

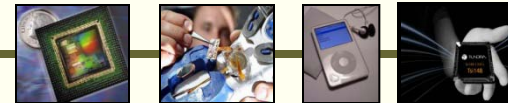
Microprocessor Fundamentals

Topic 12

Example Application

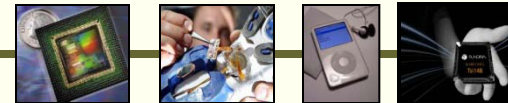
Objectives

- Gain a better understanding of the ATmega128 and assembly language programming
- Discuss an example application
- To further develop subroutines



Program

- Suppose:
 - We want to build a Logic IC Tester
 - What would we have to do?
 - How would you write the program
 - How would you connect the hardware
 - Do all logic ICs have the same pinout?



Common ICs

7432: Quad 2 Input OR Gates

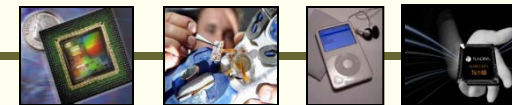
1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y

7408: Quad 2 Input AND Gates

1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y

7400: Quad 2 Input NAND Gates

1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y



Common ICs

7432: Quad 2 Input OR Gates

1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y

7408: Quad 2 Input AND Gates

1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y

7400: Quad 2 Input NAND Gates

1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y

7404: Hex Inverters

1A	1	14	Vcc
1Y	2	13	6A
2A	3	12	6Y
2Y	4	11	5A
3A	5	10	5Y
3Y	6	9	4A
Gnd	7	8	4Y

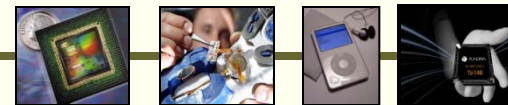
7402: Quad 2 Input NOR Gates

1Y	1	14	Vcc
1A	2	13	4Y
1B	3	12	4B
2Y	4	11	4A
2A	5	10	3Y
2B	6	9	3B
Gnd	7	8	3A



Pin-outs

- The difference in pin-outs could cause some challenges to our program
 - In a manufactured IC tester
 - You can not change the connections between the IC and the ATmega128
 - Requires a “re-definition” of input and output ports for each IC tested



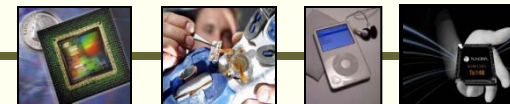
Circuit

ATMega128	PB0
	PB1
	PB2
	PB3
	PB4
	PB5
	PB6
	PB7
	PC0
	PC1
	PC2
	PC3
	PC4
	PC5
	PC6
	PC7

7432: Quad 2 Input OR Gates

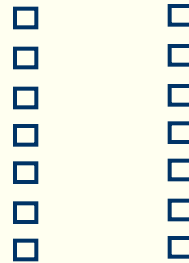
1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y

Seems to be the most common pin-out for TTL Logic ICs, so we should build our circuit with this pin-out in mind.



Circuit

ATMega128	PB0
	PB1
	PB2
	PB3
	PB4
	PB5
	PB6
	PB7
	PC0
	PC1
	PC2
	PC3
	PC4
	PC5
	PC6
	PC7



7432: Quad 2 Input OR Gates

1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y

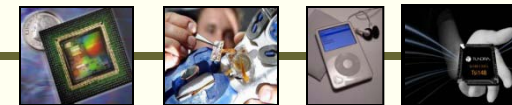
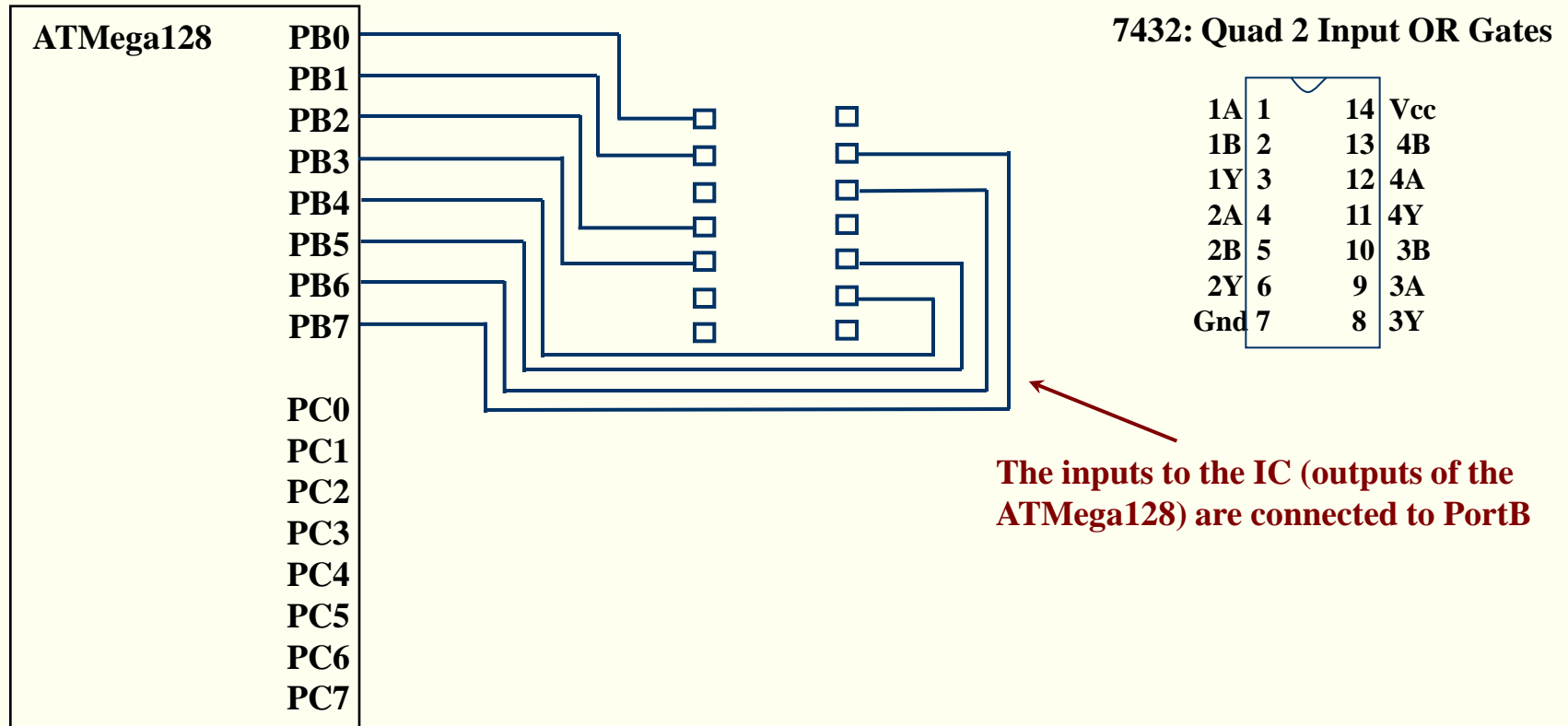


Assume this is the socket we will use in our design.

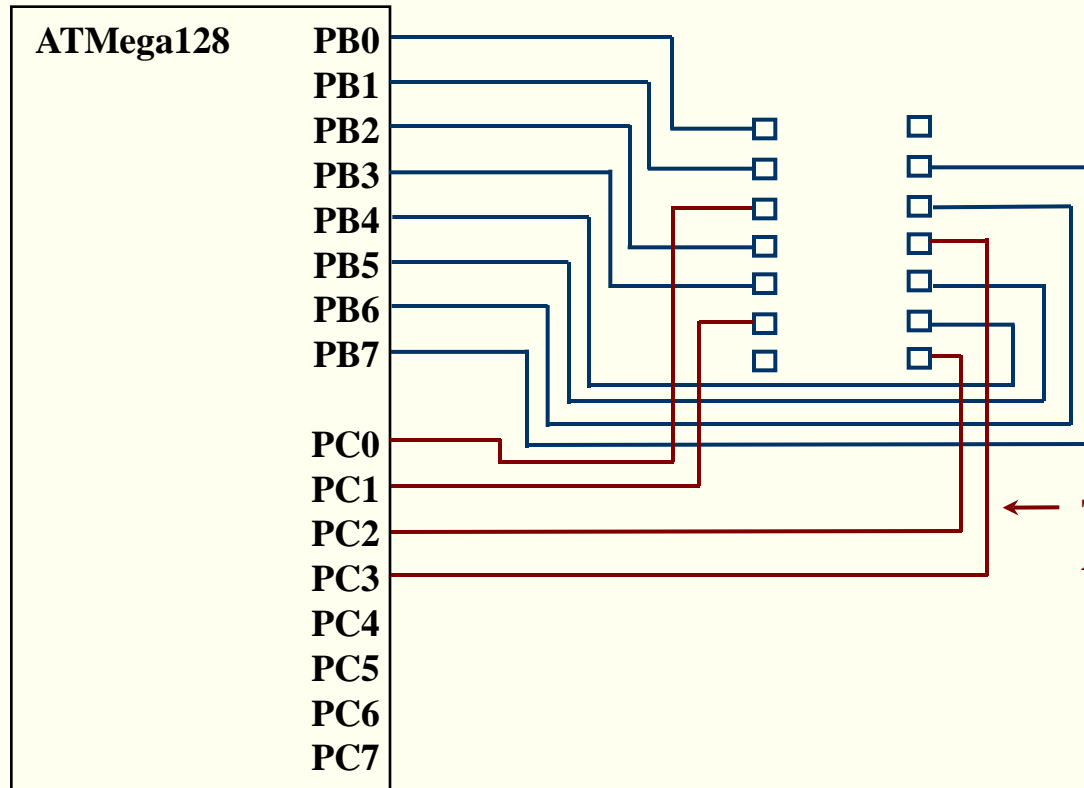
Our program will be easier to write if we use one port for the outputs and one port for the inputs



Circuit



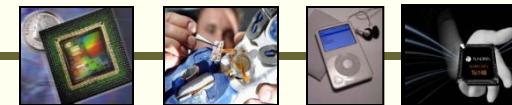
Circuit



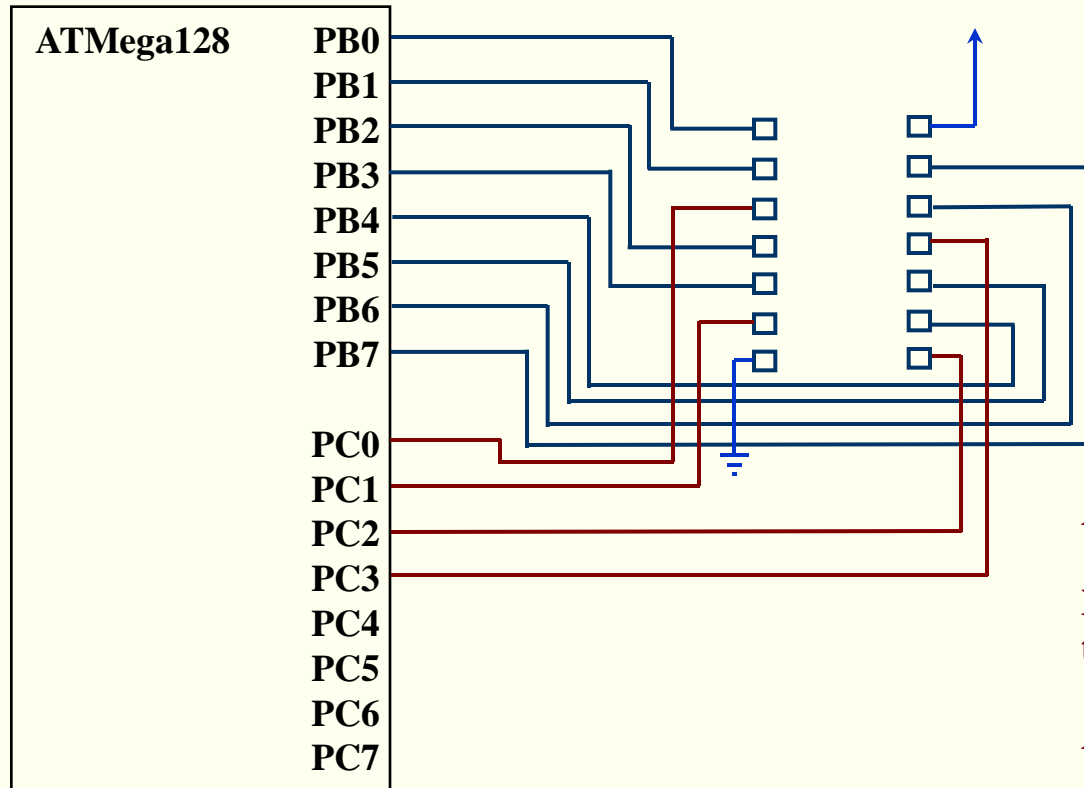
7432: Quad 2 Input OR Gates

1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y

← The outputs of the IC (inputs to the ATmega128) are connected to PortC



Circuit



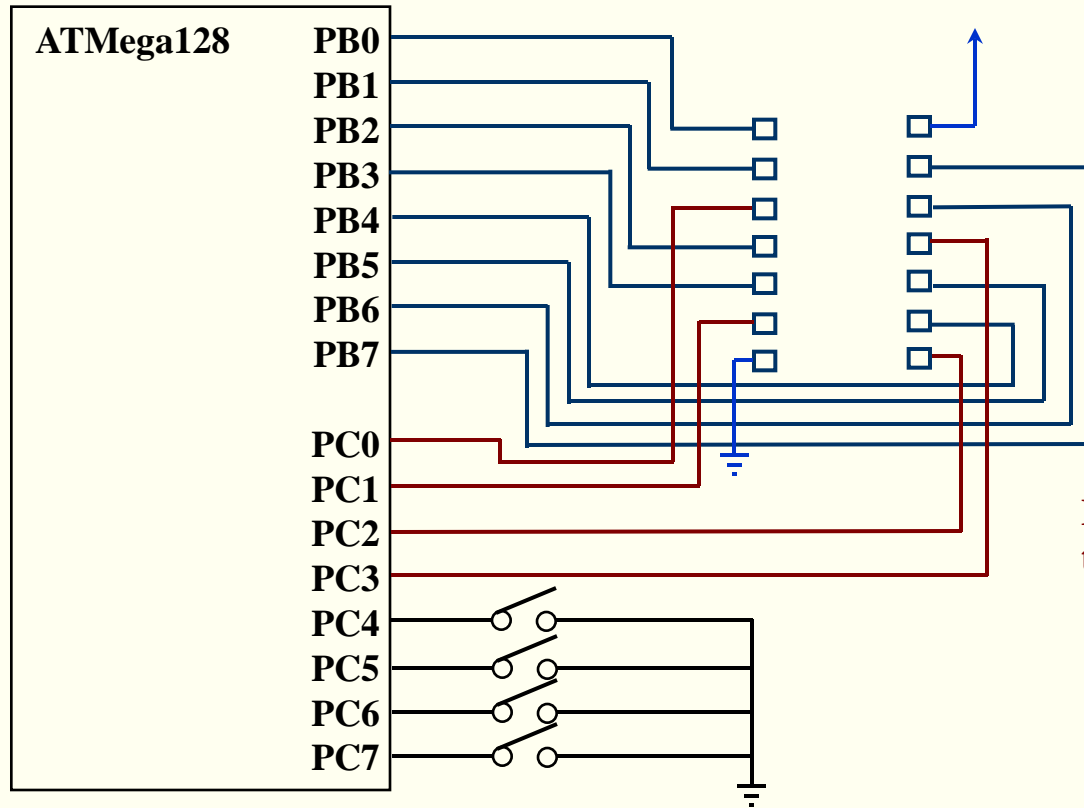
And, of course, connect Vcc and Gnd

Now we have to connect some way to call the right test based on the IC.

Any ideas?



Circuit



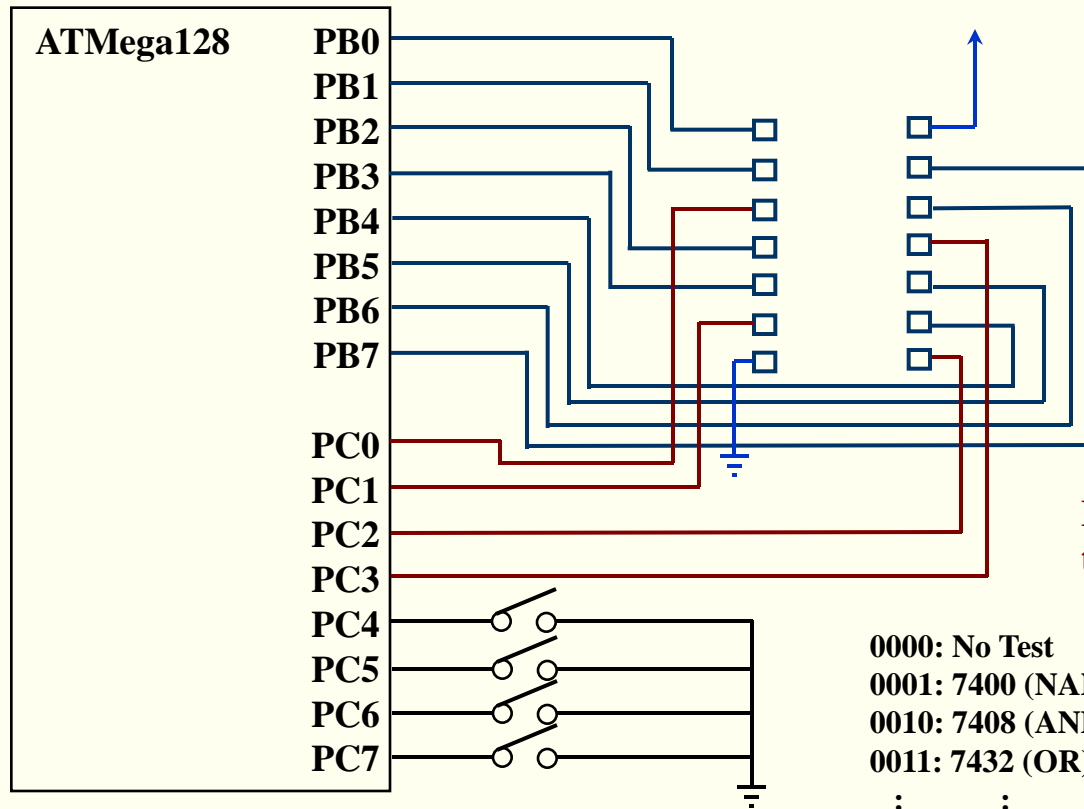
7432: Quad 2 Input OR Gates

1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y

I'll connect 4 switches. This will allow us to choose up to 15 different ICs to test.

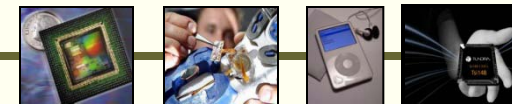


Circuit

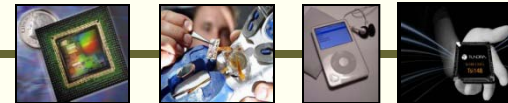
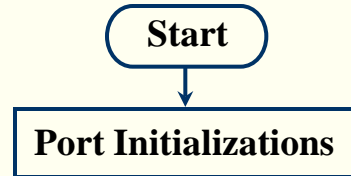


I'll connect 4 switches. This will allow us to choose up to 15 different ICs to test.

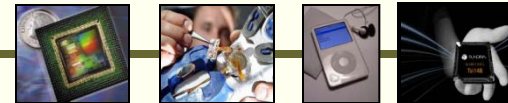
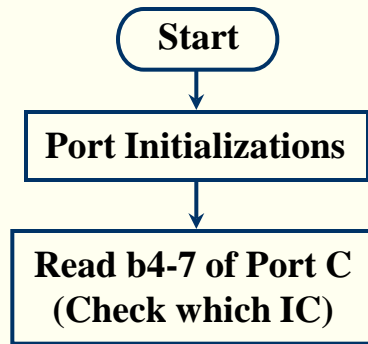
Now we have to write the program



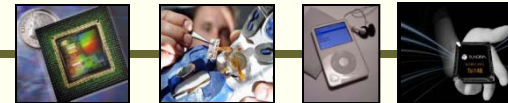
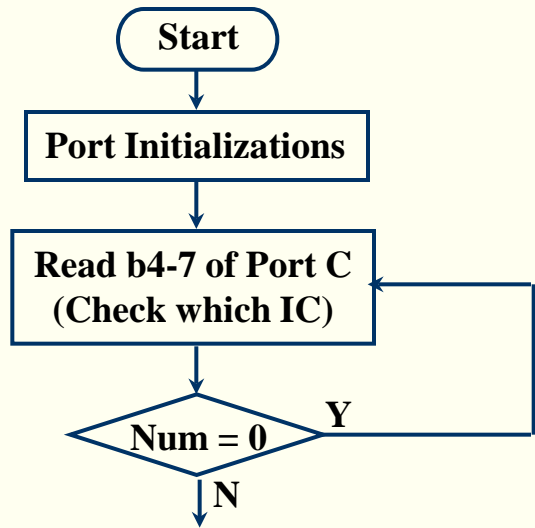
The Plan



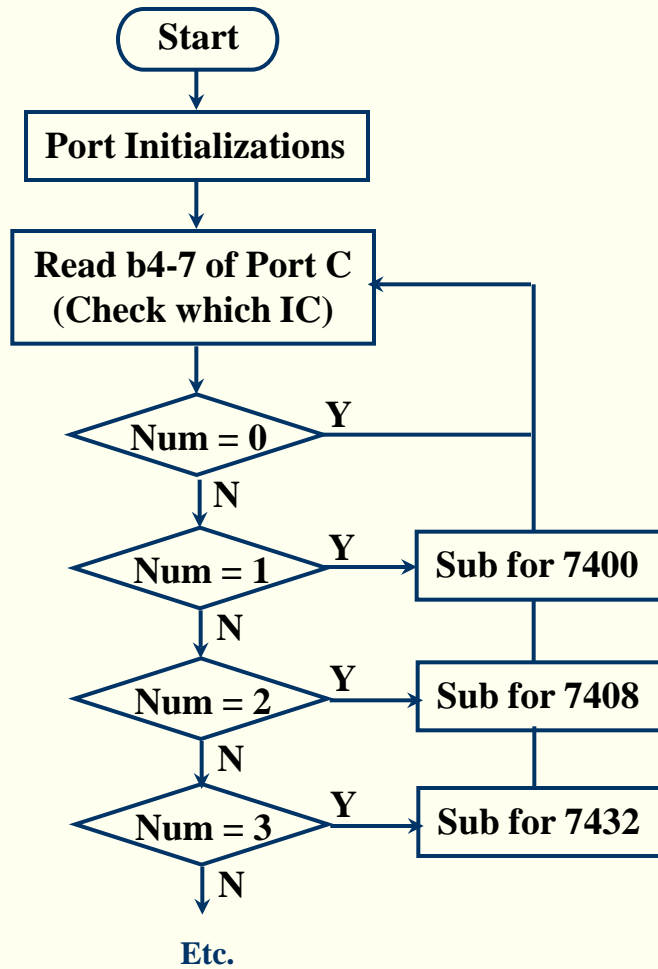
The Plan



The Plan



The Plan



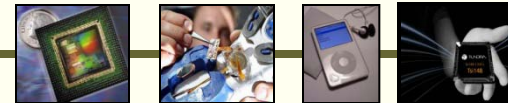
The Plan: the Subs

1. Initialize ports (other subs could have changed the I/O pins on the ATMega128)
2. Initialize a counter (how many times the test will be conducted)
3. Send 00 00 00 00 to IC inputs
4. Delay (typical delay for ICs is about 10 to 15 ns)
5. Read inputs
6. Mask bits that are not outputs from the IC
7. Correct Value?
 1. If no, send out an error message and return to the main program
 2. If yes, go to 8
8. Send next value to IC inputs (01 01 01 01, then 10 10 10 10, then 11 11 11 11)
9. Go to 4
10. Decrement the counter
 1. If counter $\neq 0$ go to 3
 2. If counter = 0 return to main program



The Plan: the Subs

1. Initialize ports (other subs could have changed the I/O pins on the ATMega128)
2. Initialize a counter (how many times the test will be conducted)
3. Send 00 00 00 00 to IC inputs
4. Delay (typical delay for ICs is about 10 to 15 ns) **Do we need a delay?**
5. Read inputs
6. Mask bits that are not outputs from the IC
7. Correct Value?
 1. If no, send out an error message and return to the main program
 2. If yes, go to 8
8. Send next value to IC inputs (01 01 01 01, then 10 10 10 10, then 11 11 11 11)
9. Go to 4
10. Decrement the counter
 1. If counter $\neq 0$ go to 3
 2. If counter = 0 return to main program

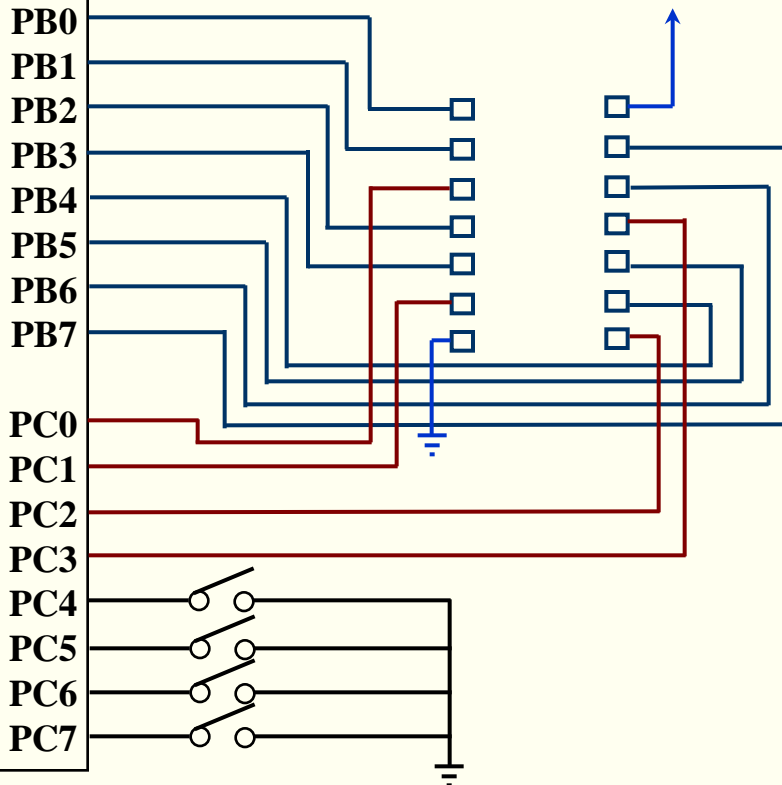


Circuit

7432: Quad 2 Input OR Gates

1A	1	14	Vcc
1B	2	13	4B
1Y	3	12	4A
2A	4	11	4Y
2B	5	10	3B
2Y	6	9	3A
Gnd	7	8	3Y

ATMega128



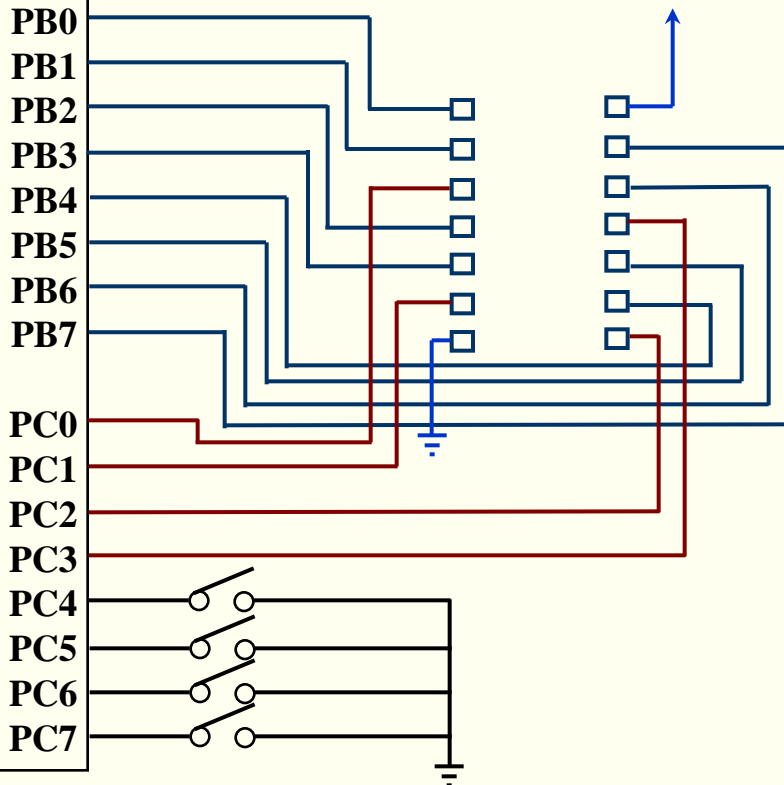
(r18) = work
(r19) = 00
(r20) = 55
(r21) = 0F
(r22) = AA
(r23) = FF

Assume these values are already in these registers



The Subroutine

ATMega128



(r18) = work
 (r19) = 00
 (r20) = 55
 (r21) = AA
 (r22) = FF
 (r23) = 00
 (r24) = 0F
 (r25) = 0F
 (r26) = 0F

Assume these values are already in these registers

T7432:

```

ser    r16          ;put all ones in r16
out    DDRB,r16    ;make B an Out Port
clr    r16
out    PortB,r16   ;init values = 0
out    DDRC,r16    ;make C an IN Port
ldi    r16,240     ;$F0
out    PortC,r16   ;pull-ups on switches
ldi    r17,50      ; setup counter

T1:    out    PortB,r19 ;set OR inputs to 00
        in    r18,PortC ;read OR outputs
        andi  r18,0b00001111 ;mask high bits
        cpse  r18,r23   ;OR outputs correct?
        rjmp  Er        ;if not jmp to error

T2:    out    PortB,r20 ;set OR inputs to 01
        in    r18,PortC ;read OR outputs
        andi  r18,0b00001111 ;mask high bits
        cpse  r18,r24   ;OR outputs correct?
        rjmp  Er        ;if not jmp to Er

T3:    out    PortB,r21 ;set OR inputs to 10
        in    r18,PortC ;read OR outputs
        andi  r18,0b00001111 ;mask high bits
        cpse  r18,r25   ;OR outputs correct?
        rjmp  Er        ;if not jmp to Er

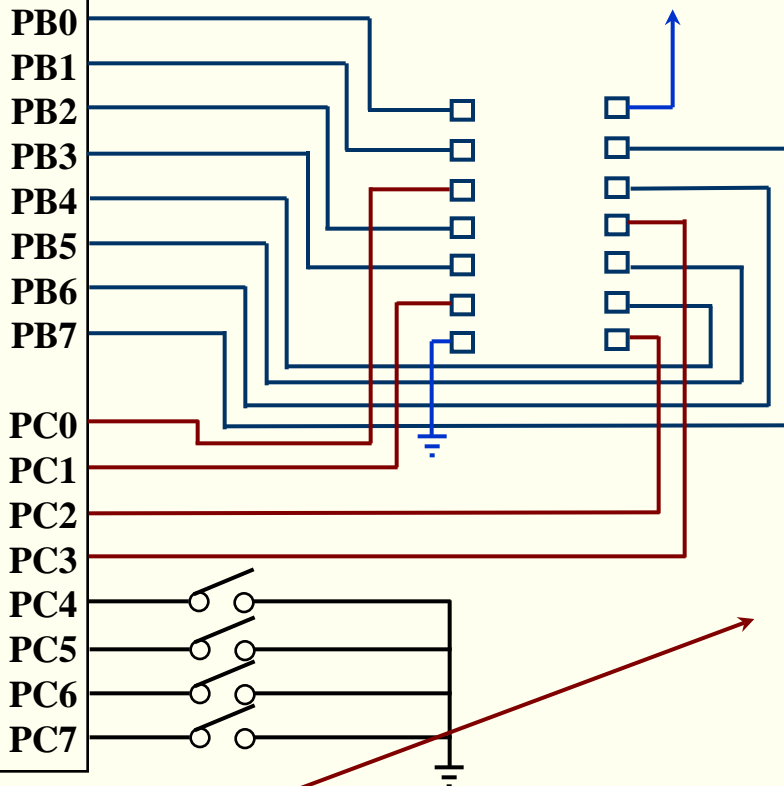
T4:    out    PortB,r22 ;set OR inputs to 11
        in    r18,PortC ;read OR outputs
        andi  r18,0b00001111 ;mask high bits
        cp    r18,r26   ;OR outputs correct?
        breq  next      ;if equal go to Next

Er:    send out error message
        rjmp  Start

Next:  dec    r17          ;decrement counter
        brne T1          ;if not 0, go thru
        ;test again
        send out a "good IC" message
        rjmp  Start
    
```

The Subroutine

ATMega128



(r18) = work
(r19) = 00
(r20) = 55
(r21) = AA
(r22) = FF
(r23) = 00
(r24) = 0F
(r25) = 0F
(r26) = 0F

This subroutine should work for any IC that has the same configuration – just set the correct values in these registers first

T7432:

```
ser    r16          ;put all ones in r16
out    DDRB,r16     ;make B an Out Port
clr    r16
out    PortB,r16    ;init values = 0
out    DDRC,r16     ;make C an IN Port
ldi    r16,240      ;$F0
out    PortC,r16    ;pull-ups on switches
ldi    r17,50       ; setup counter

T1:    out    PortB,r19 ;set OR inputs to 00
        in    r18,PortC ;read OR outputs
        andi  r18,0b00001111 ;mask high bits
        cpse  r18,r19   ;OR outputs correct?
        rjmp  Er       ;if not jmp to error

T2:    out    PortB,r20 ;set OR inputs to 01
        in    r18,PortC ;read OR outputs
        andi  r18,0b00001111 ;mask high bits
        cpse  r18,r21   ;OR outputs correct?
        rjmp  Er       ;if not jmp to Er

T3:    out    PortB,r22 ;set OR inputs to 10
        in    r18,PortC ;read OR outputs
        andi  r18,0b00001111 ;mask high bits
        cpse  r18,r21   ;OR outputs correct?
        rjmp  Er       ;if not jmp to Er

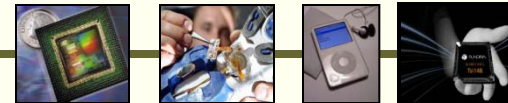
T4:    out    PortB,r23 ;set OR inputs to 11
        in    r18,PortC ;read OR outputs
        andi  r18,0b00001111 ;mask high bits
        cp    r18,r21   ;OR outputs correct?
        breq  next      ;if equal go to Next

Er:    send out error message
        rjmp  Start

Next:  dec    r17       ;decrement counter
        brne T1        ;if not 0, go thru
        ;test again
        send out a "good IC" message
        rjmp  Start
```

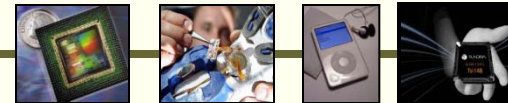
ICs with less common pin-out

- So, what do we do with the ICs with the less common pin-out?



ICs with less common pin-out

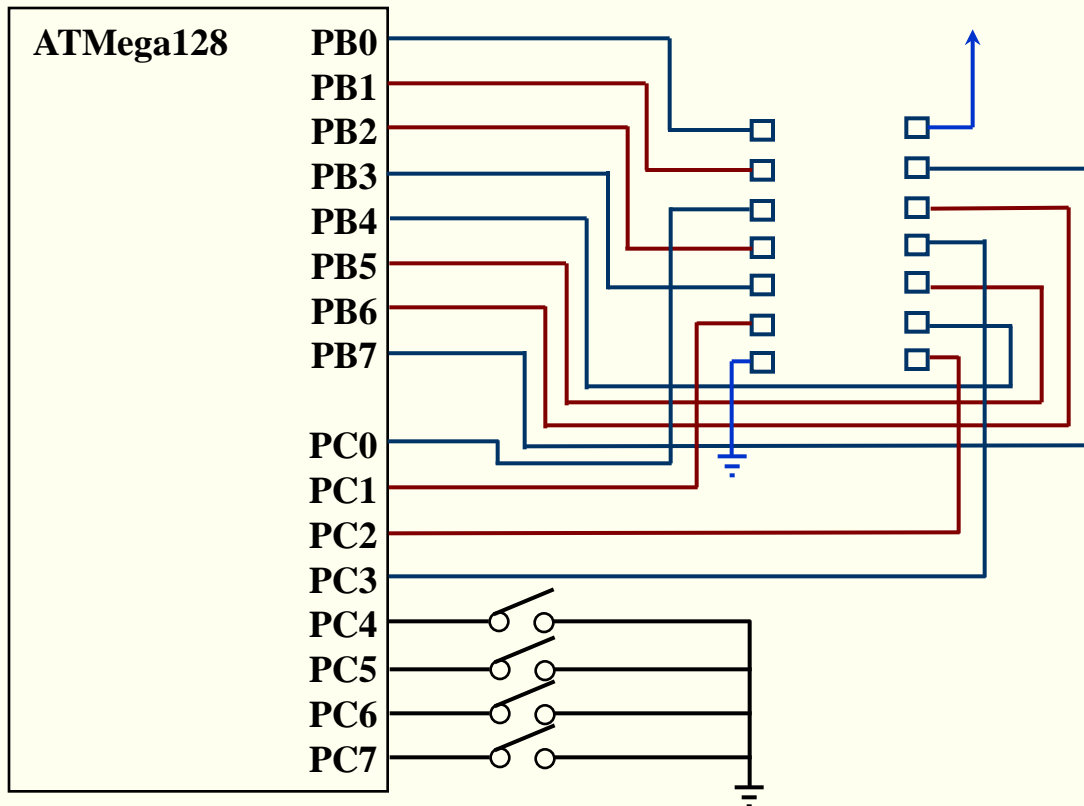
- So, what do we do with the ICs with the less common pin-out?
 - Re-initialize all Port B bits and the four lower bits of Port C
 - Send out the proper signals on output pins
 - Ensure outputs are correct without changing pins that need (or don't need) pull-up resistors
 - Read appropriate inputs
 - Masks will be different
 - May have to read both ports



7404 Example

7404: Hex Inverters

1A	1	14	Vcc
1Y	2	13	6A
2A	3	12	6Y
2Y	4	11	5A
3A	5	10	5Y
3Y	6	9	4A
Gnd	7	8	4Y



The Blue wires must be outputs on the AVR (will be inputs to the IC)

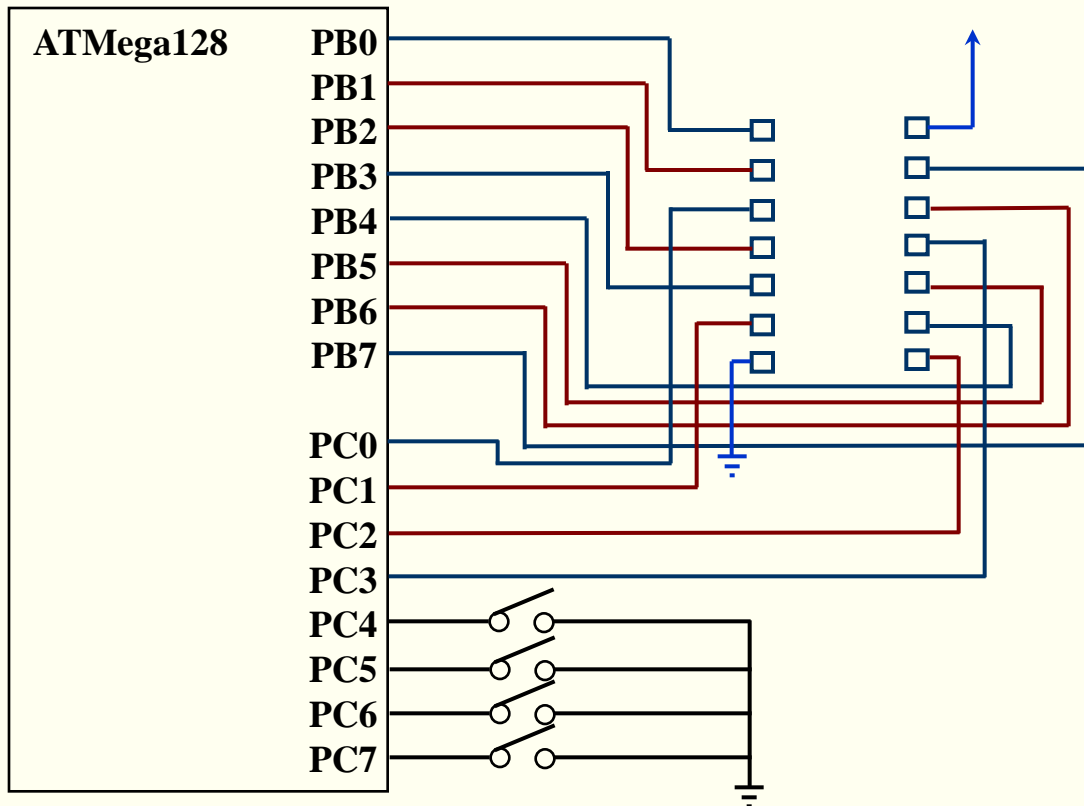
Brown wires must be inputs to the AVR (will be outputs from the IC)



7404 Example

7404: Hex Inverters

1A	1	14	Vcc
1Y	2	13	6A
2A	3	12	6Y
2Y	4	11	5A
3A	5	10	5Y
3Y	6	9	4A
Gnd	7	8	4Y



The Blue wires must be outputs on the AVR (will be inputs to the IC)

Brown wires must be inputs to the AVR (will be outputs from the IC)

None of the inputs need pull-up resistors, so it is ok to send 0s out on those pins

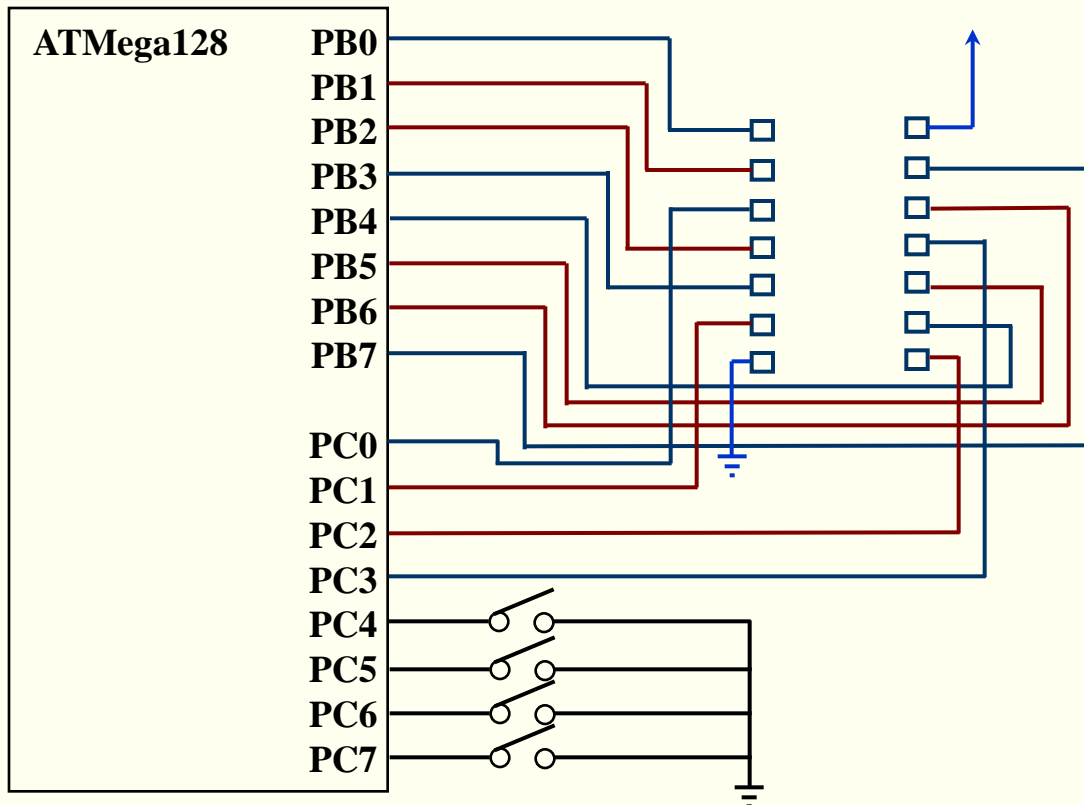
Exercise: what hex values must be sent out on Port B and Port C when testing low inputs to the inverters?



7404 Example

7404: Hex Inverters

1A	1	14	Vcc
1Y	2	13	6A
2A	3	12	6Y
2Y	4	11	5A
3A	5	10	5Y
3Y	6	9	4A
Gnd	7	8	4Y



Port B should have \$00 on it. The outputs need to be 0, and its ok to have 0s on the input pins because pull up resistors are not needed.

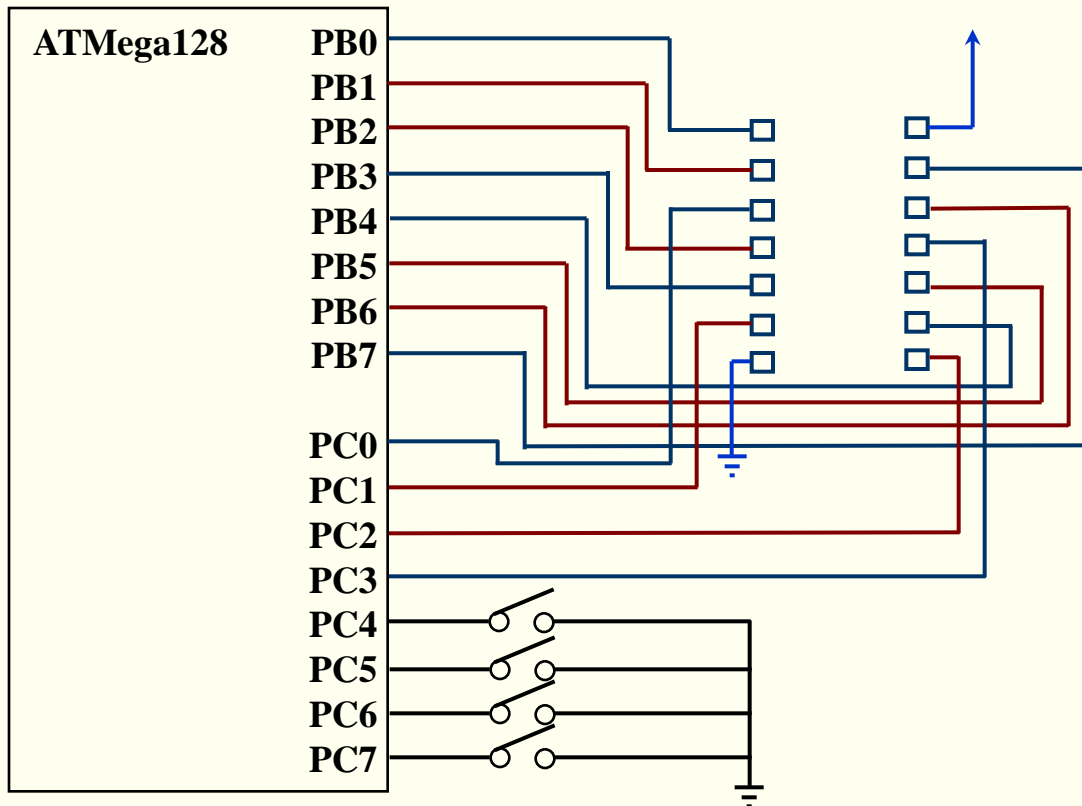
Port C should have \$F0 on it. The outputs need to be 0, and its ok to have 0s on input pins 1 & 2 because pull up resistors are not needed. However, pull-up resistors are needed on bits 4-7.



7404 Example

7404: Hex Inverters

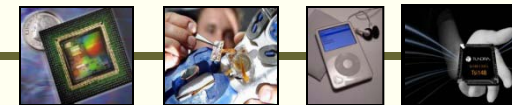
1A	1	14	Vcc
1Y	2	13	6A
2A	3	12	6Y
2Y	4	11	5A
3A	5	10	5Y
3Y	6	9	4A
Gnd	7	8	4Y



Port B should have \$00 on it. The outputs need to be 0, and its ok to have 0s on the input pins because pull up resistors are not needed.

Port C should have \$F0 on it. The outputs need to be 0, and its ok to have 0s on input pins 1 & 2 because pull up resistors are not needed. However, pull-up resistors are needed on bits 4-7.

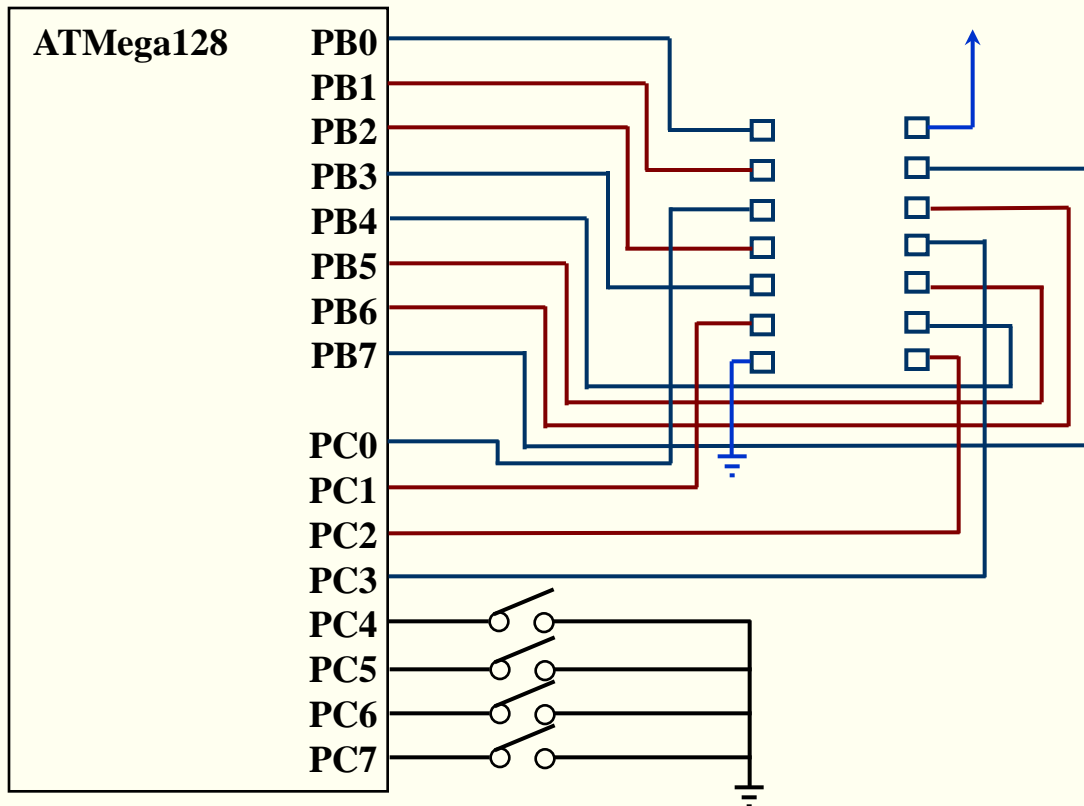
Exercise: what hex values must be sent out on Port B and Port C when testing high inputs to the inverters?



7404 Example

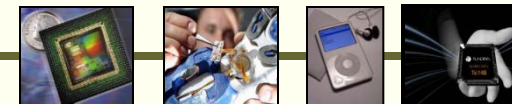
7404: Hex Inverters

1A	1	14	Vcc
1Y	2	13	6A
2A	3	12	6Y
2Y	4	11	5A
3A	5	10	5Y
3Y	6	9	4A
Gnd	7	8	4Y



Port B should have \$99 (10011001) on it. The outputs need to be 1, but the input pins should be 0 because pull up resistors are not needed.

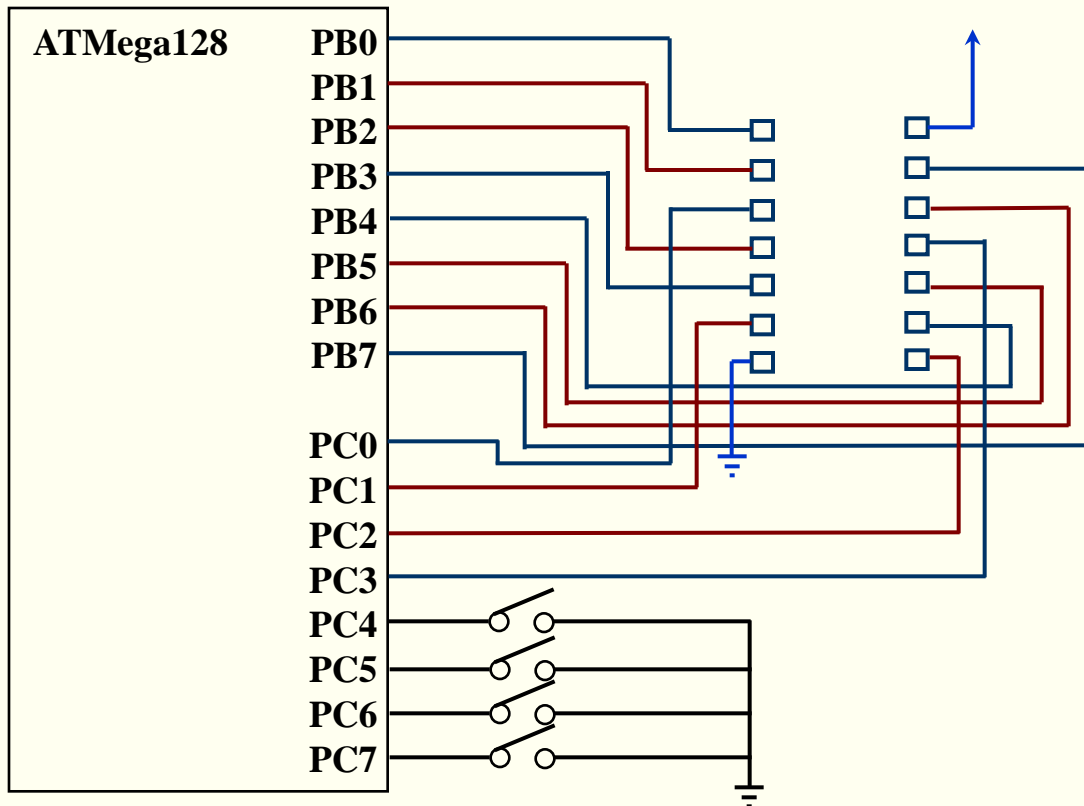
Port C should have \$F9 on it. The outputs need to be 1, and pins 1 & 2 should have 0 because pull up resistors are not needed. However, pull-up resistors are needed on bits 4-7.



7404 Example

7404: Hex Inverters

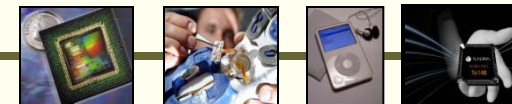
1A	1	14	Vcc
1Y	2	13	6A
2A	3	12	6Y
2Y	4	11	5A
3A	5	10	5Y
3Y	6	9	4A
Gnd	7	8	4Y



Port B should have \$99 (10011001) on it. The outputs need to be 1, but the input pins should be 0 because pull up resistors are not needed.

Port C should have \$F9 on it. The outputs need to be 1, and pins 1 & 2 should have 0 because pull up resistors are not needed. However, pull-up resistors are needed on bits 4-7.

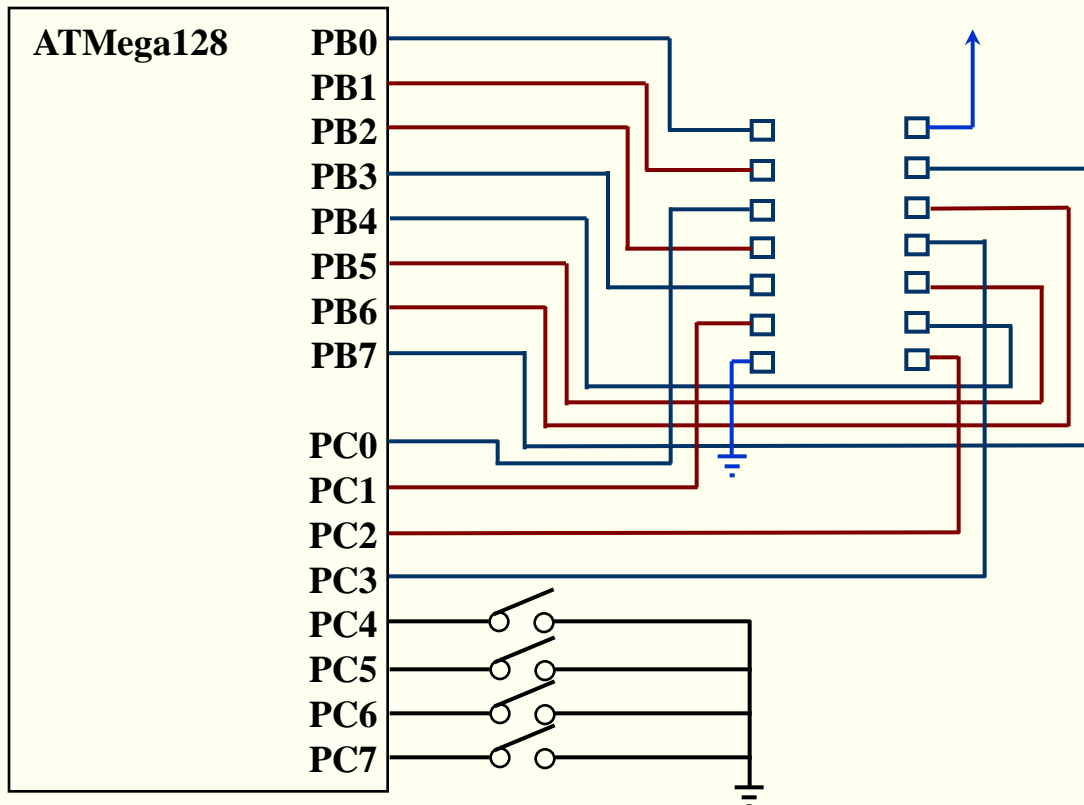
Exercise: what should the masks be when reading the inputs to the ATmega128?



7404 Example

7404: Hex Inverters

1A	1	14	Vcc
1Y	2	13	6A
2A	3	12	6Y
2Y	4	11	5A
3A	5	10	5Y
3Y	6	9	4A
Gnd	7	8	4Y



Port B should use \$66 (01100110) because we do not want to change any input values (we must use the AND operation for the mask), and we want to clear out the bits that are configured as outputs.

Port C should use \$06. The 6 for the same reason as above, the 0 because we are not sure what the switch values should be, so we want to clear those bits out.



Summary

- In looking at this example, we:
 - Gained a better understanding of the ATmega128 and assembly language programming
 - Learned more about developing subroutines
 - Especially in ways that they can be re-used
 - Were introduced to re-configuring the I/O pins on the ports
 - Discussed a practical example in which we would have to do so
 - Saw new uses for masks and determining what the mask values should be

