F28HS Hardware-Software Interface: Systems Programming

Hans-Wolfgang Loidl

School of Mathematical and Computer Sciences,
Heriot-Watt University, Edinburgh

Semester 2 — 2019/20

[0]No proprietary software has been used in producing these slides

Outline

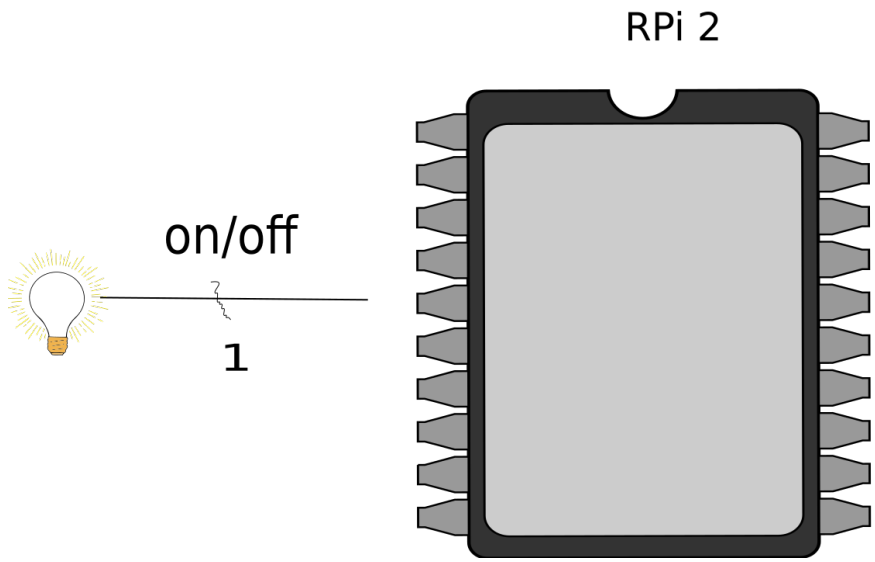
HERIOT WATT UNIVERSITY

Tutorial 2: Programming an LED

- This tutorial will deal with programming an LED output device.
- This is the “hello world” program for external devices.
- It will deal with programming techniques common to other output devices.
- The **learning objective** of this exercise is to learn how to directly control an external device through C and Assembler programs.
- We will also cover easier ways of external control, however these should only be used to test your hardware/software configuration and don’t replace the programming component.

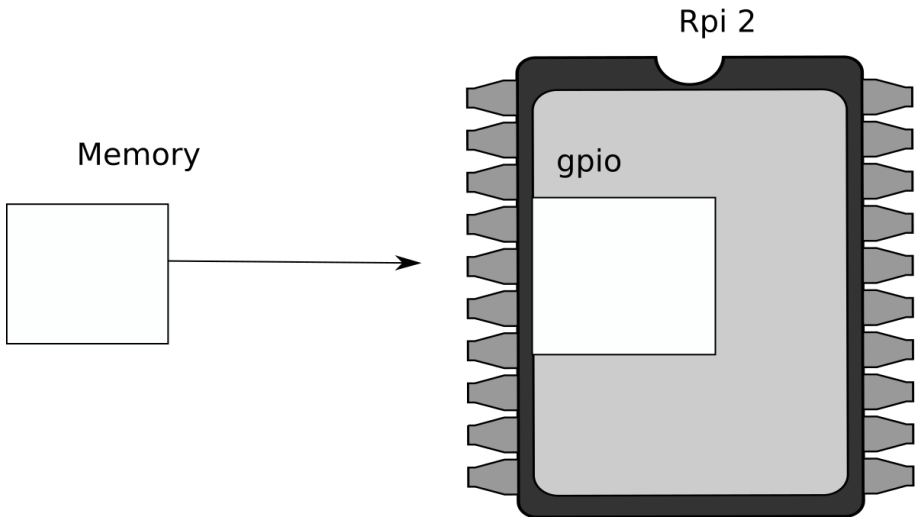
HERIOT WATT UNIVERSITY

The high-level picture



- From the main chip of the RPi2 we want to control an (external) device, here an LED.
- We use one of the **GPIO** pins to connect the device.
- Logically we want to send **1 bit** to this device to turn it **on/off**.

The low-level picture



Programmatically we achieve that, by

- memory-mapping the address space of the GPIOs into user-space
- now, we can directly access the device via memory read/writes
- we need to pick-up the meaning of the peripheral registers from the BCM2835 peripherals sheet

BCM2835 GPIO Peripherals

Base adress: 0x3F000000

0	GPFSEL	Pins 0-9	(3-bits per pin)
5		Pins 50-53	
7	GPSET	Pins 0-31	(1-bit per pin)
8		Pins 32-53	
10	GPCLR	Pins 0-31	(1-bit per pin)
11		Pins 32-53	
13	GPLEV	Pins 0-31	(1-bit per pin)
14		Pins 32-53	
...			

The meaning of the registers is (see p90ff of BCM2835 ARM peripherals):

- **GPFSEL**: function select registers (3 bits per pin); set it to 0 for input, 1 for output; 6 more alternate functions available
- **GPSET**: **set** the corresponding pin
- **GPCLR**: **clear** the corresponding pin
- **GPLEV**: return the **value** of the corresponding pin

GPIO Register Assignment

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	-	-

The GPIO has 48 32-bit registers (RPi2; 41 for RPi1).

[0]See BCM Peripherals Manual, Chapter 6, Table 6.1

GPIO Register Assignment

GPIO registers (Base address: 0x3F200000)

		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
GPFSEL0	0:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
GPFSEL1	1:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
GPFSEL2	2:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
GPFSEL3	3:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
GPFSEL4	4:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
GPFSEL5	5:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
—	6:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
GPFSET0	7:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
GPFSET1	8:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
—	9:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
GPFCLR0	10:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
GPFCLR1	11:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
—	12:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13

[0] See BCM Peripherals, Chapter 6, Table 6.1

Locating the GPFSEL register for pin 47 (ACT)

Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0
29-27	FSEL49	FSEL49 - Function Select 49 000 = GPIO Pin 49 is an input 001 = GPIO Pin 49 is an output 100 = GPIO Pin 49 takes alternate function 0 101 = GPIO Pin 49 takes alternate function 1 110 = GPIO Pin 49 takes alternate function 2 111 = GPIO Pin 49 takes alternate function 3 011 = GPIO Pin 49 takes alternate function 4 010 = GPIO Pin 49 takes alternate function 5	R/W	0
26-24	FSEL48	FSEL48 - Function Select 48	R/W	0
23-21	FSEL47	FSEL47 - Function Select 47	R/W	0
20-18	FSEL46	FSEL46 - Function Select 46	R/W	0
17-15	FSEL45	FSEL45 - Function Select 45	R/W	0
14-12	FSEL44	FSEL44 - Function Select 44	R/W	0
11-9	FSEL43	FSEL43 - Function Select 43	R/W	0
8-6	FSEL42	FSEL42 - Function Select 42	R/W	0
5-3	FSEL41	FSEL41 - Function Select 41	R/W	0
2-0	FSEL40	FSEL40 - Function Select 40	R/W	0

Table 6-6 – GPIO Alternate function select register 4

HERIOT WATT UNIVERSITY

Accessing a GPIO Pin

- Now we want to control the on-chip LED, called ACT, that normally indicates activity.
- The pin number of this device on the RPi2 is: **47**
- We need to calculate registers and bits corresponding to this pin
- The **GPFSEL** register for pin 47 is **4** (per docu, this register covers pins 40-49 (Tab 6-6, p. 94)
- For each register 3 bits are used to select the function of that pin: bits 0–2 for register 40 etc
- Thus, bits 21–23 cover register 47 (7×3)
- The function that we need to select is OUTPUT, which is encoded as the value 1
- We need to write the value `0x01` into bits 21–23 of register 4

HERIOT WATT UNIVERSITY

Accessing a GPIO Pin

- Now we want to control the on-chip LED, called ACT, that normally indicates activity.
- The pin number of this device on the RPi2 is: **47**
- We need to calculate registers and bits corresponding to this pin
- The **GPFSEL** register for pin 47 is **4** (per docu, this register covers pins 40-49 (Tab 6-6, p. 94)
- For each register 3 bits are used to select the function of that pin: bits 0–2 for register 40 etc
- Thus, bits 21–23 cover register 47 (7×3)
- The function that we need to select is OUTPUT, which is encoded as the value 1
- We need to write the value `0x01` into bits 21–23 of register 4

HERIOT WATT UNIVERSITY

Accessing a GPIO Pin

- Now we want to control the on-chip LED, called ACT, that normally indicates activity.
- The pin number of this device on the RPi2 is: **47**
- We need to calculate registers and bits corresponding to this pin
- The **GPFSEL** register for pin 47 is **4** (per docu, this register covers pins 40-49 (Tab 6-6, p. 94)
- For each register 3 bits are used to select the function of that pin: bits 0–2 for register 40 etc
- Thus, bits 21–23 cover register 47 (7×3)
- The function that we need to select is OUTPUT, which is encoded as the value 1
- We need to write the value `0x01` into bits 21–23 of register 4

HERIOT WATT UNIVERSITY

Accessing a GPIO Pin

- Now we want to control the on-chip LED, called ACT, that normally indicates activity.
- The pin number of this device on the RPi2 is: **47**
- We need to calculate registers and bits corresponding to this pin
- The **GPFSEL** register for pin 47 is **4** (per docu, this register covers pins 40-49 (Tab 6-6, p. 94)
- For each register 3 bits are used to select the function of that pin: bits 0–2 for register 40 etc
- Thus, bits 21–23 cover register 47 (7×3)
- The function that we need to select is OUTPUT, which is encoded as the value 1
- We need to write the value `0x01` into bits 21–23 of register 4

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`? **Answer:** `gpio+4`
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
  Answer: `*(gpio+4)`
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

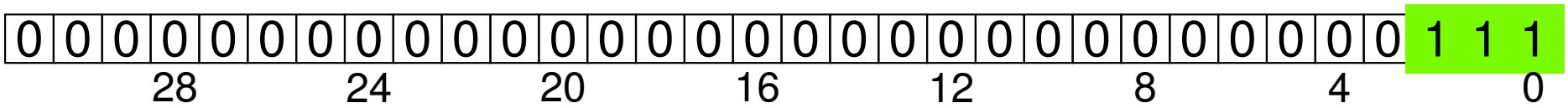
- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
  Answer: `*(gpio + 4) & ~(7 << 21)`
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?

  C code: `7`

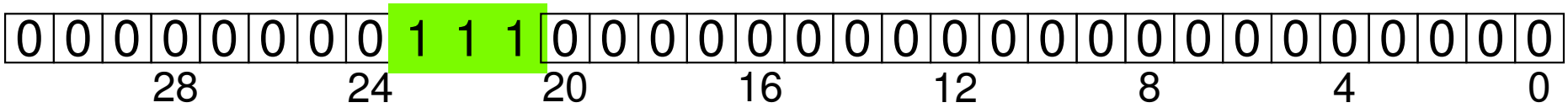


- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?

  C code: `7 << 21`

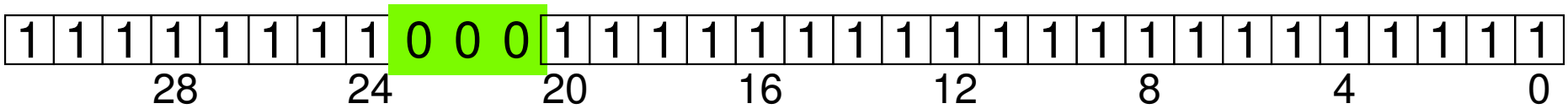


- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?

C code: `~(7 << 21)`

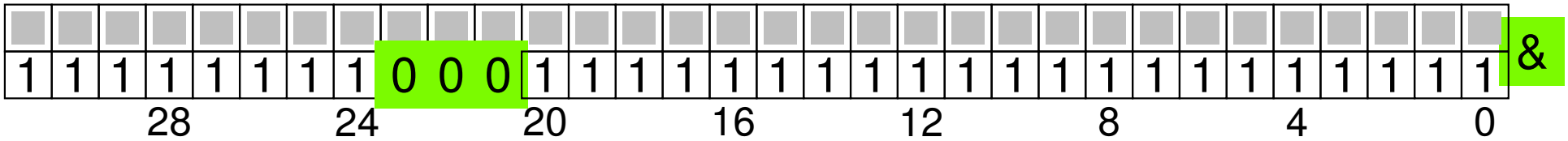


- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?

C code: `(*(gpio + 4) & ~(7 << 21))`

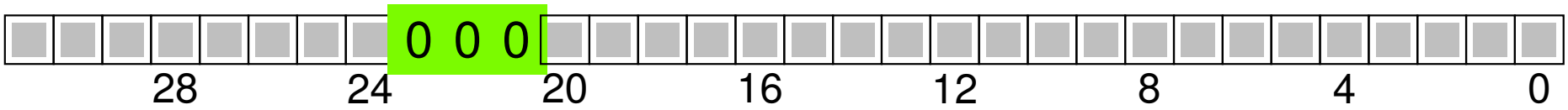


- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?

C code: `(*(gpio + 4) & ~(7 << 21))`



- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

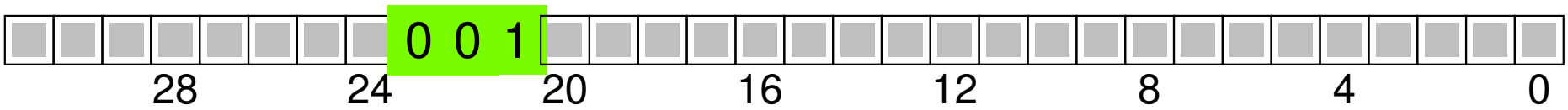
- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
  Answer: `(1 << 21)`
- How do we put **only these bits** into the contents of register 4?

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?

```
(*(gpio + 4) & ~(7 << 21)) | (1 << 21)
```



- How do we put **only these bits** into the contents of register 4?

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

HERIOT WATT UNIVERSITY

Accessing GPIO Pin 47

- We want to construct C code to write the value `0x01` into bits 21–23 of register 4
- What's the address of register 4 relative to the base address in `gpio`?
- How do we read the current value from this register?
- How do we blank out bits 21–23 from this register?
- How do we get the value `0x01` into bits 21–23 of a 32-bit word?
- How do we put **only these bits** into the contents of register 4?

```
*(gpio + 4) = (*(gpio + 4) & ~(7 << 21)) | (1 << 21)
```

HERIOT WATT UNIVERSITY

C Code: constants and memory mapping

```
// constants for RPi2
gpiobase = 0x3F200000 ;

// memory mapping
// Open the master /dev/memory device, and map it to address
    gpio

if ((fd = open("/dev/mem", O_RDWR | O_SYNC | O_CLOEXEC) )< 0)
  return failure (FALSE, "Unable␣to␣open␣/dev/mem:␣%s\n",
      strerror(errno)) ;

// gpio is the mmap'ed device memory
gpio = (uint32_t *)mmap(0, BLOCK_SIZE, PROT_READ|PROT_WRITE,
    MAP_SHARED, fd, gpiobase) ;
if ((int32_t)gpio == -1)
  return failure (FALSE, "␣mmap␣(GPIO)␣failed:␣%s\n",
      strerror(errno)) ;
```

Now, `gpio` is the address of the device memory that we can access directly (if run as root!).

Registers for the GPIO peripherals: GPFSEL

Write into these bits (21–23) to set the **function** for `pin 47`

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
0:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
1:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
2:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
3:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
4:	31	30	29	28	27	26	25	24	pin 47			20	19	18	17	16	15	14	13	12	11	10	9
5:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
6:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
7:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
8:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
9:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
10:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
11:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
12:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9

Registers for the GPIO peripherals: GPFSEL

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
0:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
1:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
2:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
3:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
5:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
6:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
7:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
8:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
9:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
10:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
11:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
12:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8

C Code: setting the mode of the pin

Essentials

Register no.: 4

Bits: 21–23

Function: 1 (output)

```
// setting the mode for GPIO pin 47
fprintf(stderr, "setting␣pin␣%d␣to␣%d␣...\n", pinACT, OUTPUT)
    ;
fSel =  4;    // GPIO 47 lives in register 4 (GPFSEL)
shift =  21;  // GPIO 47 sits in slot 7 of register 4, thus
    shift by 7*3 (3 bits per pin)
*(gpio + fSel) = (*(gpio + fSel) & ~(7 << shift)) | (1 <<
    shift) ;  // Sets bits to one = output
// *(gpio + fSel) = (*(gpio + fSel) & ~(7 << shift)) ;
                 // Sets bits to zero = input
```

Now, pin 47 (the on-board ACT LED) is set as an output device.

C Code: setting the mode of the pin

Essentials

Register no.: 4

Bits: 21–23

Function: 1 (output)

```
// setting the mode for GPIO pin 47
fprintf(stderr, "setting␣pin␣%d␣to␣%d␣...\n", pinACT, OUTPUT)
    ;
fSel =  4;    // GPIO 47 lives in register 4 (GPFSEL)
shift =  21;  // GPIO 47 sits in slot 7 of register 4, thus
    shift by 7*3 (3 bits per pin)
*(gpio + fSel) = (*(gpio + fSel) & ~(7 << shift)) | (1 <<
    shift) ;  // Sets bits to one = output
// *(gpio + fSel) = (*(gpio + fSel) & ~(7 << shift)) ;
                 // Sets bits to zero = input
```

Now, pin 47 (the on-board ACT LED) is set as an output device.

GPIO Registers for Turning the LED on/off

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	-	-

We now need to access the GPSET and GPCLR register for pin 47.

[0]See BCM Peripherals Manual, Chapter 6, Table 6.1

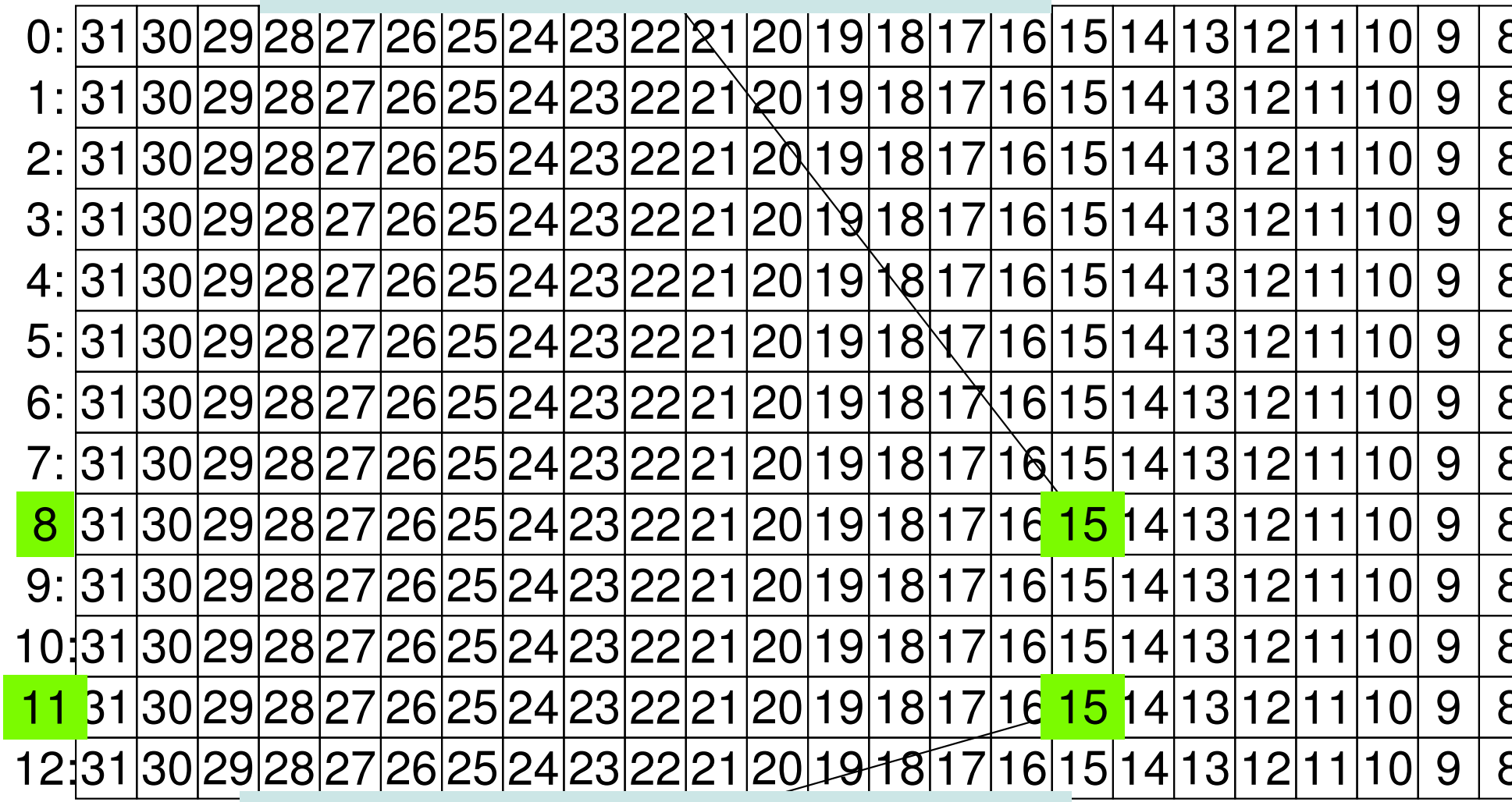
Turning the LED on or off

Write into this bit (15) to **set** `pin 47`

Write into this bit (15) to **clear** `pin 47`

Turning the LED on or off

Write into this bit (15) to **set** `pin 47`



Write into this bit (15) to **clear** `pin 47`

Code: blinking LED

```
for (j=0; j<1000; j++) {
   theValue = ((j % 2) == 0) ? HIGH : LOW;
   // write the value into the location corresp. to pin 47
   if ((pinACT & 0xFFFFFFC0) == 0)      // sanity check
     {
       if (theValue == LOW) { // GPCLR
         // GPCLR for GPIOs 32-53 is register 11
         clrOff = 11; // register for clearing a pin value
         *(gpio + clrOff) = 1 << (pinACT & 31) ;
       } else { // GPSET
         // GPSET for GPIOs 32-53 is register 8
         setOff = 8; // register for setting a pin value
         *(gpio + setOff) = 1 << (pinACT & 31) ;
       }
     } else { fprintf(stderr, "only supporting on-board pins\n
         "); exit(1); }

   // delay for howLong ms, using a Linux system function
   ...
 }
```

Discussion

- In each iteration of the loop, we toggle `theValue` between the constants `HIGH` and `LOW`
- This is **not** the value written to a register, but a flag for the control flow
- If `theValue` is `LOW`, we write a 1 into the corresponding **GPCLR** register, to turn the LED **off**
- If `theValue` is `HIGH`, we write a 1 into the corresponding **GPSET** register, to turn the LED **off**
- Note, that we determine the bit location in these registers by `pinACT & 31`, which is the same as taking pinACT modulo 32
- We then wait for a certain amount of time to control the blinking frequency

See sample source: `tut_led.c`

The main registers that you need to know about

Address	Field Name	Description	Size	Read/ Write
	GPFSEL0	GPIO Function Select 0	32	R/W
	GPFSEL0	GPIO Function Select 0	32	R/W
	GPFSEL1	GPIO Function Select 1	32	R/W
	GPFSEL2	GPIO Function Select 2	32	R/W
	GPFSEL3	GPIO Function Select 3	32	R/W
	GPFSEL4	GPIO Function Select 4	32	R/W
	GPFSEL5	GPIO Function Select 5	32	R/W
	-	Reserved	-	-
	GPSET0	GPIO Pin Output Set 0	32	W
	GPSET1	GPIO Pin Output Set 1	32	W
	-	Reserved	-	-
	GPCLR0	GPIO Pin Output Clear 0	32	W
	GPCLR1	GPIO Pin Output Clear 1	32	W
	-	Reserved	-	-

FctSelect

0
1
2
3
4
5

Set Registers

7
8

Clear Registers

10
11

HERIOT WATT UNIVERSITY

The main registers that you need to know about

Address	Field Name	Description	Size	Read/ Write
	GPFSEL0	GPIO Function Select 0	32	R/W
	GPFSEL0	GPIO Function Select 0	32	R/W
	GPFSEL1	GPIO Function Select 1	32	R/W
	GPFSEL2	GPIO Function Select 2	32	R/W
	GPFSEL3	GPIO Function Select 3	32	R/W
	GPFSEL4	GPIO Function Select 4	32	R/W
	GPFSEL5	GPIO Function Select 5	32	R/W
	-	Reserved	-	-
	GPSET0	GPIO Pin Output Set 0	32	W
	GPSET1	GPIO Pin Output Set 1	32	W
	-	Reserved	-	-
	GPCLR0	GPIO Pin Output Clear 0	32	W
	GPCLR1	GPIO Pin Output Clear 1	32	W
	-	Reserved	-	-

FctSelect

0
1
2
3
4
5

Set Registers

7
8

Clear Registers

10
11

HERIOT WATT UNIVERSITY

Controlling the LED in Assembler

```
@ ... mmap boilerplate here
ADD     R3, R3, #4                @ add 4 for block 1
LDR     R2, [SP, #16]             @ get virtual mem addr
ADD     R2, R2, #16               @ add 16 for block 4
LDR     R2, [R2, #0]              @ load R2 with value at R2
BIC     R2, R2, #0b111<<21        @ Bitwise clear of three bits
STR     R2, [R3, #0]              @ Store result in Register
LDR     R3, [SP, #16]             @ Get virtual mem address
ADD     R3, R3, #16               @ Add 16 for block 4
LDR     R2, [SP, #16]             @ Get virtual mem addr
ADD     R2, R2, #4                @ add 16 for block 4
LDR     R2, [R2, #0]              @ Load R2 with value at R2
ORR     R2, R2, #1<<21            @ Set bit....
STR     R2, [R3, #0]              @ ...and make output
LDR     R3, [SP, #16]             @ get virt mem addr
ADD     R3, R3, #32               @ add 32 to offset for  GPSET1
MOV     R4, #1                    @ get 1
MOV     R2, R4, LSL#15            @ Shift by pin number
STR     R2, [R3, #0]              @ write to memory
```

See sample source: gpio47on.s

[0] From: Bruce Smith *"Raspberry Pi Assembly Language: Raspbian"*, Ch 25

Summary

- Controlling a simple external device means **logically** sending 1 bit of information (on/off)
- Realising this control means **physically** writing into special registers which have special meaning
- The information on the special meaning is usually in bulky hardware-description documentation
- Once uncovered, the code for direct device control is fairly short
- The sample sources show a C and an Assembler version of turning pin 47 (ACT) on/off

Thanks to **Gordon Henderson** for his sterling work on the wiringPi library!

HERIOT WATT UNIVERSITY