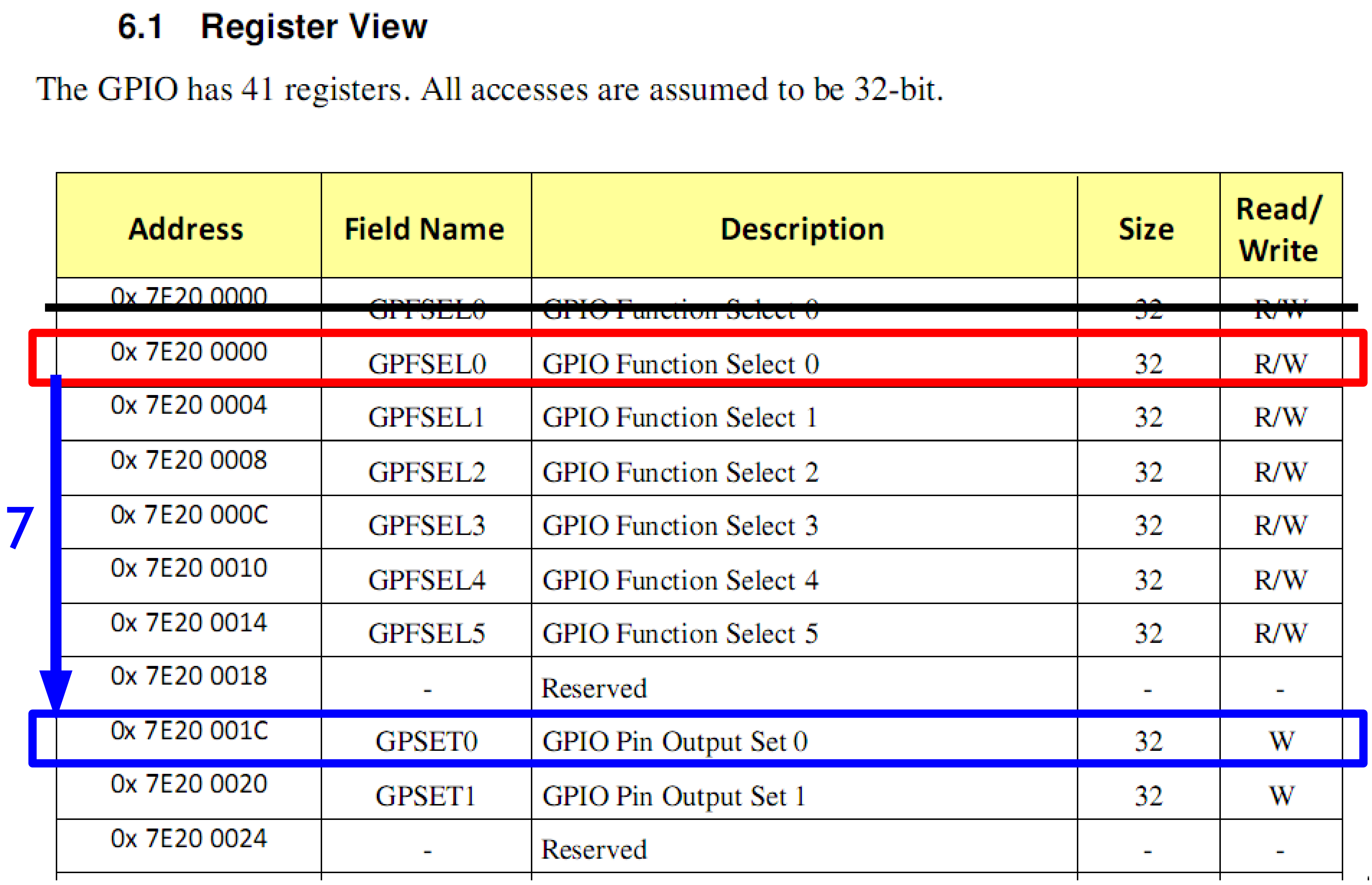
The GPIO has 41 registers. All accesses are assumed to be 32-bit.

Address	Field Name	Description	Size	Re W
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	
0x 7E20 0004	GPFSEL1	GPIO Function Select 1	32	
0x 7E20 0008	GPFSEL2	GPIO Function Select 2	32	
0x 7E20 000C	GPFSEL3	GPIO Function Select 3	32	
0x 7E20 0010	GPFSEL4	GPIO Function Select 4	32	
0x 7E20 0014	GPFSEL5	GPIO Function Select 5	32	

範例 3 ：

SET 值到某根 PIN 腳 (g=4)

Page 90



6.1 Register View

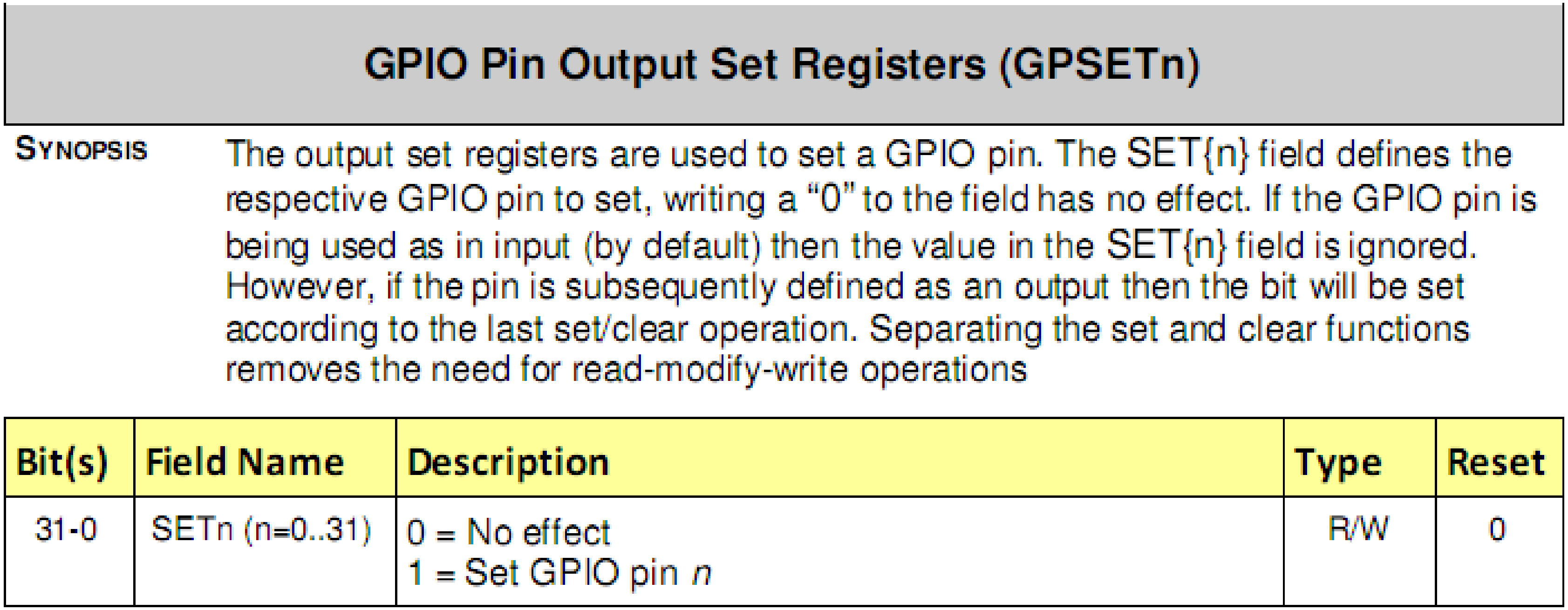
The GPIO has 41 registers. All accesses are assumed to be 32-bit.

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	-	-

再看一次吧

```
// RPI.h
#define GPIO_SET                    (*(gpio.addr + 7))
```

查表決定 GPSETn 的 bit 值設定



GPIO Pin Output Set Registers (GPSETn)

SYNOPSIS The output set registers are used to set a GPIO pin. The SET{n} field defines the respective GPIO pin to set, writing a “0” to the field has no effect. If the GPIO pin is being used as in input (by default) then the value in the SET{n} field is ignored. However, if the pin is subsequently defined as an output then the bit will be set according to the last set/clear operation. Separating the set and clear functions removes the need for read-modify-write operations

Bit(s)	Field Name	Description	Type	Reset
31-0	SETn (n=0..31)	0 = No effect 1 = Set GPIO pin *n*	R/W	0

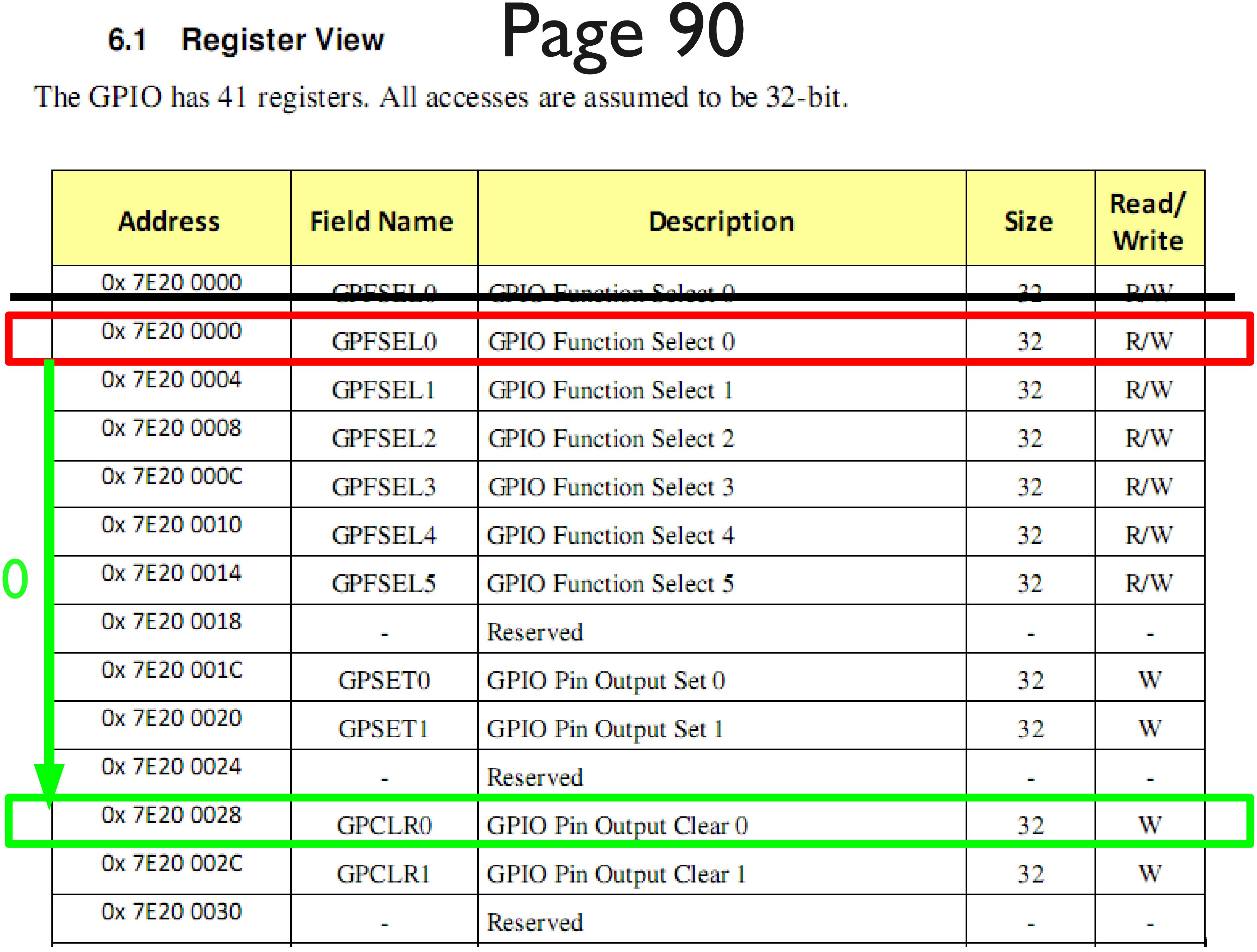
Table 6-8 – GPIO Output Set Register 0

Page 95

- 0 - 31 號 Pin 都是看 GPSET0
- 寫入 1 表示 Set, 寫入 0 無效果

範例 4 ：
CLEAR 某根 PIN 腳 (g=4)

Page 90



6.1 Register View

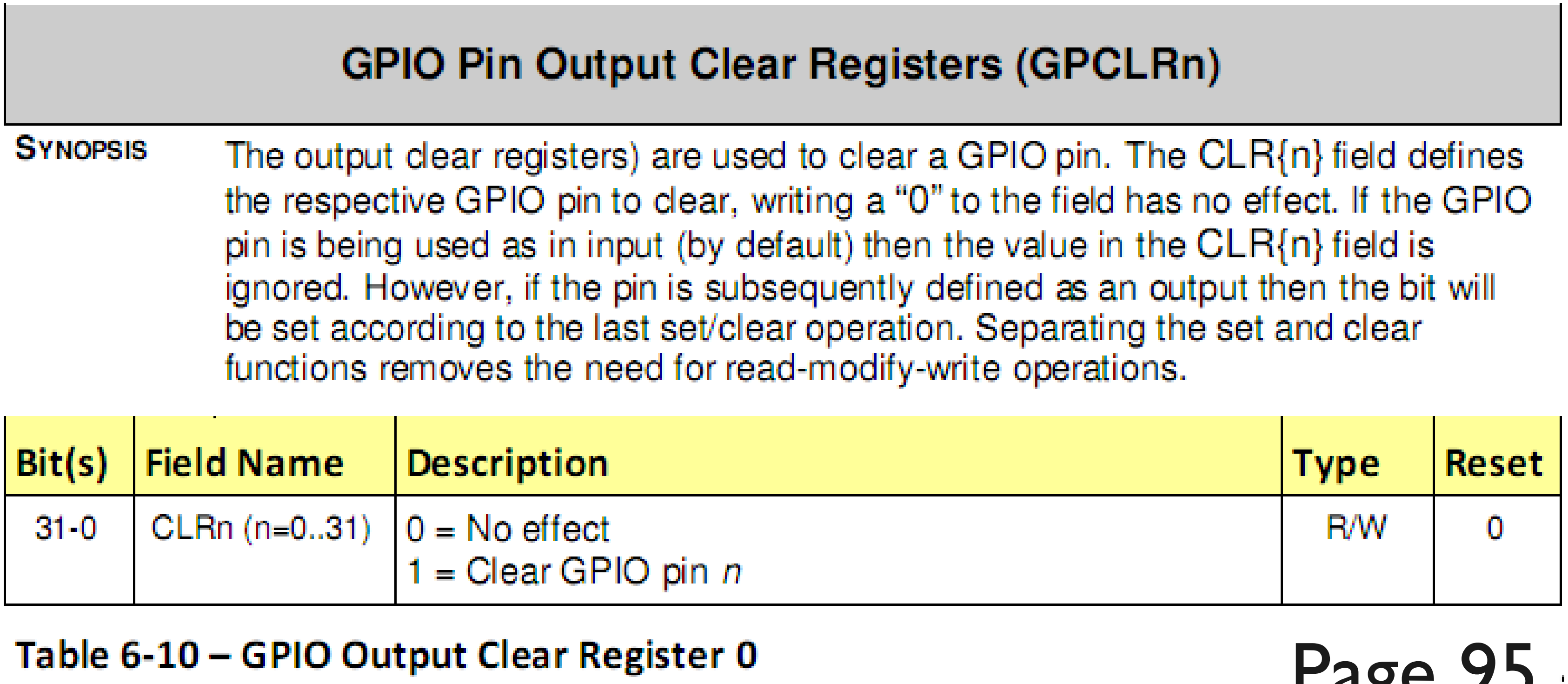
The GPIO has 41 registers. All accesses are assumed to be 32-bit.

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	-	-
0x 7E20 0028	GPCLR0	GPIO Pin Output Clear 0	32	W
0x 7E20 002C	GPCLR1	GPIO Pin Output Clear 1	32	W
0x 7E20 0030	-	Reserved	-	-

最後一次機會

```
// RPI.h
#define GPIO_CLR            (*(gpio.addr + 10))
```

查表決定 GPCLRn 的 bit 值設定



GPIO Pin Output Clear Registers (GPCLRn)

SYNOPSIS The output clear registers) are used to clear a GPIO pin. The CLR{n} field defines the respective GPIO pin to clear, writing a “0” to the field has no effect. If the GPIO pin is being used as in input (by default) then the value in the CLR{n} field is ignored. However, if the pin is subsequently defined as an output then the bit will be set according to the last set/clear operation. Separating the set and clear functions removes the need for read-modify-write operations.

Bit(s)	Field Name	Description	Type	Reset
31-0	CLRn (n=0..31)	0 = No effect 1 = Clear GPIO pin *n*	R/W	0

Table 6-10 – GPIO Output Clear Register 0

Page 95

- 0 - 31 號 Pin 都是看 GPCLR0
- 寫入 1 表示 Clear, 寫入 0 無效果

寫了這幾個 macro

然後呢？

存取 register = 在記憶體位置讀寫值

先定義週邊成一個 structure

```
// RPI.h
struct bcm2835_peripheral {
        unsigned long addr_p;              //  指到實體記憶體位址
        int mem_fd;                        //  開啟 /dev/mem 的 fd
        void *map;                         // memory map 的回傳
        volatile unsigned int *addr;       //  指到 register 的位址
};


// RPI.c
struct bcm2835_peripheral gpio = {GPIO_BASE};
```

1. 開啟記憶體裝置
2. 映射到實體記憶體空間

```
// RPI.c
fd = open(“/dev/mem”, O_RDWR|O_SYNC);
mmap(NULL,
          BLOCK_SIZE,
          PROT_READ,
          MAP_SHARED,
          mem_fd,
          addr_p);
```

準備的差不多了

寫個用 C 控制 GPIO 的 Hello World 吧

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 INPUT
- 跑一個無窮迴圈 while

```
{
    SET 該 PIN 為 HIGH
    休息一秒

    CLEAR 該 PIN
    休息一秒
}
```

實際程式

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

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

DEMO

透過 driver 進行操作

那就是另外一個故事了

用 Python 就快樂多了

https://github.com/raspberrypi-tw/tutorial/tree/master/gpio/led/python

安裝 RPi.GPIO 套件

- 自動安裝：使用 APT 套件管理系統

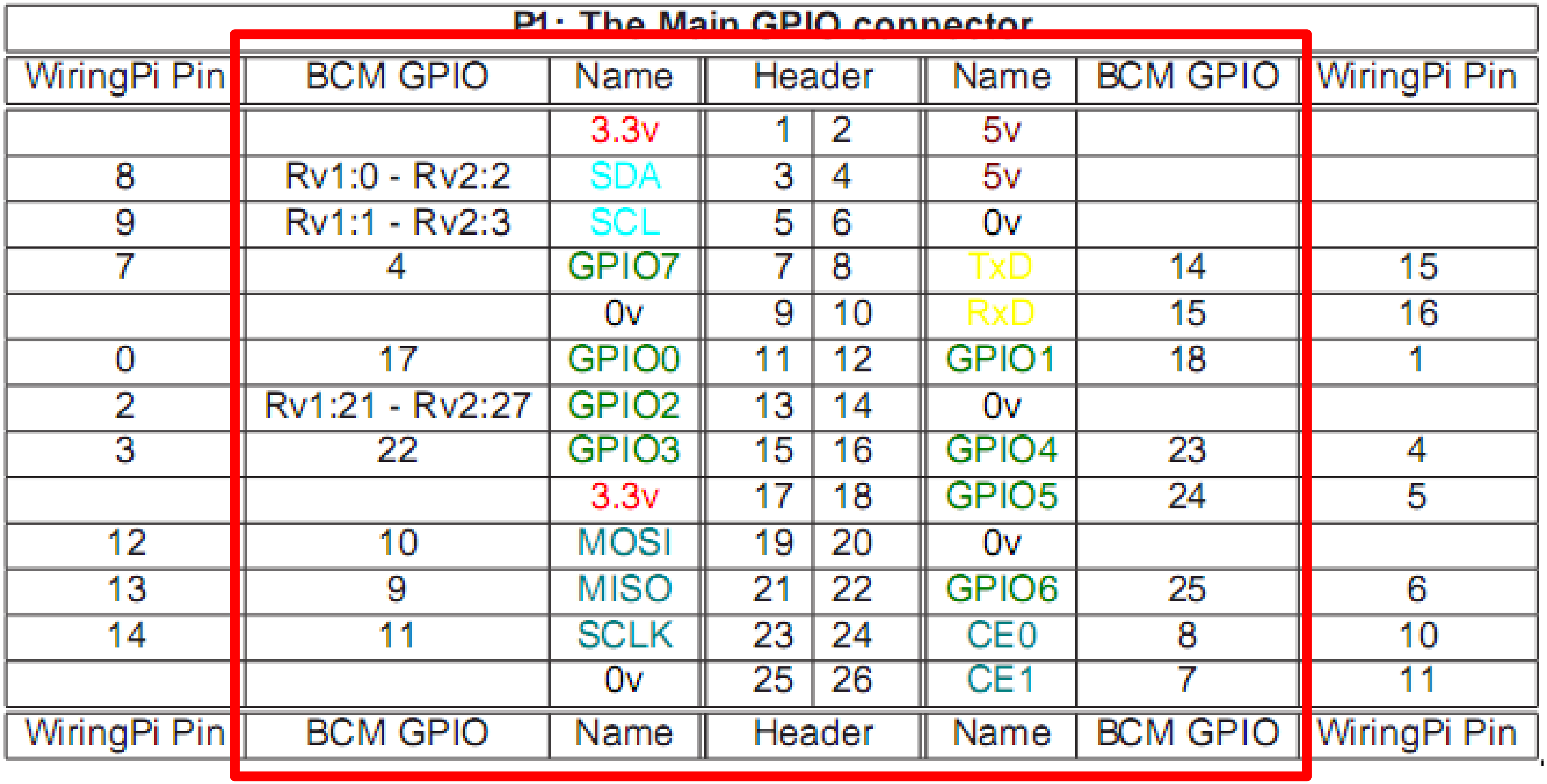
```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get install python-rpi.gpio python3-rpi.gpio
```

- 客製化安裝：下載原始檔並安裝

```
$ wget http://raspberry-gpio-
python.googlecode.com/files/RPi.GPIO-0.5.3a.tar.gz
$ sudo apt-get install python-dev python3-dev
$ sudo python setup.py install
```

http://code.google.com/p/raspberry-gpio-python/

Broadcom 腳位定義



P1: The Main GPIO connector

WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin
		3.3v	1	2	5v		
8	Rv1:0 - Rv2:2	SDA	3	4	5v		
9	Rv1:1 - Rv2:3	SCL	5	6	0v		
7	4	GPIO7	7	8	TxD	14	15
		0v	9	10	RxD	15	16
0	17	GPIO0	11	12	GPIO1	18	1
2	Rv1:21 - Rv2:27	GPIO2	13	14	0v		
3	22	GPIO3	15	16	GPIO4	23	4
		3.3v	17	18	GPIO5	24	5
12	10	MOSI	19	20	0v		
13	9	MISO	21	22	GPIO6	25	6
14	11	SCLK	23	24	CE0	8	10
		0v	25	26	CE1	7	11
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin

http://wiringpi.com/wp-content/uploads/2013/03/pins.pdf

Python Code

- 載入模組 (Import module)
- 選擇系統 (Define pin numbering)
- 定義腳位 (Setup up a channel)
- 讀取輸入 / 寫入輸出 (Input/Output)
- 清理 (Cleanup)

http://code.google.com/p/raspberry-gpio-python/wiki/BasicUsage

Python Code

```
#!/usr/bin/python
import RPi.GPIO as GPIO       # 載入模組
import time
GPIO.setmode(GPIO.BCM)       # 選擇系統
LED_PIN = 4
GPIO.setup(LED_PIN, GPIO.OUT)  # 定義腳位
while True:
    print("LED is on")
    GPIO.output(LED_PIN, GPIO.HIGH)   # 讀取輸入 / 寫入輸出
    time.sleep(1)
    print("LED is off")
    GPIO.output(LED_PIN, GPIO.LOW)    # 讀取輸入 / 寫入輸出
    time.sleep(1)
GPIO.cleanup()  # 清理
```

參考資料

- RPi Low-level peripherals
  - http://elinux.org/RPi_Low-level_peripherals
- Raspberry Pi | Wiring | Gordons Projects
  - https://projects.drogon.net/raspberry-pi/wiringpi/
- Low Level Programming of the Raspberry Pi in C
  - http://www.pieter-jan.com/node/15

http://code.google.com/p/raspberry-gpio-python/wiki/BasicUsage

Raspberry Pi Rocks the World

Thanks