The University of Texas at Arlington

Lecture 5
PIC I/O

CSE 3442/5442

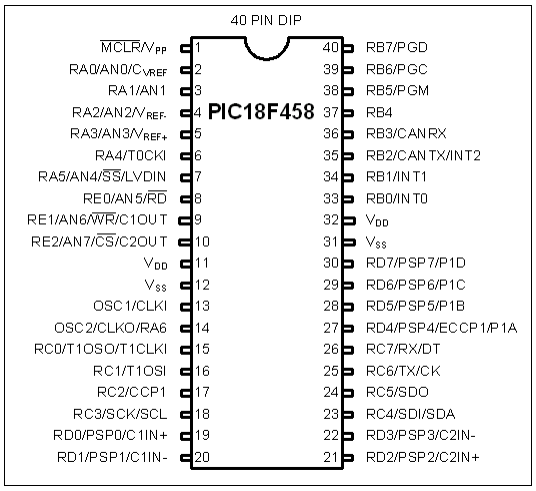
Embedded Systems I

Based heavily on slides by Dr. Roger Walker

Chapter 4 – PIC I/O PORT PROGRAMMING

- Ports are not only used for simple I/O, but also can be used other functions such as ADC, timers, interrupts, and serial communication pins. The following figure (Figure 4-1) shows the alternate functions for the PIC18F458 pins.

Figure 4-1 PICF458 Pin Diagram



PIC18F458/452 (40 Pins) has 5 ports, other Family Members Can Have More or Less Number

Table 4-1: Number of Ports in PIC18 Family Members

Pins	18-pin	28-pin	40-pin	64-pin	80-pin
Chip	PIC18F1220	PIC18F2220	PIC18F458	PIC18F6525	PIC18F8525
Port A	X	X	X	X	X
Port B	X	X	X	X	X
Port C		X	X	X	X
Port D			X	X	X
Port E			X	X	X
Port F				X	X
Port G				X	X
Port H				X	X
Port J				X	X
Port K					X
Port L					X

Note: X indicates that the port is available.

Number of Individual Port Pins

- For example, for the PIC18F458, Port A has 7 pins; Ports B, C, and D each have 8 pins; and Port E has only 3 pins.
- Each port has three SFRs associated with it. -- PORTx, TRISx (TRIState), and LATx (LATch).

Using PIC18F458 A-E Ports for Input/Output

- Each of the Ports A-E in the PIC18F458 can be used for input or output. The TRISx SFR is used solely for the purpose of making a given port an input or output port. To make a port an output, write Os to the TRISx register. Or, to output data to any of the pins of the Port B, first put Os into the TRISB register to make it an output port. Then send the data to the Port B SFR itself.

Addresses of SFR, PORTx, TRISx (TRIState), and LATx (LATch).

- See Table 4-2
- PORTA F80H
- PORTB F81H
-
-
-
- TRISA F92H
-

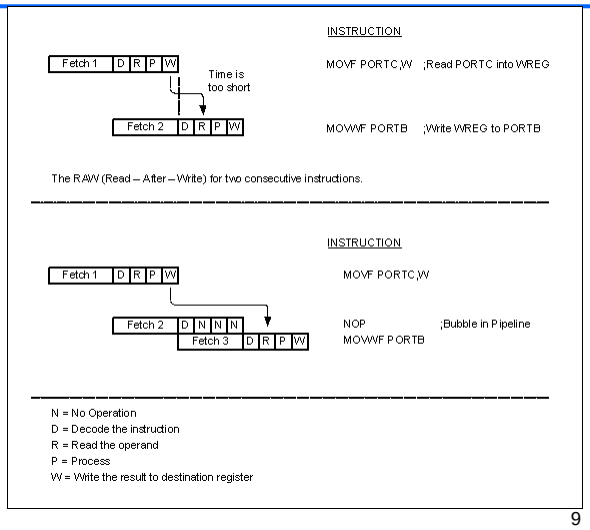
Example Using Port A as Input

In order to make all the bits of Port A an input, TRISA must be programmed by writing 1 to all the bits. In the next slide code, Port A is configured first as an input port by writing all 1 s to register TRISA, and then data is received from Port A and saved in some RAM location of the file registers:

```
MYREG    EQU 0X20    ;Program location (RAM)
MOVLW    B‘11111111’  ;All 1’s to WREG
MOVWF    TRISA  ;Port A as input port (1 for In)
MOVF   PORTA,W ;move from filereg of Port A to WREG
MOVWF  MYREG   ;save in fileReg of MYREG
```

Read After Write

- Need to add NOP between read from port and write to port, or use 4 byte MOVFF instruction



Read-Modify-Write

- Any instruction which performs a write operation actually does a read followed by a write operation. The BCF and BSF instructions, for example, read the register into the CPU, execute the bit operation, and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a BSF operation on bit5 of PORTB will cause all eight bits of PORTB to be read into the CPU. Then the BSF operation takes place on bit5 and PORTB is written to the output latches. If another bit of PORTB is used as a bi-directional I/O pin (e.g., bit0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and rewritten to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the input mode, no problem occurs. However, if bit0 is switched to an output, the content of the data latch may now be unknown.
- The LATx registers could/should be used in this case.

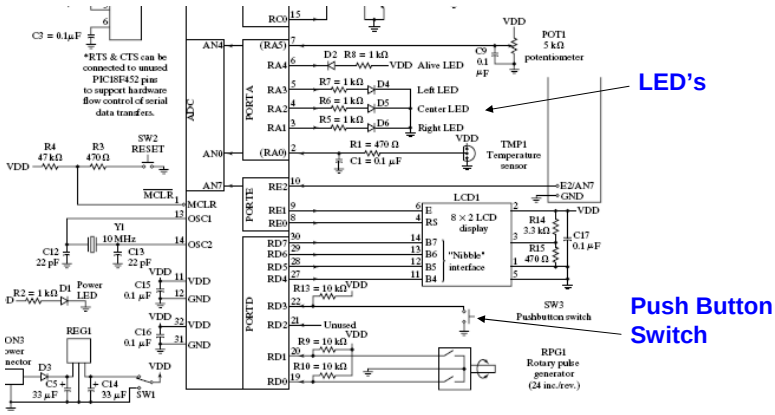
Register bit manipulation

- Bit set flag BSF filereg, bit
- Bit clear flag BCF filereg, bit
- Bit toggle flag BTF filereg, bit
- Bit test filereg skip next instruction if clear BTFSC filereg, bit
- Bit test filereg skip next instruction if set BTFSS filereg, bit
- Work for all file registers but especially helpful for PortA(RA0-RA5), PortB, PortC, PortD, and PortE(RE0-RE2)

Fan-out

- Current can flow in (pin at 0 level) and out (pin at 1 level) of port pins.
- This current is limited by the design of the IC.
- For PIC18 pins can drain a total of 8.5mA and source a total of 3mA.
- Fan-out is really the number of logic gates a pin can drive but is closely connected to the total current of pins. (total current / current drained by the input of the next logic gates)
- Arguably, for microcontrollers it is more important to remember the total current drawn (see LEDs driven in QwikFlash)

Example of Interfacing PIC to LED/Switches on QuickFlash



Example: Parallel Digital Output

Driving LCD Controllers

(textbook chapter 12,
PICBook chapter 7)

LCDs

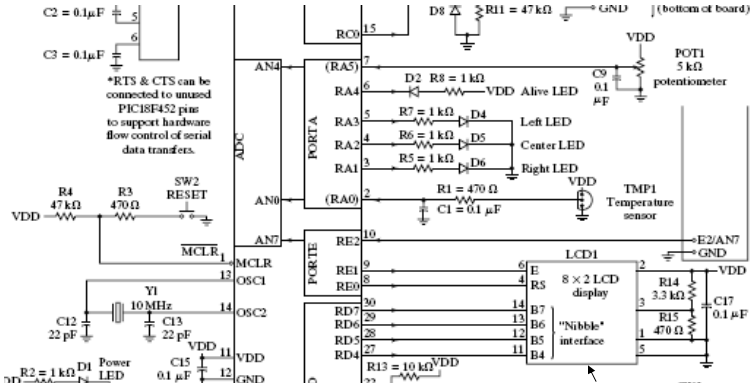
- Liquid Crystal Displays are frequently used with PICs
- Commonly used LCD modules have their own controller
- LCD modules commonly have parallel digital inputs where interfacing is done (there are some with serial interfaces as well)
- LCD modules usually go through some own initialization thus a couple hundred milliseconds after reset should pass before talking to them.
- LCD modules usually have to be first initialized before use.

LCD Common Pins

- Supply, “ground,” and LCD contrast voltage
- Register Select (RS); RS=0 for sending instructions (such as clear screen, or defining characters); RS=1 for sending data to be displayed.
- Enable (E) , essentially a clock input; a high-low transition will cause the LCD to latch in the data on the data pins.
- Data (D0-D7) or (D0-D3), the parallel interface pins – byte versus nibble interface
- Read/Write (R/W) direction of I/O (if used only as a display, “grounding” this is necessary)

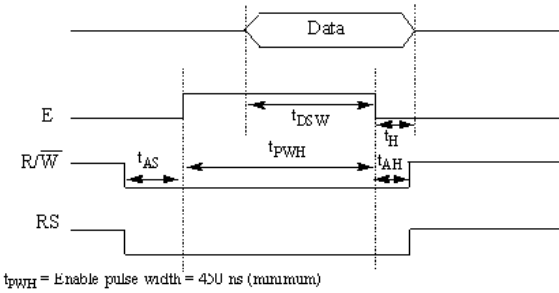
Hitachi HD44780 LCD controllers are by far the most used

Connections to QuickFlash



LCD Module – Hitachi HD44780 onboard controller

Typical LCD Timing for Displaying (Write)



t_{PWH} = Enable pulse width = 450 ns (minimum)
t_{DSW} = Data setup time = 195 ns (minimum)
t_H = Data hold time = 10 ns (minimum)
t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

Example Initialization (Nibble Interface)

1. Wait 100ms to make sure own initialization has occurred
2. RS=0, (all commands)
3. 3 times: E=1,D=3,E=0, wait
4. 2 times: E=1,D=2,E=0, wait (set nibble iface)
5. E=1,D=8,E=0, wait, E=1,D=0,E=0 (two line display)
6. E=1,D=1,E=0, wait, E=1,D=0,E=0 (clear display)
7. E=1,D=0xC,E=0, wait, E=1,D=0,E=0 (Turn off cursor, turn on display)
8. E=1,D=1,E=0, wait (auto cursor increment)

Cursor Positioning

- All commands (RS=0) where the MSB is set are cursor positioning commands
- Row 1 begins with 0x80 (1000 0000)
- Row 2 begins with 0XC0 (1100 0000)
- (Row 3 would be a continuation of row 1 and row 4 a continuation of row 2)
- Positions are counted left to right and auto increment can be enabled (no need for cursor positioning for short strings)

Hitachi HD44780 LCD controllers are by far the most used

Special Characters

- ASCII lower 128 characters are easy to display (just send ASCII codes) with a few exceptions
- Japanese characters at codes 0xa0 to 0xff
- Eight user defined characters 0x0 to 0x7
- All command codes (RS=0) with MSBs ’01’ are character generating commands
- 5x8 characters are then defined by sending their bitmaps (sending 8 bytes where upper three bits are always ignored)

Debugging

- LCDs (as they are displays) are a great tool for debugging embedded code.
- Of course we need to assume that the microcontroller works
- Displaying variables and port statuses can be very helpful