RTC II

ECE 3710

When everything is coming your way, you're in the wrong lane.

- Steven Wright

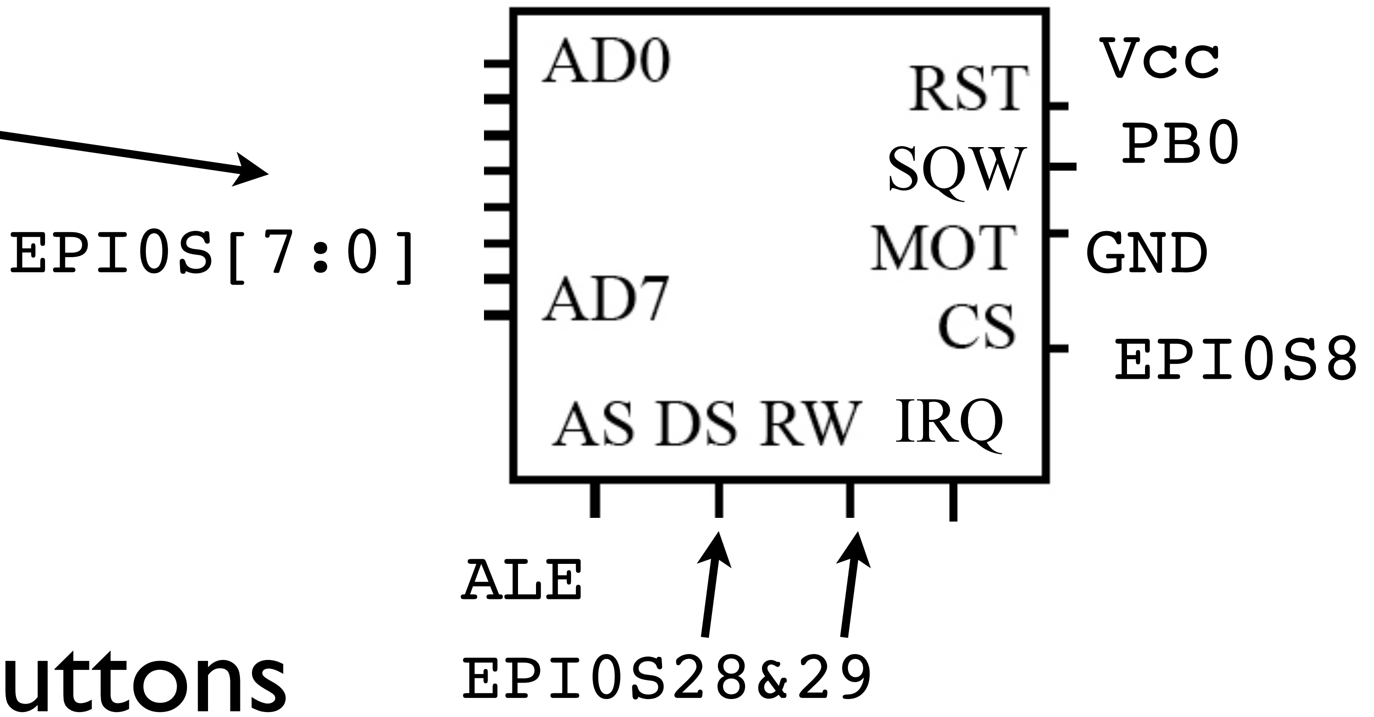
Q: stop watch (minutes+seconds)

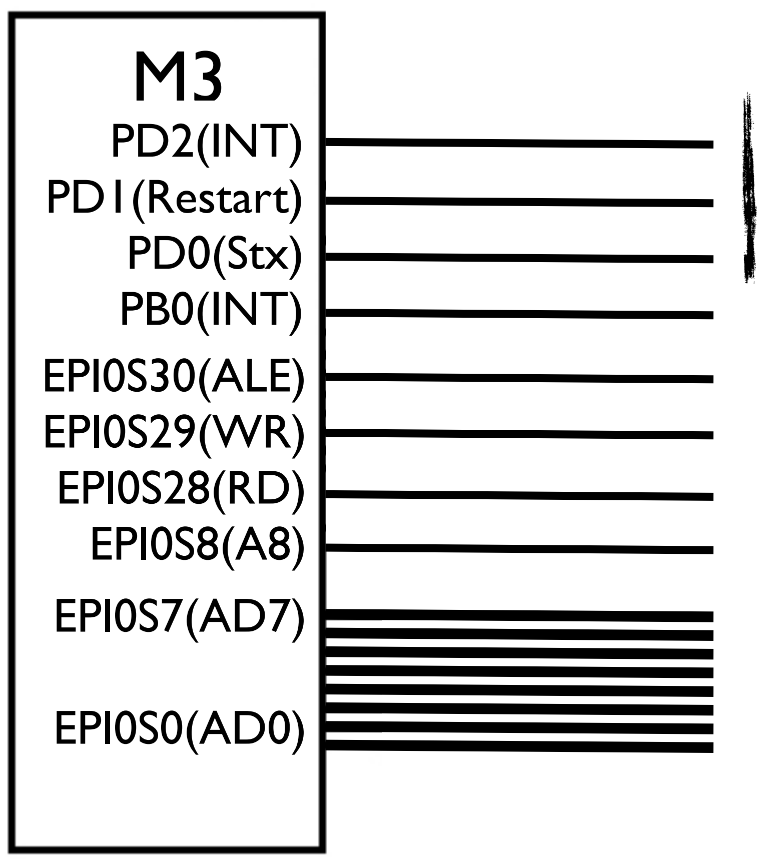
1. stop/start and reset push buttons
   a. use interrupt
   b. debouncing ← 30 ms
2. display accurate to half second
   a. use interrupt

requirements

stopwatch: shared, multiplexed bus

we do our own latching





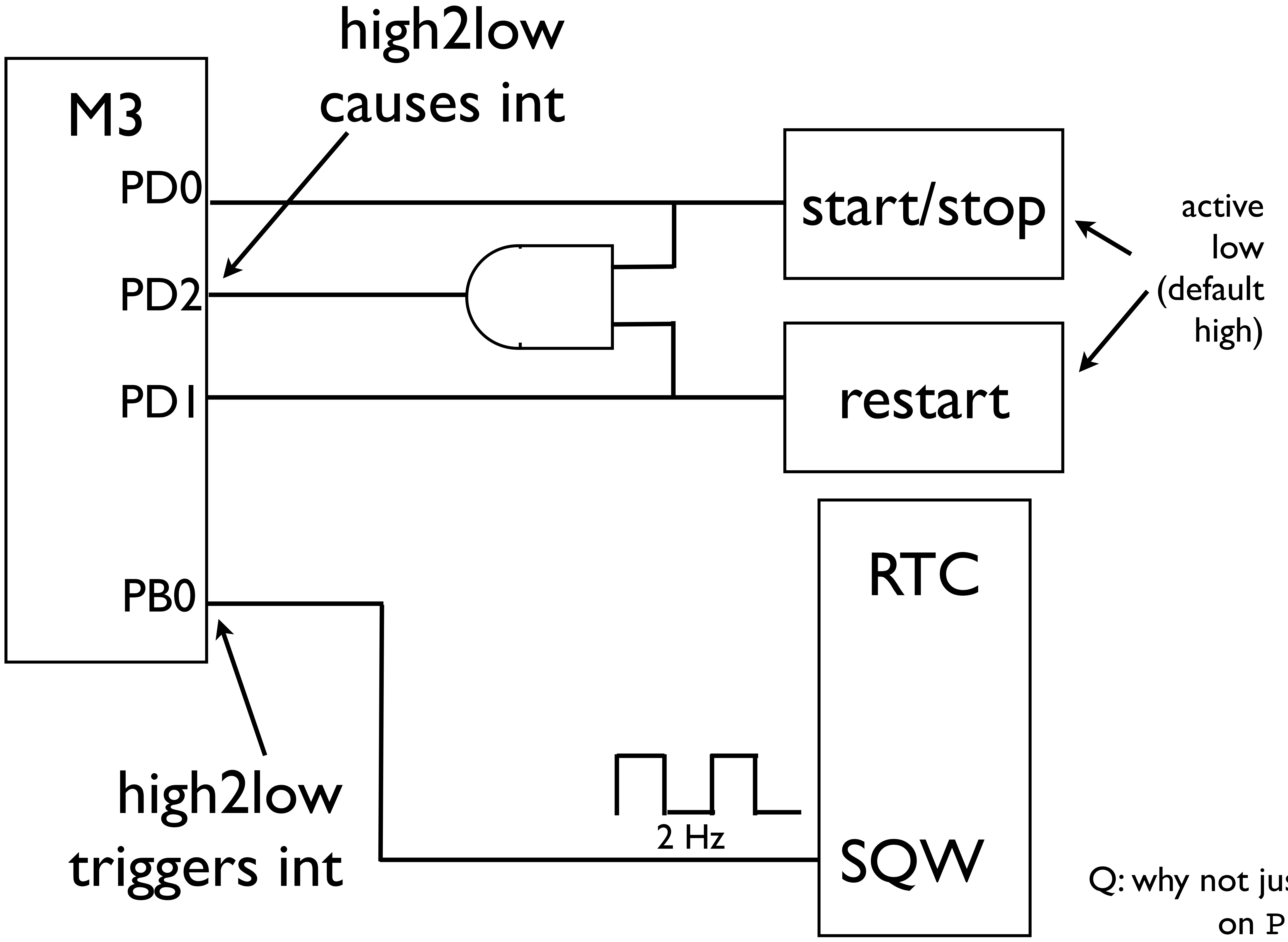
buttons

want:

1. MM
2. ext. int.

not due to alarm, though

stopwatch: connections



Q: why not just level on `PD0/1`?

rtc: stop watch

```
unsigned char SEC __attribute__((at(0xA0000100)));
unsigned char MIN __attribute__((at(0xA0000102)));
unsigned char HOUR __attribute__((at(0xA0000104)));
...
unsigned char D __attribute__((at(0xA000010D)));
unsigned char *RTC = (unsigned char *) 0xA0000100;
unsigned char *RTC_MEM = (unsigned char *) 0xA000010E;

void main()
{
  /* port & interrupt setup */
  PBInit(); //PB0 as input; high2low interrupts
  PDInit(); //PB0--2 as inputs; PB2 high2low interrupts

  /* SysTick setup */
  SysTickInit(); //expire after 30 ms; interrupts; not running

  /* rtc setup */
  A = 0x2F; //0b00101111: turn rtc on and set sqr wave output to 2 Hz (500 ms
  B = 0x8; //0b00001000: enable square wave output

  setDisplay(MIN,SEC); //set display to 00:00; assume init. to zero
  while(1);  //wait for button press
}
```

rtc: stop watch

```
// m3 core peripherals base
unsigned char *M3CP = (unsigned char *) 0xE000E000;
// base addr for port b
unsigned char *PB = (unsigned char *) 0x40005000;
// base addr for port d
unsigned char *PD = (unsigned char *) 0x40007000;

/* update display */
void GPIOPortB_Handler(void)
{
  // ack interrupt
  PB[0x41C] = 1;

  // call a function to update the display connect to uC
  setDisplay(MIN,SEC);
}

/* button press */
void GPIOPortD_Handler(void);
{
  //to avoid interrupts from bounces disable PD2 interrupts;
  PD[0x410] = PD[0x410] & 0xFB; //0xFB=0b11111011

  // need time for switches to close to figure out which one caused int
  //  get SysTick going
  M3CP[0x10] = 0x7; //0b111: start counting w/interrupts, CURRENT=RELOAD
}
```

rtc: stop watch

notice, don’t change register B

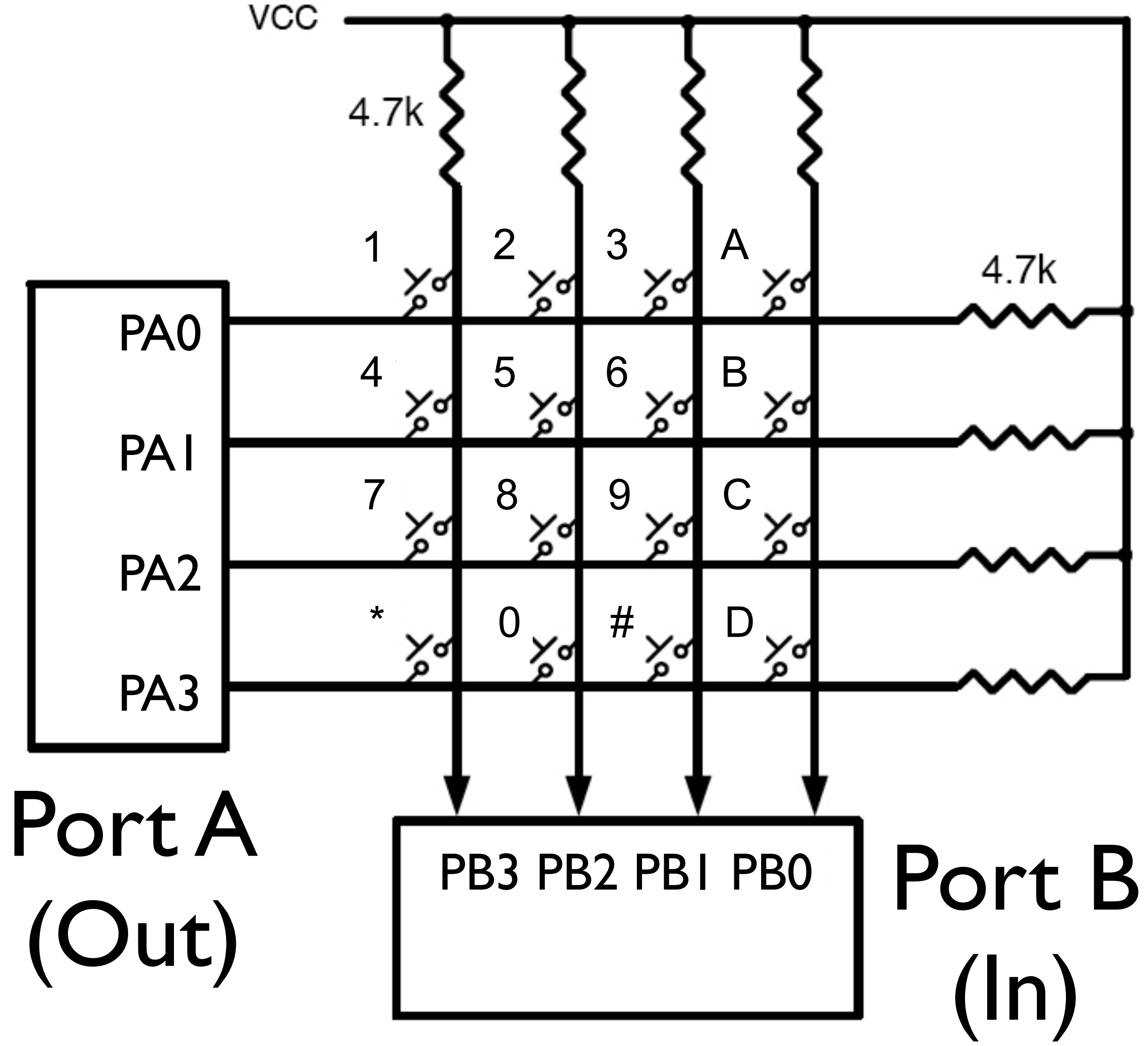
```
// bit addr for port d, pin one (reset)
unsigned char RESET __attribute__((at(0x420E7F84)));
unsigned char CNT = 0; //signifies whether stop watch is counting(1) or not(0)

void SysTick_Handler(void)
{
  M3CP[0x10] = 0x0; //stop SysTick counting

 /* which button was pressed, act */
  if(RESET == 0)
    {
      B = B | 0x80; //0b10000000: no updates
      MIN = 0;
      SEC = 0;
      B = B & 0x7F; //0b01111111: enable updates (start counting)
    }
  else //start/stop button pressed: if counting stop, if not start
    {
      if(CNT==1) //we need to stop rtc
        {
          B = B | 0x80; //0b10000000: no updates
          CNT=0; //next time button pressed we’ll start counting
        }
      else  //we need to start rtc
        {
          B = B & 0x7F; //0b01111111: enable updates
          CNT = 1; //next time button pressed we’ll stop counting
        }
    }

  // reenable interrupts for buttons: get ready for next button press
  PD[0x410] = PD[0x410] | 0x4; //0x4=0b100
}
```

Keypads & Seven Segment Displays

ECE 3710

keypads

an engineer's keypad?



1. keys are switches
2. to determine key: need row and column

odd keypad...

Steve Wozniak's *Blue Box*

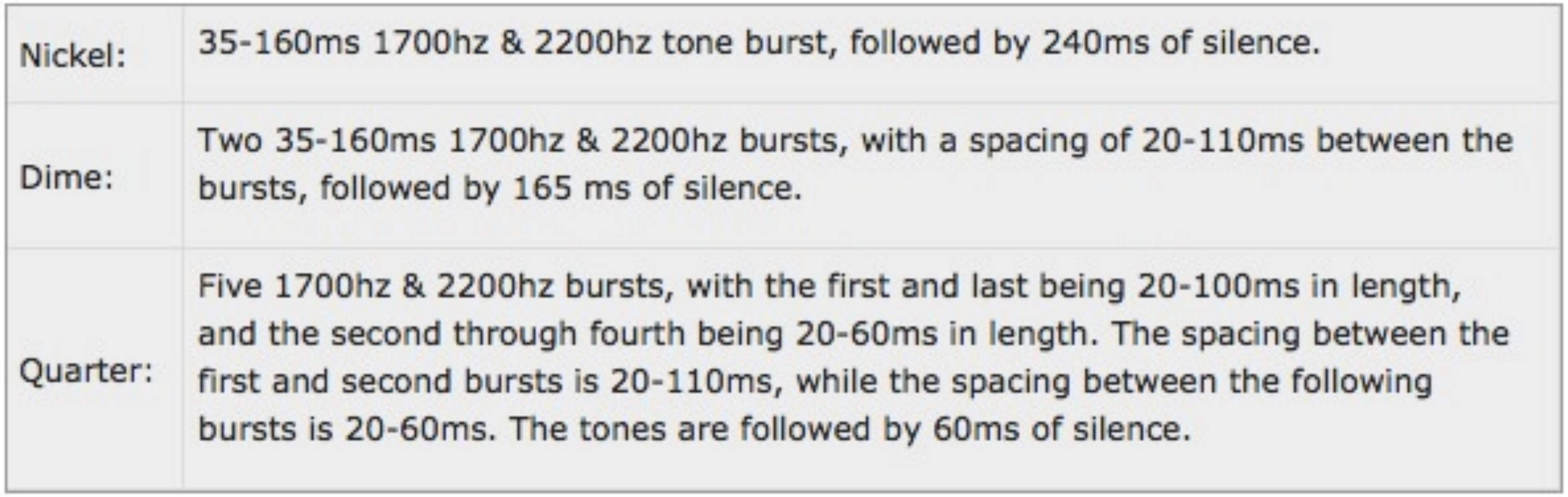
phone hacking
(phreaking):
need extra keys for
special functions

note: stopped working
before my time

back when we still used pay phones...

how the phone knew you'd deposited money

(Automated Coin Toll System)

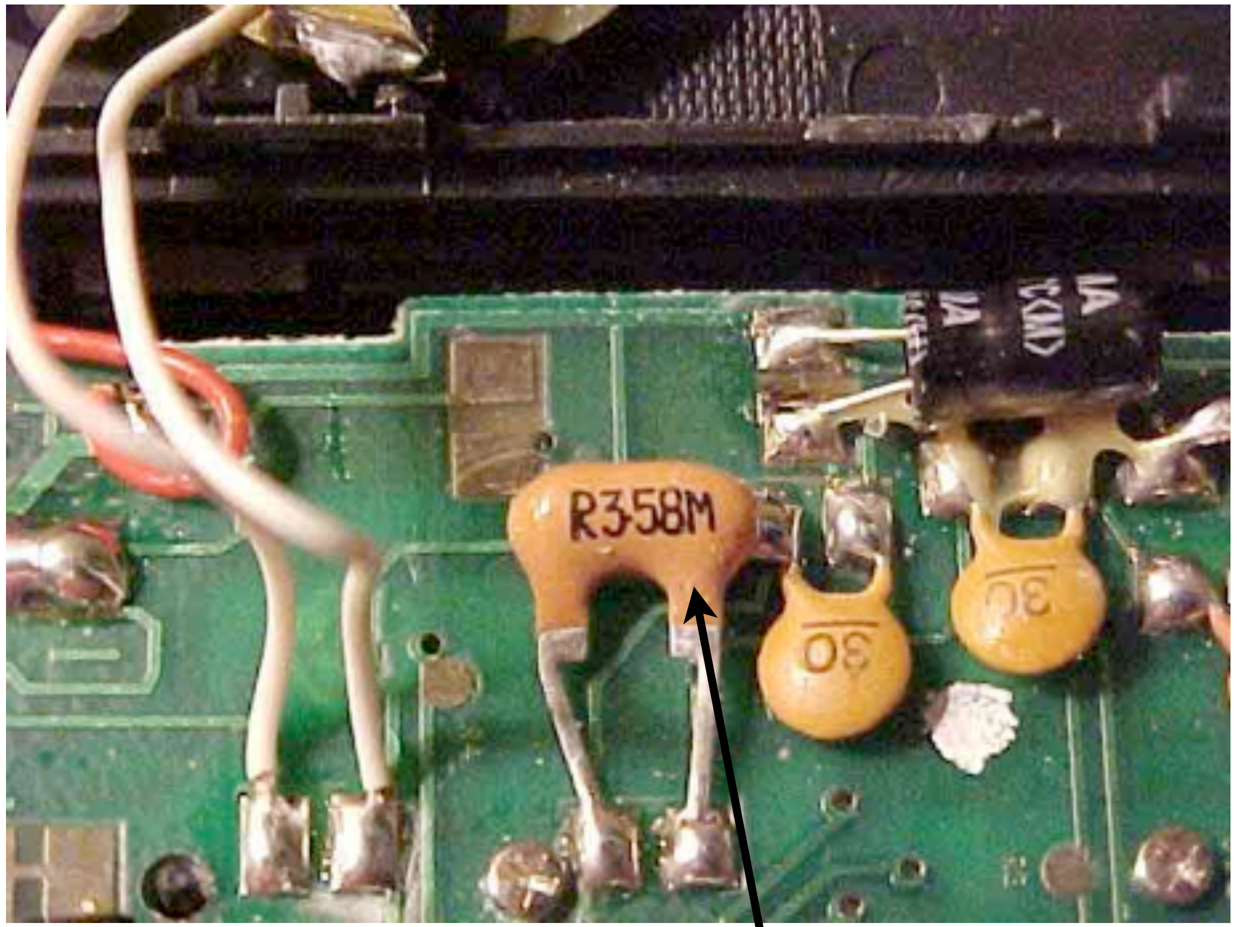


Nickel:	35-160ms 1700hz & 2200hz tone burst, followed by 240ms of silence.
Dime:	Two 35-160ms 1700hz & 2200hz bursts, with a spacing of 20-110ms between the bursts, followed by 165 ms of silence.
Quarter:	Five 1700hz & 2200hz bursts, with the first and last being 20-100ms in length, and the second through fourth being 20-60ms in length. The spacing between the first and second bursts is 20-110ms, while the spacing between the following bursts is 20-60ms. The tones are followed by 60ms of silence.

what happens if we reproduce these?

back when we still used pay phones...

$\longrightarrow$



stock XTAL = 3.579545 MHz
"*" = 941 Hz + 1209 Hz

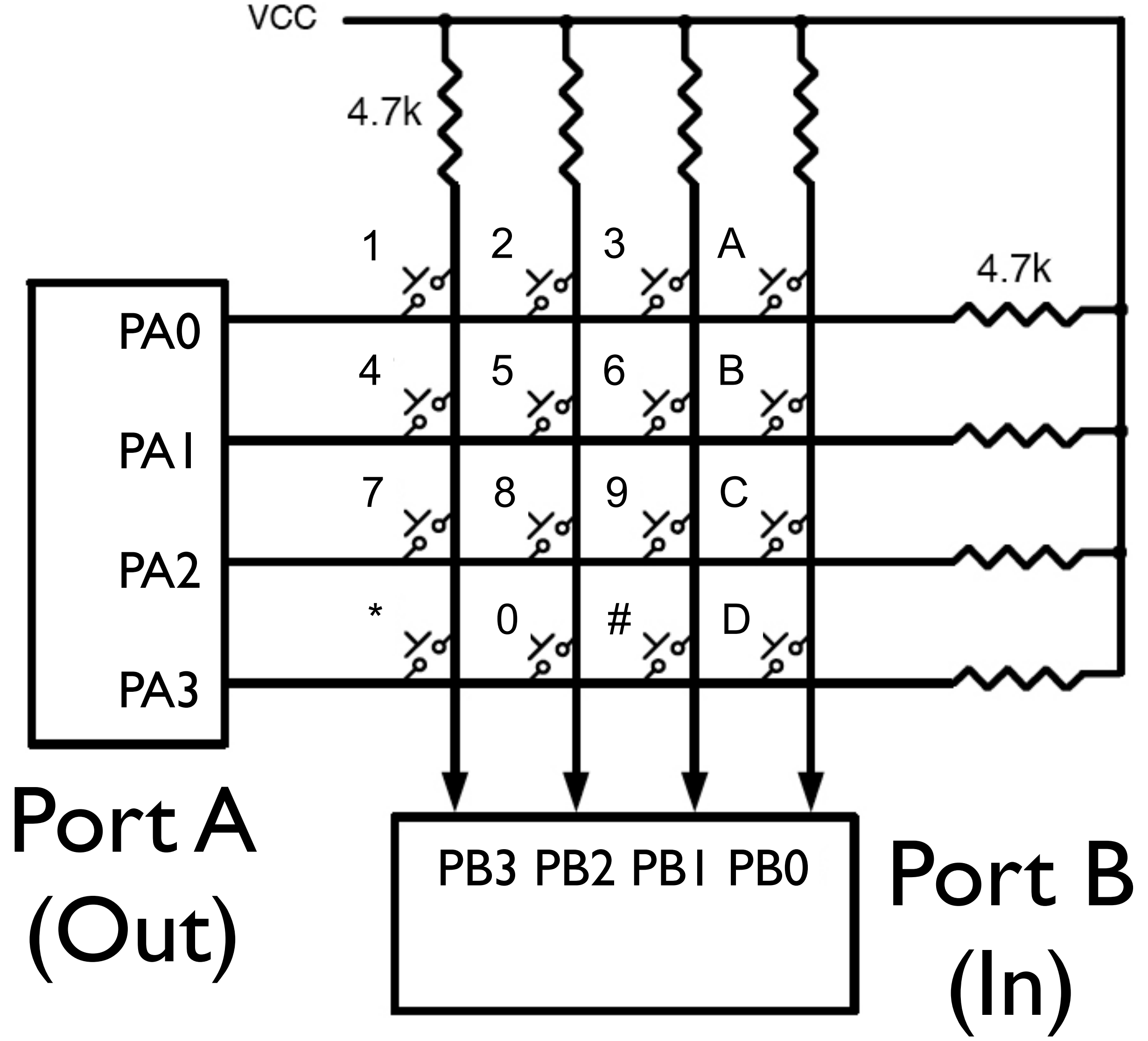
new XTAL = 6.5536 MHz
"*" = ~1700 Hz + ~2200 Hz ⟵ a nickel

Wednesday, November 27, 13

if a key is pressed

(assume 5 is pressed)

uC checks very quickly



0. both ports = 0xF

(PA: push-pull; pull-up)

1. if we set PA1 = 0

(pin is grounded)

2. then PB2= 0

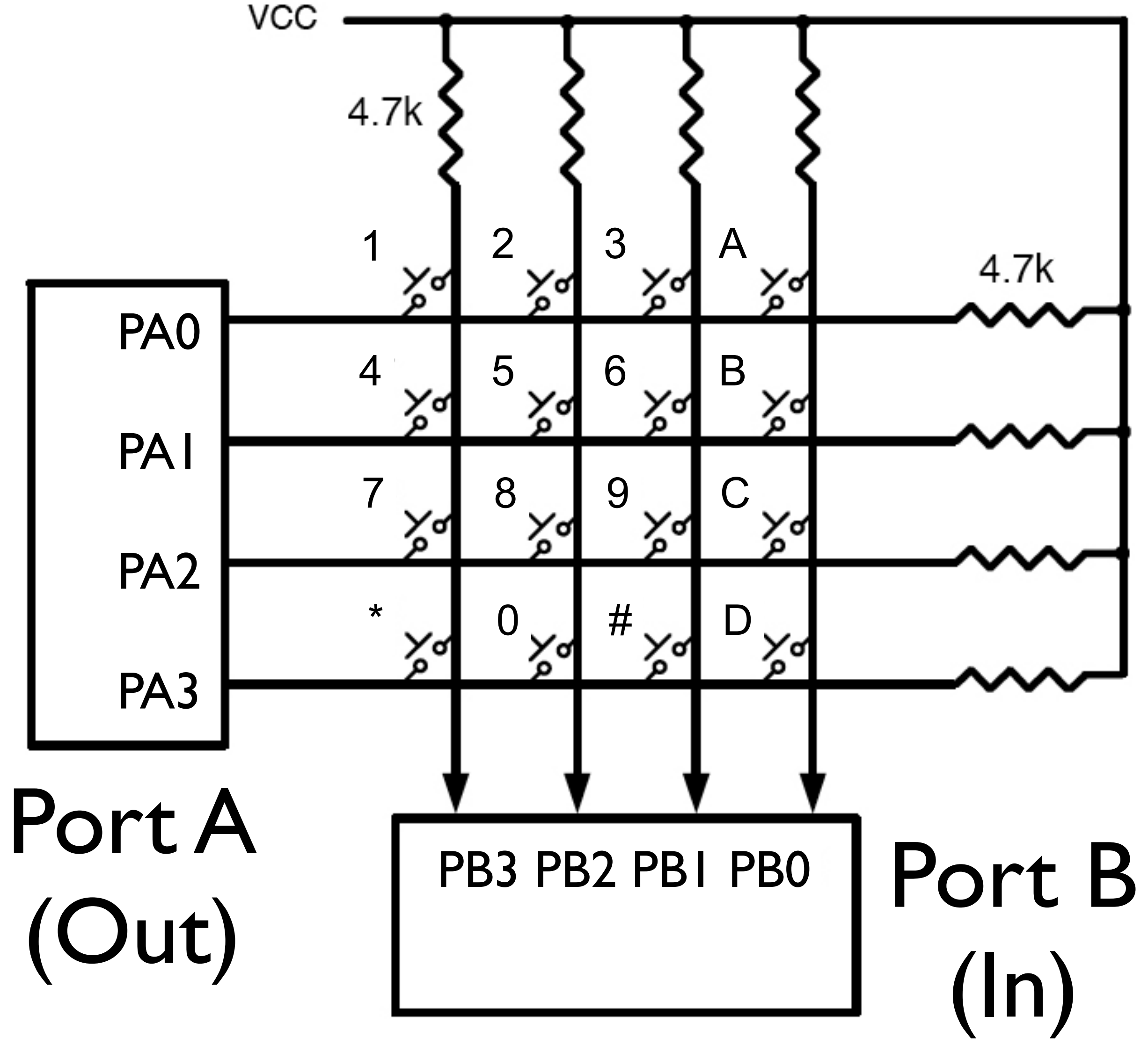
(pin is brought low)

3. all other pins unaffected

(pins still one)

if key is pressed, when its row is grounded so is its column

finding a key press



```
void keyScan()
{
//PA[3:0]=1;
//PB[3:0]=1;
 for(i=0;i<4;i++)
   {
     PA[i]=0;

     for(j=0;j<4;j++)
       if(PB[j]==0)
         keyPress(i,j);

     PA[i]=1;
   }
 }
```

wouldn't this be nice

`PA[i]=0;`

wait, we can:

(bit banding)

PA data register

(each bit corresponds to pin)

`0x400043FC`

bit band addr

pin zero

```
0x42000000 + 32*0x43FC + 4*0 = 0x42087F80
0x42000000 + 32*0x43FC + 4*1 = 0x42087F84
0x42000000 + 32*0x43FC + 4*2 = 0x42087F88
0x42000000 + 32*0x43FC + 4*3 = 0x42087F8C
```

pin three

`PA[i]=0;`

```
0x42000000 + 32*0x43FC + 4*0 = 0x42087F80
0x42000000 + 32*0x43FC + 4*1 = 0x42087F84
0x42000000 + 32*0x43FC + 4*2 = 0x42087F88
0x42000000 + 32*0x43FC + 4*3 = 0x42087F8C
```

pin

byte addr increase by four per pin

```
unsigned int *PA_B = (unsigned int *) 0x42087F80;
```

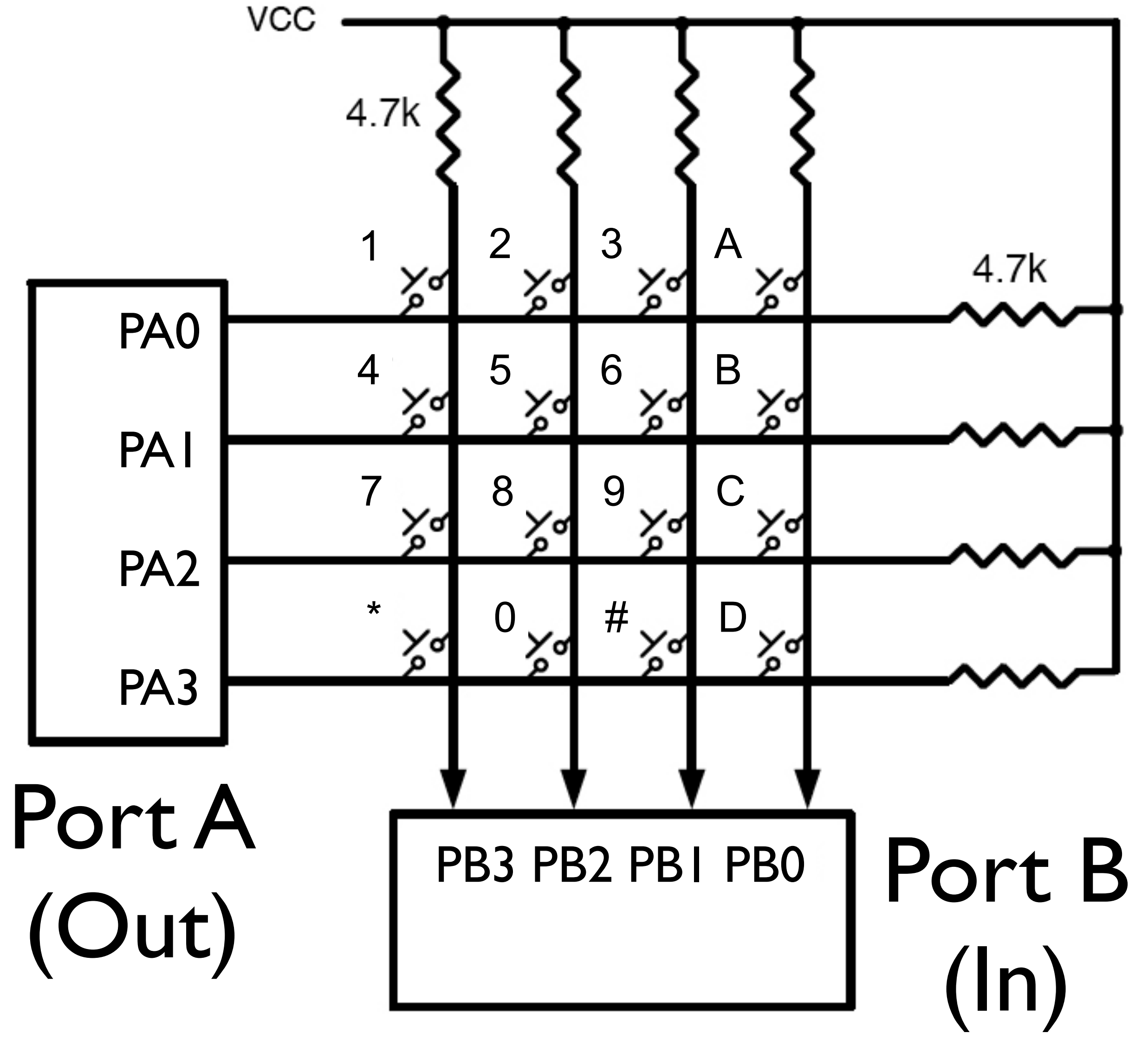
byte addr increase by four per index

(int = 32 bits = 4 bytes)

```
PA_B[0] => 0x42087F80
PA_B[1] => 0x42087F84
```

Q: how to know if key has been pressed?

how to know if key has been pressed?
(method one)



0. both ports = 0xF
1. `PA=0x0`
2. if `PB < 0xF` then a key is pressed

still have to figure out which one, though

ex: how to know if key has been pressed (C)

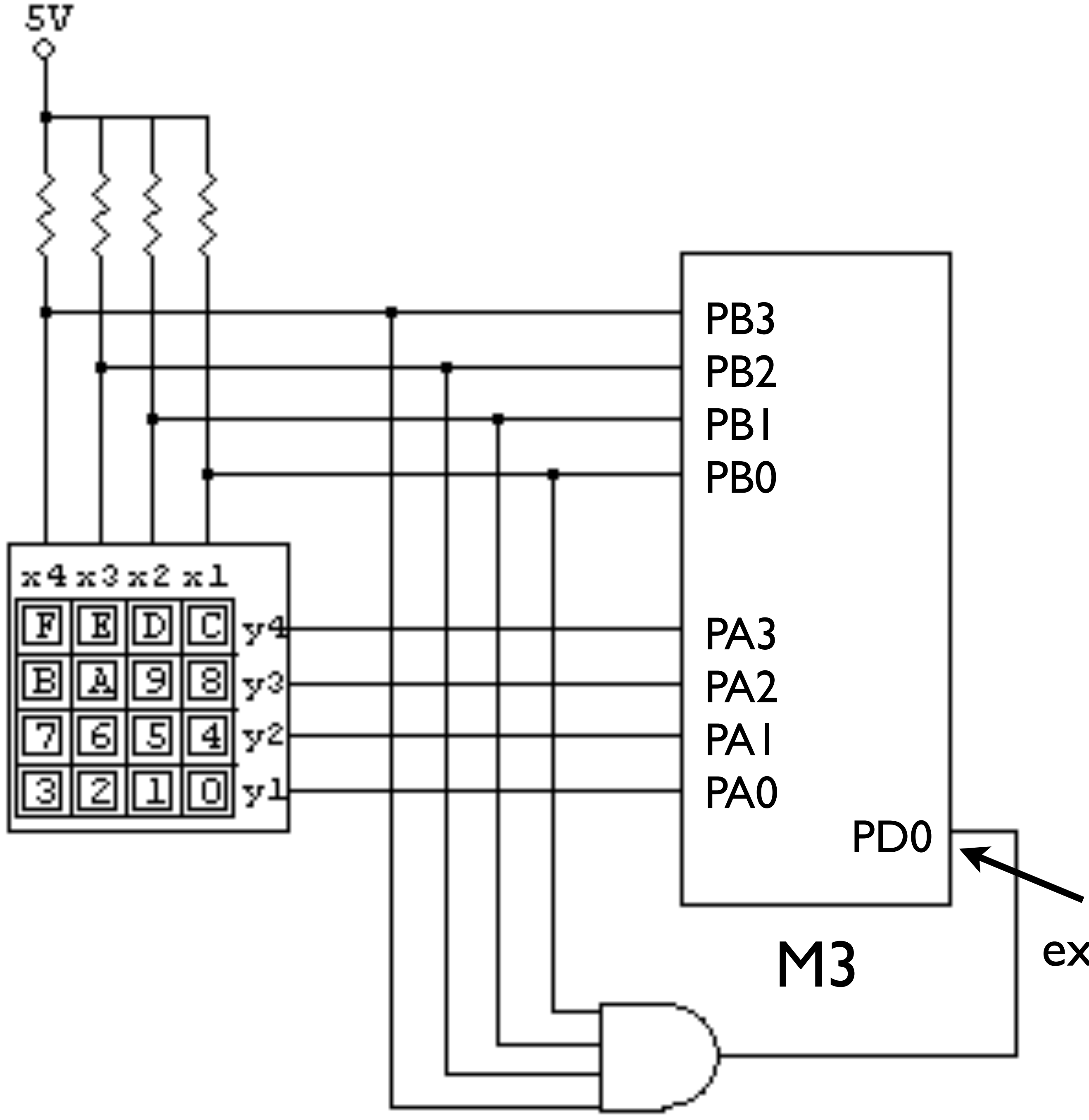
(method one)

not bb for pin
but entire port

```
// port a pins
unsigned char PA_D __attribute__((at(0x400043FC)));
// port b pins
volatile unsigned char PB_D __attribute__((at(0x400053FC)));

void keyScan(); //determines which key was pressed and does something

int main(void)
{
    PABInit();

  while(1)
    {
        PA_D &= 0xF0; //PA_D = PA_D & 0b11110000; preserve upper bits

        if((PB_D & 0xF) != 0xF) //PB_B will be all 1's unless key pressed;
                                // ignore upper bits
            keyScan();
        else
            PA_D |= 0x0F; //PA_D = PA_D | 0xb00001111;
                          // preserve upper bits but output ones
    }
}
```

how to know if key has been pressed?

(method two)



configured for external interrupt

0. both ports = 0xF

1. `PA&=0b11110000`

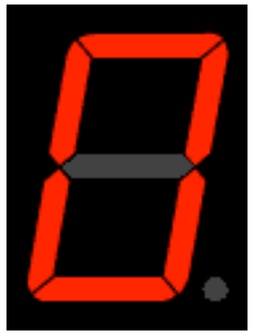
2. interrupt if a key is pressed

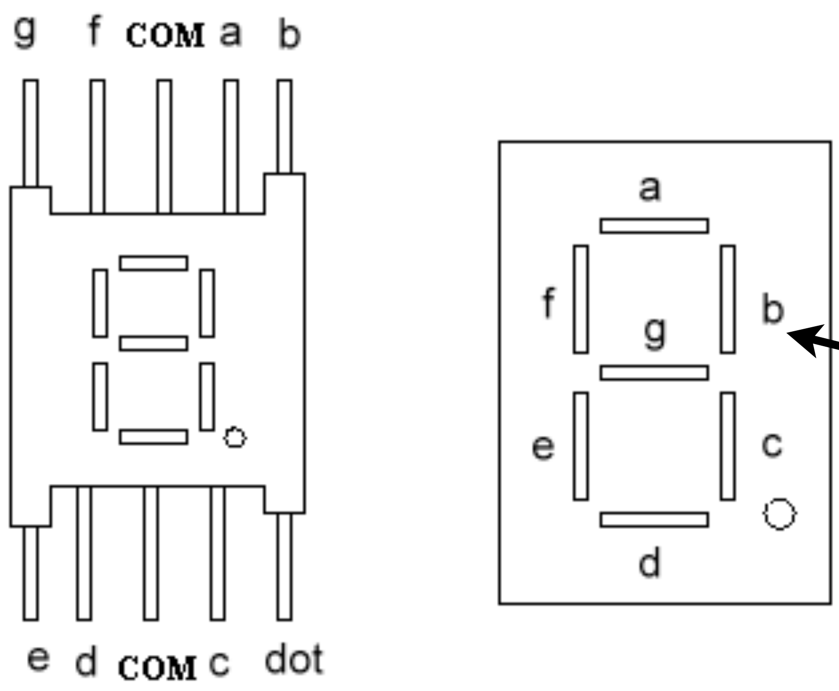
(any of PB[3:0]=0 then AND outputs 0)

still have to figure out which one, though

interlude: seven segment display

almost as impressive as a seven segment display, eh?

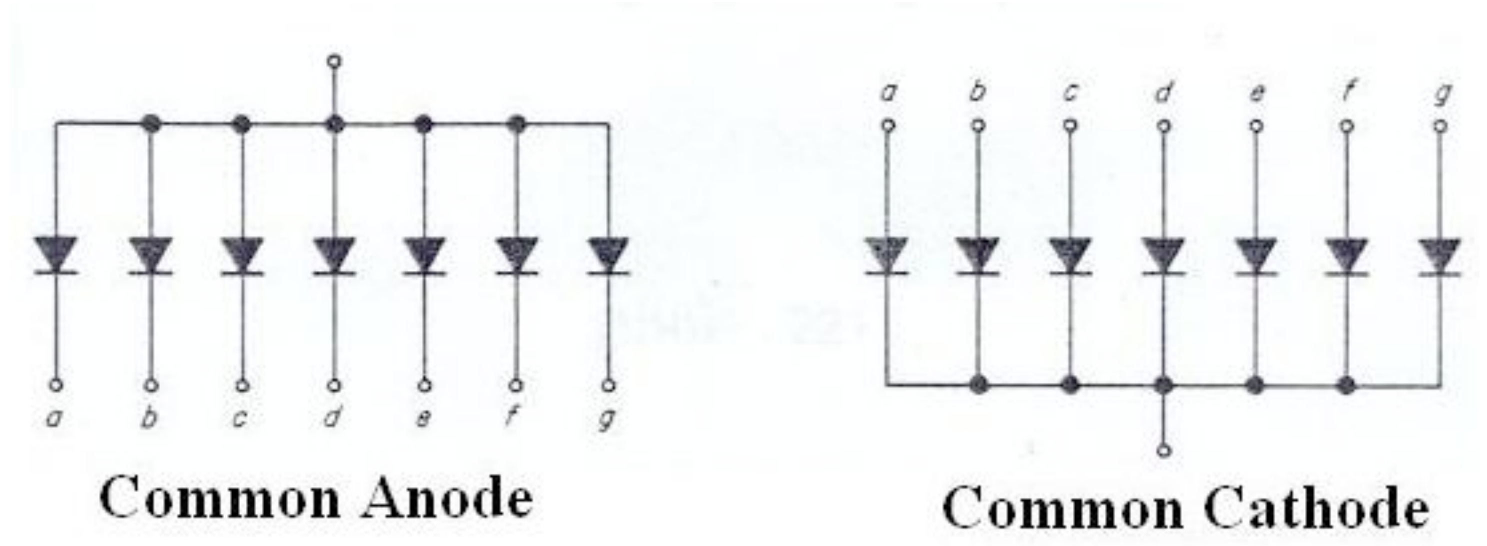




Seven-Segment Display

name of LED

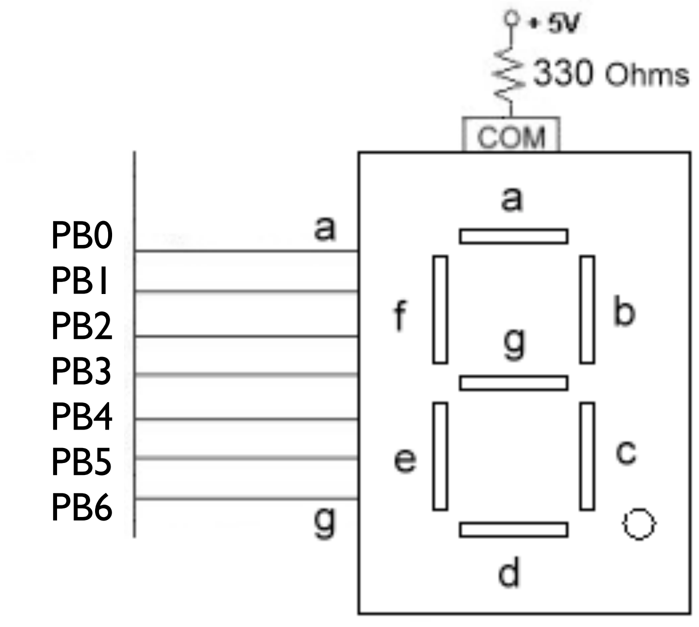
connecting seven segment display



use uC as sink

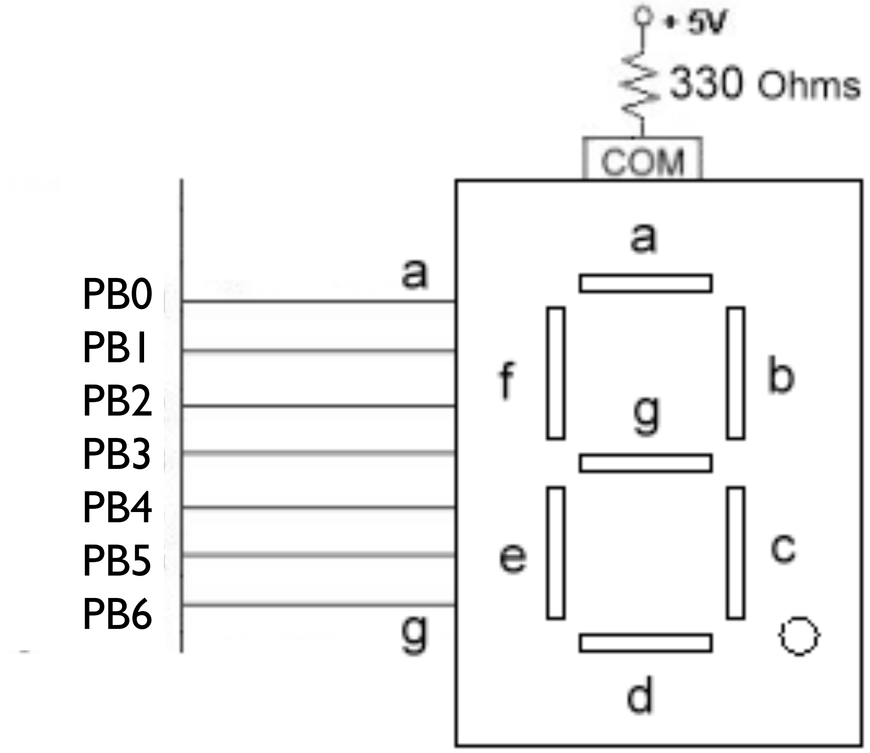
use uC as src

if `PB=0x00`, all lit up

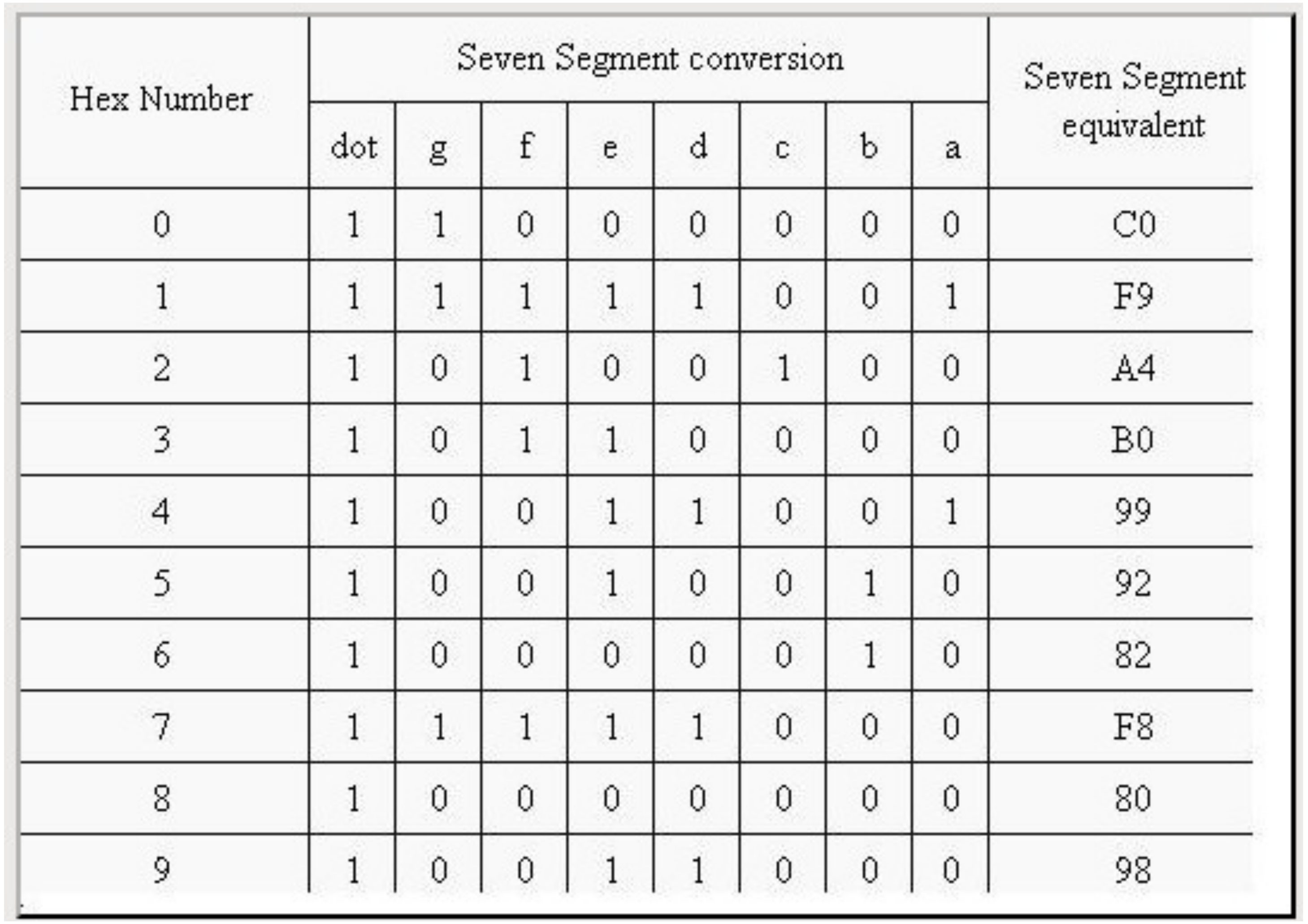


assume this config

displaying hex using seven segment display



Segment num	Pin
a	PB0
b	PB1
c	PB2
d	PB3
e	PB4
f	PB5
g	PB6
h(dp)	PB7



Hex Number	Seven Segment conversion								Seven Segment equivalent
	dot	g	f	e	d	c	b	a	
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	1	0	0	0	98

A = 0x88
b = 0x83
C = 0xC6
d = 0xA1
E = 0x86
F = 0x8E

Q: display key press using ISR and 7SD (C)

Q: display key press using ISR and 7SD (C)

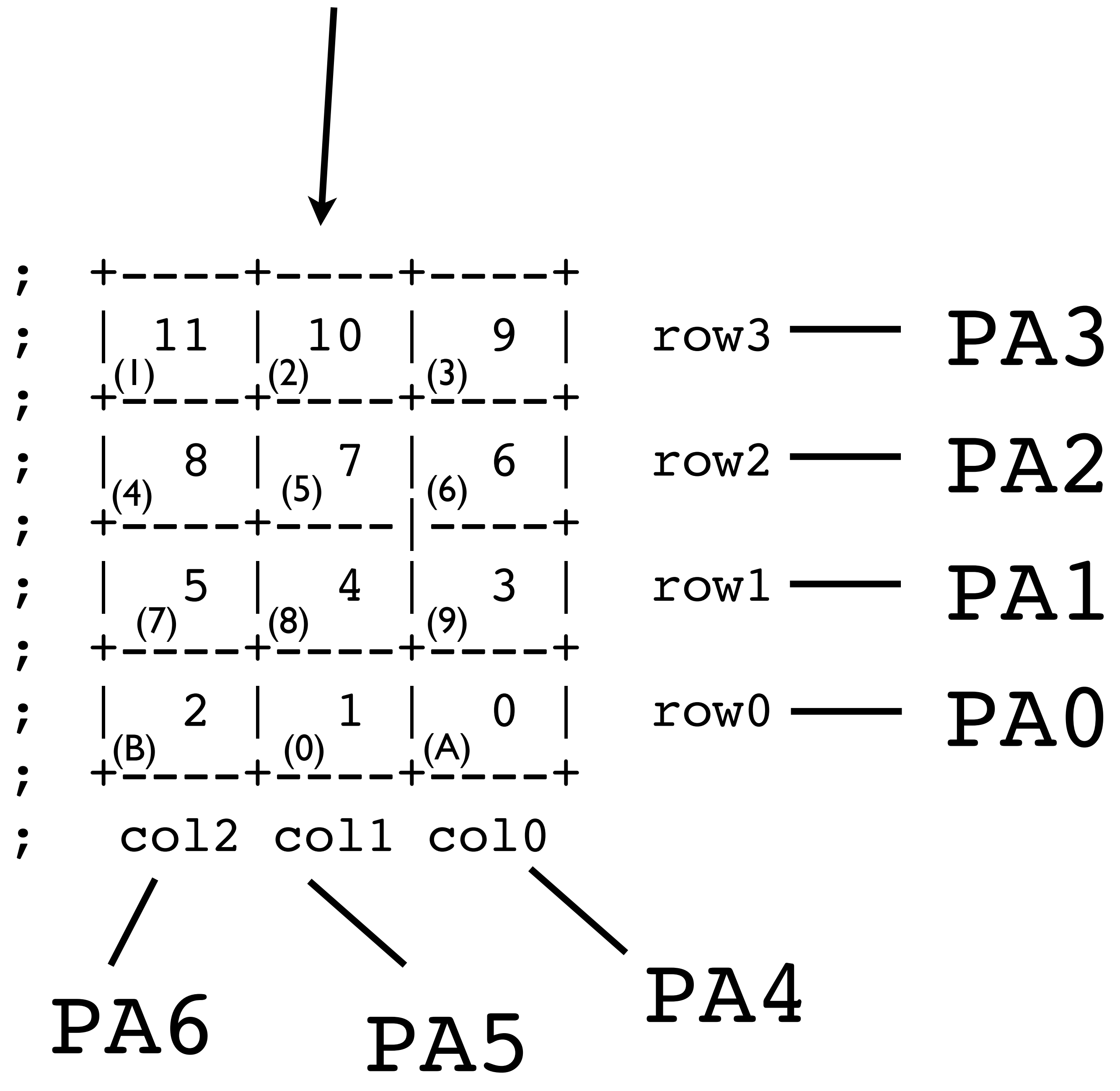
A:

1. keyScan in C
2. ISR

ex: finding a key press (C)

4x3 generic keypad

(numbers refer to key position)



```
;  +----+----+----+
;  | 11 | 10 |  9 |   row3
;  +----+----+----+
;  |  8 |  7 |  6 |   row2
;  +----+----|----+
;  |  5 |  4 |  3 |   row1
;  +----+----+----+
;  |  2 |  1 |  0 |   row0
;  +----+----+----+
;   col2 col1 col0
```

to find key:

use array with bit combination for each key

key zero

(port looks like this)

```
keys={0b11101110,
      0b11011110,
      ...,
      0b10110111};
```