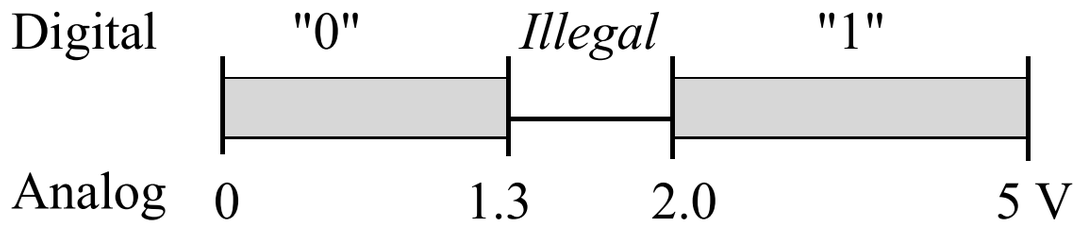
GPIO I

ECE 3710

On the other hand...
You have different fingers.

- Steven Wright



Digital I/O (GPIO)

represented by

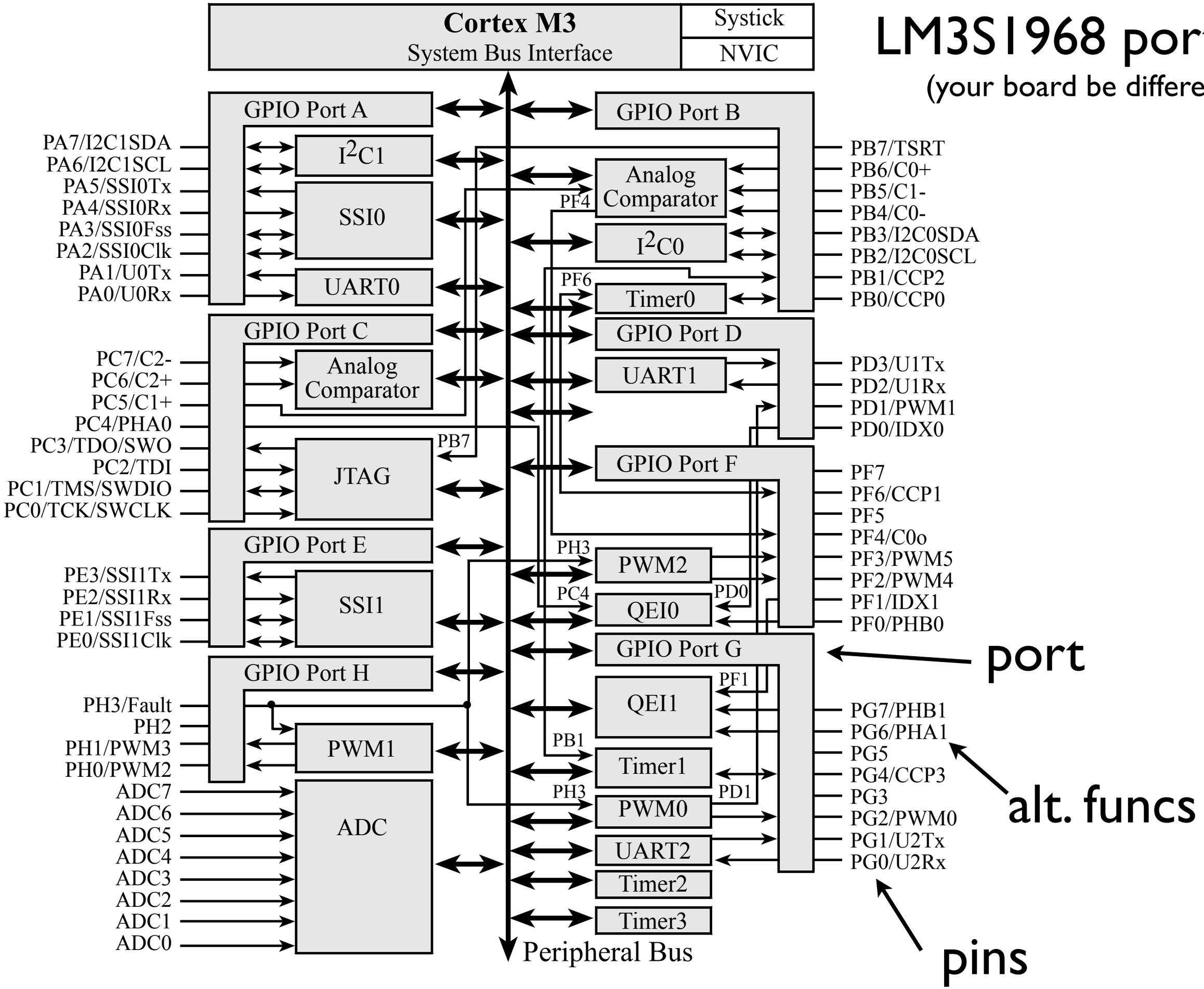
input/output: *ones* and *zeros*

happens through

ports
(consist of pins)

LM3S1968 ports

(your board be different)

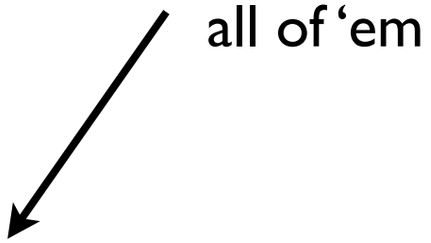


a reflection on
the dual nature of pins:

alt func: UART, timers, interrupts

(fun stuff...or evil, your decision)

this works for all of 'em



gpio on cortex m3:

1. enable clock on port
2. disable *alternative function*
3. configure pins as input or output
4. set speed and pull-up/down
5. enable pins
6. read or write

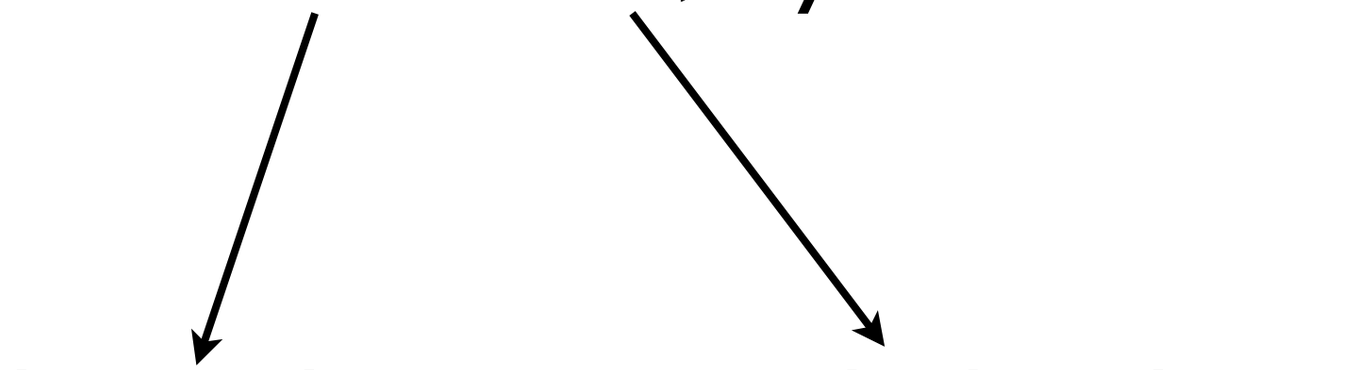
(move stuff to/from registers)

i.e. we configure ports by flipping bits at certain addresses

remember: I/O is memory mapped

we input/output what?

what: 0s and 1s, my friends



low voltage

high voltage

got that?

that's the chapter, really

(besides for a little EE stuff;
only useful if you want to actually connect the uController to something)

if so....

...we all go home

or is this your response?

output: set pin to 1

input: we set pin to 0

LM3S1968 ports A--E memory map

define use EQU

Address	7	6	5	4	3	2	1	0	Name
$400F.E108	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGC2_R
$4000.43FC			DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTA_DATA_R
$4000.4400			DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTA_DIR_R
$4000.4420			SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTA_AFSEL_R
$4000.451C			DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTA_DEN_R
$4000.53FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTB_DATA_R
$4000.5400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTB_DIR_R
$4000.5420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTB_AFSEL_R
$4000.551C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTB_DEN_R
$4000.63FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTC_DATA_R
$4000.6400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTC_DIR_R
$4000.6420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTC_AFSEL_R
$4000.651C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTC_DEN_R
$4000.73FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTD_DATA_R
$4000.7400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTD_DIR_R
$4000.7420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTD_AFSEL_R
$4000.751C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTD_DEN_R
$4002.43FC							DATA	DATA	GPIO_PORTE_DATA_R
$4002.4400							DIR	DIR	GPIO_PORTE_DIR_R
$4002.4420							SEL	SEL	GPIO_PORTE_AFSEL_R
$4002.451C							DEN	DEN	GPIO_PORTE_DEN_R

preferred naming convention: GPIO_PORTX => PX

note: each memory location refers to word;
we consider only bits 7--0

1. enable clock on port/

set bit for port we want to work (bit = 1)

```
ldr R1,=SYSCTL_RCGC2_R ;get addr. of clock
;enable PA
mov R0,#0x1 ;0x1=0b1
str R0,[R1]
;enable PA & PH
mov R0,#0x81 ;0x81=0b10000001
str R0,[R1]
;enable PE (don't disturb previous settings)
ldr R0,[R1] ;get current clock config for ports
orr R0,#0x10 ;0x10 = 0b00010000
str R0,[R1]
nop
nop
```

$400F.E108	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_

2. disable alternative function

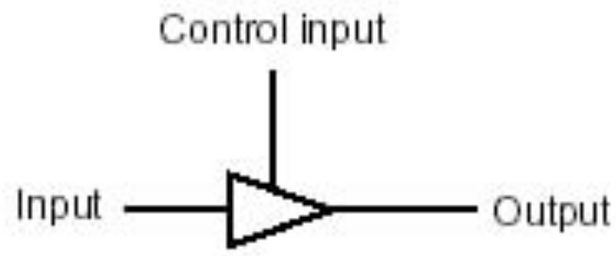
clear bits for pin(s) on port(s)
we want for GPIO
(bit = 0)

```
;configure AF for port b
ldr R1,=PB_AFSEL_R ;get addr. alt. func reg
;disable AF for port
mov R0,#0x0
str R0,[R1]
;enable AF for pins zero and seven
;(don't disturb previous settings)
ldr R0,[R1] ;get AF config for port
orr R0,#0x81 ;0x81 = 0b10000001
str R0,[R1]
```

$4000.5420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTB_AFSEL_R

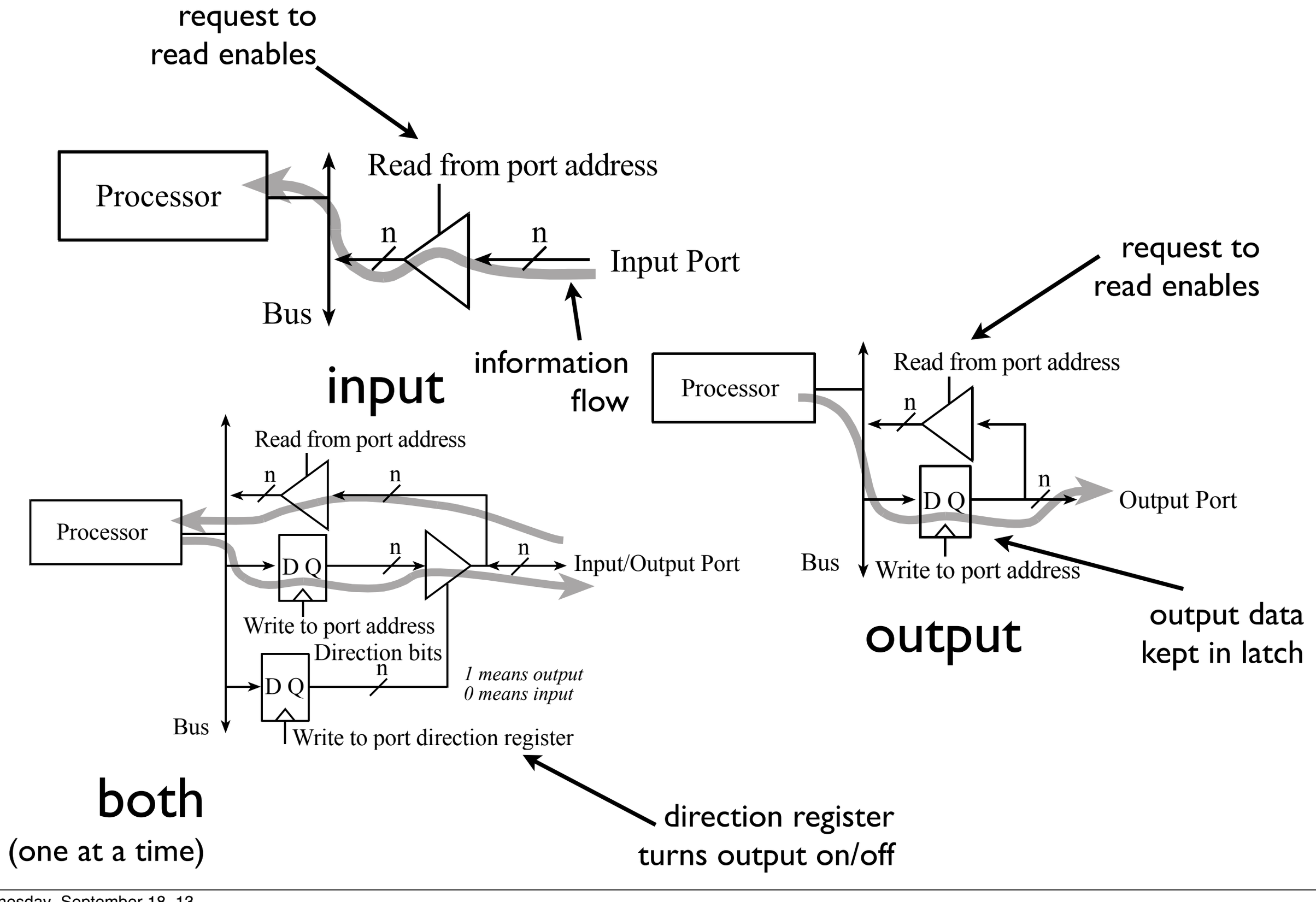
3. configure pins as input or output

tri-state buffer:



In	Control	Out
0	0	Hi Z
1	0	Hi Z
0	1	0
1	1	1

3. configure pins as input or output



3. configure pins as input or output

for a pin to be:
output = 1 (set)
input = 0 (clear)

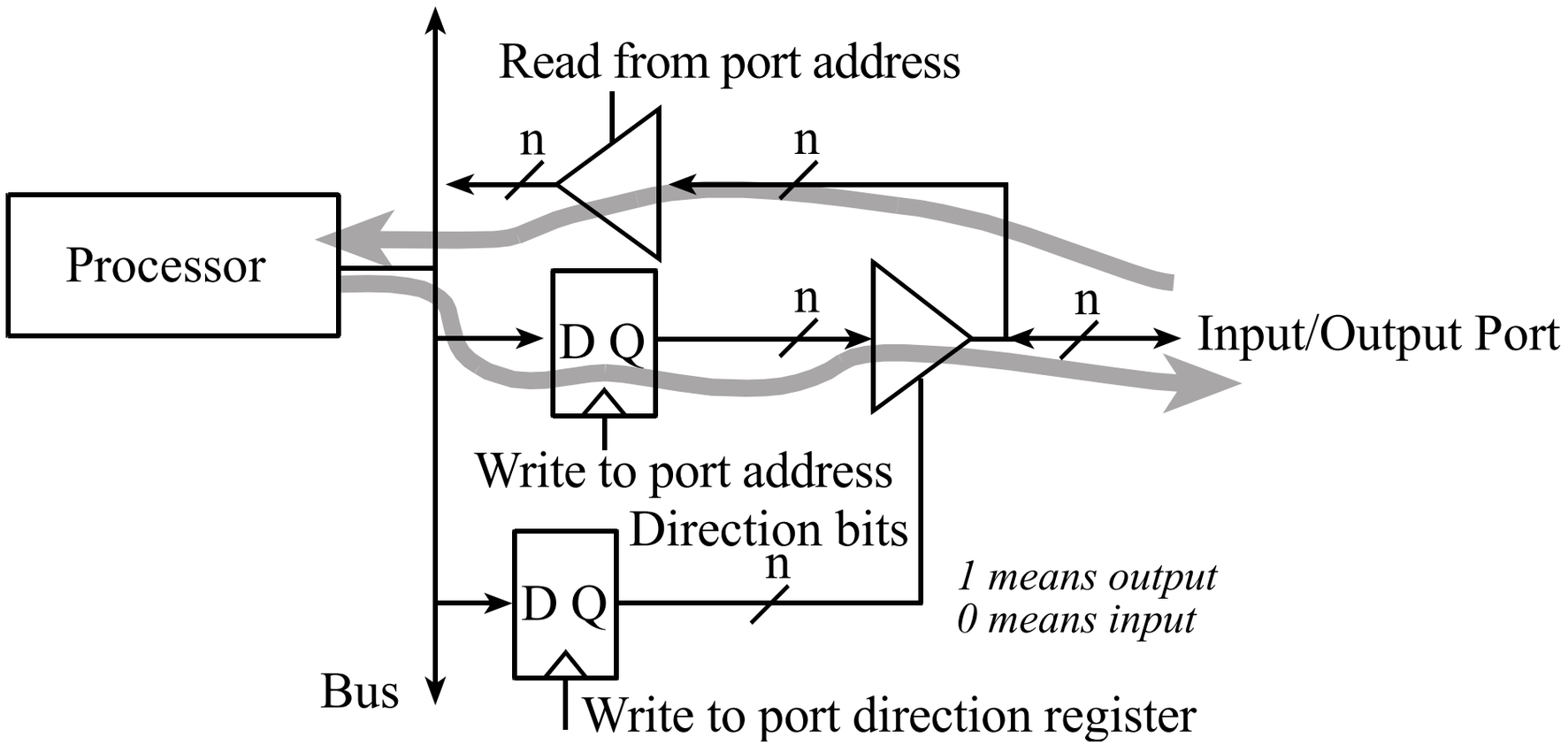
```
ldr R1,=PB_DIR_R ;get addr. for direction reg
;port b as input
mov R0,#0x0
str R0,[R1]
;port b as output
mov R0,#0xFF
str R0,[R1]
;pins 2--0 input; 7--5 output
;(don't disturb previous settings)
ldr R0,[R1] ;get I/O config for port
orr R0,#0xE0 ;output: 0xE0=0b11100000
and R0,#0xF8 ;input: 0xF8=0b11111000
str R0,[R1]
```

specify input/output;
(pin can be either but only one at at a time [not quite])

$4000.5400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTB_DIR_R

an I/O port

(conceptual)

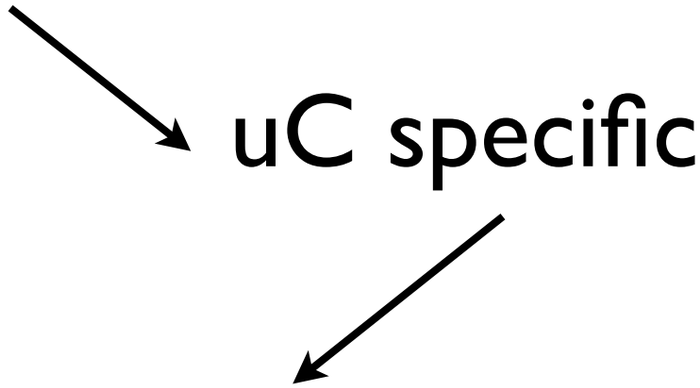


for a pin to be:

output `DIR` = 1 (set)

input `DIR` = 0 (clear)

4. set physical parameters



uC specific

so: how ports work

modes:

1. input

a. pull-up

(default state is high)

b. pull-down

(default state is low)

2. output

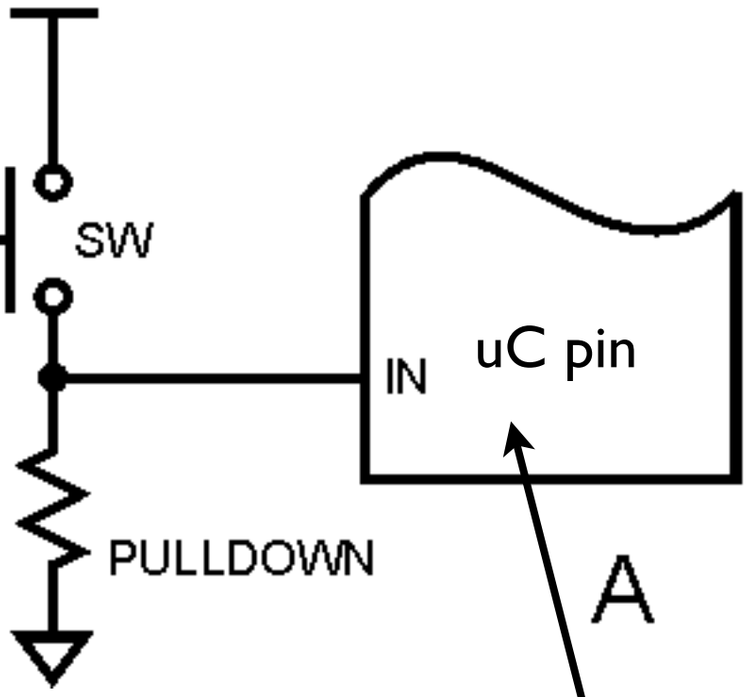
a. open drain

b. open drain w/pull-up

c. push-pull

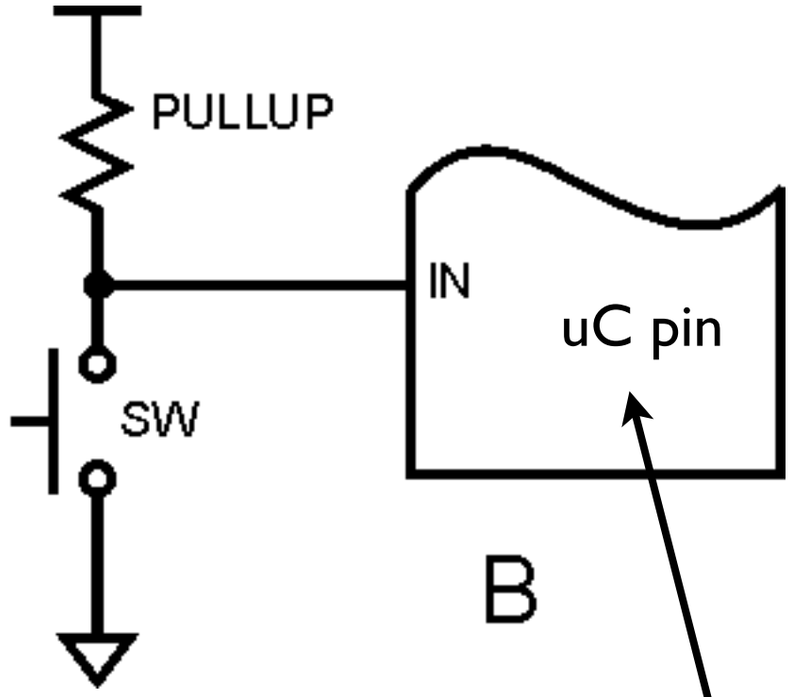
input/output: pull-up/down

Vcc = 5 V



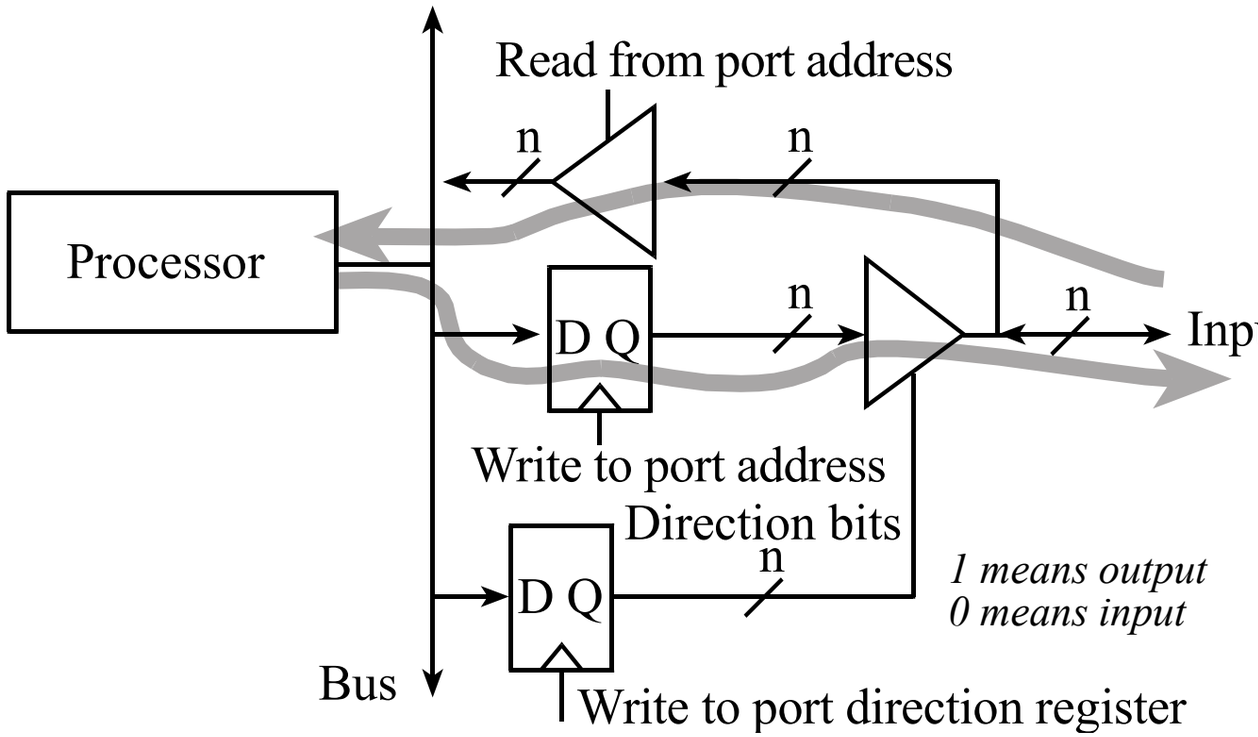
sees 0 V
(switch open)
sees 5 V
(switch closed)

Vcc = 5 V



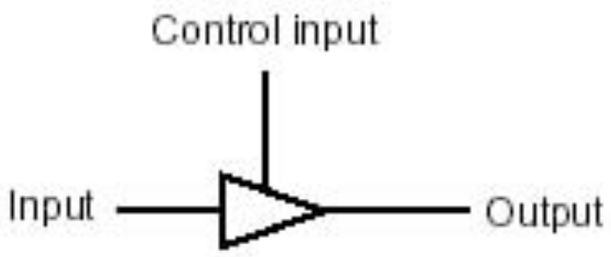
sees 5 V
(switch open)
sees 0 V
(switch closed)

Q: where does I/O pin fit in?

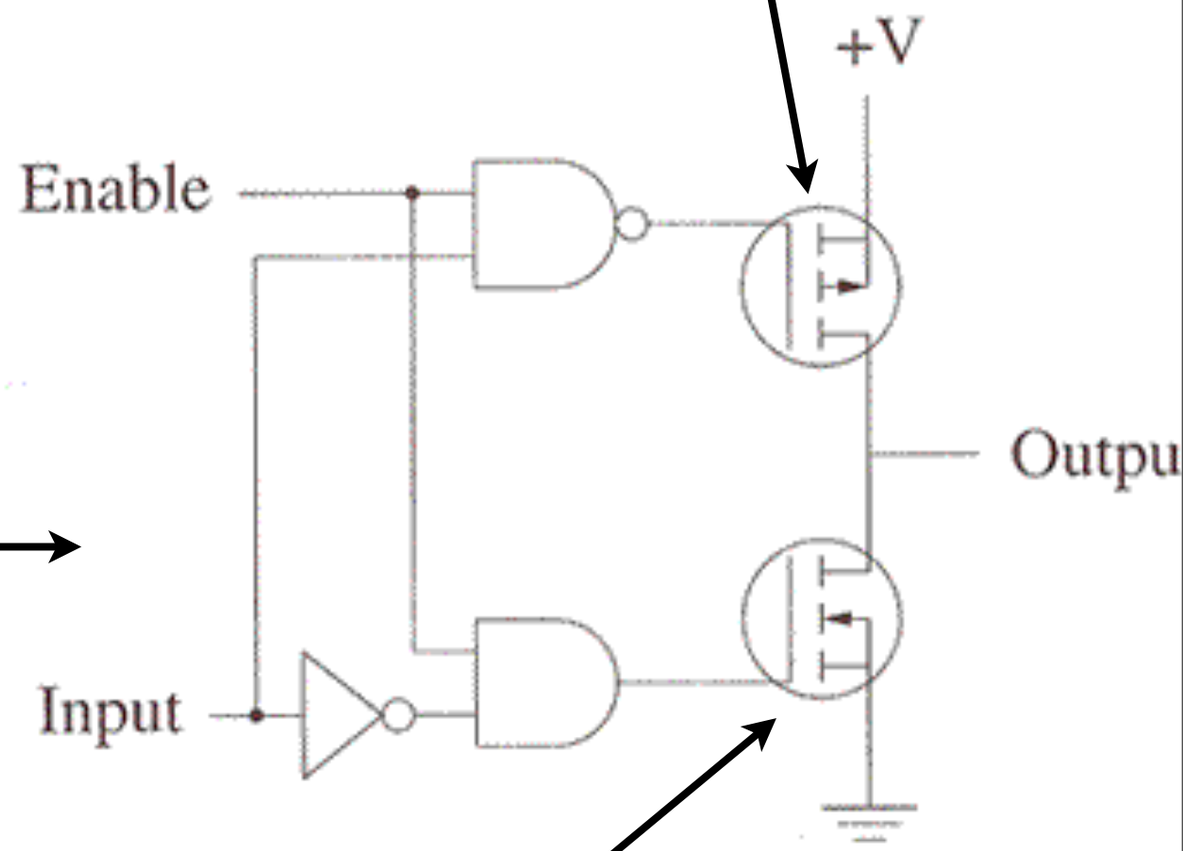


simplified
(doesn't handle
open drain case)

on if input=0
off if input=1

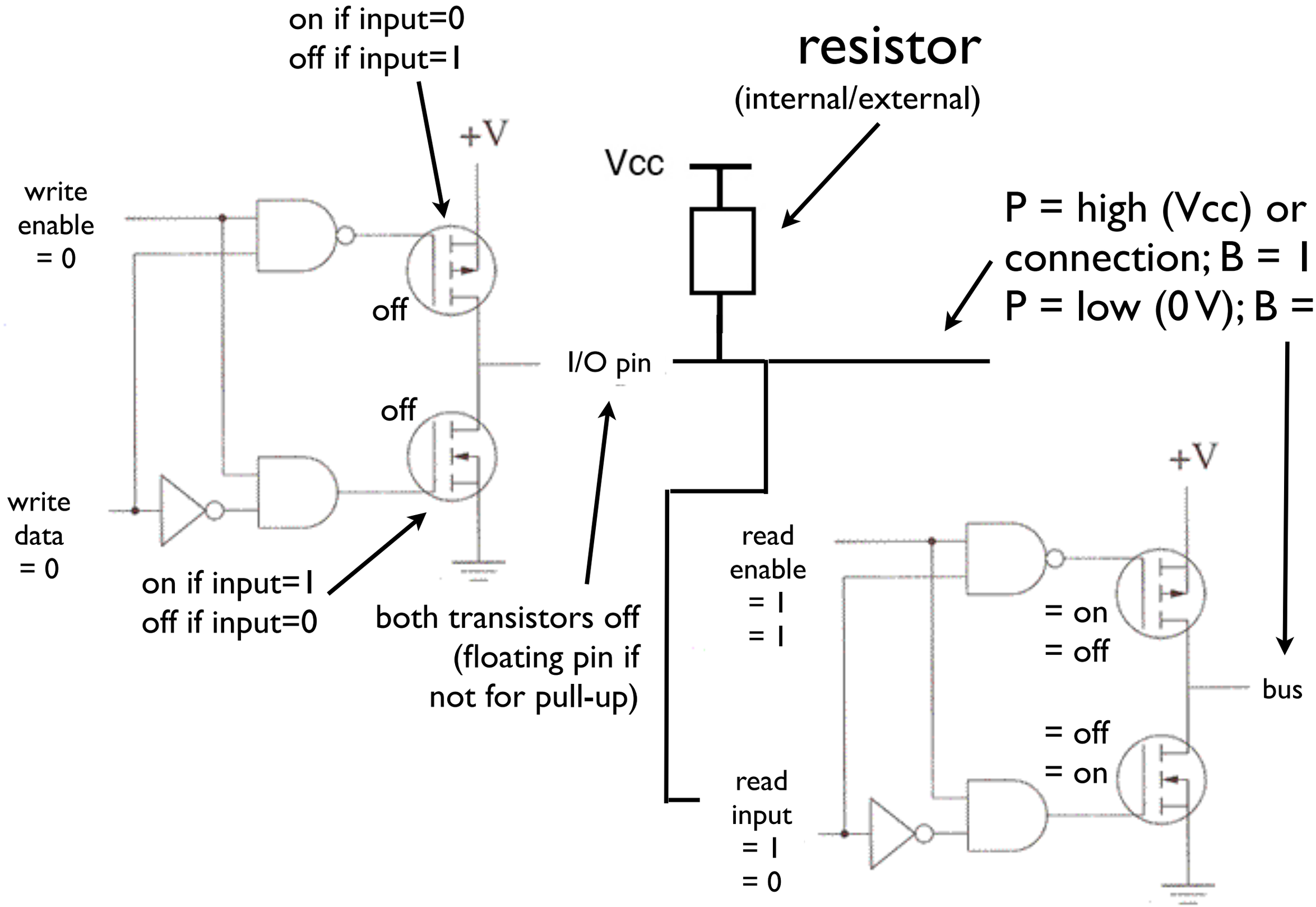


In	Control	Out
0	0	Hi Z
1	0	Hi Z
0	1	0
1	1	1



on if input=1
off if input=0

input: pull-up

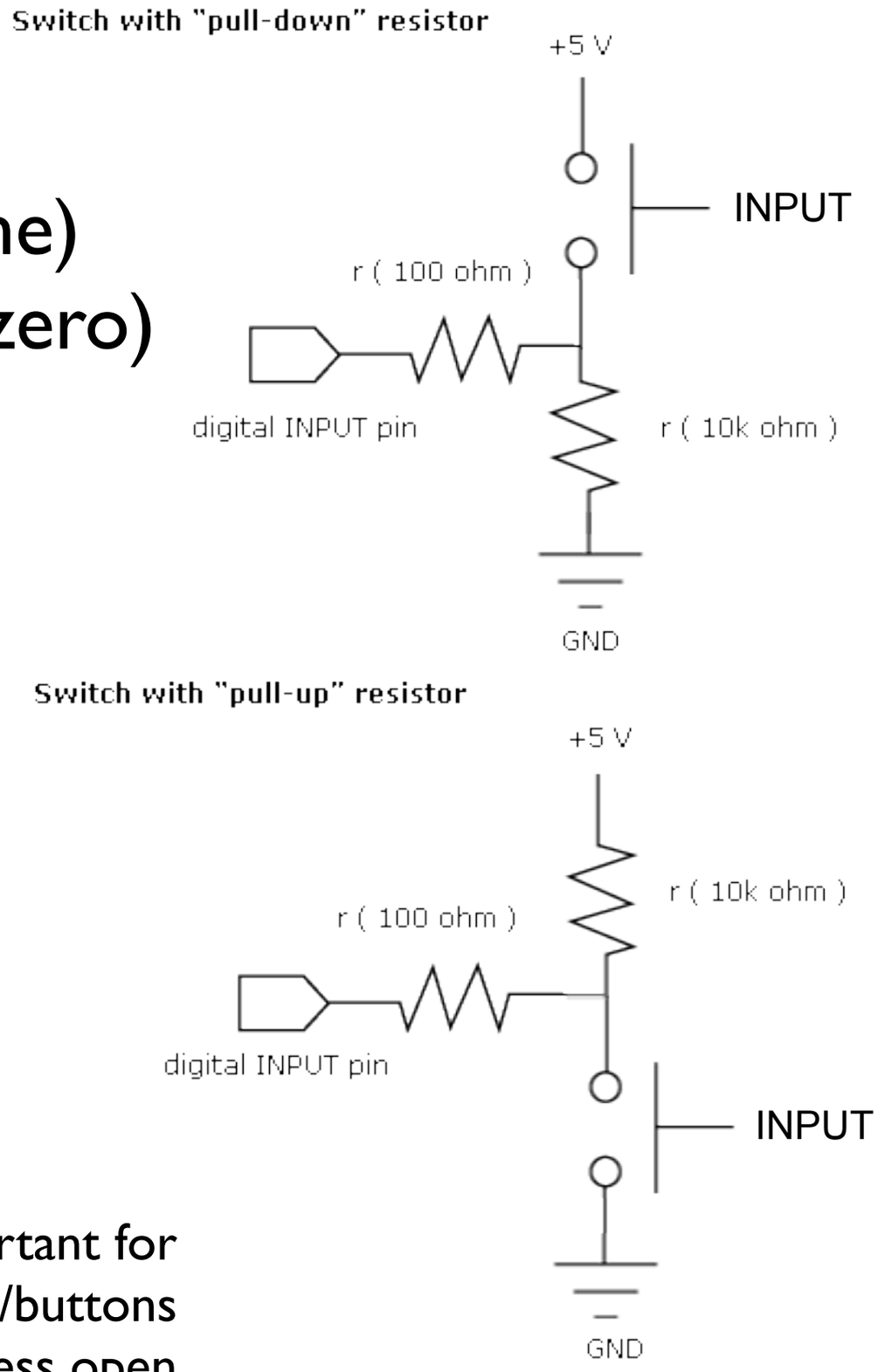


input: pull-up/down

output from
external device = Vcc

if input (connected) high: PX.y = Vcc (logic one)
if input (connected) low: PX.y = GND (logic zero)

output from
external device = 0

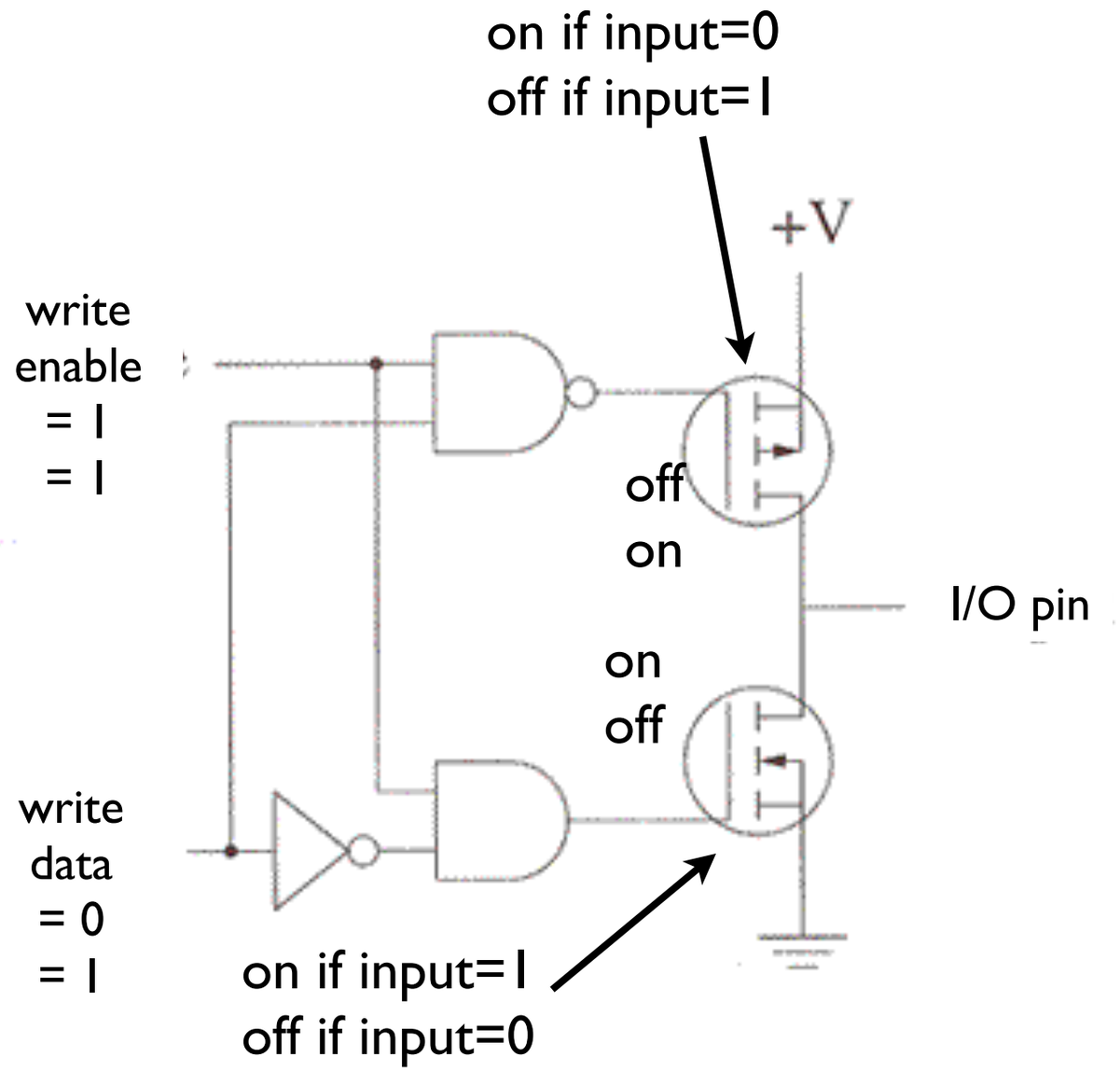


no connection:

pull-up = high
pull-down = low

important for switches/buttons (i.e. does press open or close switch?)

output: push-pull → when one is on, the other is off



P = 0 V

P = Vcc

if output one: P = Vcc (logic one)

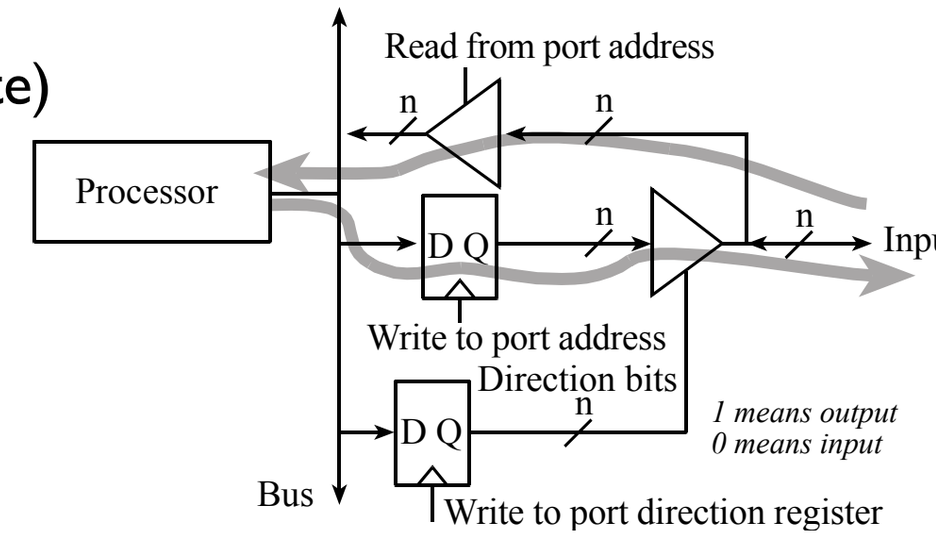
if output zero: P = GND (logic zero)

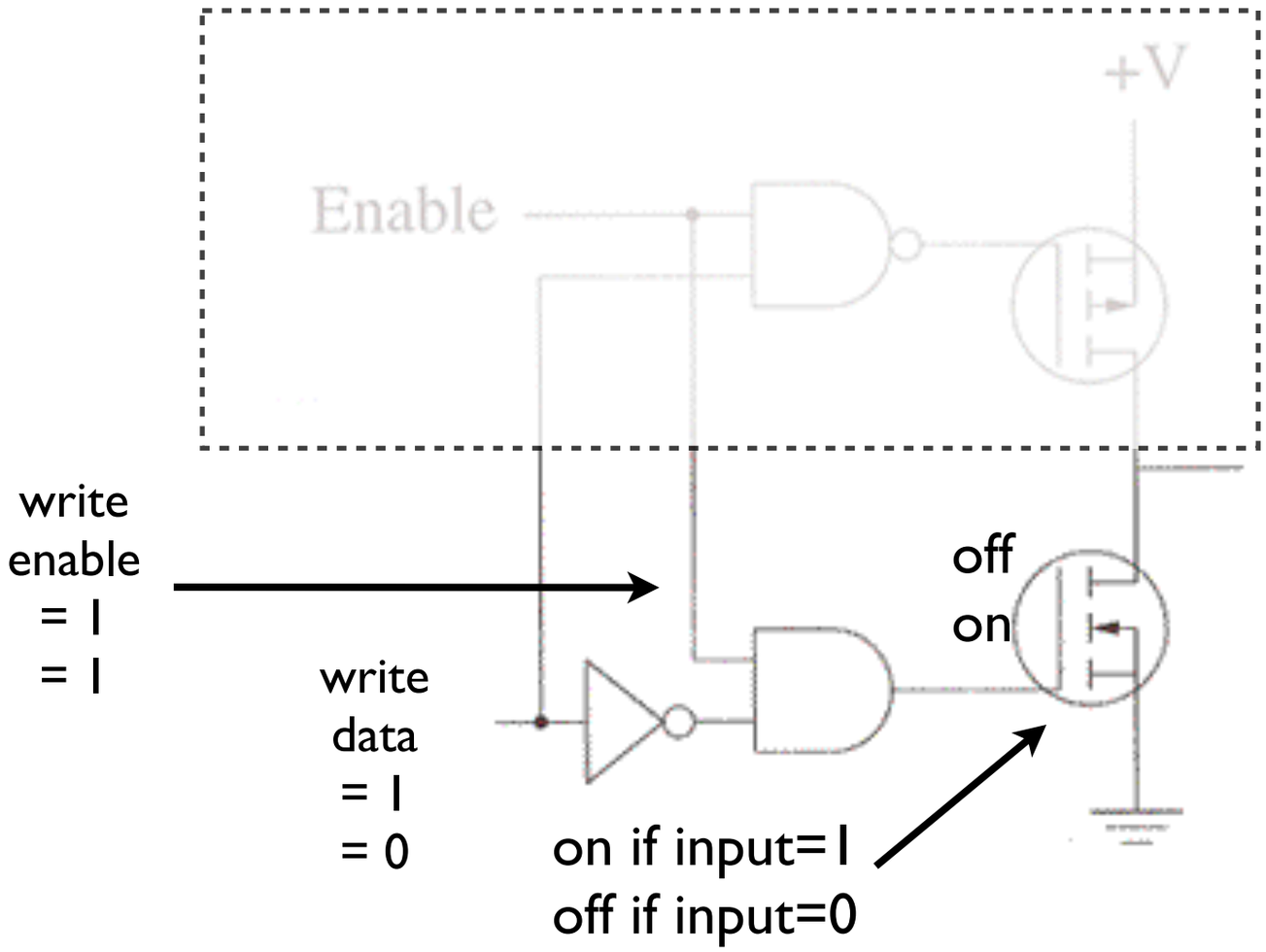
open drain

(primarily concerned w/write)

disables top transistor

`DIR = 1`



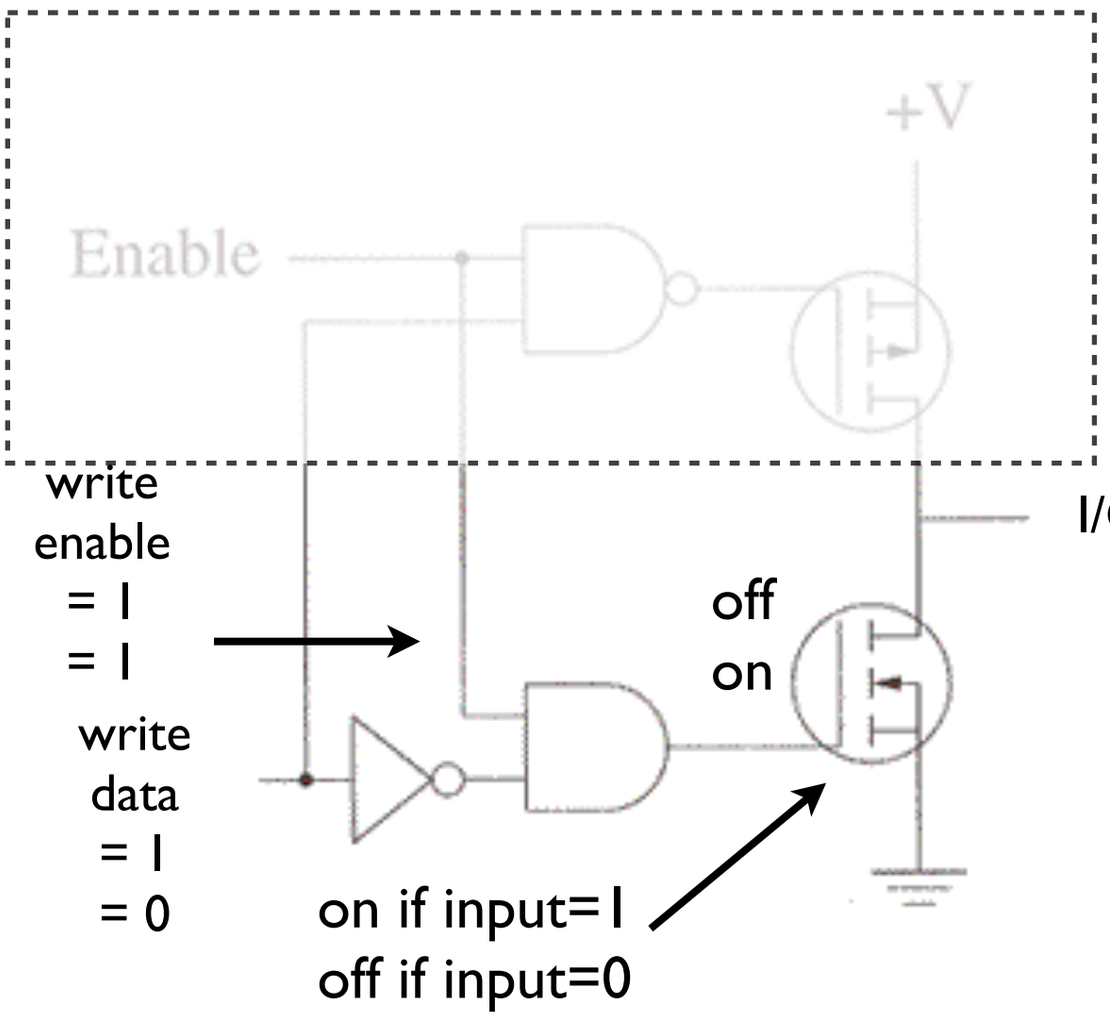


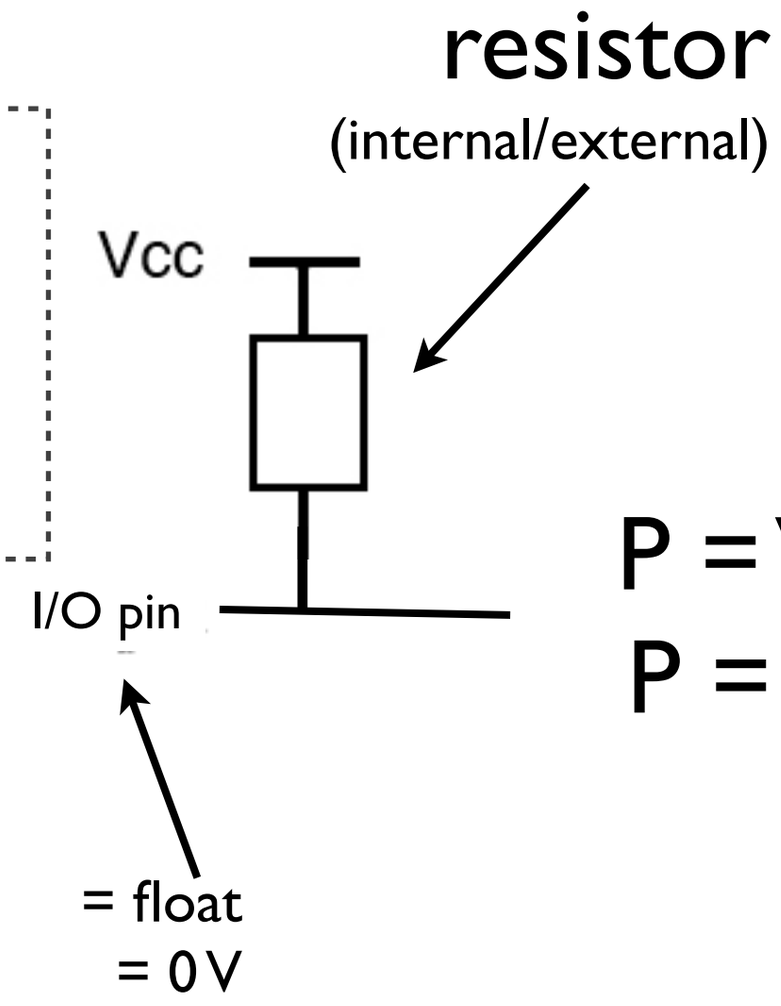
P = float (high Z)

P = 0

output: pull-up

(assumes open drain)





P = Vcc

P = 0 V

if output one: PX.y = Vcc (logic one)

if output zero: PX.y = GND (logic zero)

simulator treats this as high

(essentially pull-up; so I don't set in example code)

defaults:

input is floating

(control disabled: high-impedance [open drain])

output is push-pull

current source

(be careful)

use open drain
to sink

got it?

5. enable pins

set bits for pin(s) on port(s) we want for GPIO

(bit = 1)

```
;enable port b
ldr R1,=PB_DEN_R ;get addr. enable reg
;enable i/o for port
mov R0,#0xFF
str R0,[R1]
;disable i/o for pins zero and seven
;(don't disturb previous settings)
ldr R0,[R1] ;get i/o config for port
and R0,#0x7E ;0x7E = 0b01111110
str R0,[R1]
```

$4000.551C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTB_DEN_R

6. read or write

get/send words from/to port

```
ldr R1,=PB_DATA_R ;addr. for data reg, port b
;assume port b input--get data from it
ldr R0,[R1] ;R0 = PortB
;assume port b output--send data to it
mov R0,#0xABCDEF
str R0,[R1] ;PortB = R0
```

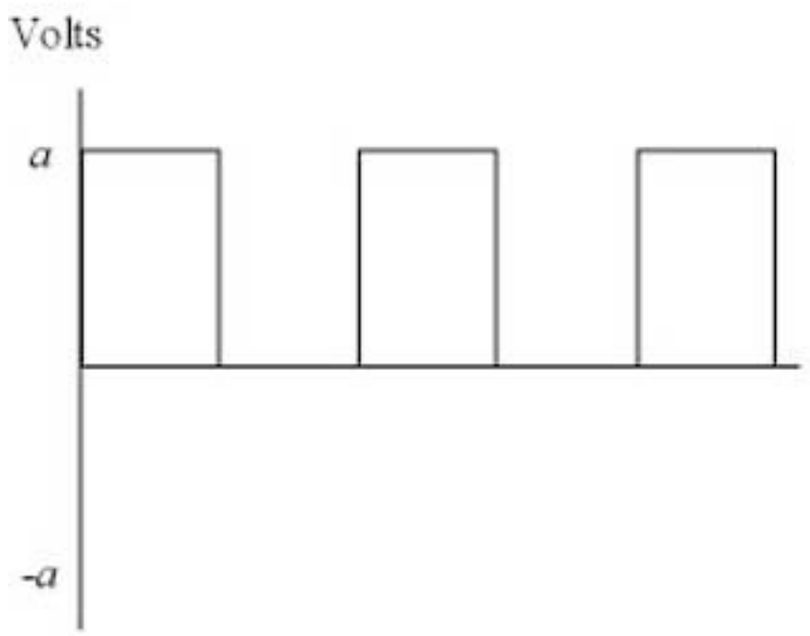
$4000.53FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTB_DATA_R

example: read from one port (PD), write to another (PA)

```
; 3. set port pins as INPUT or OUTPUT
LDR R1,=PD_DIR_R ;PD is input
MOV R0,#0x0
STR R0,[R1]
LDR R1,=PA_DIR_R ;PA is output: only need four bits
MOV R0,#0xF
STR R0,[R1]
; 5. enable ports: only need four bits for each
LDR R1,=PD_EN_R
MOV R0,#0xF
STR R0,[R1]
  LDR R1,=PA_EN_R
  STR R0,[R1]

  ; let's get data from PORT D and output on PORT A
  LDR R1,=PD_DATA_R
  LDR R2,=PA_DATA_R

loop
  LDR R0,[R1] ;get from PD
  STR R0,[R2] ;send to PA
  b loop
```

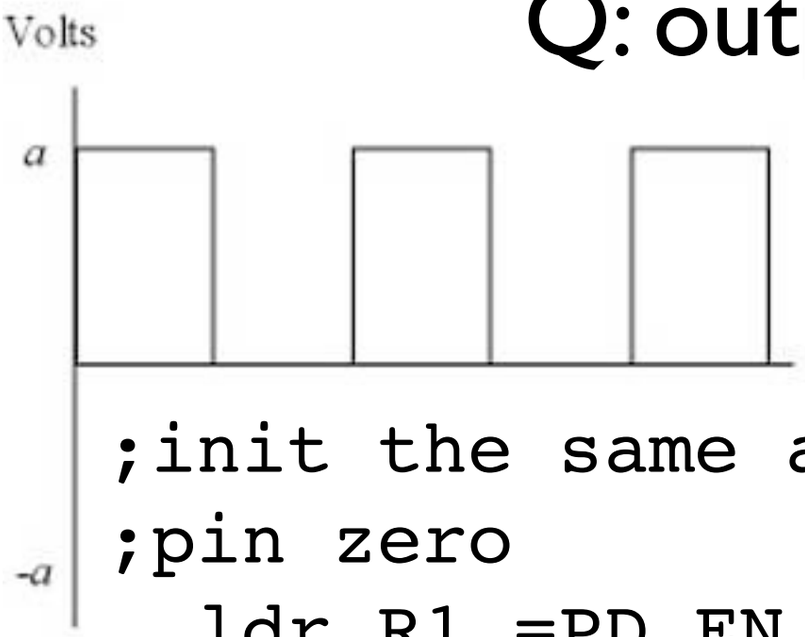
Q: what if port is input and output?

Q: output a square wave on a pin



for output: set pin to 0 (low voltage) or 1 (high voltage)

Q: output a square wave on a pin ⟶ PD.0



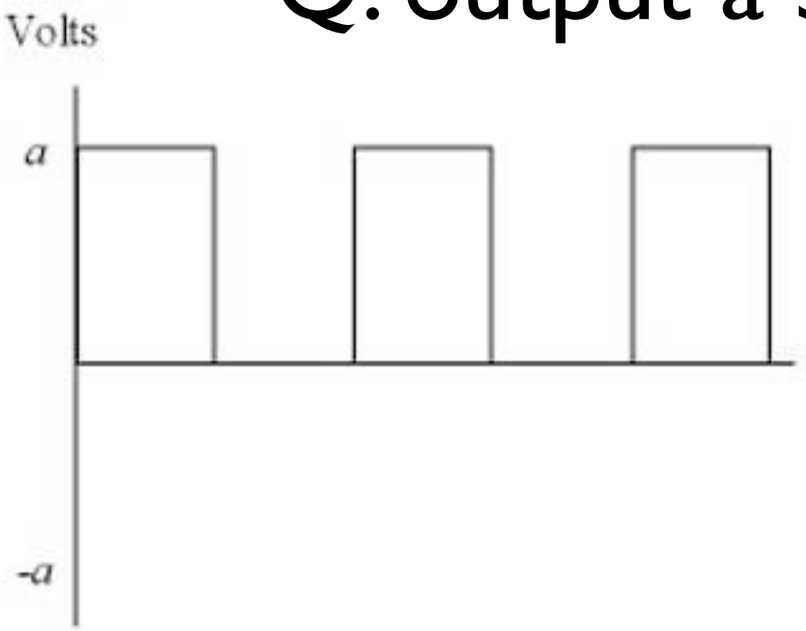

```
;init the same as before, except only enable
;pin zero
  ldr R1,=PD_EN_R
  mov R0,#0x1
  str R0,[R1] ;enable pin zero

  ldr R1,=PD_DATA_R ;port d data here
loop
  mov R0,#0x1
  str R0,[R1]
  bl delay1
  mov R0,#0x0
  str R0,[R1]
  bl delay2
  b loop
```

for 50% duty cycle:

time(delay2) = time(delay1) - time(b loop)

Q: output a square wave on a pin: better way

(50% duty cycle)




```
mov R0,#0x0
loop
   mvn R0,R0
   str R0,[R1]
   bl delay
   b loop
```

complement register: $X = \bar{X}$