02-3 GPIO via mmap()

GPIO via sysfs

- So far we've been access the GPIO pins via sysfs
- You can turn a USR LED on with

bone\$ cd /sys/class/leds/beaglebone\:green\:usr3

bone\$ echo none > trigger

- bone\$ echo 1 > brightness
- What if speed is needed?

• sysf is portable, but can be slow

GPIO via mmap()

- All the IO on the am335x is memory mapped
- You can look them up on the am335xTechnical Reference Manual (TRM)

USR3 LED

bone\$ cd ~/exercises/gpio
bone\$ sudo ./findGPIO.js USR3

```
{ name: 'USR3', gpio: 56, led: 'usr3', mux: 'gpmc_a8', key: 'USR3', muxRegOffset: '0x060', options: [ 'gpmc_a8', 'gmii2_rxd3', 'rgmii2_rd3', 'mmc2_dat6', 'gpmc_a24', 'pr1_mii1_rxd0', 'mcasp0_aclkx', 'gpio1_24' ] } USR3 (gpio 56) mode: 7 (gpio1_24) 0x060 pullup pin 24 (44e10860): (MUX UNCLAIMED) (GPIO UNCLAIMED)
```

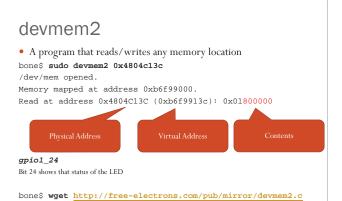
From Table 2-3 of TRM

Table 2-3. L4_PER Peripheral Memory Map (continued)

Device Name	Start_address (hex)	End_address (hex)	Size	Description
DMTIMERS .	0x4804_6000	0x4804_6FFF	4KB	DMTimer5 Registers
	0×4804_7000	0x4804_7FFF	4KB	Reserved
DMTIMER6	0x4804_8000	0x4804_8FFF	4KB	DMTimer6 Registers
	0x4804_9000	0x4804_9FFF	4KB	L4 Interconnect
DMTIMER7	0x4804_AD00	0x4804_AFFF	4108	DMTimer7 Registers
	0x4804_8000	0x4804_BFFF	4KB	Reserved
GPIO1	0x4804_C000	0x4804_CFFF	4KB	GPIO1 Registers
	0x4004_D000	0x4804_DFFF	4KB	Reserved
Reserved	0x4804_E000	0x4804_FFFF	8KB	Reserved

- Base address is **0x4804_C000.**
- Click on GPIO1

	Section
GPIO_REVISION	Section 25.4.1.1
字母:Paoning ケーケ OT IRIVI	Section 25.4.1.2
	Section 25.4.1.3
GPIO_IRQSTATUS_RAW_0	Section 25.4.1.4
GPIO_IRQSTATUS_RAW_1	Section 25.4.1.5
GPIO_IRQSTATUS_0	Section 25.4.1.6
GPIO_IRQSTATUS_1	Section 25.4.1.7
GPIO_IROSTATUS_SET_0	Section 25.4.1.8
GPIO_IRQSTATUS_SET_1	Section 25.4.1.9
GPIO_IRQSTATUS_CLR_0	Section 25.4.1.10
GPIO_IRQSTATUS_CLR_1	Section 25.4.1.11
GPIO_IRQWAKEN_0	Section 25.4.1.12
GPIO_IRQWAKEN_1	Section 25.4.1.13
GPIO_SYSSTATUS	Section 25.4.1.14
GPIO_CTRL	Section 25.4.1.15
GPIO_OE	Section 25.4.1.16
GPIO_DATAIN	Section 25.4.1.17
GPIO_DATAOUT	Section 25.4.1.18
GPIO LEVELDETECTO	Section 25.4.1.19
GPIO_LEVELDETECT1	Section 25.4.1.20
GPIO_RISINGDETECT	Section 25.4.1.21
GPIO_FALLINGDETECT	Section 25.4.1.22
GPIO_DEBOUNCENABLE	Section 25.4.1.23
GPIO_DEBOUNCINGTIME	Section 25.4.1.24
GPIO_CLEARDATAOUT	Section 25.4.1.25
GPIO_SETDATAOUT	Section 25.4.1.26
	CPIC_PROVIDED TO CPIC_P



Toggle the LED - PIC

- The PIC way
 - Read register
 - XOR with (1<<24)
 - Write register
- 3 operations

Toggle the LED

bone\$ gcc -o devmem2 devmem2.c

• Use GPIO_SETDATAOUT and GPIO_CLEARDATAOUT

ı	150h	GPIO_DEBOUNCENABLE	Section 25.4.1.23
I	154h	GPIO_DEBOUNCINGTIME	Section 25.4.1.24
İ	190h	GPIO_CLEARDATAOUT	Section 25.4.1.25
I	194h	GPIO_SETDATAOUT	Section 25.4.1.26

- Write to **GPIO_SETDATAOUT** a value with 1's for the pins to be set to 1
- Write to **GPIO_CLEARDATAOUT** a value with 1's for the pins to be cleared to 0
- Use 0x190 to Clear

Turn LED off then on

GPIO_DATAOUT

Off

bone\$ sudo devmem2 0x4804c190 w 0x01000000 /dev/mem opened.

Memory mapped at address 0xb6f53000.

Read at address 0x4804C190 (0xb6f53190): 0x01800000 Write at address 0x4804C190 (0xb6f53190): 0x010000000, readback 0x01000000

• On

bone\$ sudo devmem2 0x4804c194 w 0x01000000/dev/mem opened.

Memory mapped at address 0xb6f9f000.

Read at address 0x4804C194 (0xb6f9f194): 0x00800000 Write at address 0x4804C194 (0xb6f9f194): 0x01800000, readback 0x01800000

mmap()

- The same can be done more quickly from a C program using mmap()
- mmap() is a way of mapping a physical address space into a user-space program

Exercise GPIO via mmap

- Homework has you work through some examples
- gpioThru.c copies an input pin to an output

