Touchscreen Interfacing

ECE 3710

I used to have an open mind but my brains kept falling out.

- Steven Wright

resistive touchscreens

assuming:

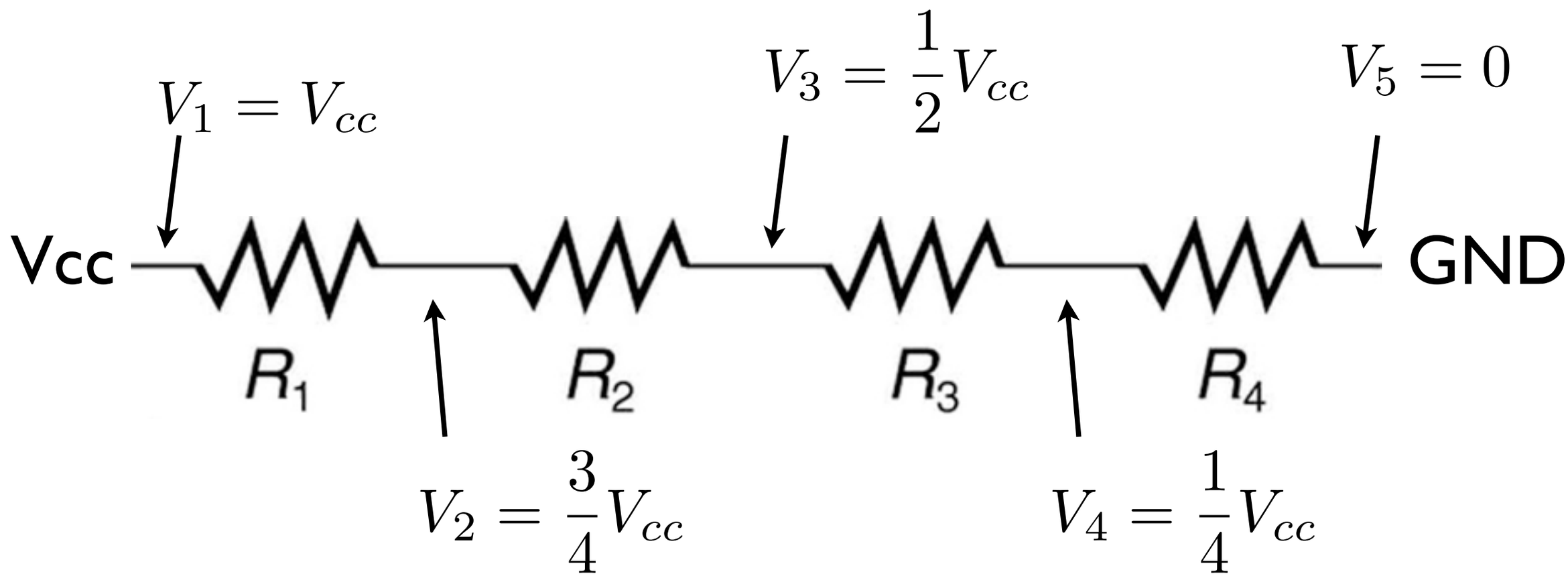
if given V_x:

can we deduce where on resistive chain the measurement is made?

$$R_1 = R_2 = R_3 = R_4$$

resistive touchscreens

assuming:



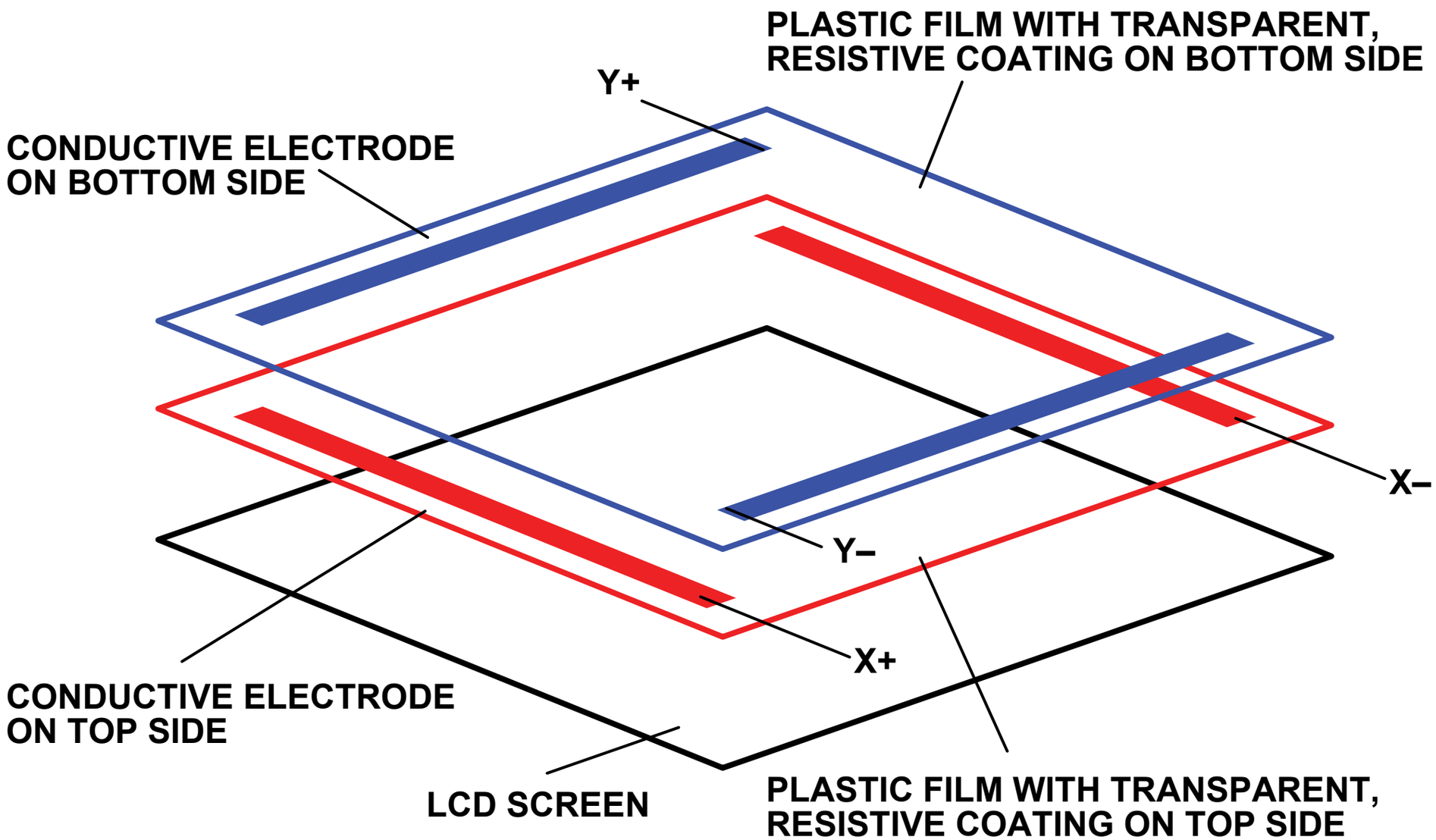
if given V_x:

we can deduce where on resistive chain the measurement is made

$$R_1 = R_2 = R_3 = R_4$$

resistive touchscreens

if we embed resistors
in a screen overlay:

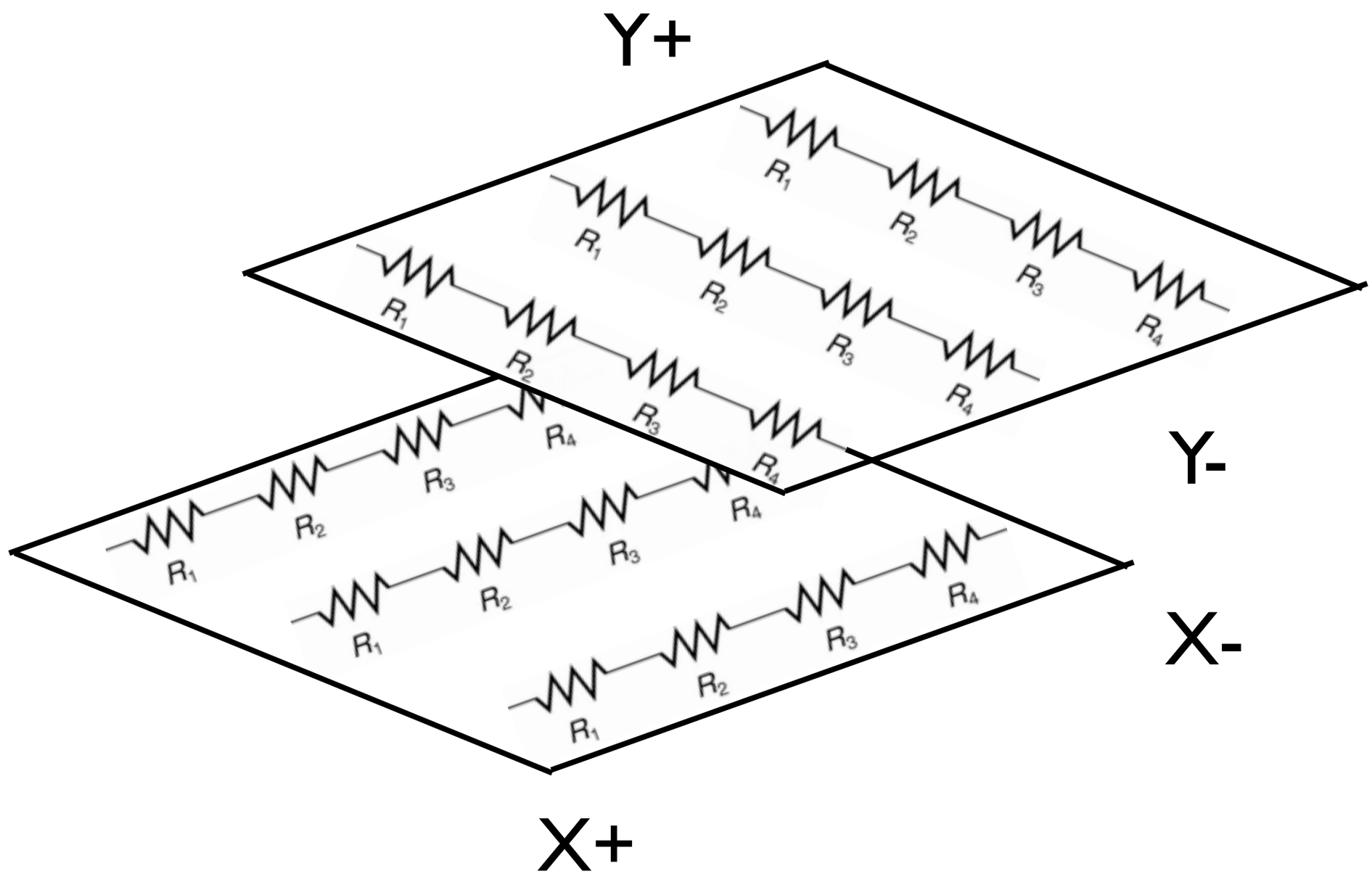


resistive coating: resistance per unit length

resistive touchscreens

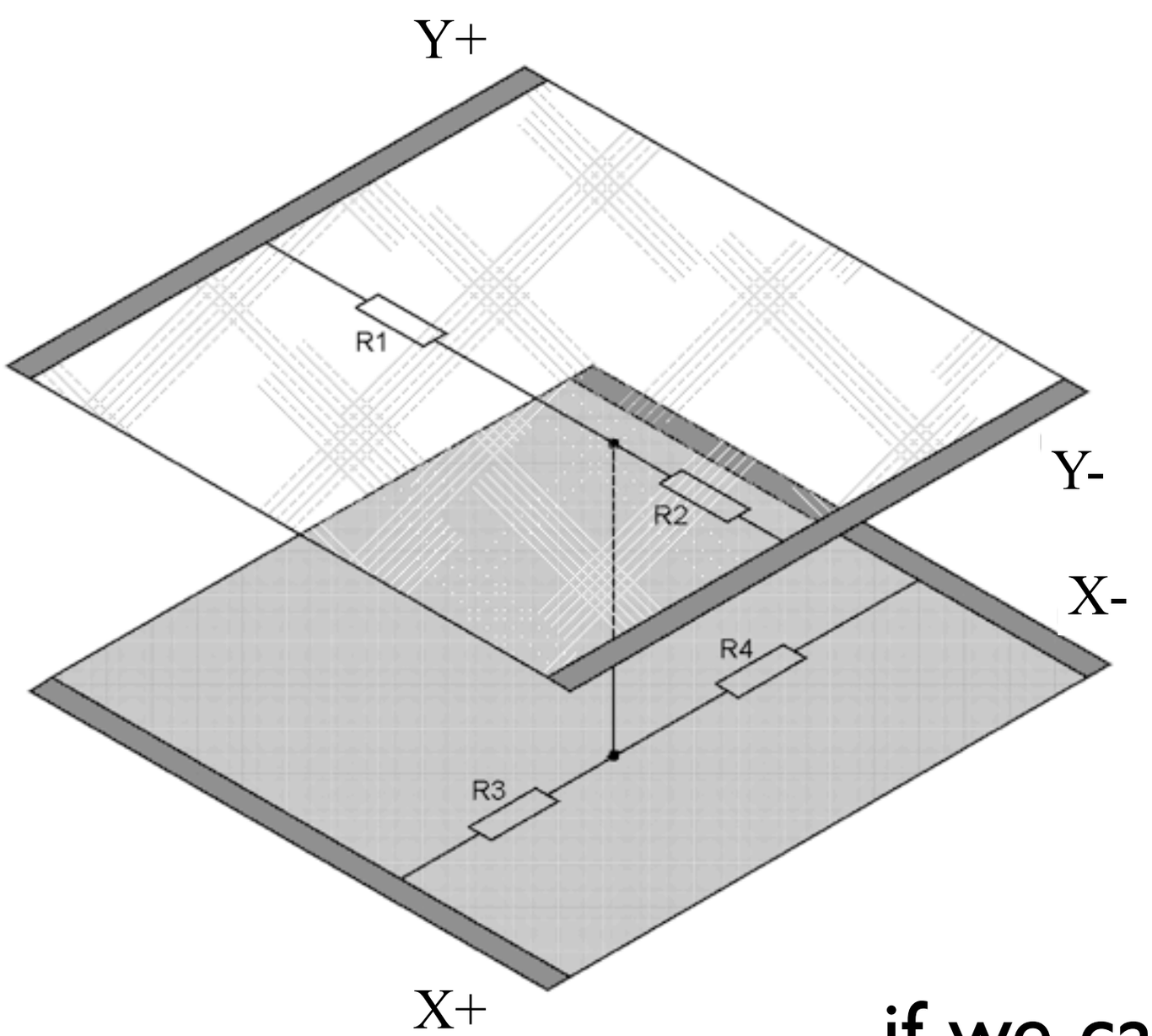
resistive coating:

resistance per unit length



resistive touchscreens

when the screen is touched:



given:

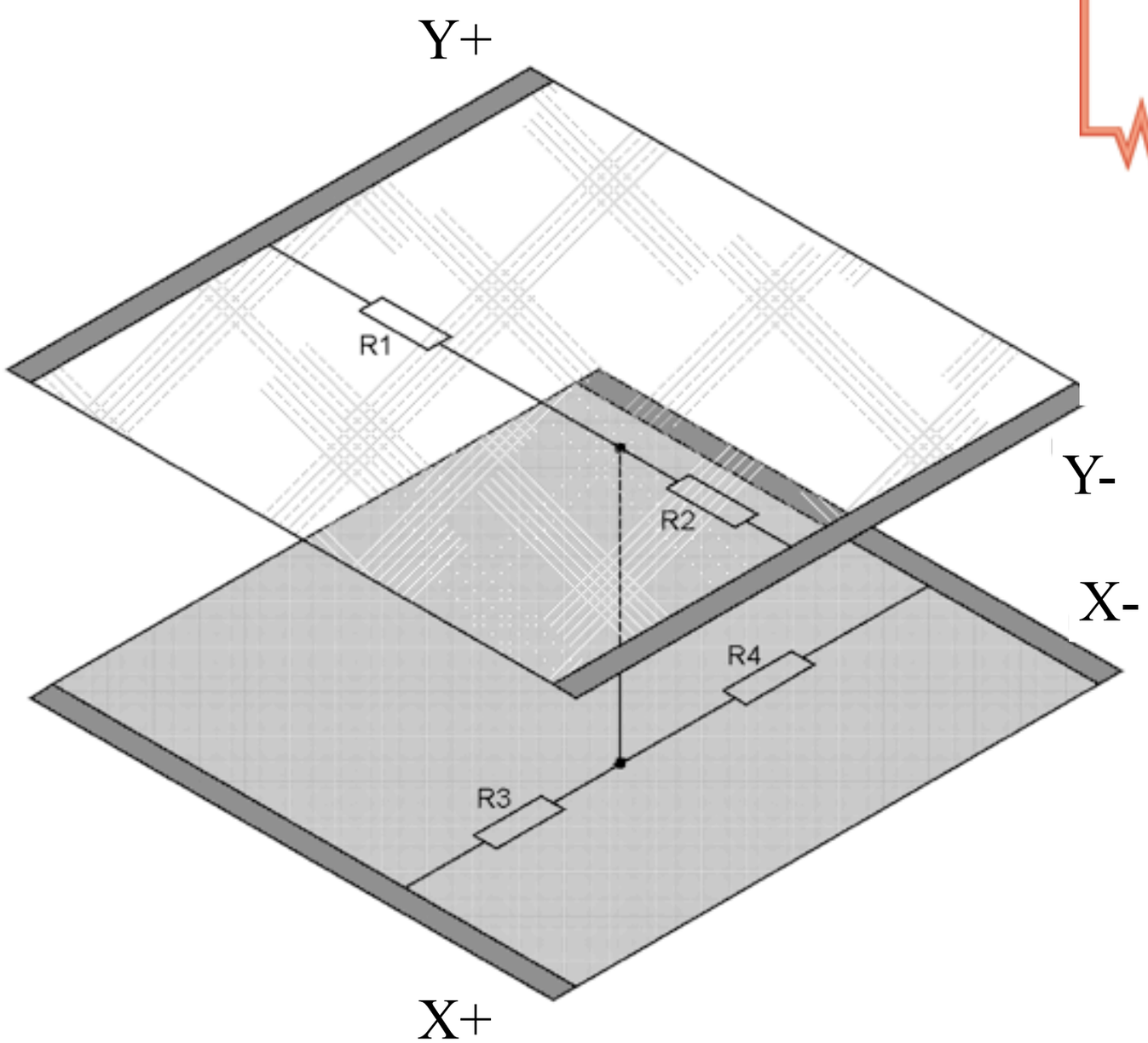
1. Y+ = Vcc & Y- = GND
2. V = voltage at point of contact
3. smaller V means closer to Y- edge; larger V means closer to Y+ edge

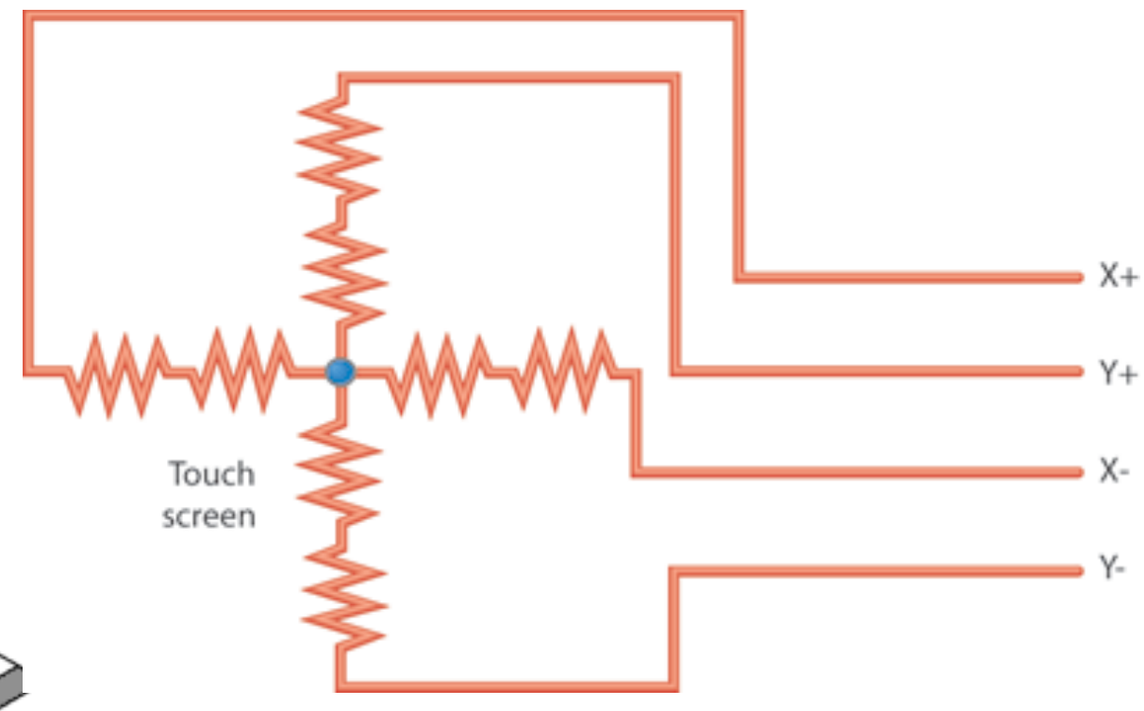
↓

can determine position if we can measure voltage at point

resistive touchscreens

when the screen
is touched:

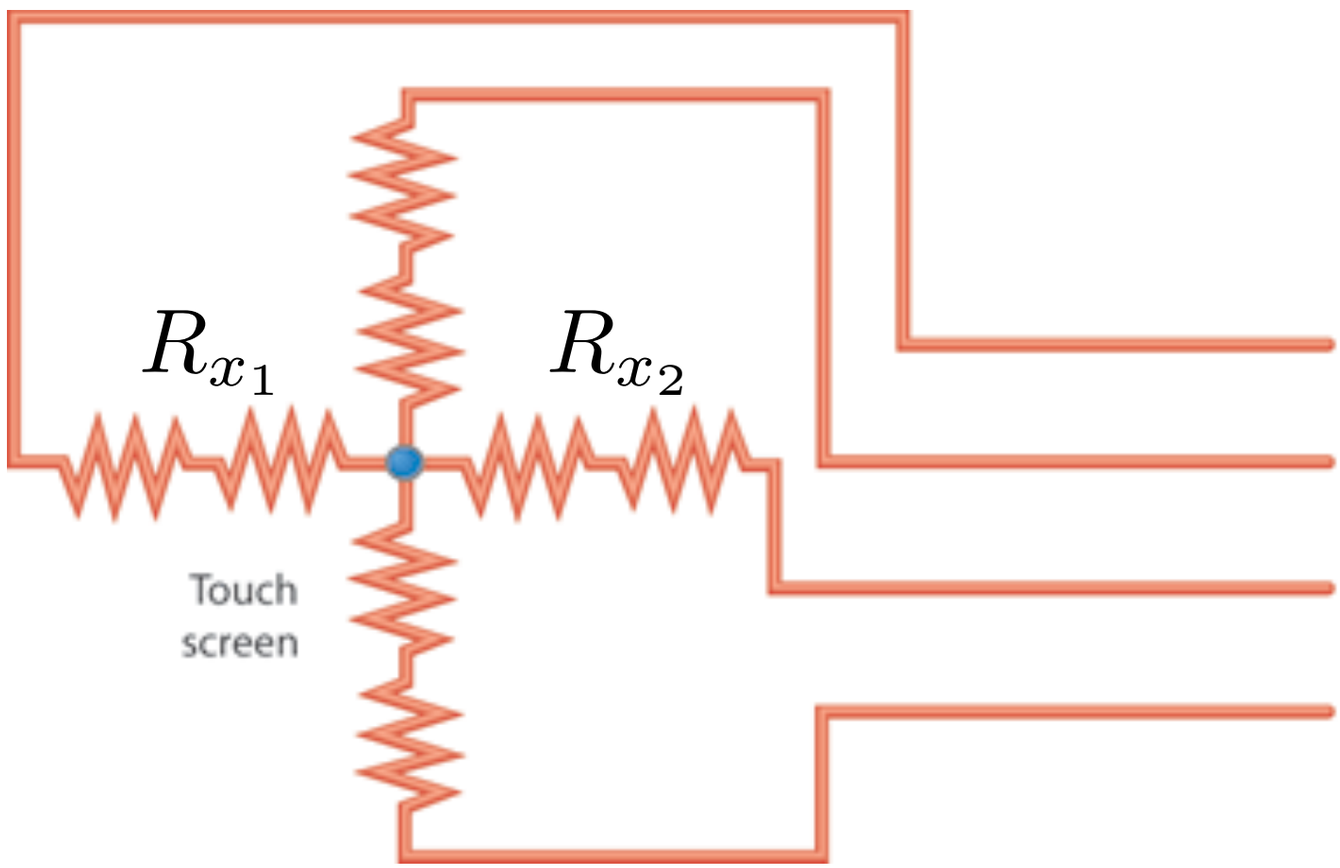




equivalent
circuits

resistive touchscreens

to determine x coordinate:



X+ = Vcc

Y+ = $V_{cc}\left(\frac{R_{x_2}}{R_{x_1}+R_{x_2}}\right)$

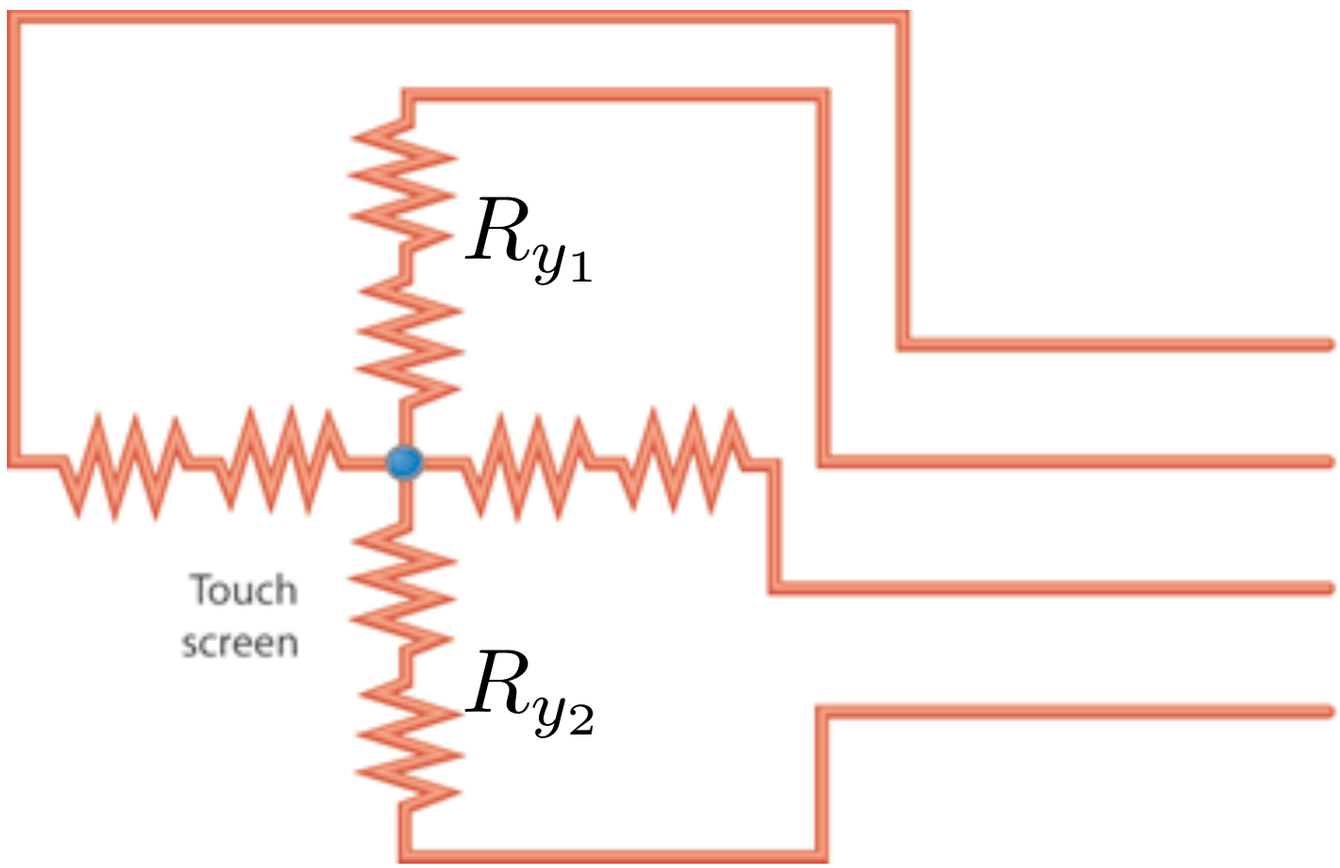
X- = GND

Y- = floating

need x,y coords:
two measurements

resistive touchscreens

to determine y coordinate:



X+ = $V_{cc}\left(\frac{R_{y_2}}{R_{y_1}+R_{y_2}}\right)$

Y+ = Vcc

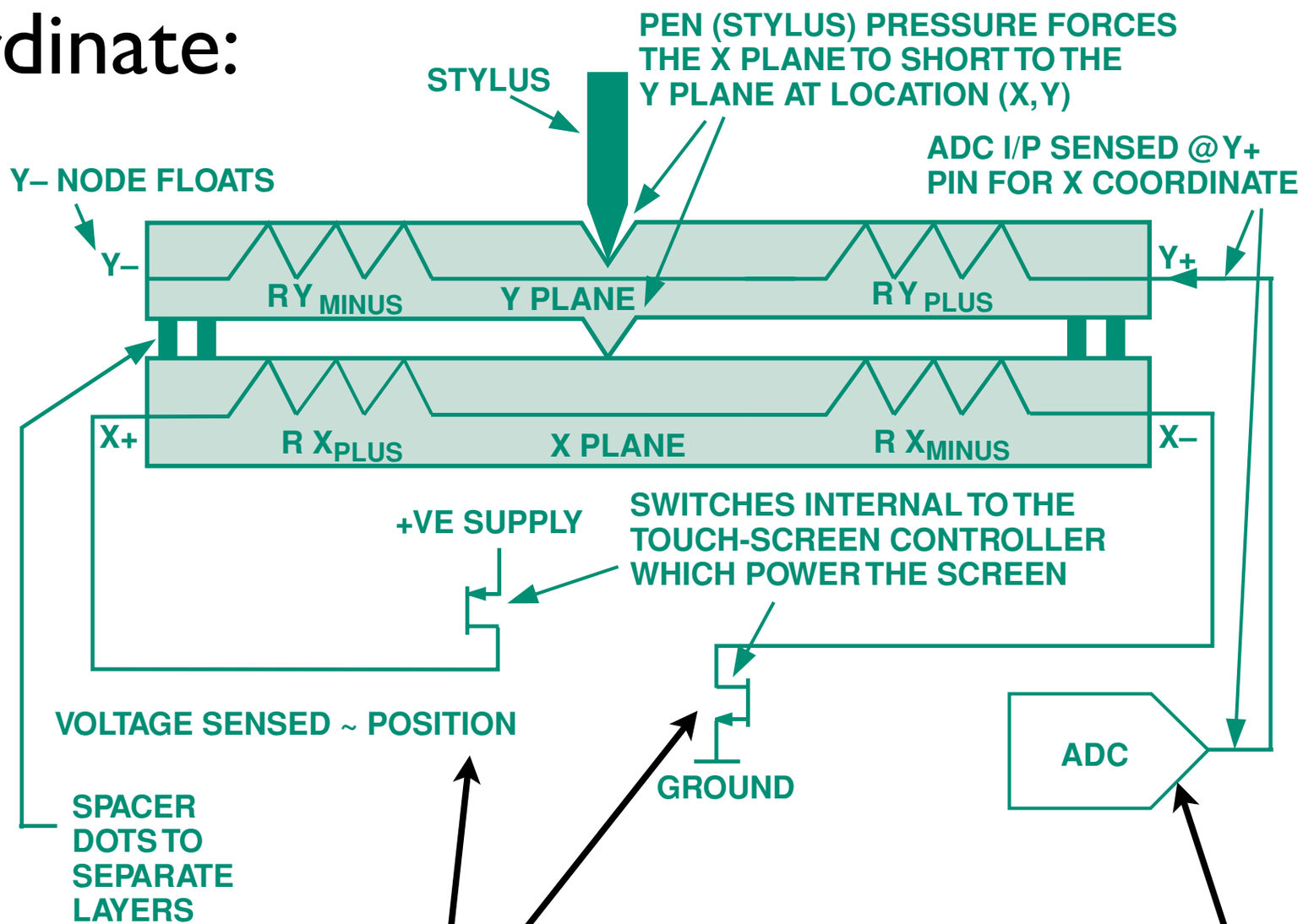
X- = floating

Y- = GND

need x,y coords:
two measurements

resistive touchscreens

to determine x coordinate:



touchscreen controller controls these switches

measures voltage; converts to proportional binary number

you, on theory operation:

Q: how do we actually use this?

generic touchscreen controller

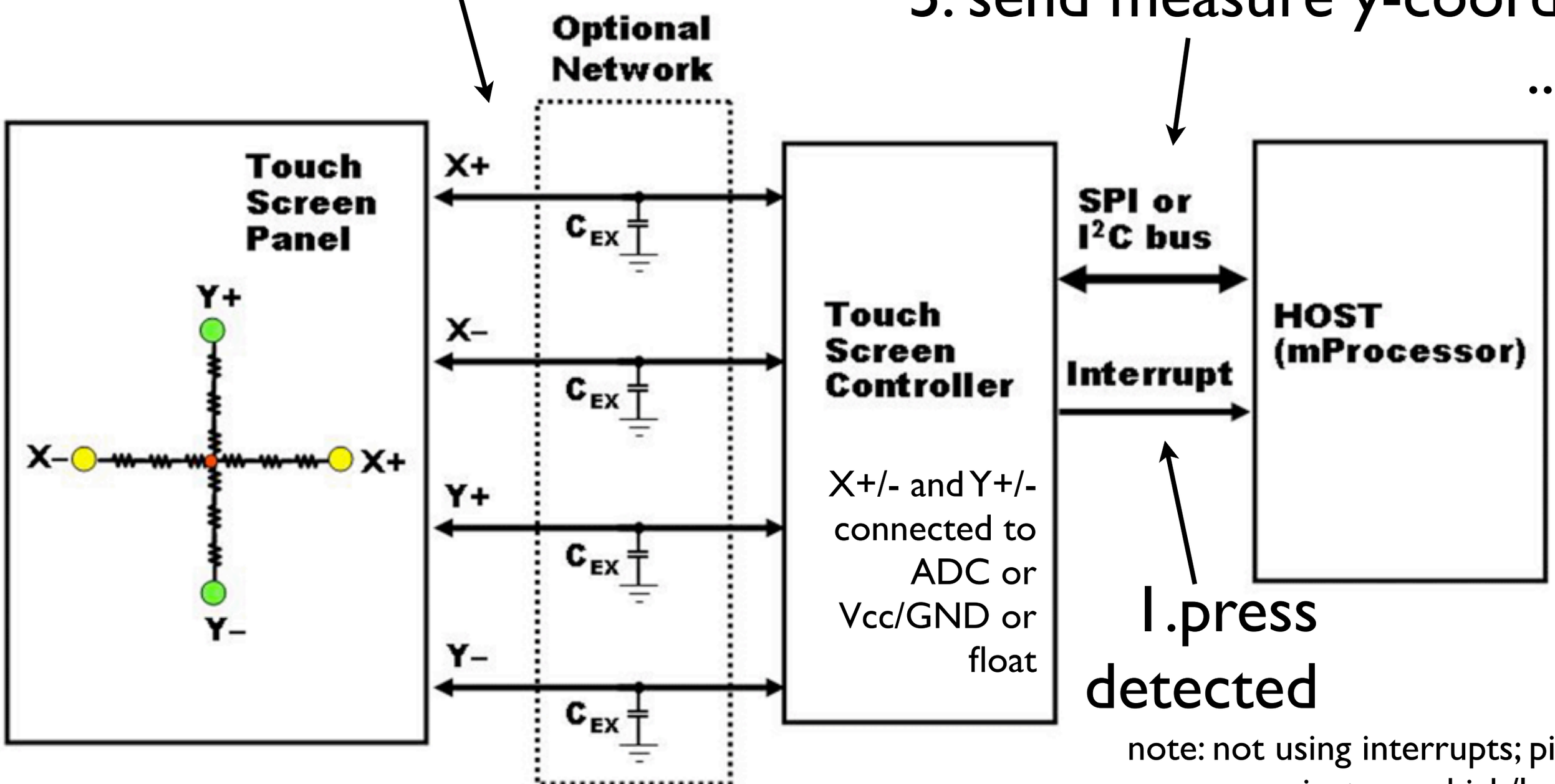
3. connect Y+ to ADC; Y- is floating; X+ to VCC; X- to GND

2.send measure x-coord command

4. receive x-coord

5. send measure y-coord

...



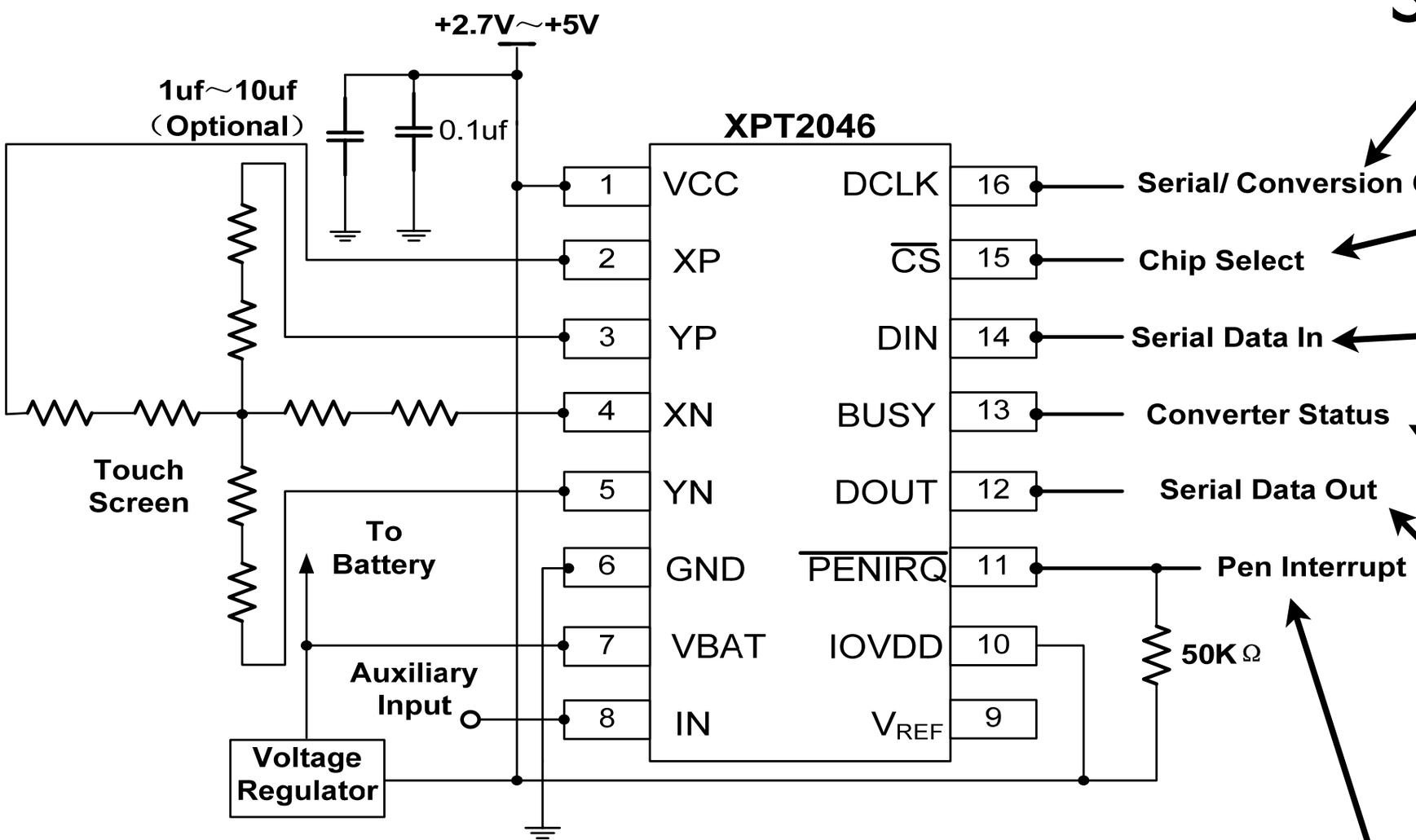
X+/- and Y+/- connected to ADC or Vcc/GND or float

1.press detected

note: not using interrupts; pin just goes high/low

XPT2046 touchscreen

interface via SPI



SCLK

enable I/O

data latched on rising edge

high during conversion

data shifted on falling edge

low while touch is detected

note: refer to schematic to find pins on LCD board

p13 & p9

XPT2046 touchscreen

one command

pressure
temperature

very simple:

1. tx: get x,y, or z/t (control byte)
2. rx: requested coord

note: the control byte also contains configuration options

XPT2046 touchscreen: control byte

what is sent to touchscreen controller

coordinate:

x: 101

y: 001

conversion resolution:

1: 8-bits

0:12-bits

BIT7(MSB)	BIT 6	BIT 5	BIT 4	BIT 3	BIT2	BIT 1	BIT 0(LSB)
S	A2	A1	A0	MODE	SER/$\overline{\text{DFR}}$	PD1	PD0

always 1

conversion reference type:

1: single

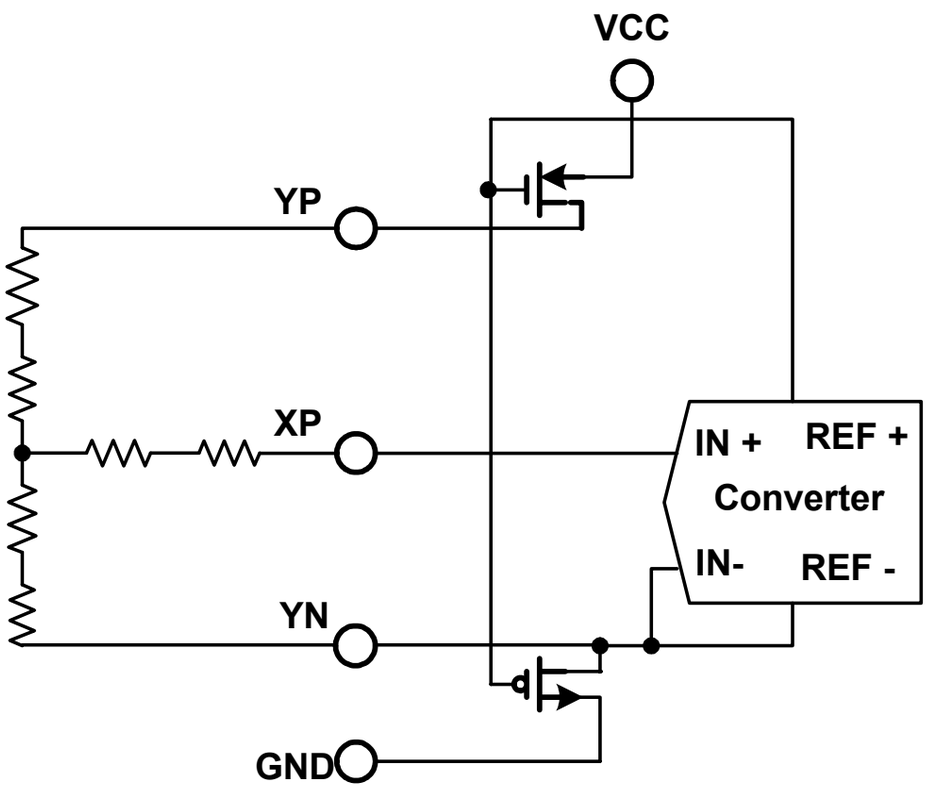
0:differential

power/interrupt pin:

00: unit off between conversions but / PENIRQ active

XPT2046 touchscreen: A[2:0]

(get y)



X+ input to ADC

=> measure y coord

of course,
depends on orientation

A2	A1	A0	+REF	−REF	YN	XP	YP	Y-POSITION	X-POSITION	Z_1-POSITION	Z_2-POSITION	DRIVERS
0	0	1	YP	YN		+IN		M				YP, YN
0	1	1	YP	XN		+IN				M		YP,
1	0	0	YP	XN	+IN						M	YP,
1	0	1	XP	XN			+IN		M			XP,

XPT2046 touchscreen: control byte

CB[6:4] depend on if you want x or y

recommended configuration (CB[3:0]):

1. differential reference
2. 12-bit conversion
3. power-down between conversions

recommended functions:

1. `getX()`
2. `getY()`

1. issue appropriate control byte
2. get response

this is the tricky bit → need routines for SPI TX/RX

XPT2046 touchscreen: SPI

(12 bit)

recommended
config

24 clock cycles to get single coord:

1. TX control byte (clocks 1--8)
2. RX coord (clocks 9--24)

coord: [0C[11:0]000]

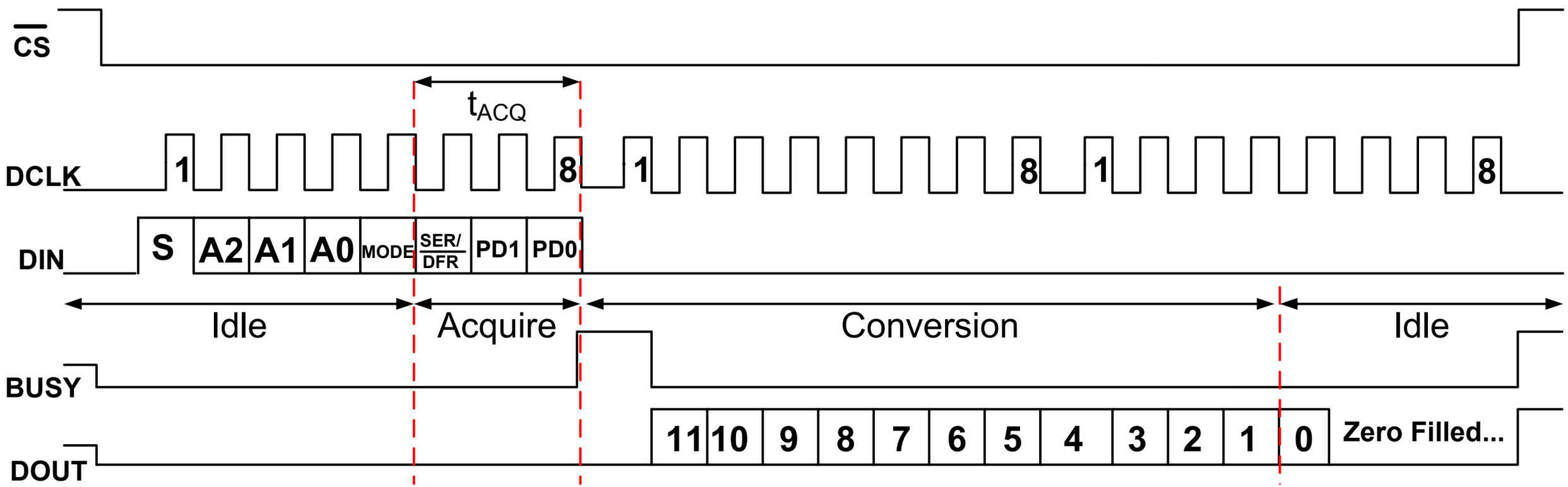
preceded by 0

followed by 000

p21 & fig 12, p23

XPT2046 touchscreen: SPI

note: not all of these pins are connected on your board



touchscreen RX on rising ⟶ uC change data on falling

touchscreen change data on falling ⟶ uC RX on rising

fig 12, p23

response?

to receive coordinate

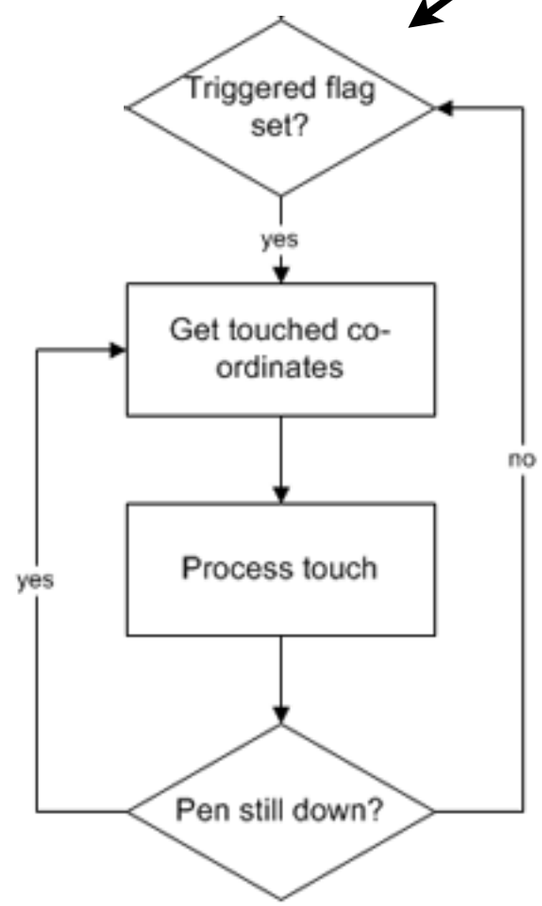
use RX queue in SPI module:

1. init conversion
2. wait
3. read DR
   (A11--1)
4. read DR
   (A0)
5. concatenate

XPT2046 touchscreen: approach

polling
/PENIRQ

basic approach:
(triggered flag is /PENIRQ)



1. don't update until screen is released

2. data is noisy: average or take median of coords while pressed

LCD Adapter

ECE 3710

better adapter would
save 2 more

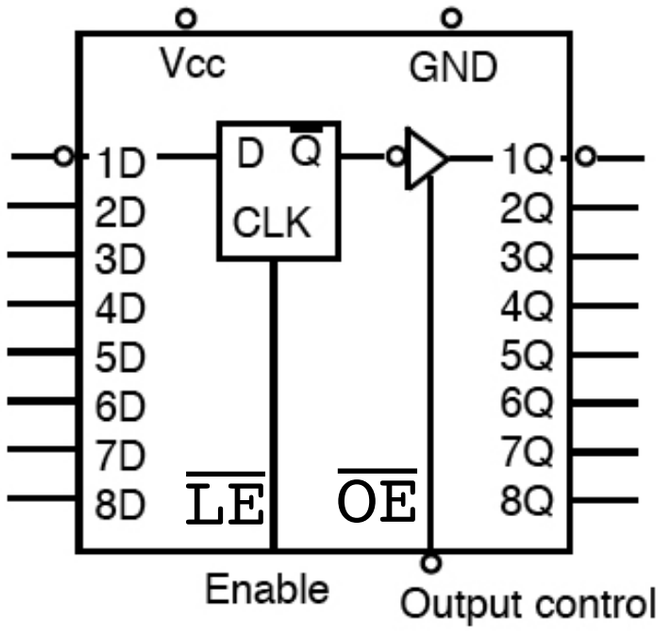
LCD adapter: 16 to 11 pins

to save
pins

more components

ah, more complexity:

latch



Funtion Table

Output control	Enable $\overline{LE}$	Enable D	Output
L	H	H	H
L	H	L	L
L	L	X	Q0
H	X	X	Z

if $\overline{OE}$=0

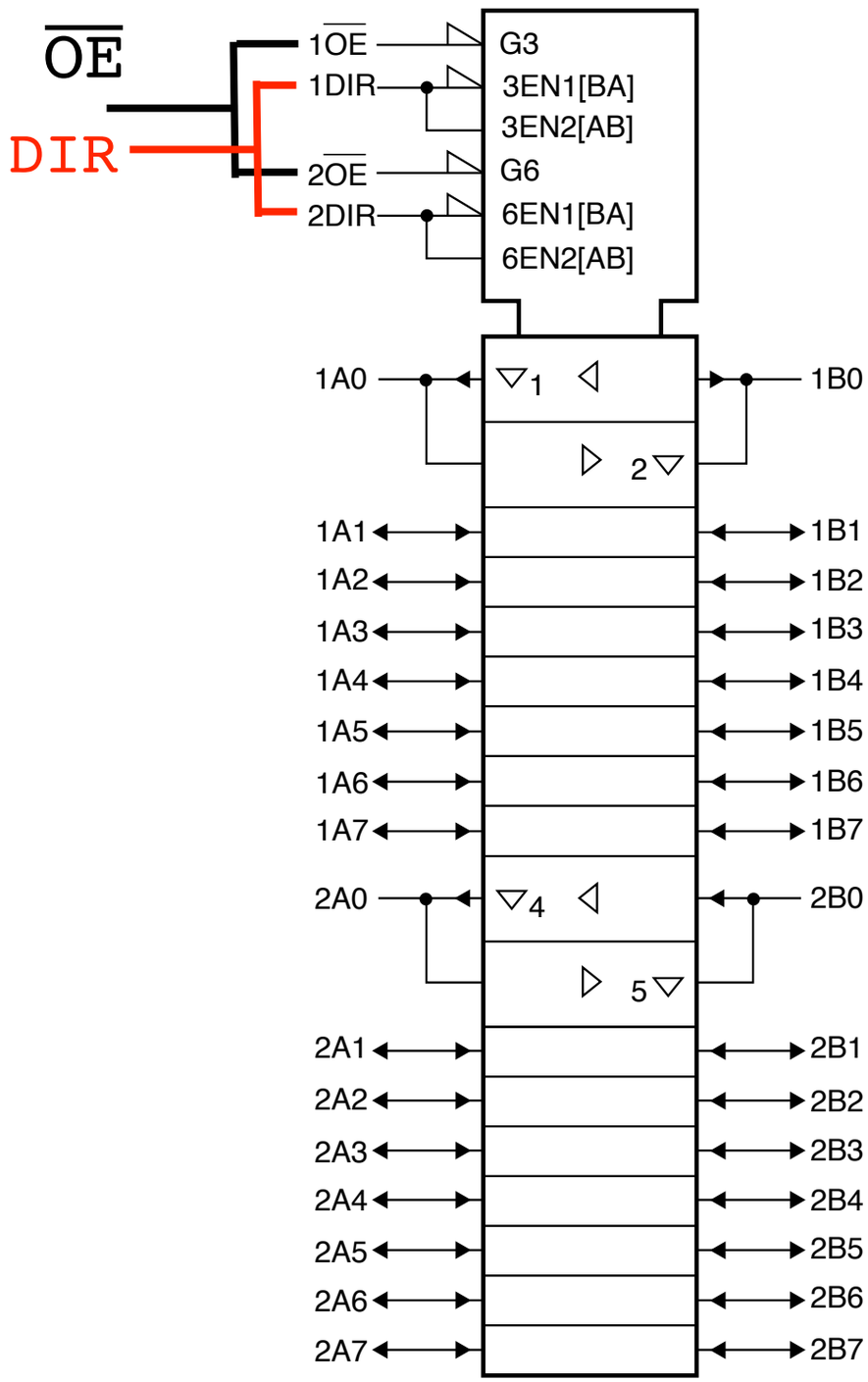
when $\overline{LE}$=1 Output=D

when $\overline{LE}$=0 Output=Q

stored value

transceiver

(16-bit mode)



if $\overline{OE}$=0

when DIR=1 A=>B

when DIR=0 A<=B

$\overline{LE}$

data goes from A to B

data goes from B to A

$\overline{LE}$

to write data: first step

(lower byte)

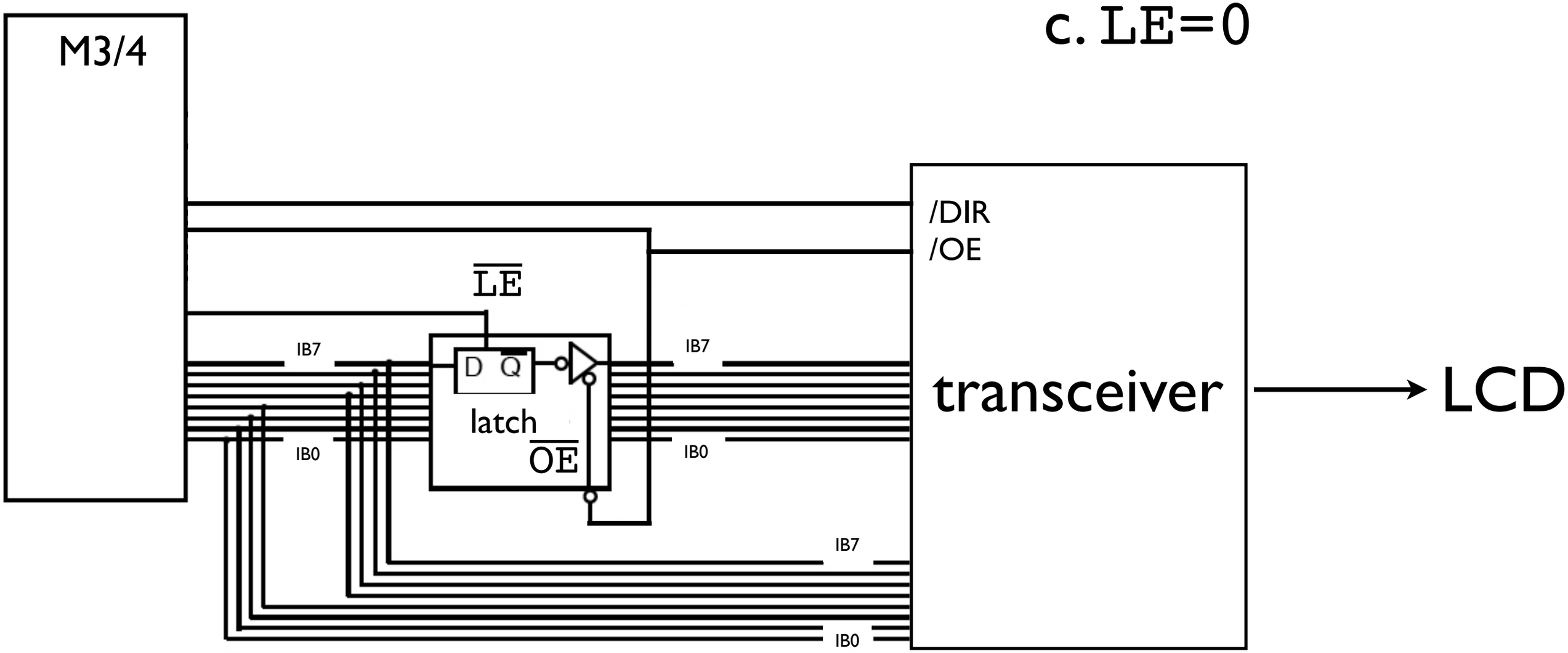
uC does:

1. outputs lower byte: D[7:0]

a. `LE=1`

b. `D[7:0]=IB[7:0]`

c. `LE=0`



to write data: second step

(upper byte)

uC does:

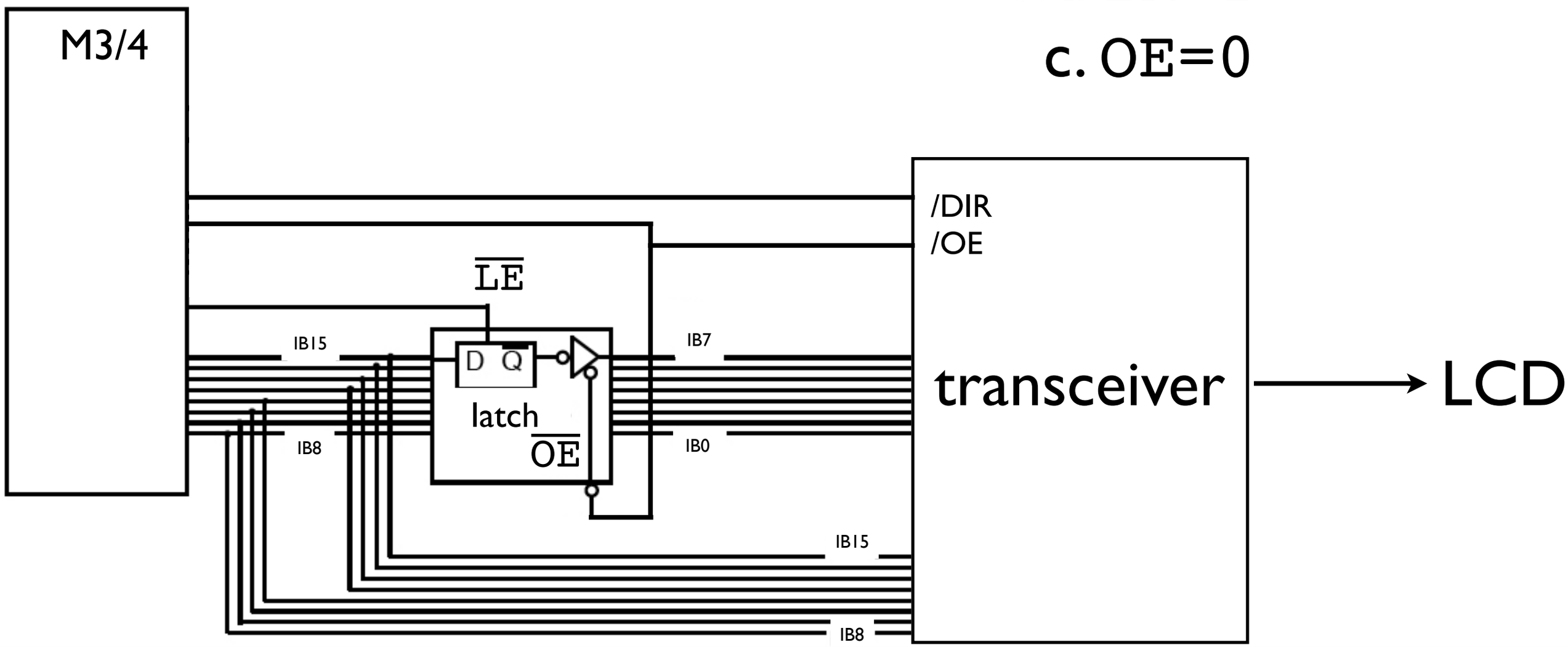
1. outputs upper byte: D[7:0]

b. `D[7:0]=IB[15:8]`

c. `LE=0`

c. `DIR=1`

c. `OE=0`



for LCD adapter pins

note:

1. `OE = DEN`
2. `LE = DLE`
3. `DIR = DDIR`