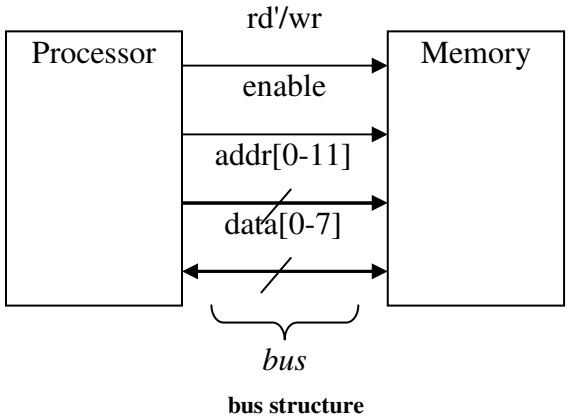
Input/Output

Introduction

- Embedded system functionality aspects
  - Processing
    - Transformation of data
    - Implemented using processors
  - Storage
    - Retention of data
    - Implemented using memory
  - Communication
    - Transfer of data between processors and memories
    - Implemented using buses
    - Called *interfacing*

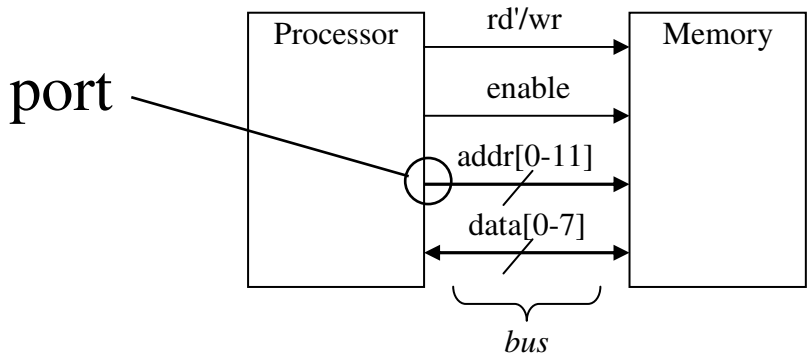
A simple bus

- Wires:
  - Uni-directional or bi-directional
  - One line may represent multiple wires
- Bus
  - Set of wires with a single function
    - Address bus, data bus
  - Or, entire collection of wires
    - Address, data and control
    - Associated protocol: rules for communication



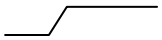
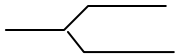
bus structure

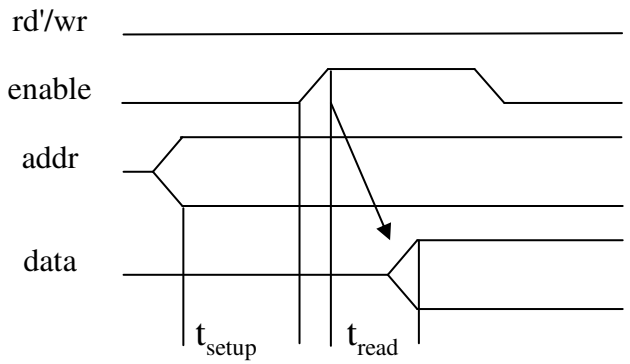
Ports



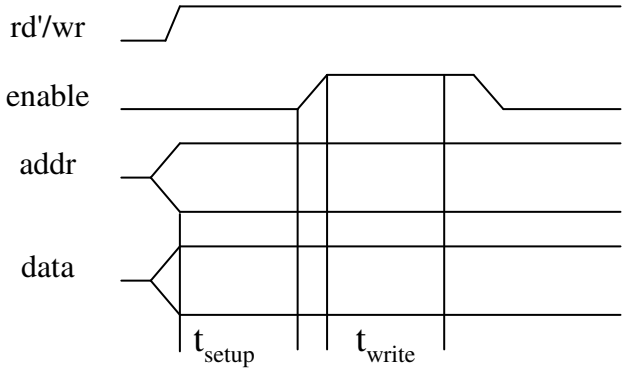
- Conducting device on periphery
- Connects bus to processor or memory
- Often referred to as a *pin*
  - Actual pins on periphery of IC package that plug into socket on printed-circuit board
  - Sometimes metallic balls instead of pins
  - Today, metal “pads” connecting processors and memories within single IC
- Single wire or set of wires with single function
  - E.g., 12-wire address port

Timing Diagrams

- Most common method for describing a communication protocol
- Time proceeds to the right on x-axis
- Control signal: low or high 
  - May be active low (e.g., go', /go, or go_L)
  - Use terms *assert* (active) and *deassert*
  - Asserting go' means go=0
- Data signal: not valid or valid 
- Protocol may have subprotocols
  - Called bus cycle, e.g., read and write
  - Each may be several clock cycles
- Read example
  - *rd'/wr* set low,address placed on *addr* for at least t_{setup} time before *enable* asserted, enable triggers memory to place data on *data* wires by time t_{read}



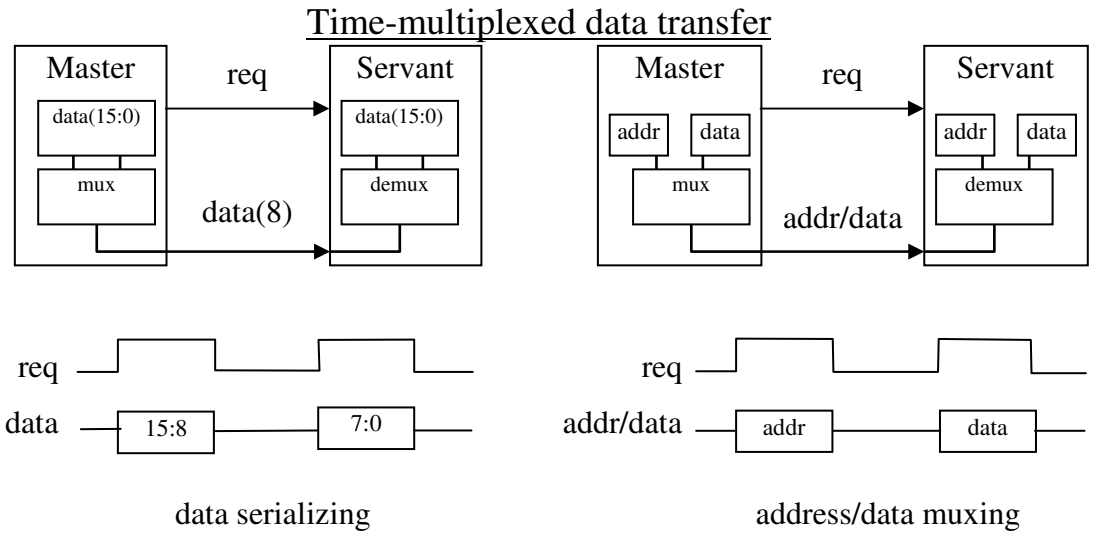
read protocol



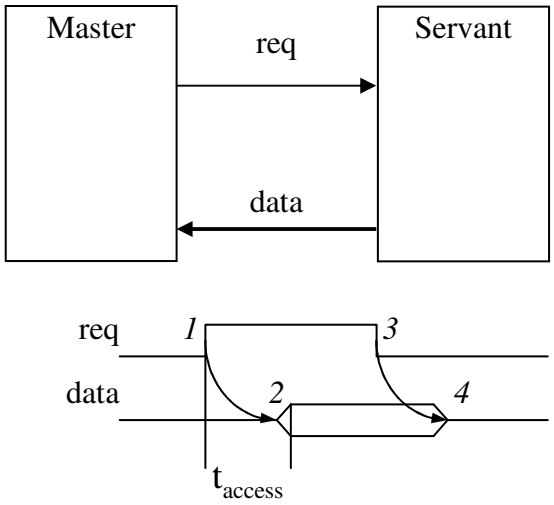
write protocol

Basic protocol concepts

- Actor: master initiates, servant (slave) respond
- Direction: sender, receiver
- Addresses: special kind of data
  - Specifies a location in memory, a peripheral, or a register within a peripheral
- Time multiplexing
  - Share a single set of wires for multiple pieces of data
  - Saves wires at expense of time

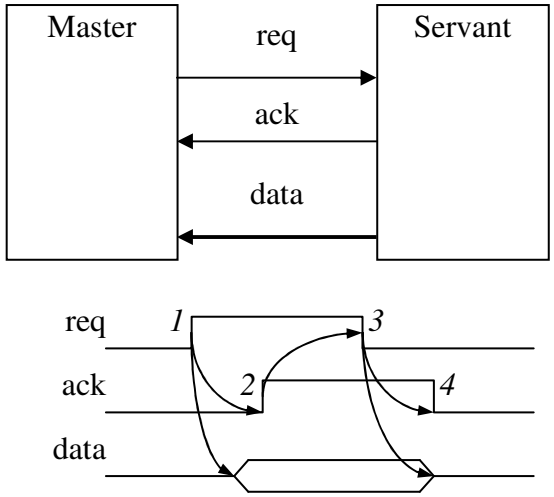


Basic protocol concepts: control methods



1. Master asserts *req* to receive data
2. Servant puts data on bus **within time t_{access}**
3. Master receives data and deasserts *req*
4. Servant ready for next request

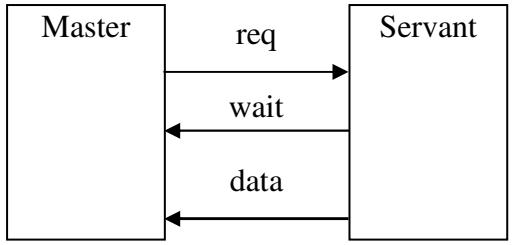
Strobe protocol

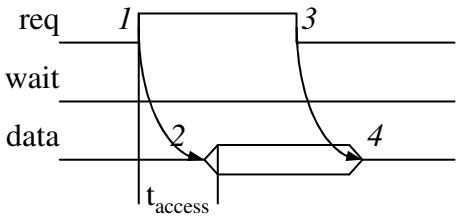


1. Master asserts *req* to receive data
2. Servant puts data on bus **and asserts *ack***
3. Master receives data and deasserts *req*
4. Servant ready for next request

Handshake protocol

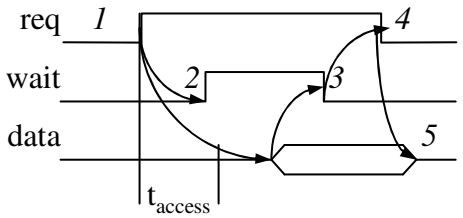
A strobe/handshake compromise





1. Master asserts *req* to receive data
2. Servant puts data on bus **within time t_{access}** (wait line is unused)
3. Master receives data and deasserts *req*
4. Servant ready for next request

Fast-response case

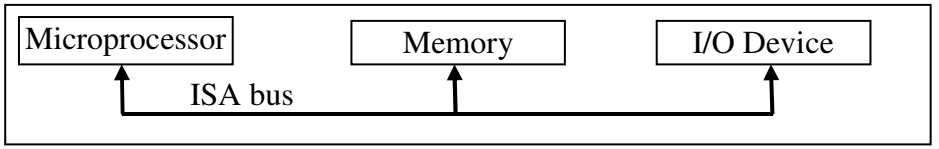


1. Master asserts *req* to receive data
2. Servant can't put data within **t_{access}**, **asserts *wait*** ack
3. Servant puts data on bus and **deasserts *wait***
4. Master receives data and deasserts *req*
5. Servant ready for next request

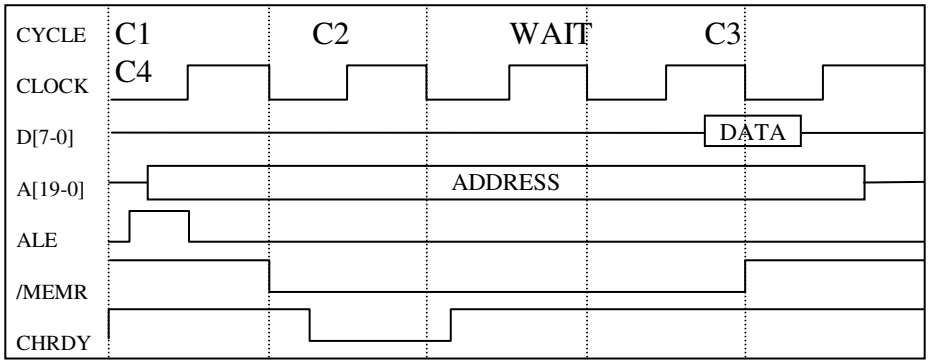
Slow-response case

ISA bus protocol – memory access

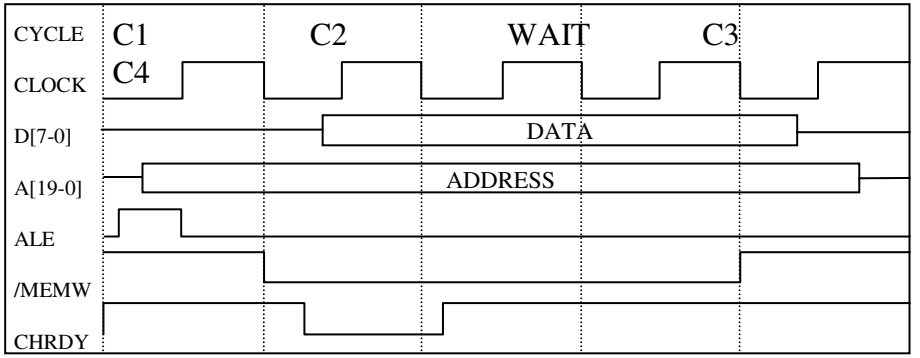
- ISA: Industry Standard Architecture
  - Common in 80x86's
- Features
  - 20-bit address
  - Compromise strobe/handshake control
    - 4 cycles default
    - Unless CHRDY deasserted – resulting in additional wait cycles (up to 6)



memory-read bus cycle



memory-write bus cycle

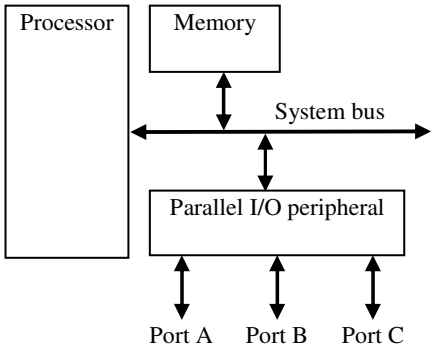


Microprocessor interfacing: I/O addressing

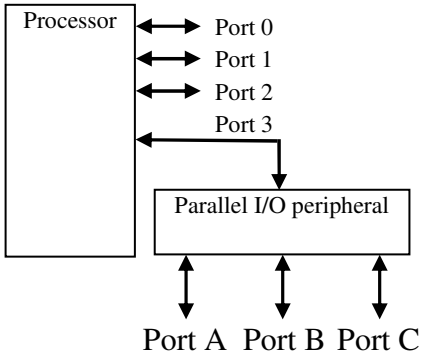
- A microprocessor communicates with other devices using some of its pins
  - Port-based I/O (parallel I/O)
    - Processor has one or more N-bit ports
    - Processor's software reads and writes a port just like a register
    - E.g., P0 = 0xFF; v = P1.2; -- P0 and P1 are 8-bit ports
  - Bus-based I/O
    - Processor has address, data and control ports that form a single bus
    - Communication protocol is built into the processor
    - A single instruction carries out the read or write protocol on the bus

Compromises/extensions

- Parallel I/O peripheral
  - When processor only supports bus-based I/O but parallel I/O needed
  - Each port on peripheral connected to a register within peripheral that is read/written by the processor
- Extended parallel I/O
  - When processor supports port-based I/O but more ports needed
  - One or more processor ports interface with parallel I/O peripheral extending total number of ports available for I/O
  - e.g., extending 4 ports to 6 ports in figure



Adding parallel I/O to a bus-based I/O processor



Extended parallel I/O

Types of bus-based I/O: memory-mapped I/O and standard I/O

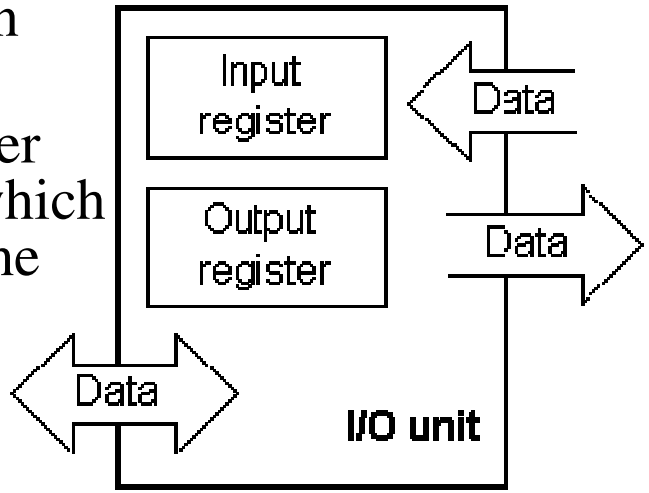
- Processor talks to both memory and peripherals using same bus – two ways to talk to peripherals
  - Memory-mapped I/O
    - Peripheral registers occupy addresses in same address space as memory
    - e.g., Bus has 16-bit address
      - lower 32K addresses may correspond to memory
      - upper 32k addresses may correspond to peripherals
  - Standard I/O (I/O-mapped I/O)
    - Additional pin (*M/IO*) on bus indicates whether a memory or peripheral access
    - e.g., Bus has 16-bit address
      - all 64K addresses correspond to memory when *M/IO* set to 0
      - all 64K addresses correspond to peripherals when *M/IO* set to 1

Memory-mapped I/O vs. Standard I/O

- Memory-mapped I/O
  - Requires no special instructions
    - Assembly instructions involving memory like MOV and ADD work with peripherals as well
    - Standard I/O requires special instructions (e.g., IN, OUT) to move data between peripheral registers and memory
- Standard I/O
  - No loss of memory addresses to peripherals
  - Simpler address decoding logic in peripherals possible
    - When number of peripherals much smaller than address space then high-order address bits can be ignored
      - smaller and/or faster comparators

I/O on 16F877

- Term "port" refers to a group of pins on a microcontroller which can be accessed simultaneously, or on which we can set the desired combination of zeros and ones, or read from them an existing status.
- Physically, port is a register inside a microcontroller which is connected by wires to the pins of a microcontroller.
- Ports represent physical connection of Central Processing Unit with an outside world.
- Microcontroller uses them in order to monitor or control other components or devices.

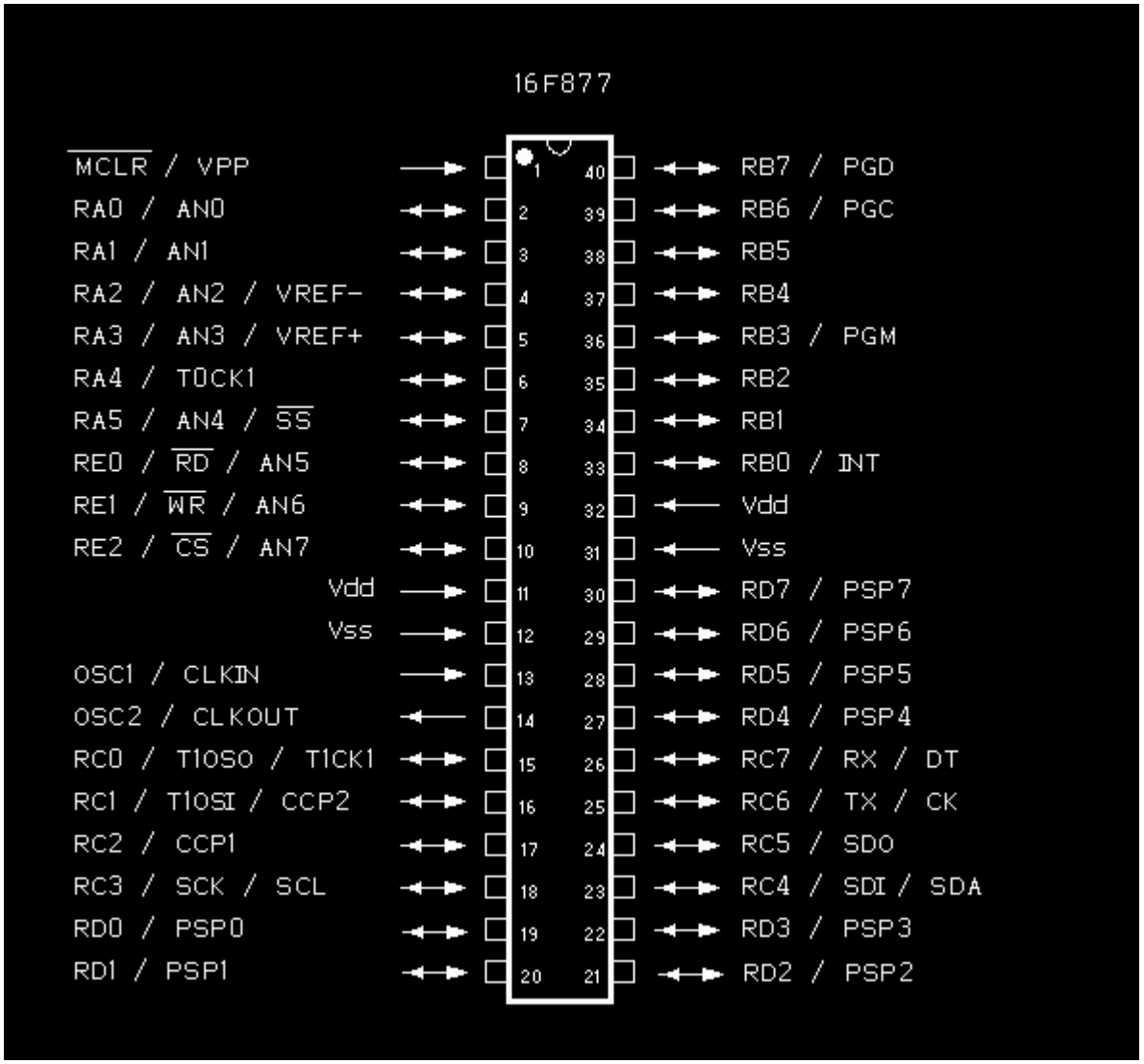


Example of a simplified input-output unit that provides communication with external world

- All input and output on the PIC is performed via the I/O ports.
- I/O ports are like RAM locations with wires leading from the bits to the pins of the microchip.

- PORTA is a 6-bit wide, bi-directional port.
- PORTB, PORTC, and PORTD are an 8-bit wide, bi-directional ports.
- PORTE is a 3-bit wide, bi-directional port.

- Pin functionality “overloaded” with other features
- Each pin can be individually configured for input or output.
- Configuration controlled through TRISx Register
  - Note that these registers are all in bank 1
- Port data controlled through PORTx register


16F877
MCLR / VPP
RA0 / AN0
RA1 / AN1
RA2 / AN2 / VREF-
RA3 / AN3 / VREF+
RA4 / T0CK1
RA5 / AN4 / SS
RE0 / RD / AN5
RE1 / WR / AN6
RE2 / CS / AN7
Vdd
Vss
OSC1 / CLKIN
OSC2 / CLKOUT
RC0 / T1OSO / T1CK1
RC1 / T1OSI / CCP2
RC2 / CCP1
RC3 / SCK / SCL
RD0 / PSP0
RD1 / PSP1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
RB7 / PGD
RB6 / PGC
RB5
RB4
RB3 / PGM
RB2
RB1
RB0 / INT
Vdd
Vss
RD7 / PSP7
RD6 / PSP6
RD5 / PSP5
RD4 / PSP4
RC7 / RX / DT
RC6 / TX / CK
RC5 / SDO
RC4 / SDI / SDA
RD3 / PSP3
RD2 / PSP2

PORTA (address 0x05 - Bank 0)

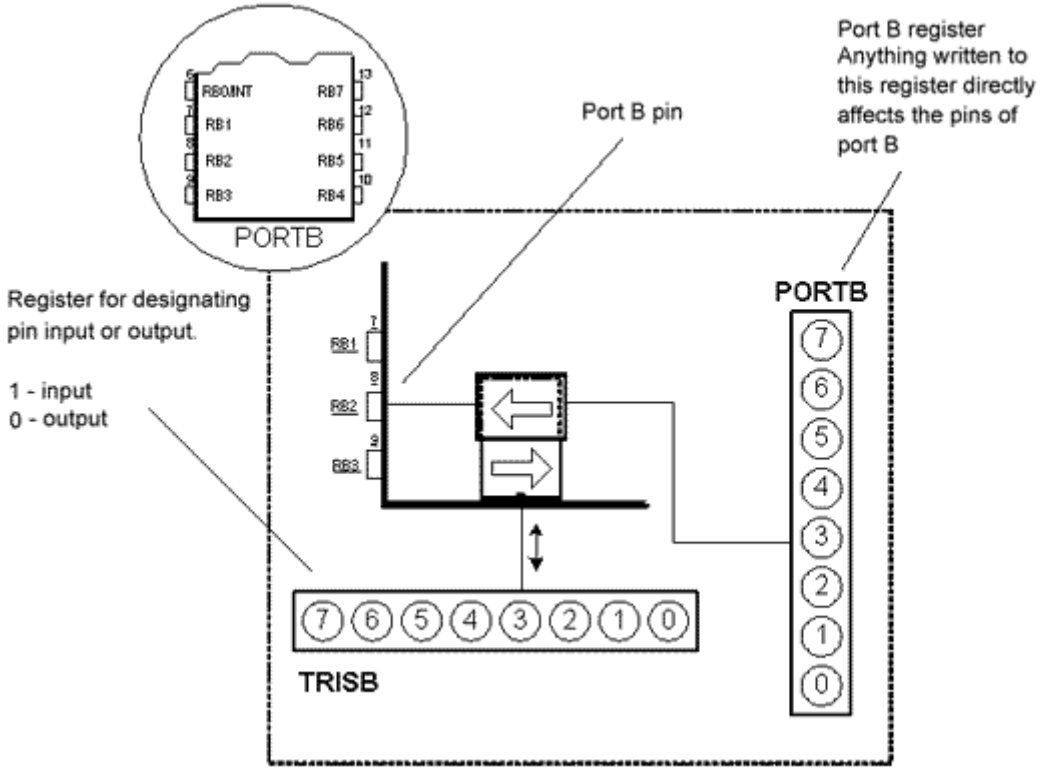
Bit	7	6	5	4	3	2	1	0
Name	-	-	RA5	RA4*	RA3	RA2	RA1	RA0
Hardware Pin Connection:			7	6	5	4	3	2

- I/O ports on a PIC are memory mapped.
  - This means that to read from or write to a port on a PIC the program must read from or write to a special RAM location.
  - To access PORTA on the 16F877 the program must access the RAM at address location 5.

- The I/O ports on the PIC can be used as either inputs or outputs.
- Configuring a port for input or output is done by setting or clearing the data direction register for the port.
- On the PIC each bit of each port has a data direction bit.
  - Therefore it is possible to set some bits within a port as inputs and others within the same port as outputs.
- For most ports, the I/O pin's direction (input or output) is controlled by the data direction register, called the TRIS register.
  - TRIS<x> controls the direction of PORT<x>.
  - A '1' in the TRIS bit corresponds to that pin being an input, while a '0' corresponds to that pin being an output.
  - An easy way to remember is that a '1' looks like an I (input) and a '0' looks like an O (output).
  - Note that these registers are all in bank 1

PORTA	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
TRISA	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**
DIRECTION	**OUT**	**OUT**	**IN**	**OUT**	**OUT**	**IN**	**OUT**	**IN**

To access TRISA on the 16F877 the program must access the RAM at address location 133 (this is actually 5 + 128).


RB0/INT
RB1
RB2
RB3
RB7
RB6
RB5
RB4
PORTB
Port B pin
Port B register
Anything written to
this register directly
affects the pins of
port B
Register for designating
pin input or output.
1 - input
0 - output
RB1
RB2
RB3
PORTB
7
6
5
4
3
2
1
0
7 6 5 4 3 2 1 0
TRISB

- PORTA is a little different. Since PORTA can be analog or digital, you also need to tell the PIC that PORTA is digital.
- This is done by writing 0x06 to ADCON1

ADCON1 (address 0x9F - Bank 1)

Bit	7	6	5	4	3	2	1	0
Value for binary I/O on PORTA	0	0	0	0	0	1	1	0

```
        call    Bank1
        clrf    0x85    ; TRISA = 0
        clrf    0x86    ; TRISB = 0
        clrf     0x87   ; TRISC = 0
        movlw   6
        movwf   0x9F    ; ADCON1 = 6
        call    Bank0
        clrf    7       ; PORTC = 0
        clrf    6       ; PORTB = 0
        clrf    5       ; PORTA = 0
L0:
        goto    L0      ; stop (infinite loop)
Bank0:
        bcf     3,5     ; bcf STATUS, RP0
        bcf     3,6     ; bcf STATUS, RP1
        return
Bank1:
        bsf     3,5      ; bsf STATUS, RP0
        bcf     3,6      ; bcf STATUS, RP0
        return
```

```
bsf     STATUS, RP0     ;Bank1
movlw   0x0F            ;Defining input and output pins
movwf   TRISB           ;Writing to TRISB register
bcf     STATUS, RP0     ;Bank0
bsf     PORTB, 4        ;PORTB <7:4>=0
bsf     PORTB, 5
bsf     PORTB, 6
bsf     PORTB, 7
```

The above example shows how pins 0, 1, 2, and 3 are designated input, and pins 4, 5, 6, and 7 for output, after which PORTB output pins are set to one.

initialize_portc:

```
clrf    PORTC       ; clear the data register to cause
                    ; outputs to be 0
bsf     STATUS,RP0  ; switch to bank 1
movlw   0xBF        ; each bit with a 1 is input,
                    ; otherwise output – Bit 6 output
movwf   TRISC       ; setup the port
bcf     STATUS,RP0  ; switch back to bank 0
return              ; exit
```

- The RA4 pin is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers.

- To add flexibility and functionality to a device, some pins are multiplexed with an alternate function(s).
- These functions depend on which peripheral features are on the device.
- In general, when a peripheral is functioning, that pin may not be used as a general purpose I/O pin.

- PORTA is a 6-bit I/O port (shared with A/D converter)
  - The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

- Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin.

- PORTD can be configured as an 8-bit wide microprocessor port (parallel slave port) by setting control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.
- 8-bit Parallel Slave Port - function allow another processor to read from a data buffer in the PIC.
- In Slave mode, it is asynchronously readable and writable by the external world through RD control input pin RE0/RD and WR control input pin RE1/WR.

Summary