The University of Texas at Arlington

Lecture 5
PIC I/O and LCD Control

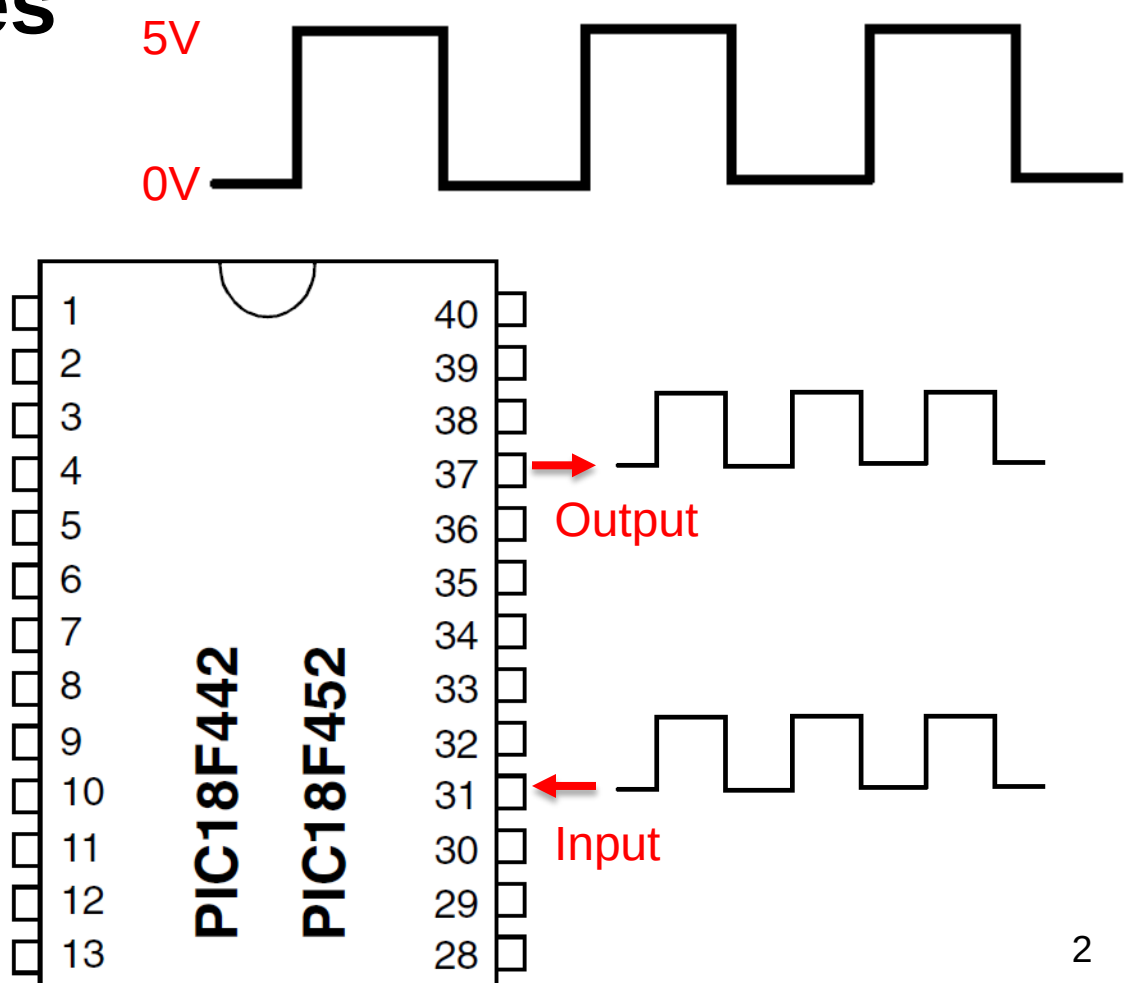
CSE 3442/5442

Embedded Systems I

Based heavily on slides by Dr. Gergely Záruba and Dr. Roger Walker

Digital Input/Output

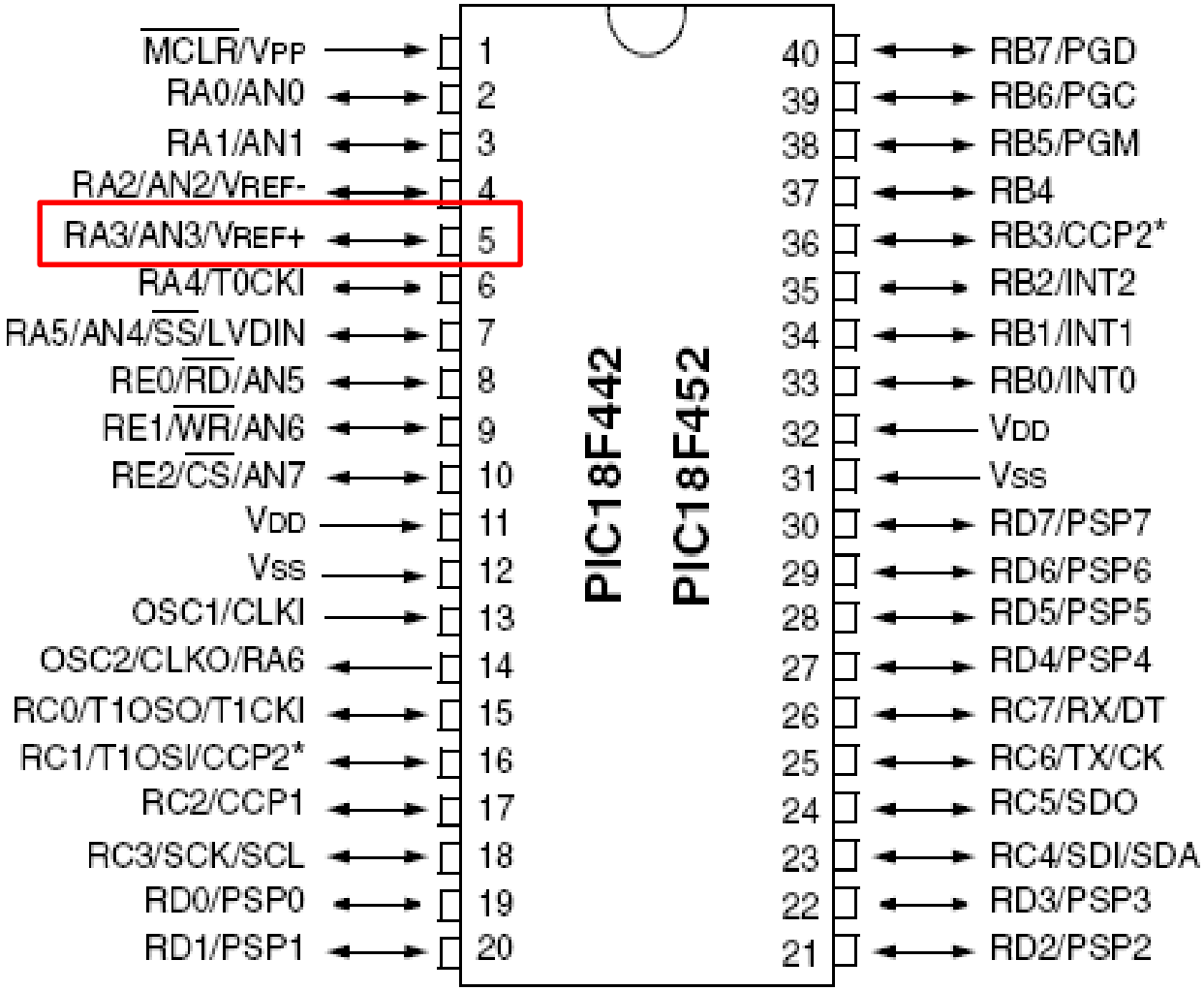
- **Only two states**
  - ON / OFF
  - HIGH / LOW
  - 1 / 0
  - 5V / 0V
  - 3.3V / 0V
  - etc.



Chapter 4 – PIC I/O PORT PROGRAMMING

- Ports are not only used for simple I/O, but also can be used other functions
  - ADC (analog-to-digital conversion)
  - Timers
  - Oscillator Input
  - Interrupts
  - Serial communication
  - Capturing and Generating PWM Signals
  - Programming the PIC

PIC18F452 Pin Diagram



Pin 5: RA3/AN3/Vref+

						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19			
RA0				I/O	TTL	Digital I/O.
AN0				I	Analog	Analog input 0.
RA1/AN1	3	4	20			
RA1				I/O	TTL	Digital I/O.
AN1				I	Analog	Analog input 1.
RA2/AN2/VREF-	4	5	21			
RA2				I/O	TTL	Digital I/O.
AN2				I	Analog	Analog input 2.
VREF-				I	Analog	A/D Reference Voltage (Low) input.
RA3/AN3/VREF+	5	6	22			
RA3				I/O	TTL	Digital I/O.
AN3				I	Analog	Analog input 3.
VREF+				I	Analog	A/D Reference Voltage (High) input.
RA4/T0CKI	6	7	23			
RA4				I/O	ST/OD	Digital I/O. Open drain when configured as output.
T0CKI				I	ST	Timer0 external clock input.
RA5/AN4/$\overline{SS}$/LVDIN	7	8	24			
RA5				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
$\overline{SS}$				I	ST	SPI Slave Select input.
LVDIN				I	Analog	Low Voltage Detect Input.
RA6						(See the OSC2/CLKO/RA6 pin.)

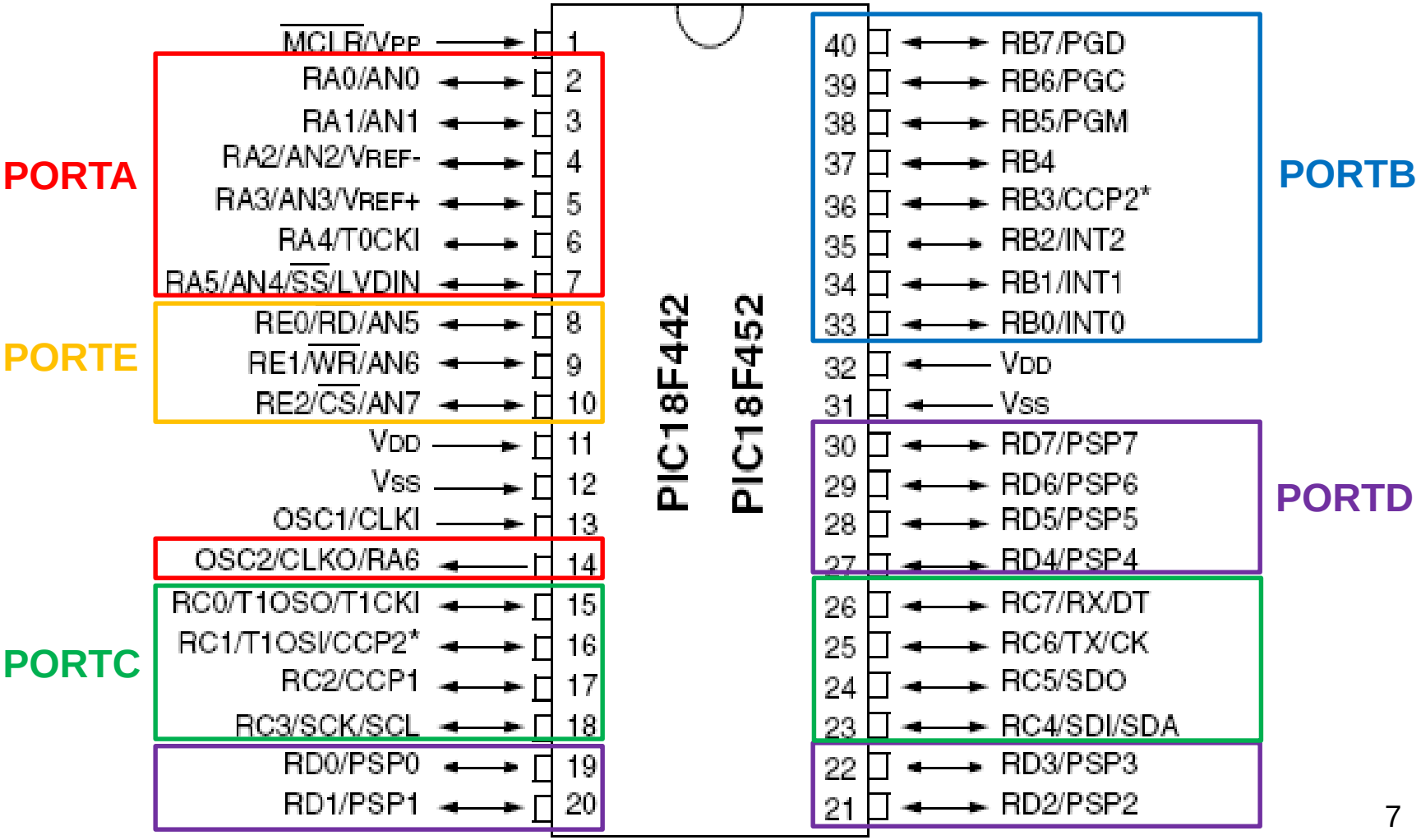
PIC18F452 (40 Pins) has 5 ports, other Family Members Can Have More or Less

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.

PIC18F452 Pin Diagram
5 Ports

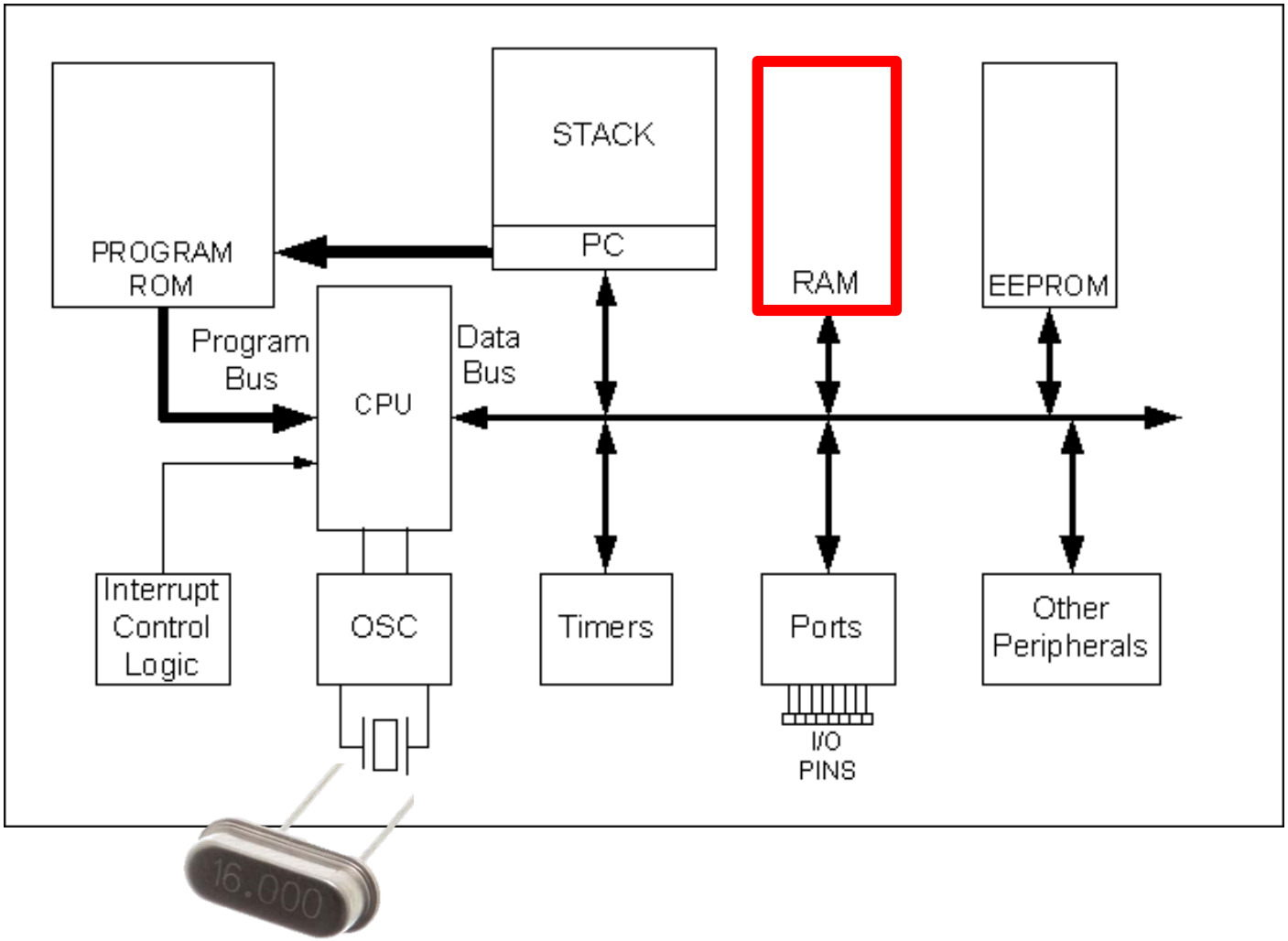


Number of Individual Port Pins

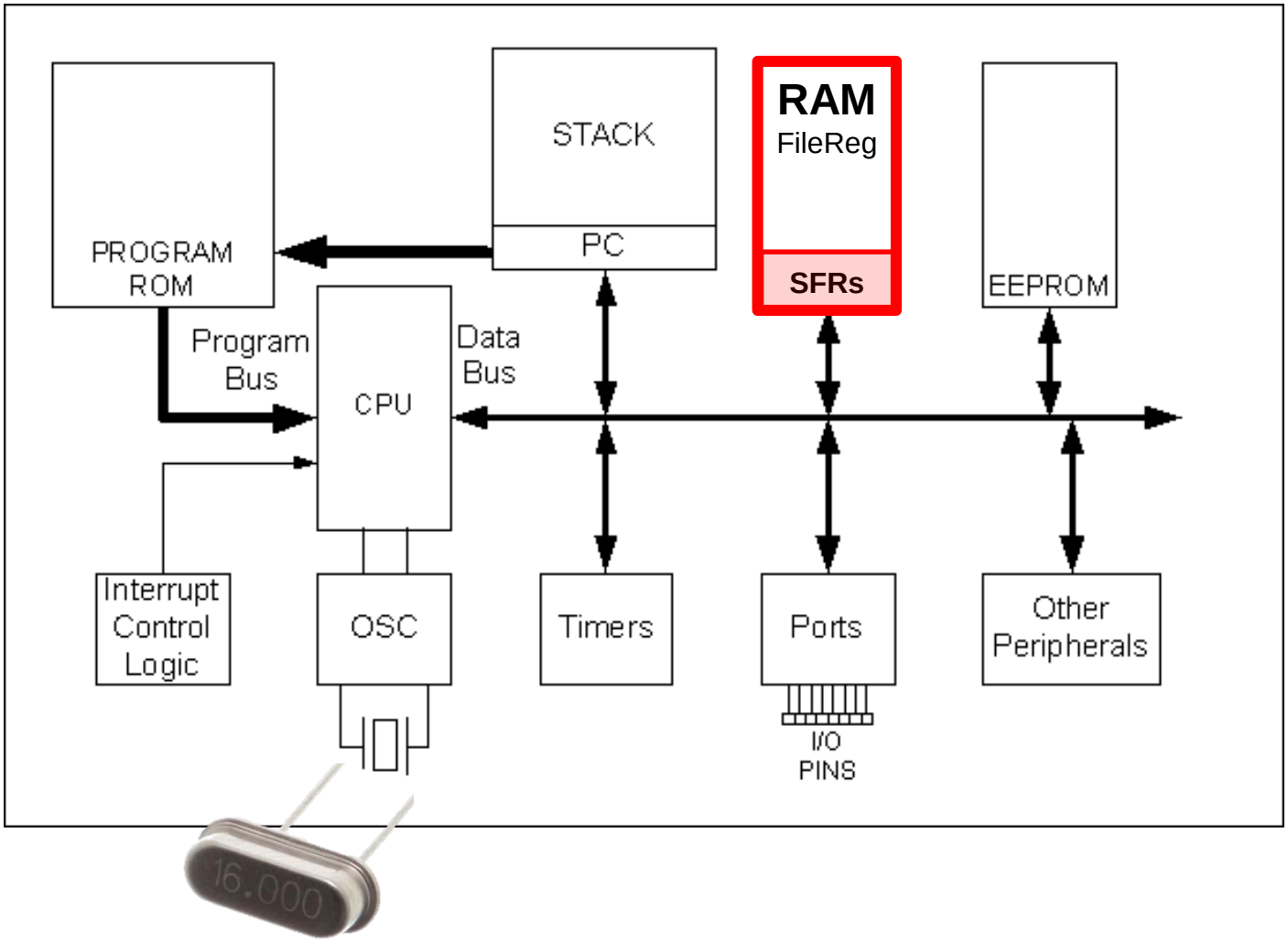
- For example, the PIC18F452
  - Port A has 7 pins
  - Ports B, C, and D each have 8 pins
  - Port E has only 3 pins

    →34 total digital IO pins
- Each port has three SFRs associated
  - **PORTx**
  - **TRISx (TRIState)**
  - **LATx (LATch)**

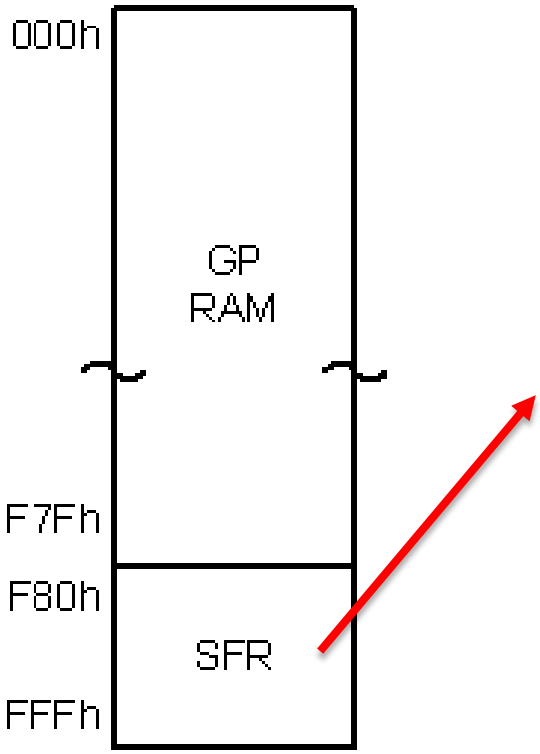
SFRs in the File Registers



SFRs in the File Registers



SFRs in the File Registers



F80h	PORTA	FA0h	PIE2	FC0h	----	FE0h	BSR
F81h	PORTB	FA1h	PIR2	FC1h	ADCON1	FE1h	FSR1L
F82h	PORTC	FA2h	IPR2	FC2h	ADCON0	FE2h	FSR1H
F83h	PORTD	FA3h	----	FC3h	ADRESL	FE3h	PLUSW1 *
F84h	PORTE	FA4h	----	FC4h	ADRESH	FE4h	PREINC1 *
F85h	----	FA5h	----	FC5h	SSPCON2	FE5h	POSTDEC1 *
F86h	----	FA6h	----	FC6h	SSPCON1	FE6h	POSTINC1 *
F87h	----	FA7h	----	FC7h	SSPSTAT	FE7h	INDF1 *
F88h		FA8h	----	FC8h	SSPADD	FE8h	WREG
F89h	LATA	FA9h	----	FC9h	SSPBUF	FE9h	FSR0L
F8Ah	LATB	FAAh	----	FCAh	T2CON	FEAh	FSR0H
F8Bh	LATC	FABh	RCSTA	FCBh	PR2	FEBh	PLUSW0 *
F8Ch	LATD	FACh	TXSTA	FCCh	TMR2	FECh	PREINC0 *
F8Dh	LATE	FADh	TXREG	FCDh	T1CON	FEDh	POSTDEC0 *
F8Eh	----	FAEh	RCREG	FCEh	TMR1L	FEEh	POSTINC0 *
F8Fh	----	FAFh	SPBRG	FCFh	TMR1H	FEFh	INDF0 *
F90h	----	FB0h	----	FD0h	RCON	FF0h	INTCON3
F91h	----	FB1h	T3CON	FD1h	WDTCON	FF1h	INTCON2
F92h	TRISA	FB2h	TMR3L	FD2h	LVDCON	FF2h	INTCON
F93h	TRISB	FB3h	TMR3H	FD3h	OSCCON	FF3h	PRODL
F94h	TRISC	FB4h	----	FD4h	----	FF4h	PRODH
F95h	TRISD	FB5h	----	FD5h	T0CON	FF5h	TABLAT
F96h	TRISE	FB6h	----	FD6h	TMR0L	FF6h	TBLPTRL
F97h	----	FB7h	----	FD7h	TMR0H	FF7h	TBLPTRH
F98h	----	FB8h	----	FD8h	STATUS	FF8h	TBLPTRU
F99h	----	FB9h	----	FD9h	FSR2L	FF9h	PCL
F9Ah	----	FBAh	CCP2CON	FDAh	FSR2H	FFAh	PCLATH
F9Bh	----	FBBh	CCPR2L	FDBh	PLUSW2 *	FFBh	PCLATU
F9Ch	----	FBCh	CCPR2H	FDCh	PREINC2 *	FFCh	STKPTR
F9Dh	PIE1	FBDh	CCP1CON	FDDh	POSTDEC2 *	FFDh	TOSL
F9Eh	PIR1	FBEh	CCPR1L	FDEh	POSTINC2 *	FFEh	TOSH
F9Fh	IPR1	FBFh	CCPR1H	FDFh	INDF2 *	FFFh	TOSU

TRISx SFR

- Each of the Ports A-E in the PIC18F452 can be used for **input** or **output**
  - **TRISx** is used solely for the purpose of making a given port an **input** or **output** port
    - **TRISx bit = 0** → PORTx bit is an **OUTPUT**
      - Can now **write to** the PORTx bit(s)
    - **TRISx bit = 1** → PORTx bit is an **INPUT**
      - Can now **read in** from the PORTx bit(s)
  - Can set I/O bit-by-bit or whole TRIS byte at once

PORTx and LATx SFRs

- **PORTx**
  - For reading input coming into the PIC
    - Digital High (1) or Low (0)
  - For writing output from the PIC
    - Writing a 1 → pin is High, 0 → pin is Low
- **LATx**
  - For writing output from the PIC
    - Writing a 1 → pin is High, 0 → pin is Low
- Point of the Latch??

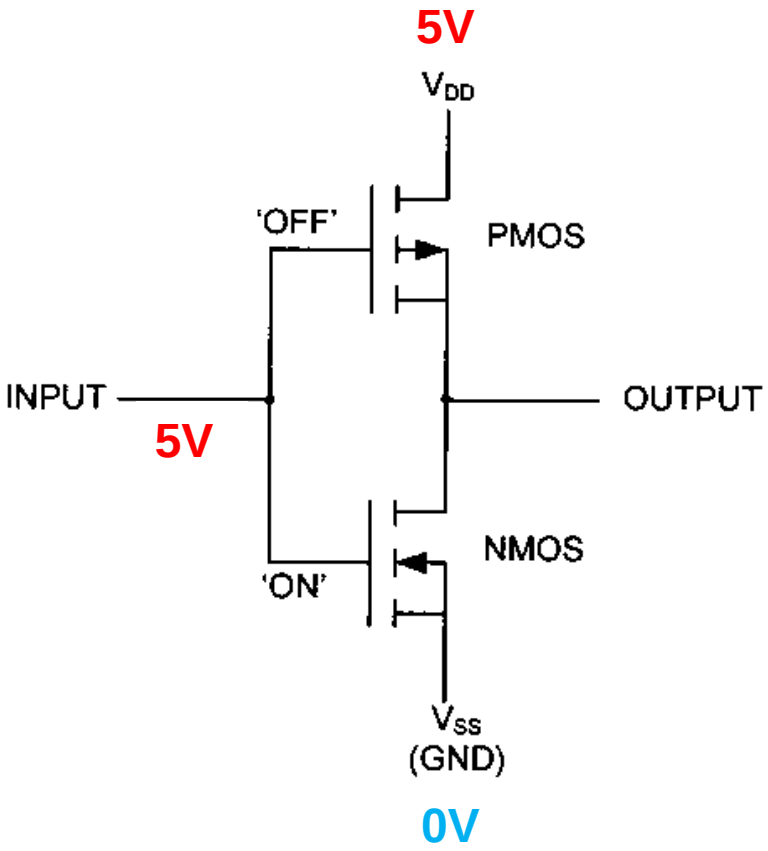
PORTB Example

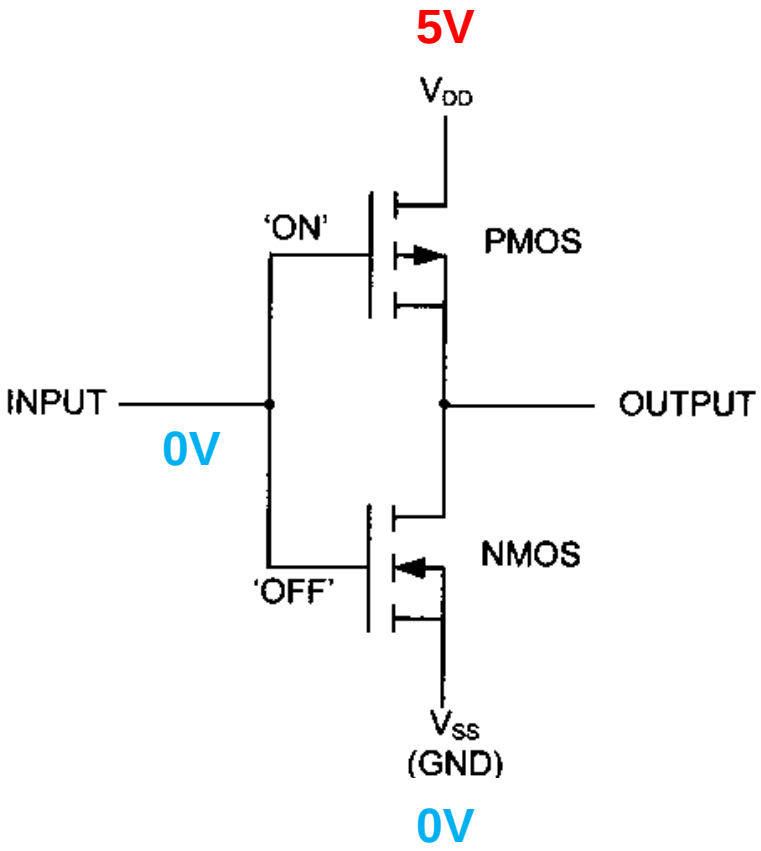
TABLE 9-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
LATB	LATB Data Output Register								xxxx xxxx	uuuu uuuu
TRISB	PORTB Data Direction Register								1111 1111	1111 1111
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1	1111 -1-1
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	11-0 0-00

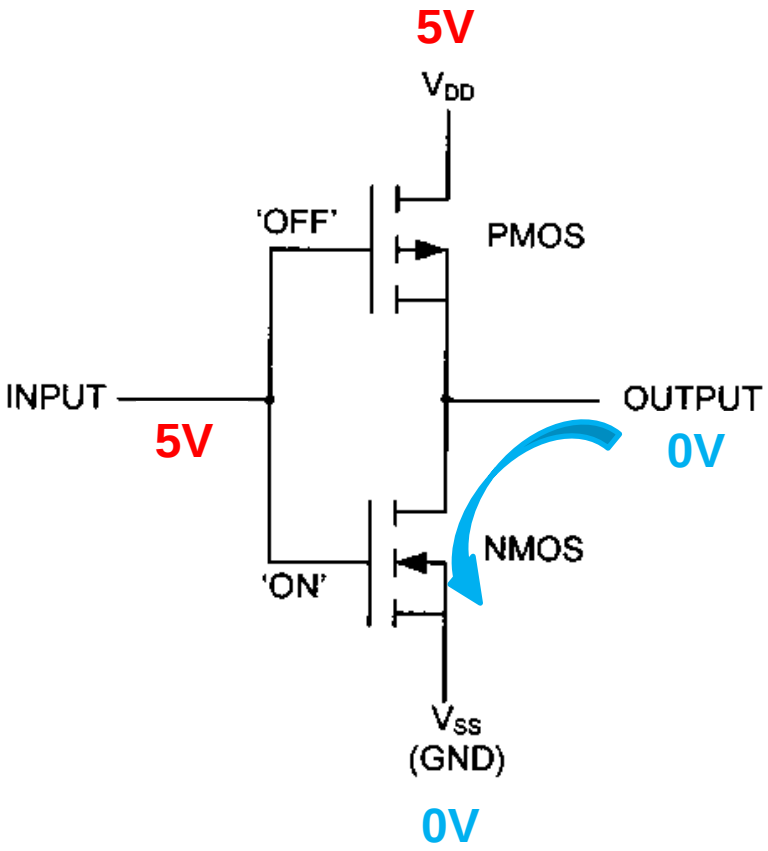
Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

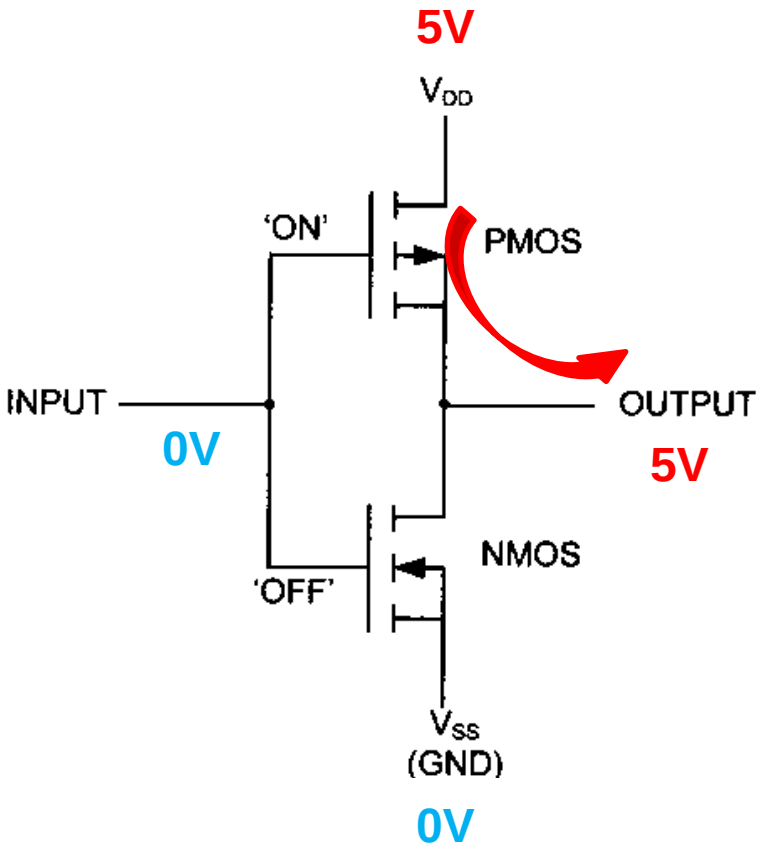
N and P Transistors (MOSFET Logic)



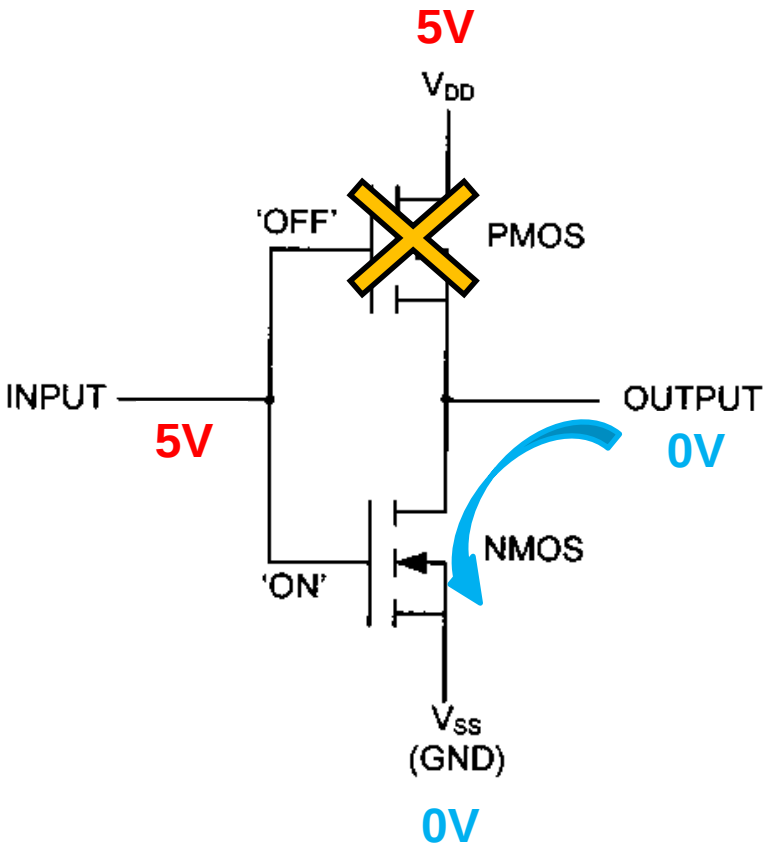


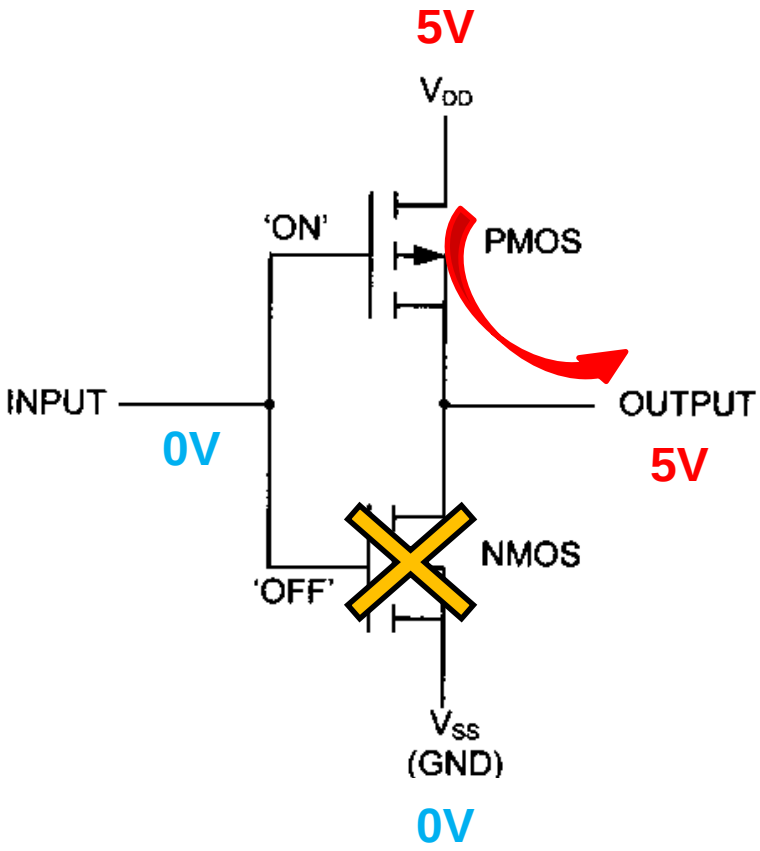
N and P Transistors (MOSFET Logic)





N and P Transistors (MOSFET Logic)





Outputting a 0

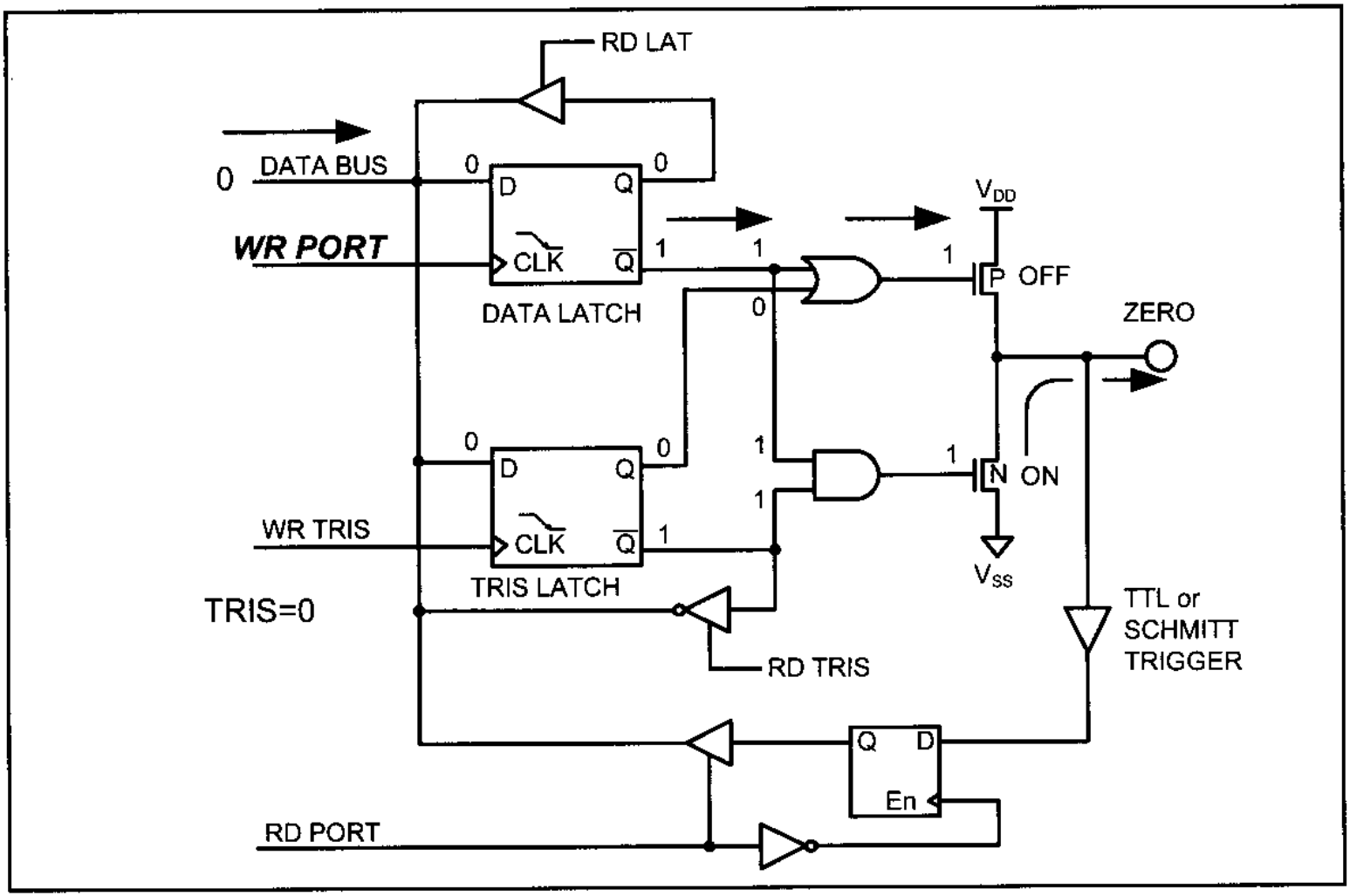
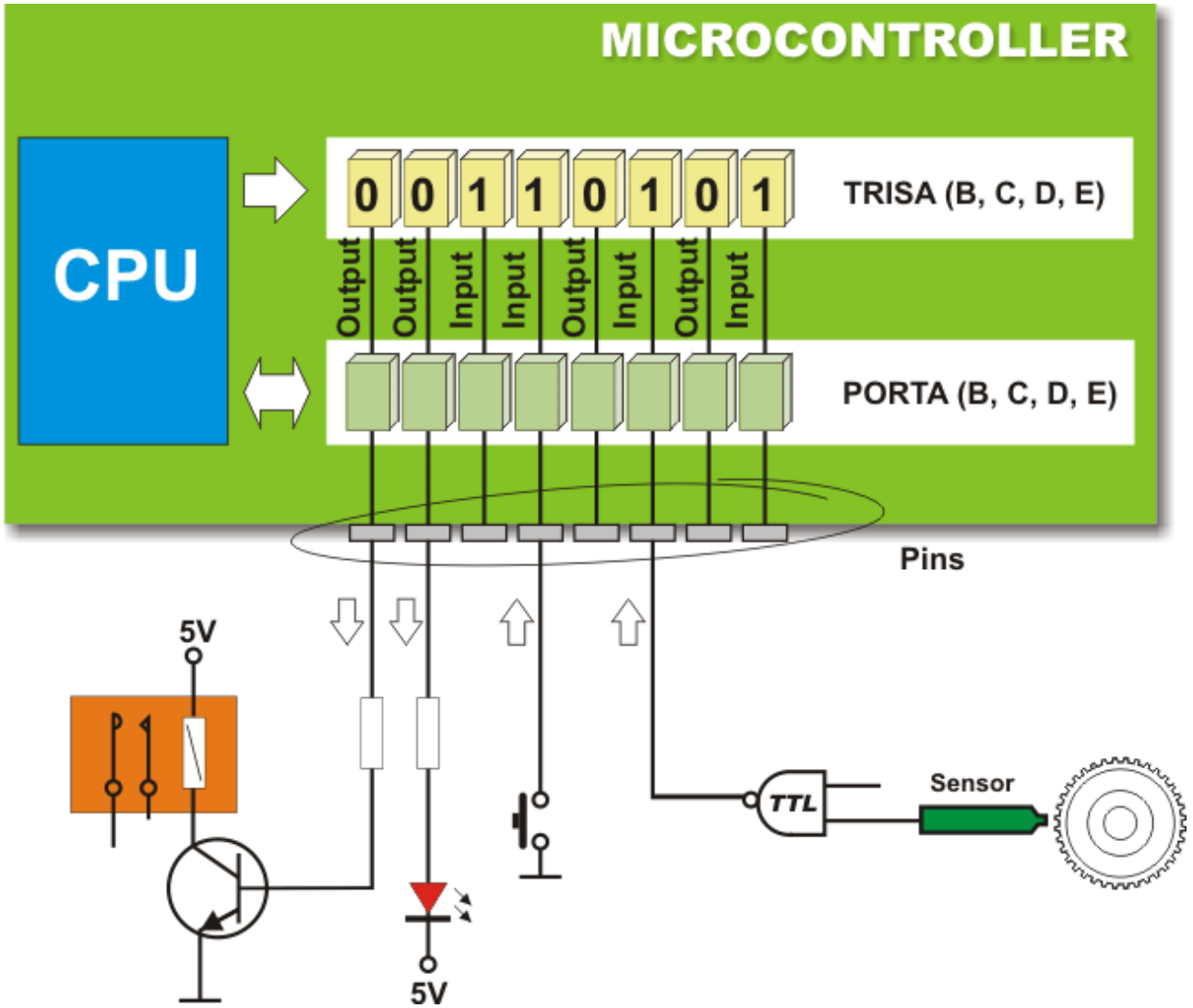


Figure 4-3. Outputting (Writing) 0 to a Pin in the PIC18

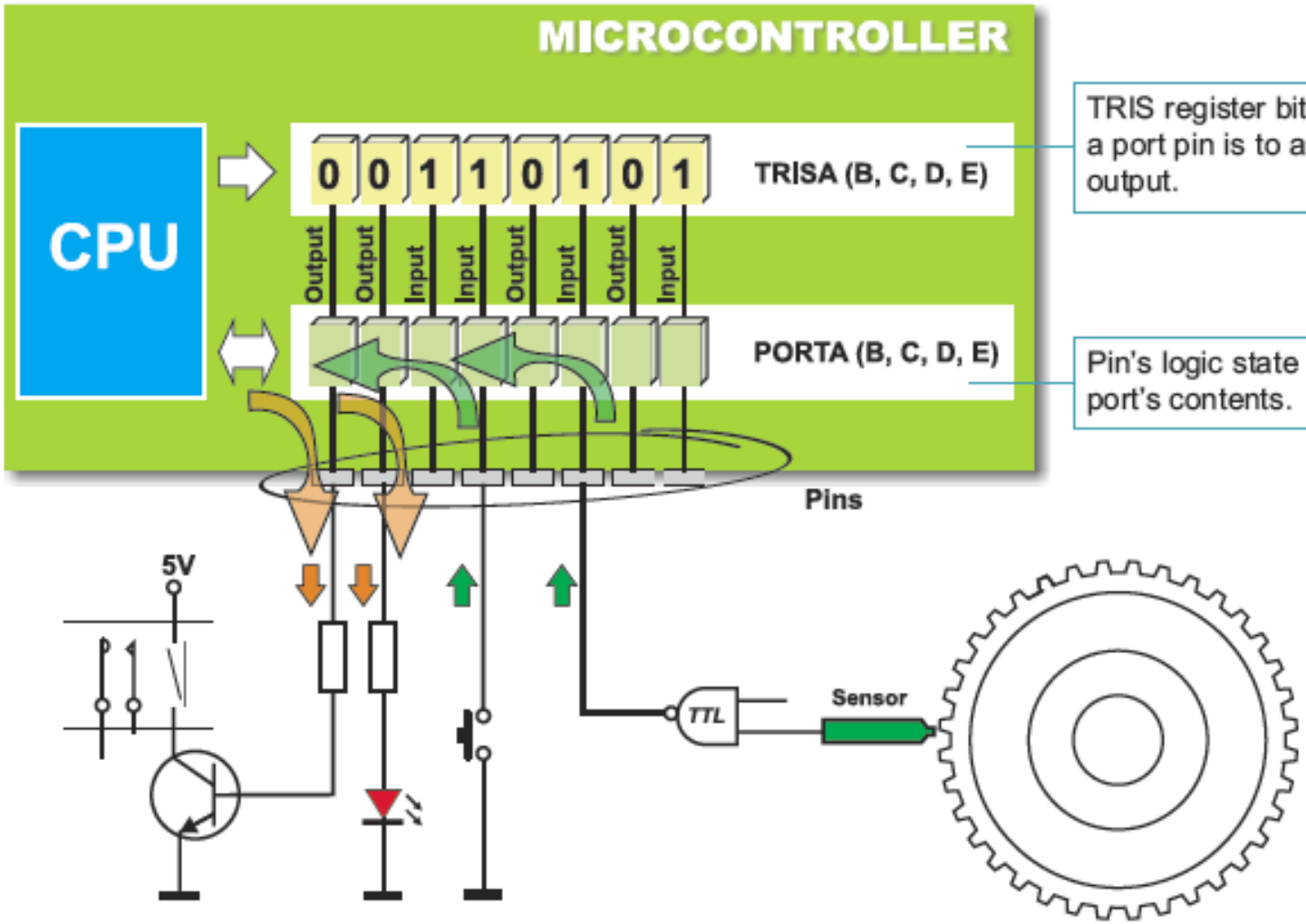
MikroElektronika (img source)

http://learn.mikroe.com/ebooks/picbasicprogramming/chapter/input-output-ports/

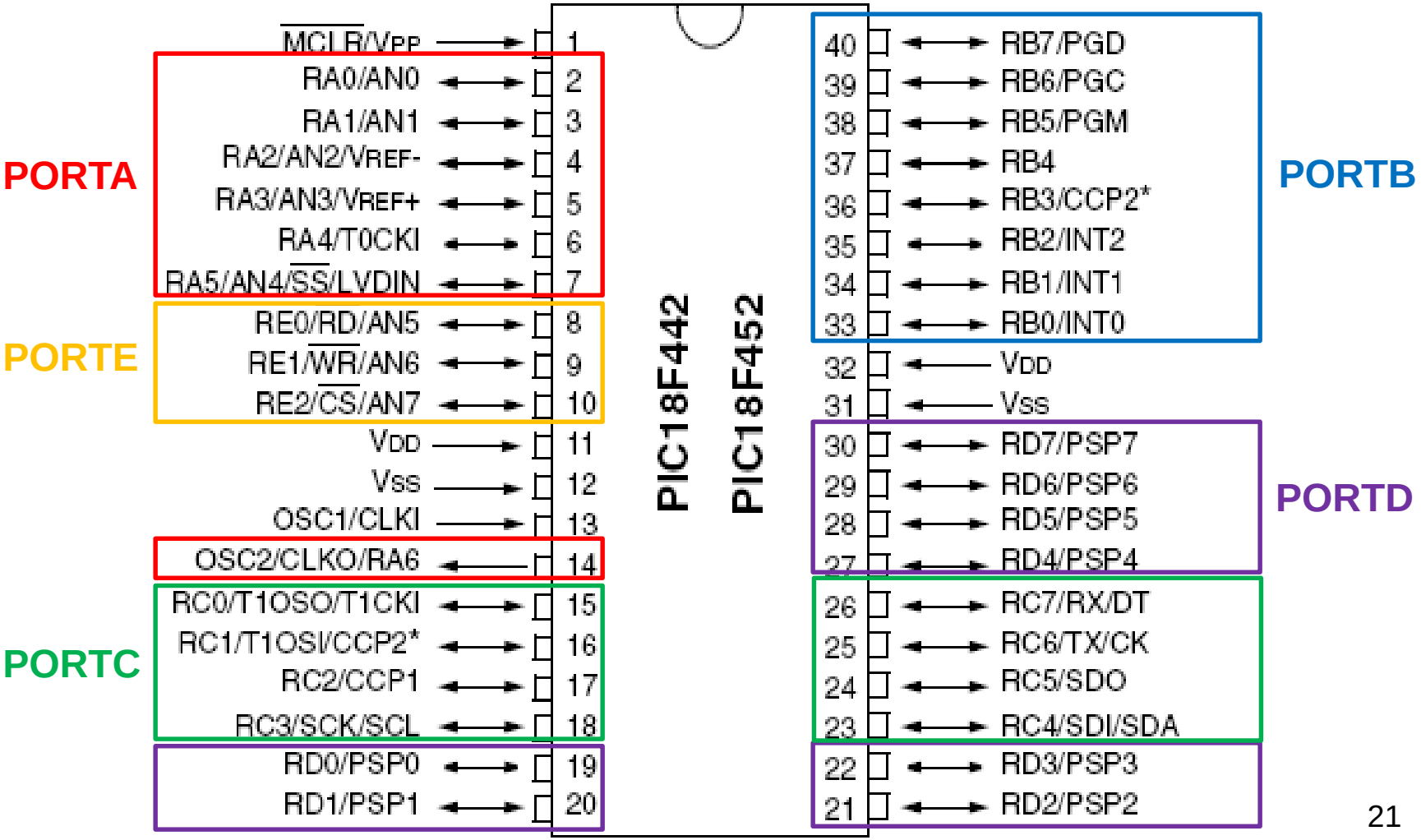


MikroElektronika (img source)

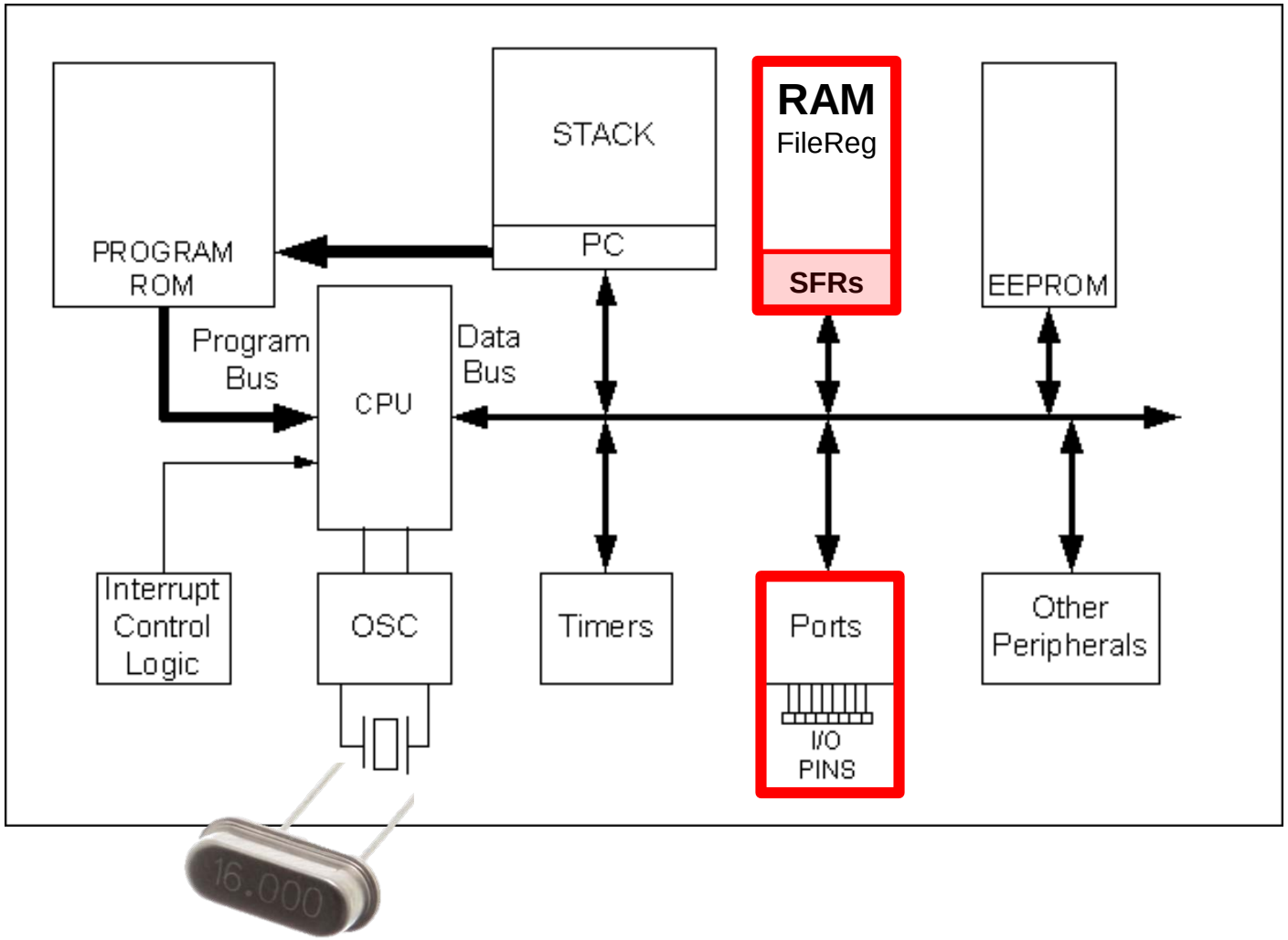
http://learn.mikroe.com/ebooks/picbasicprogramming/chapter/input-output-ports/



PIC18F452 Pin Diagram
5 Ports



PORT/TRIS Functionality is Mapped to the SFRs



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

CSE@UTA

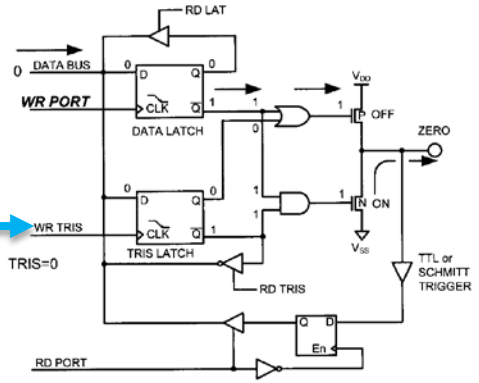
TABLE 4-1: SPECIAL FUNCTION REGISTER MAP

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(3)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(3)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(3)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(3)	FBCh	CCPR2H	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2(3)	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE(2)
FF5h	TABLAT	FD5h	T0CON	FB5h	—	F95h	TRISD(2)
FF4h	PRODH	FD4h	—	FB4h	—	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0(3)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0(3)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0(3)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(2)
FECh	PREINC0(3)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(2)
FEBh	PLUSW0(3)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—
FE7h	INDF1(3)	FC7h	SSPSTAT	FA7h	EECON2	F87h	—
FE6h	POSTINC1(3)	FC6h	SSPCON1	FA6h	EECON1	F86h	—
FE5h	POSTDEC1(3)	FC5h	SSPCON2	FA5h	—	F85h	—
FE4h	PREINC1(3)	FC4h	ADRESH	FA4h	—	F84h	PORTE(2)
FE3h	PLUSW1(3)	FC3h	ADRESL	FA3h	—	F83h	PORTD(2)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

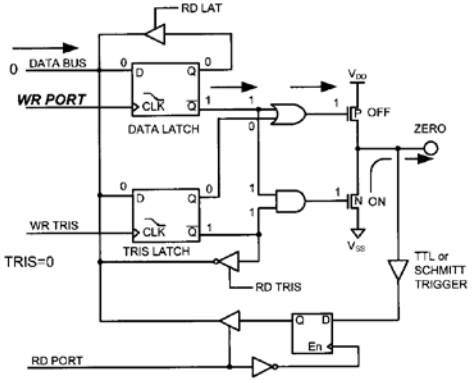
Accessing SFRs in .ASM
PORTB as an OUTPUT

CSE@UTA

F98h	—
F97h	—
F96h	TRISE(2)
F95h	TRISD(2)
F94h	TRISC
F93h	TRISB
F92h	TRISA
F91h	—
F90h	—
F8Fh	—
F8Eh	—
F8Dh	LATE(2)
F8Ch	LATD(2)
F8Bh	LATC
F8Ah	LATB
F89h	LATA
F88h	—
F87h	—
F86h	—
F85h	—
F84h	PORTE(2)
F83h	PORTD(2)
F82h	PORTC
F81h	PORTB
F80h	PORTA



...

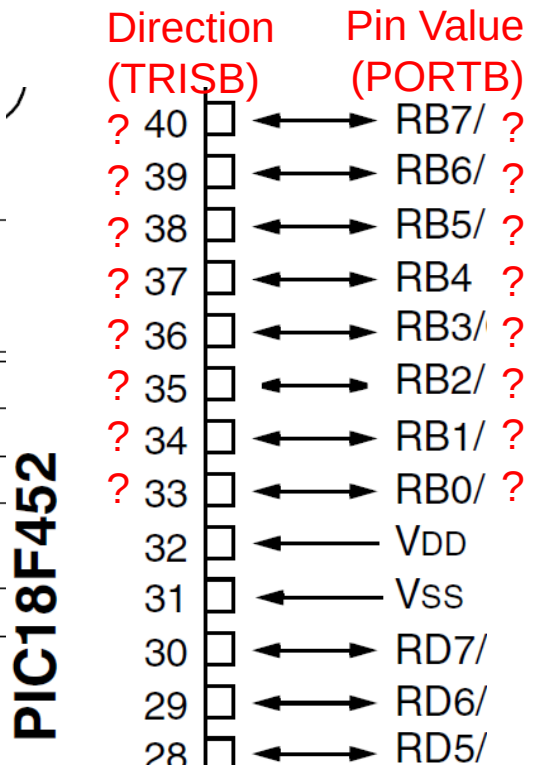



```
PORTB   EQU   0XF81          ;in .H header file
TRISB   EQU   0XF93          ;in .H header file

ORG 0x00

MOVLW   0                    ;All 0's to WREG
MOVWF   TRISB                ;All of PORTB is an OUTPUT

MOVLW   B'10101010'
MOVWF   PORTB                ;Write 1/0 to PORTB pins
```

PORTB Example

TABLE 9-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

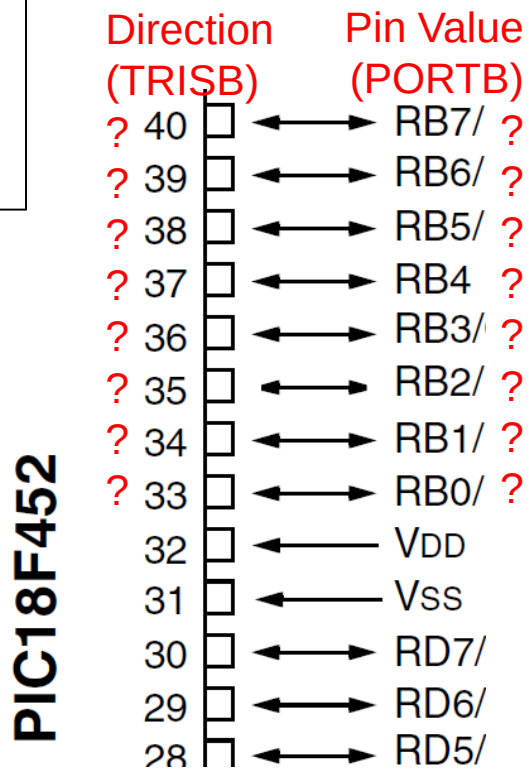
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
LATB	LATB Data Output Register							
TRISB	PORTB Data Direction Register							
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF



Accessing SFRs in .ASM
PORTB as an OUTPUT

```
PORTB   EQU   0XF81    ;in .H header file
TRISB   EQU   0XF93    ;in .H header file

ORG 0x00

MOVLW   0              ;All 0's to WREG
MOVWF   TRISB          ;PORTB is an OUTPUT

MOVLW   B'10101010'
MOVWF   PORTB          ;Write 1/0 to PORTB pins
```

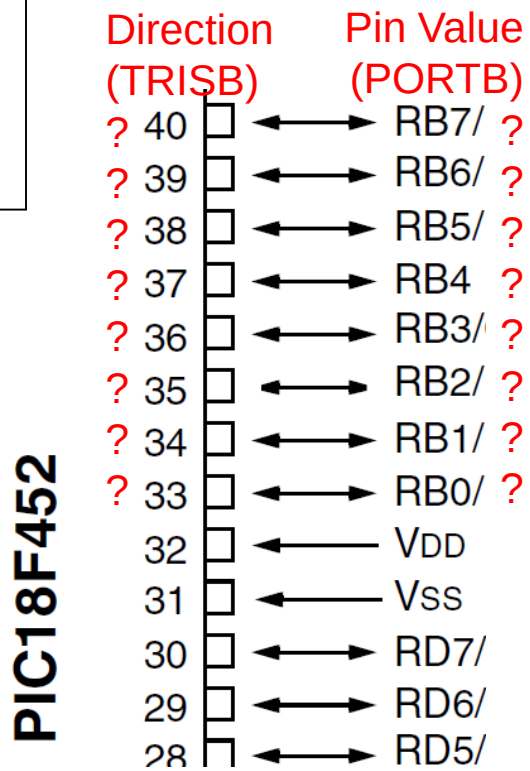
WREG = ?
TRISB = ?
PORTB = ?



Accessing SFRs in .ASM
PORTB as an OUTPUT

```
PORTB   EQU   0XF81    ;in .H header file
TRISB   EQU   0XF93    ;in .H header file

ORG 0x00

MOVLW   0              ;All 0's to WREG
MOVWF   TRISB          ;PORTB is an OUTPUT

MOVLW   B'10101010'
MOVWF   PORTB          ;Write 1/0 to PORTB pins
```

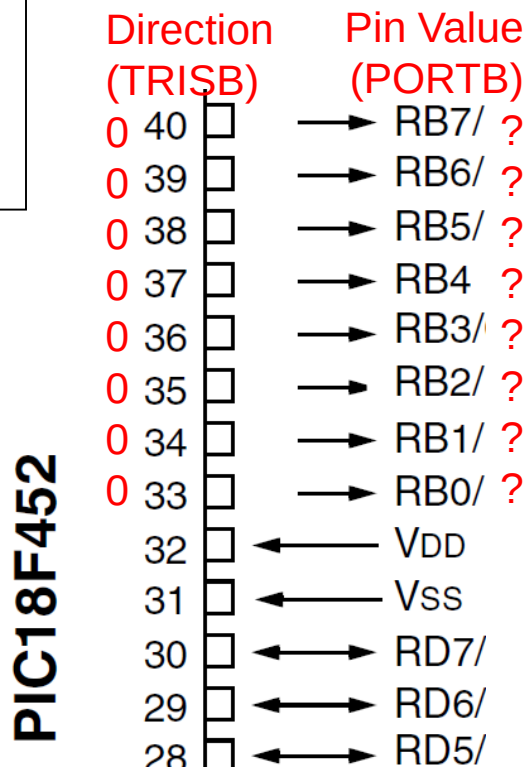
WREG = 0000 0000
TRISB = ?
PORTB = ?



Accessing SFRs in .ASM
PORTB as an OUTPUT

```
PORTB   EQU   0XF81    ;in .H header file
TRISB   EQU   0XF93    ;in .H header file

ORG 0x00

MOVLW   0              ;All 0's to WREG
MOVWF   TRISB          ;PORTB is an OUTPUT

MOVLW   B'10101010'
MOVWF   PORTB          ;Write 1/0 to PORTB pins
```

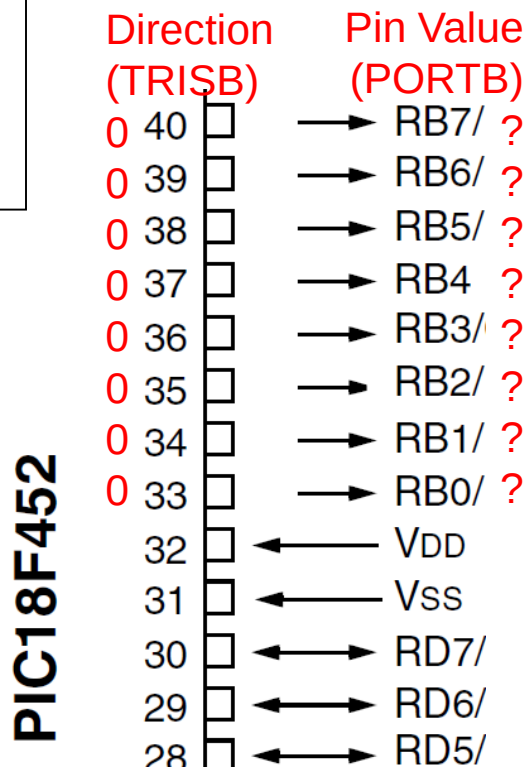
WREG = 0000 0000
TRISB = 0000 0000
PORTB = ?



Accessing SFRs in .ASM
PORTB as an OUTPUT

```
PORTB   EQU   0XF81    ;in .H header file
TRISB   EQU   0XF93    ;in .H header file

ORG 0x00

MOVLW   0              ;All 0's to WREG
MOVWF   TRISB          ;PORTB is an OUTPUT

MOVLW   B'10101010'
MOVWF   PORTB          ;Write 1/0 to PORTB pins
```

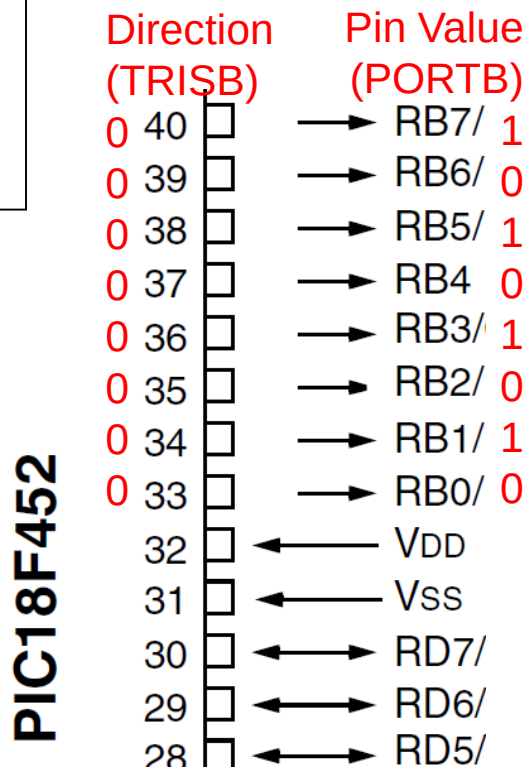
WREG = 1010 1010
TRISB = 0000 0000
PORTB = ?



Accessing SFRs in .ASM
PORTB as an OUTPUT

```
PORTB   EQU   0XF81    ;in .H header file
TRISB   EQU   0XF93    ;in .H header file

ORG 0x00

MOVLW   0              ;All 0's to WREG
MOVWF   TRISB          ;PORTB is an OUTPUT

MOVLW   B'10101010'
MOVWF   PORTB          ;Write 1/0 to PORTB pins
```

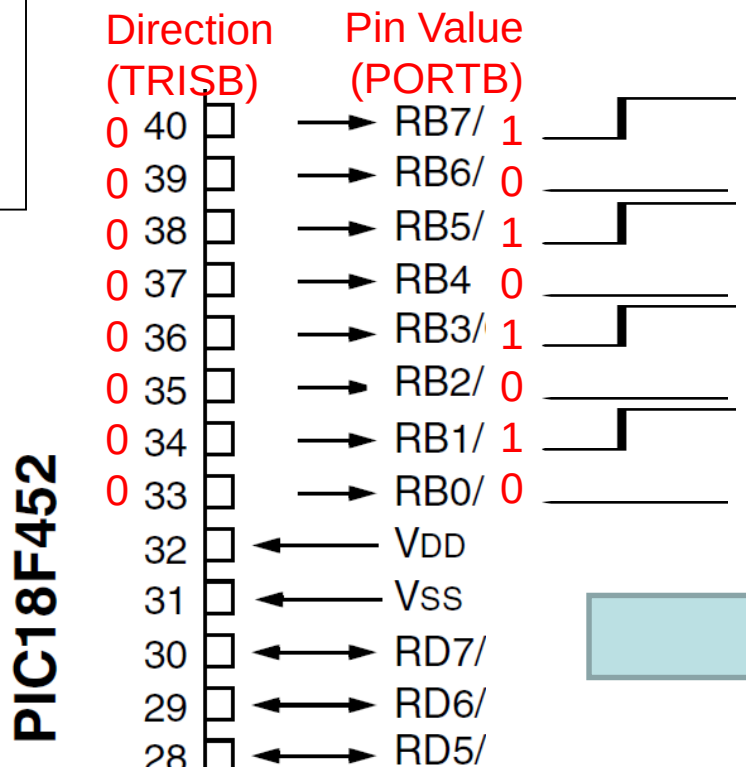
WREG = 1010 1010
TRISB = 0000 0000
PORTB = 1010 1010



Accessing SFRs in .ASM
PORTB as an OUTPUT

```
PORTB   EQU   0XF81    ;in .H header file
TRISB   EQU   0XF93    ;in .H header file

ORG 0x00

MOVLW   0              ;All 0's to WREG
MOVWF   TRISB          ;PORTB is an OUTPUT

MOVLW   B'10101010'
MOVWF   PORTB          ;Write 1/0 to PORTB pins
```

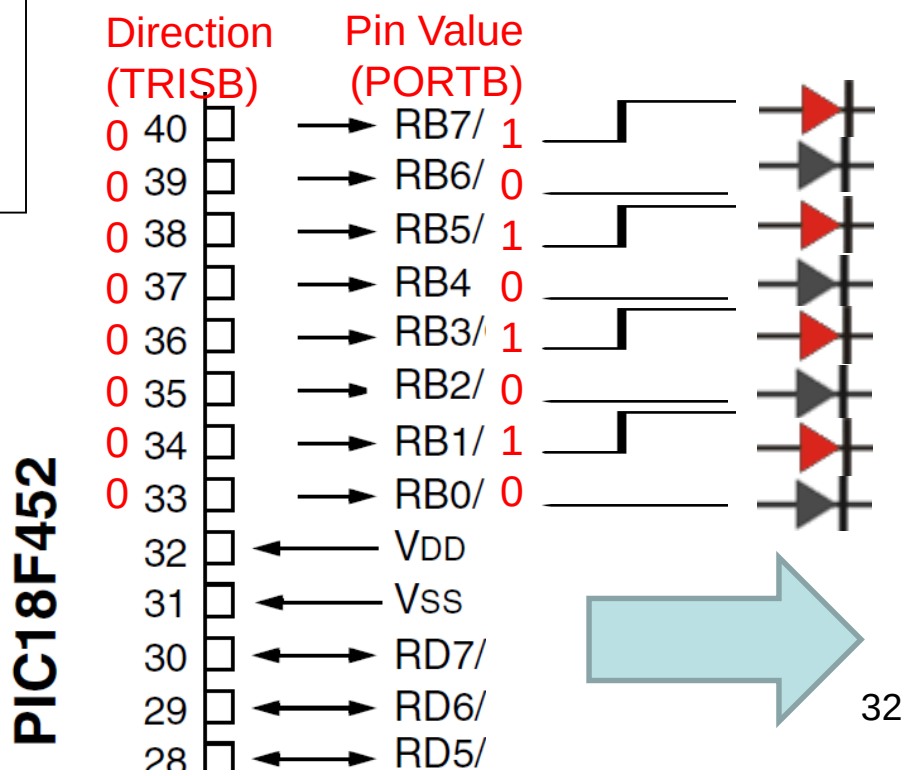
WREG = 1010 1010
TRISB = 0000 0000
PORTB = 1010 1010



Accessing SFRs in .ASM
PORTB as an OUTPUT

```
PORTB   EQU   0XF81    ;in .H header file
TRISB   EQU   0XF93    ;in .H header file

ORG 0x00

MOVLW   0              ;All 0's to WREG
MOVWF   TRISB          ;PORTB is an OUTPUT

MOVLW   B'10101010'
MOVWF   PORTB          ;Write 1/0 to PORTB pins
```

WREG = 1010 1010
TRISB = 0000 0000
PORTB = 1010 1010



PORTB as an INPUT

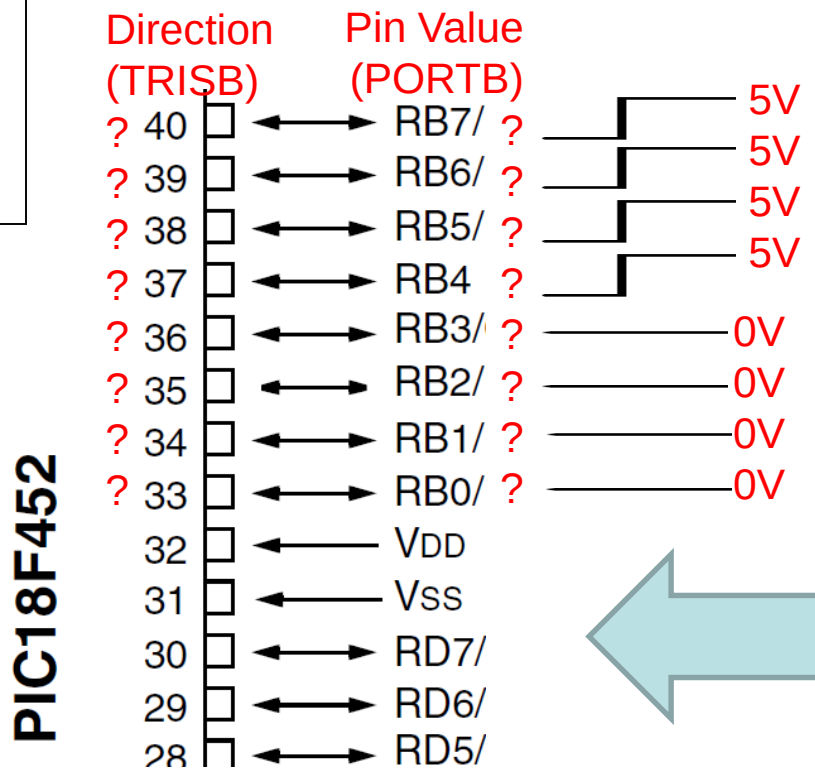
In order to make all the bits of PORTB an input, TRISB must be programmed by writing 1 to all the bits. In the code below, PORTB is configured first as an input port by writing all 1s to register TRISB, and then data is received from PORTB and saved in some RAM location of the file registers:

```
MYREG   EQU 0X20        ;Program location (RAM)

MOVLW   B‘11111111’     ;All 1’s to WREG
MOVWF   TRISB           ;PORTB as INPUT port (1 for In)

MOVF    PORTB, W        ;move from filereg of PORTB to WREG
MOVWF   MYREG           ;save in fileReg of MYREG
```

Accessing SFRs in .ASM
PORTB as an INPUT

```
ORG 0x00

MYREG   EQU   0X20 ;Program location (RAM)

MOVLW   B'11111111' ;All 1's to WREG
MOVWF   TRISB       ;PORTB as INPUT port

MOVF    PORTB, W ;move from filereg of
                  PORTB to WREG
MOVWF   MYREG    ;save in fileReg of
                  MYREG
```

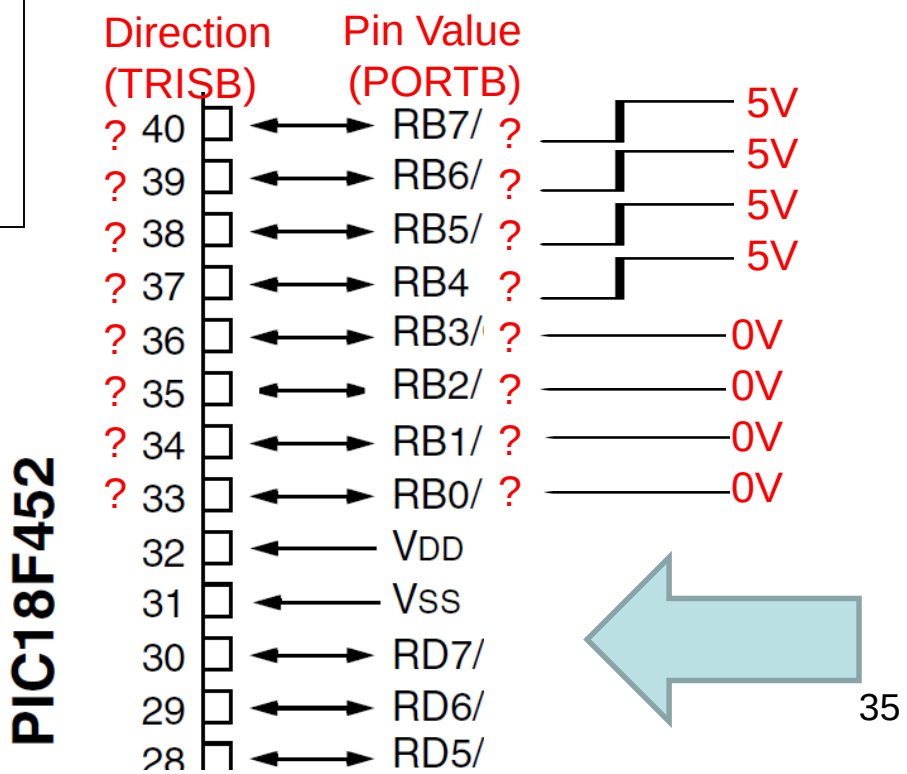
MYREG = ?
WREG = ?
TRISB = ?
PORTB = ?



Accessing SFRs in .ASM
PORTB as an INPUT

```
ORG 0x00

MYREG   EQU   0X20 ;Program location (RAM)

MOVLW   B‘11111111’ ;All 1’s to WREG
MOVWF   TRISB       ;PORTB as INPUT port

MOVF    PORTB, W ;move from filereg of
                  PORTB to WREG
MOVWF   MYREG    ;save in fileReg of
                  MYREG
```

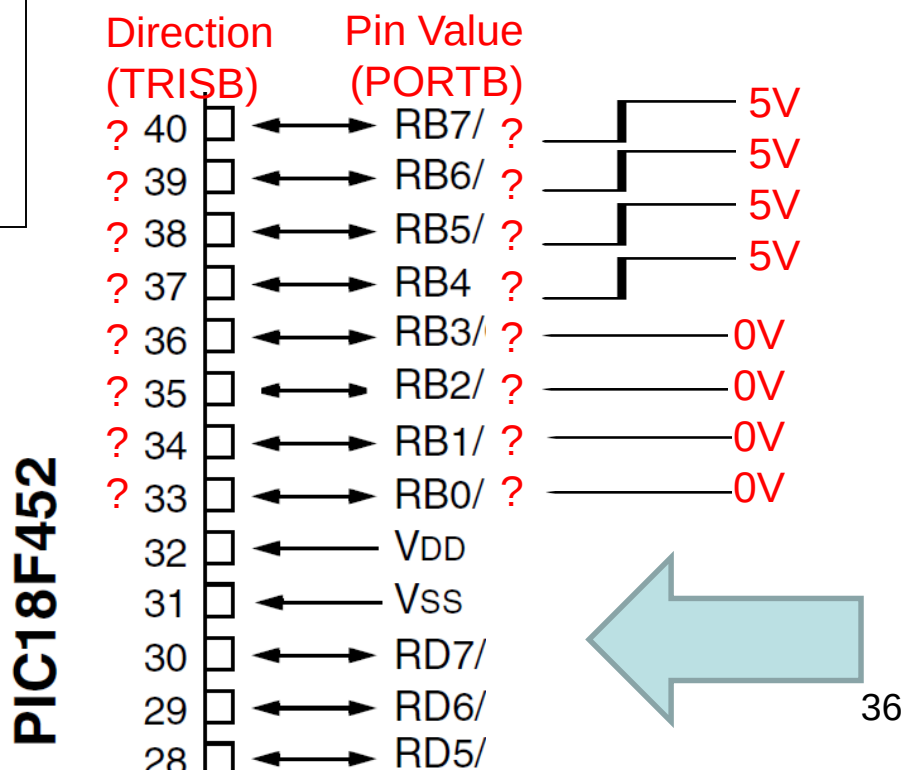
MYREG = 0000 0000 (at 0x20)
WREG = ?
TRISB = ?
PORTB = ?



Accessing SFRs in .ASM
PORTB as an INPUT

```
ORG 0x00

MYREG   EQU   0X20 ;Program location (RAM)

MOVLW   B‘11111111’ ;All 1’s to WREG
MOVWF   TRISB       ;PORTB as INPUT port

MOVF    PORTB, W ;move from filereg of
                  PORTB to WREG
MOVWF   MYREG    ;save in fileReg of
                  MYREG
```

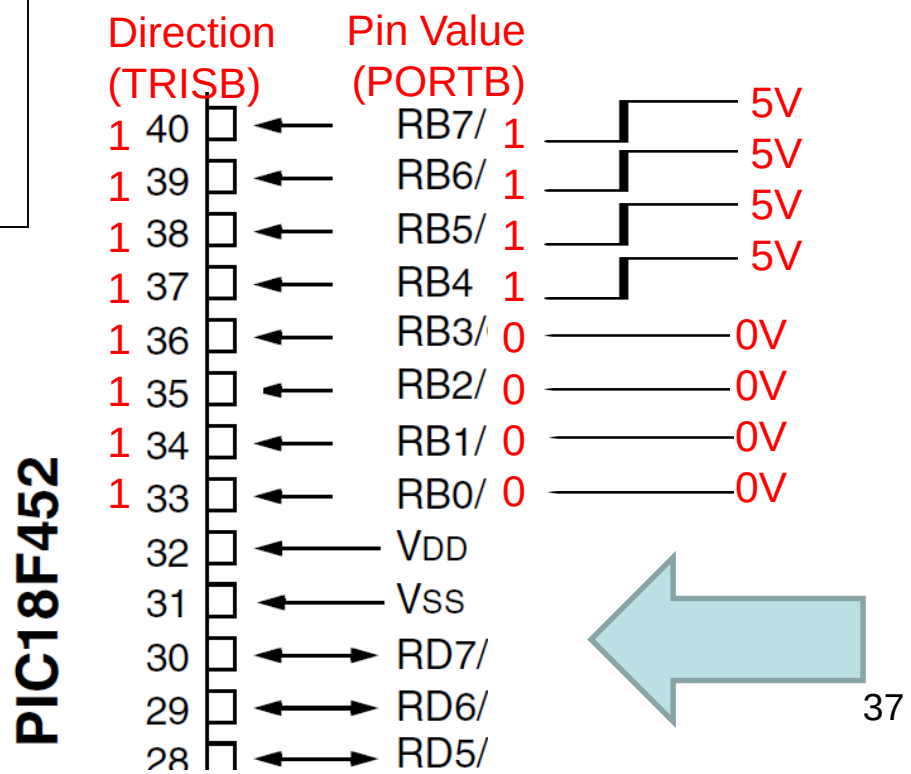
MYREG = 0000 0000 (at 0x20)
WREG = 1111 1111
TRISB = ?
PORTB = ?



Accessing SFRs in .ASM
PORTB as an INPUT

```
ORG 0x00

MYREG   EQU   0X20 ;Program location (RAM)

MOVLW   B'11111111' ;All 1's to WREG
MOVWF   TRISB       ;PORTB as INPUT port

MOVF    PORTB, W ;move from filereg of
                  PORTB to WREG
MOVWF   MYREG    ;save in fileReg of
                  MYREG
```

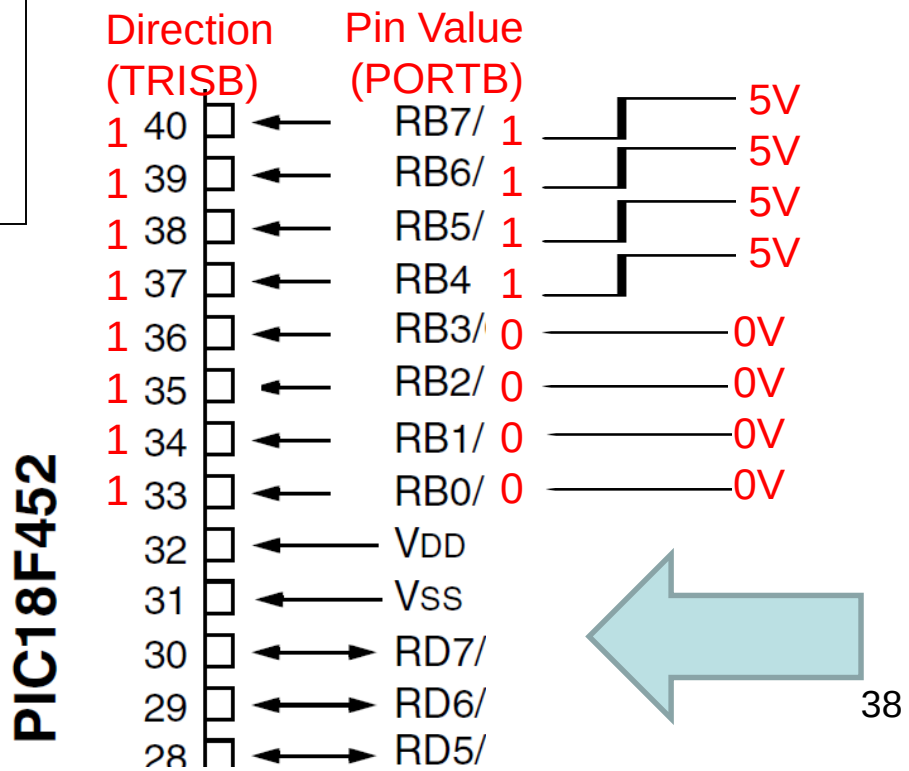
MYREG = 0000 0000 (at 0x20)
WREG = 1111 1111
TRISB = 1111 1111
PORTB = 1111 0000



Accessing SFRs in .ASM
PORTB as an INPUT

```
ORG 0x00

MYREG   EQU   0X20 ;Program location (RAM)

MOVLW   B‘11111111’ ;All 1’s to WREG
MOVWF   TRISB       ;PORTB as INPUT port

MOVF    PORTB, W ;move from filereg of
                  PORTB to WREG
MOVWF   MYREG    ;save in fileReg of
                  MYREG
```

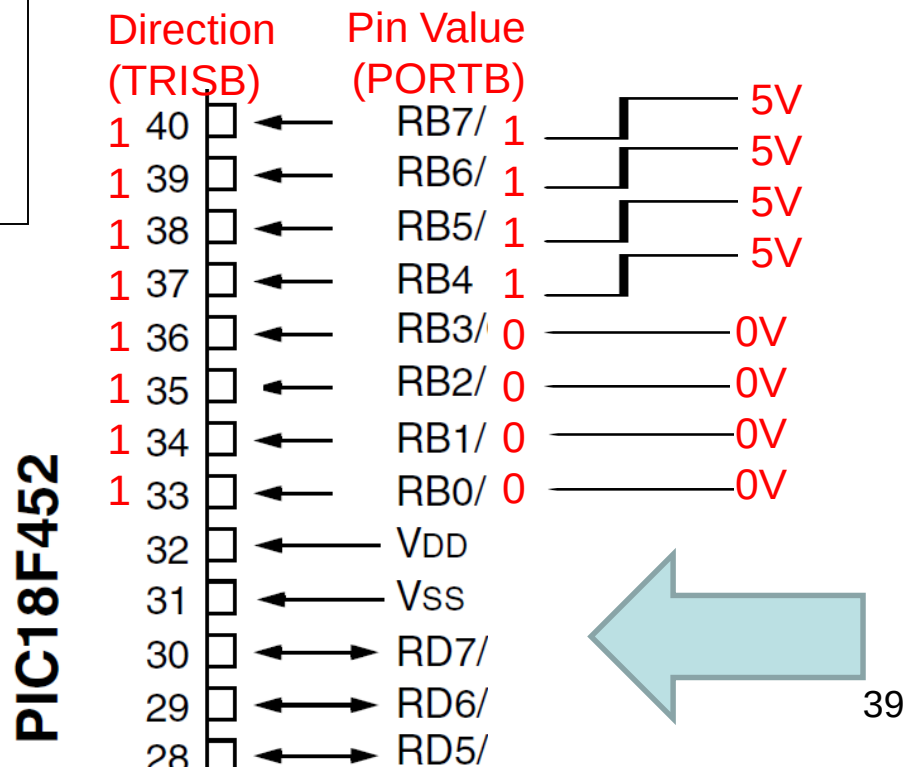
MYREG = 0000 0000 (at 0x20)
WREG = 1111 0000
TRISB = 1111 1111
PORTB = 1111 0000



Accessing SFRs in .ASM
PORTB as an INPUT

```
ORG 0x00

MYREG   EQU   0X20 ;Program location (RAM)

MOVLW   B‘11111111’ ;All 1’s to WREG
MOVWF   TRISB       ;PORTB as INPUT port

MOVF    PORTB, W  ;move from filereg of
                   PORTB to WREG
MOVWF   MYREG     ;save in fileReg of
                   MYREG
```

MYREG = 1111 0000 (at 0x20)
WREG = 1111 0000
TRISB = 1111 1111
PORTB = 1111 0000



Register bit manipulation

- Bit set flag
  - **BSF filereg, bit** **BSF TRISB, 4**
- Bit clear flag
  - **BCF filereg, bit** **BCF PORTB, 2**
- Bit toggle flag
  - **BTF filereg, bit**
- Bit test filereg skip next instruction if clear (0)
  - **BTFSC filereg, bit**
- Bit test filereg skip next instruction if set (1)
  - **BTFSS filereg, bit**

MPLAB Example

- http://omega.uta.edu/~nbb0130/misc_files/Main5_1.asm

Working with I/O Ports in C
Whole BYTES at a Time

```
#include <xc.h>          //OLD → #include <p18F452.h>

void main(void)
{
    unsigned char mybyte;
    TRISC = 0b11111111;  //PORTC is input
    TRISB = 0b00000000;  //PORTB is output
    TRISD = 0b00000000;  //PORTD is output

    while(1)
    {
        mybyte = PORTC;   //load the value of PORTC

        if(mybyte < 100)
            PORTB = mybyte;  //send it to PORTB is it is less than 100
        else
            PORTD = mybyte; //otherwise, send to PORTD
    }
}
```

Working with I/O Ports in C
Single BITS at a Time

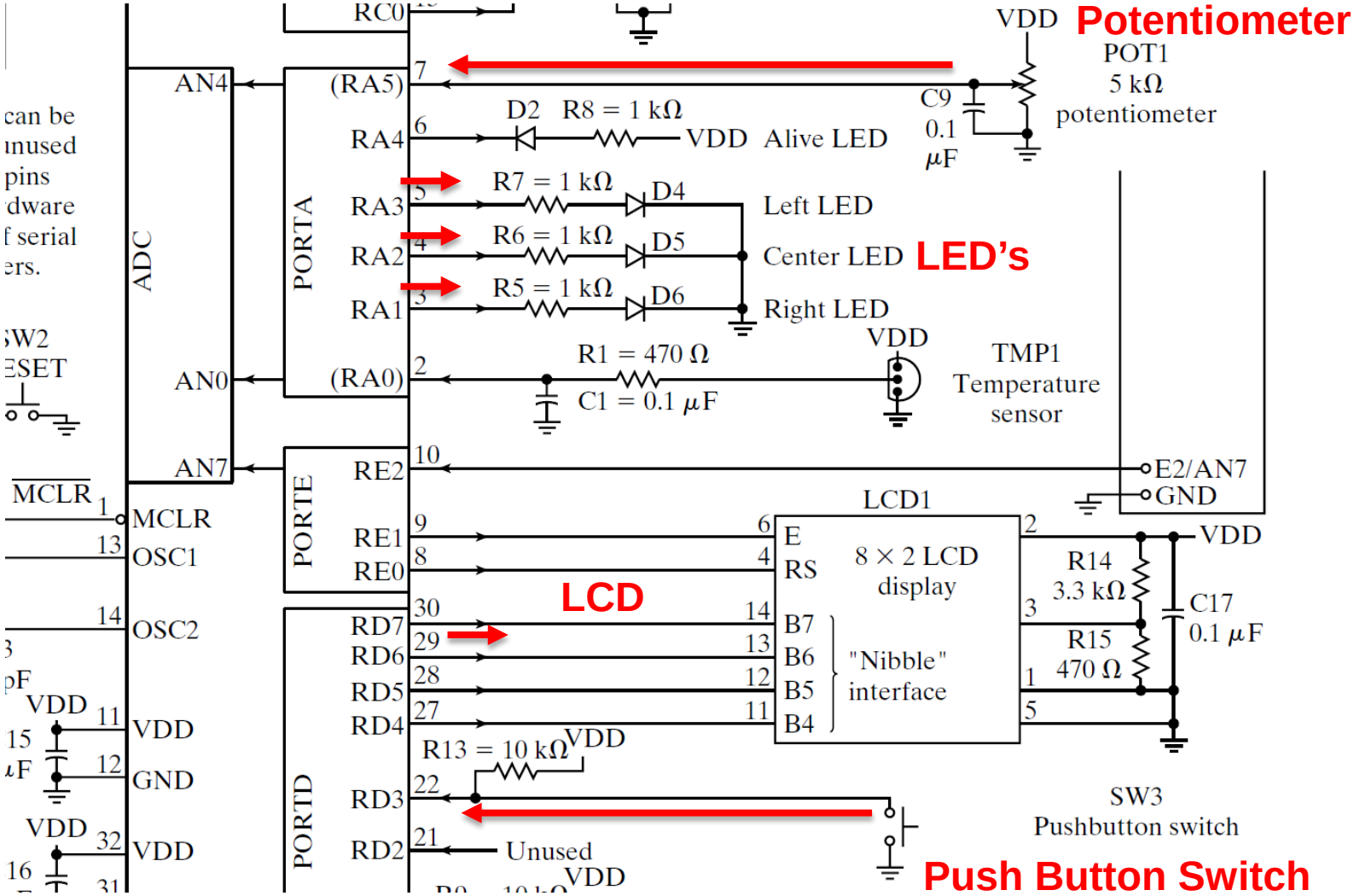
```
#include <xc.h>                    //OLD → #include <p18F452.h>

void main(void)
{
    unsigned char mybyte;
    TRISC = 0b11111111;  //PORTC is input
    TRISB = 0b00000000;  //PORTB is output
    TRISBbits.RB4 = 1;
    TRISD = 0b00000000;  //PORTD is output

    while(1)
    {
        mybyte = PORTC;   //load the value of PORTC

        if(mybyte < 100)
            PORTB = mybyte;  //send it to PORTB is it is less than 100
        else
            PORTD = mybyte; //otherwise, send to PORTD

        mybyte = PORTCbits.RC1;
    }
}
```

Fan-out

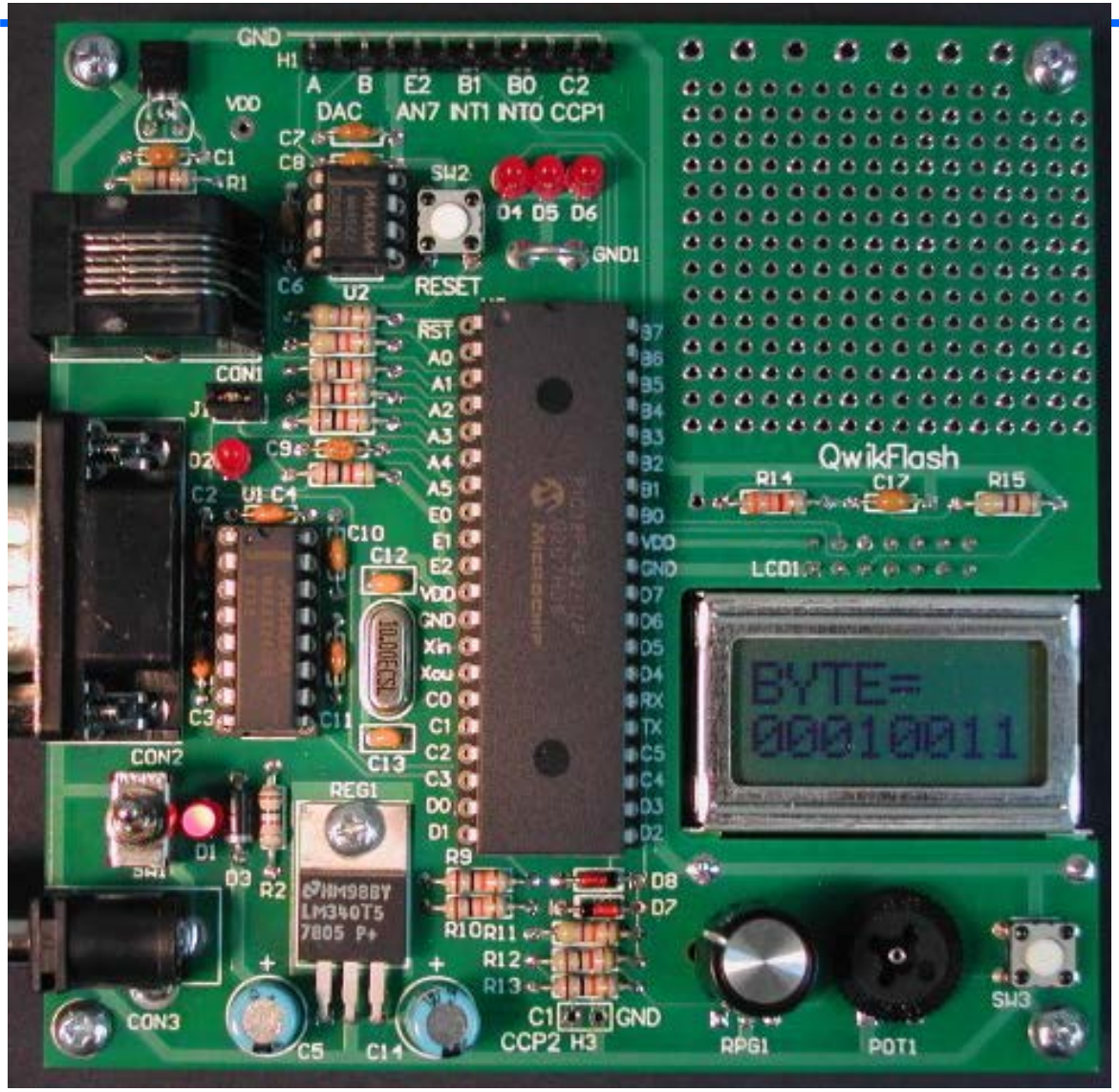
- Current can flow in (pin at 0 level) and out (pin at 1 level) of port pins.
- This current is limited by the design of the IC.
- Fan-out is really the number of logic gates a pin can drive but is closely connected to the total current of pins.
- Arguably, for microcontrollers it is more important to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

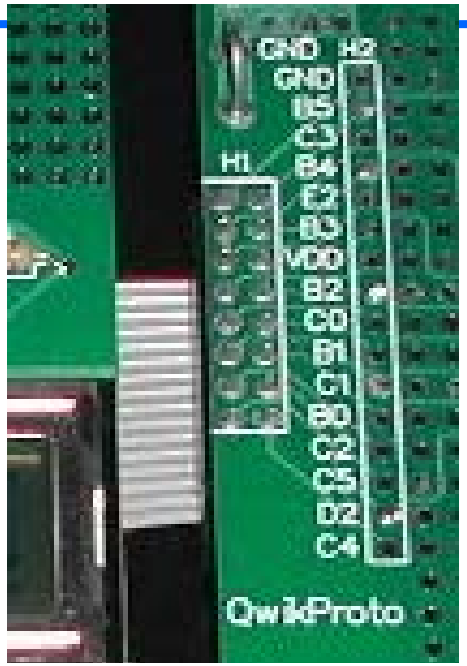
Maximum output current sunk by any I/O pin	25 mA
Maximum output current sourced by any I/O pin	25 mA
Maximum current sunk by PORTA, PORTB, and PORTE **(Note 3)** (combined)	200 mA

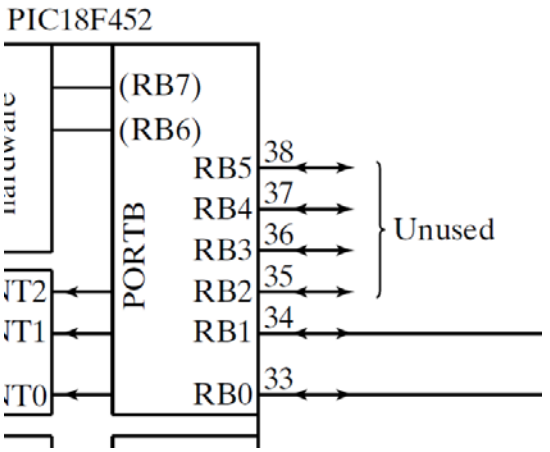
Example of Interfacing PIC to Components on QwikFlash



Example of Interfacing PIC to Components on QwikFlash







Example: Parallel Digital Output LCD Control

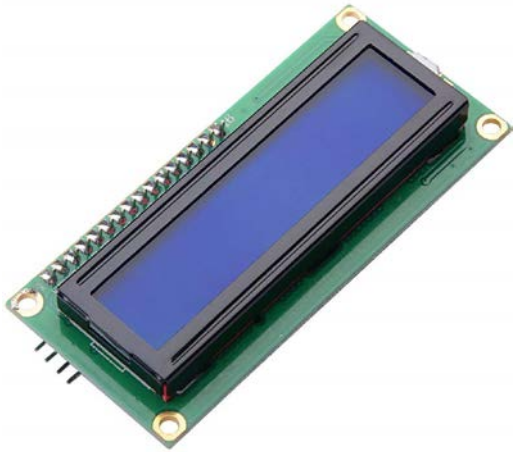
Driving LCD Controllers

(textbook chapter 12,
PICBook chapter 7)

LCDs

- **L**iquid **C**rystal **D**isplays are frequently used with microcontrollers and embedded devices
- Usually have their own controller for logic and receiving commands
- Commonly have **parallel** digital inputs for interfacing
  - Some have **serial** interfaces instead (SPI, I^2C, etc.)
- LCD modules usually require an **initialization sequence** when powered up before regular commands can be sent
  - So some small wait time should be expected (ms range)

Different LCD Types



Sample Image (not a PIC)

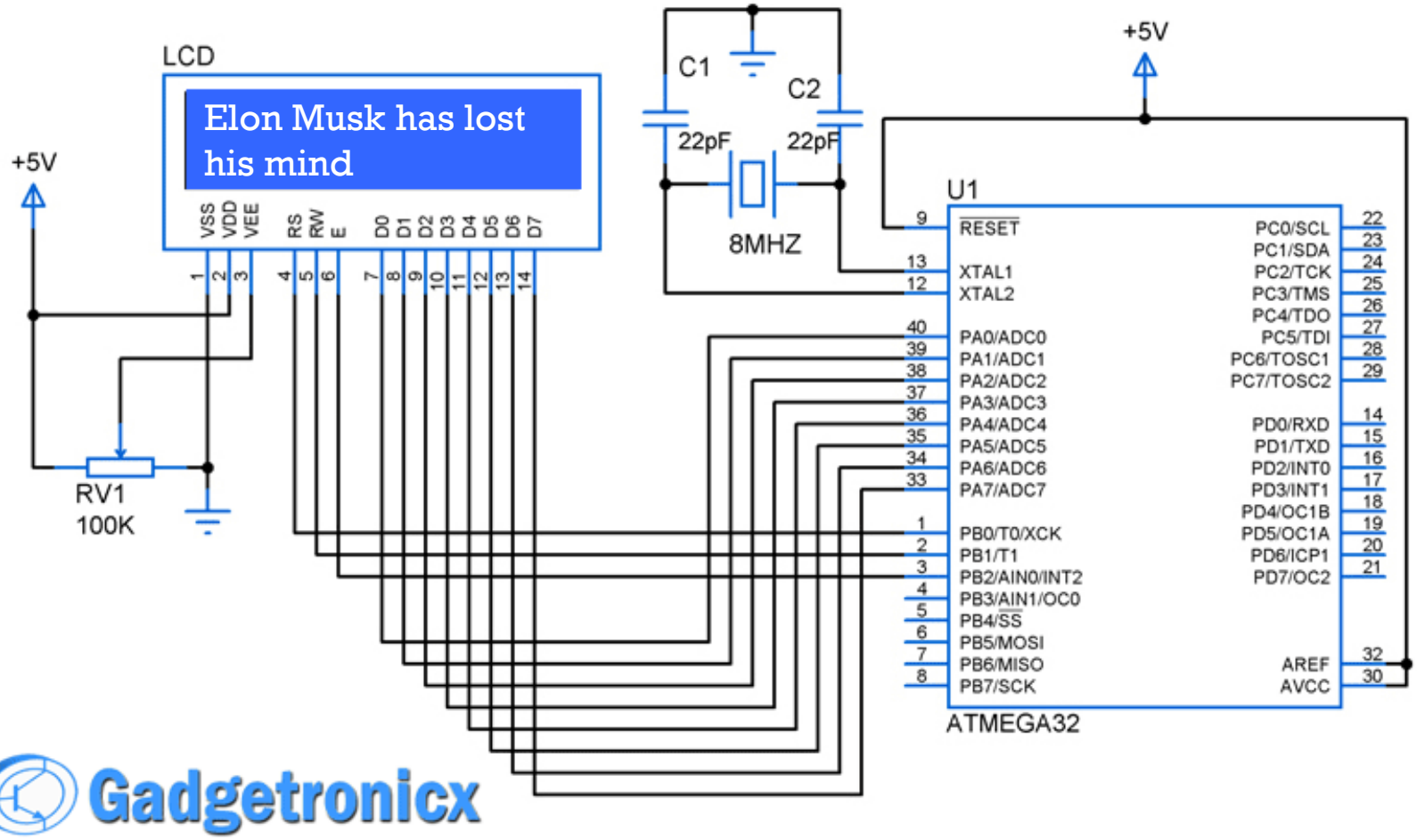
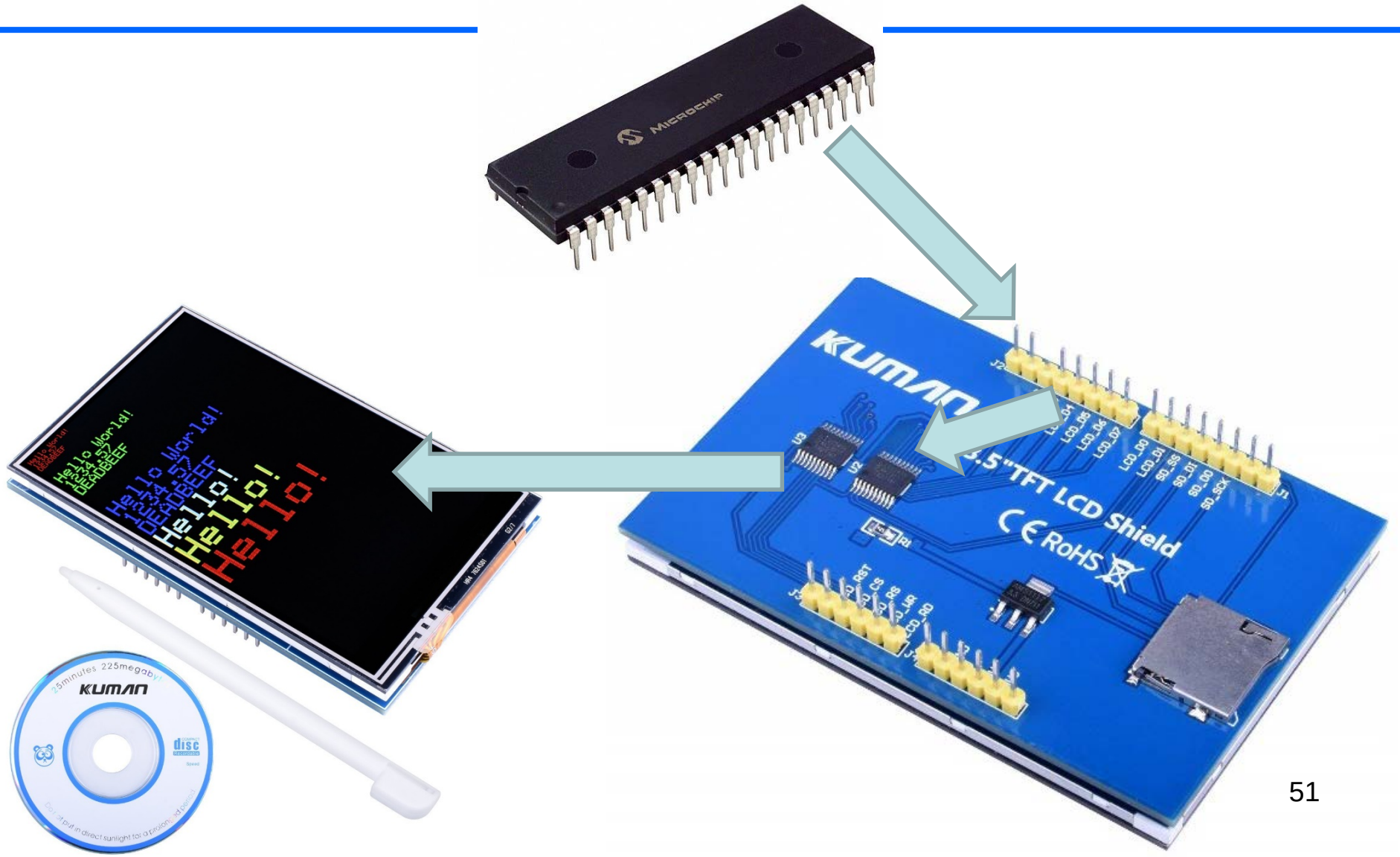
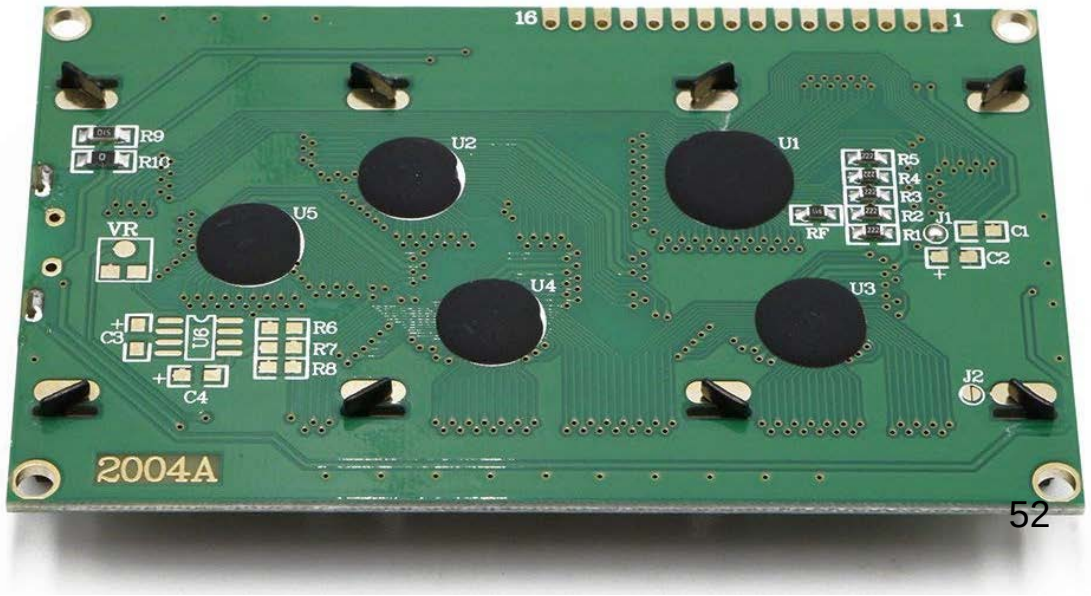


Image Source: http://www.gadgetronicx.com/lcd-interface-with-atmega32-avr/

LCD Controller



LCD Controller

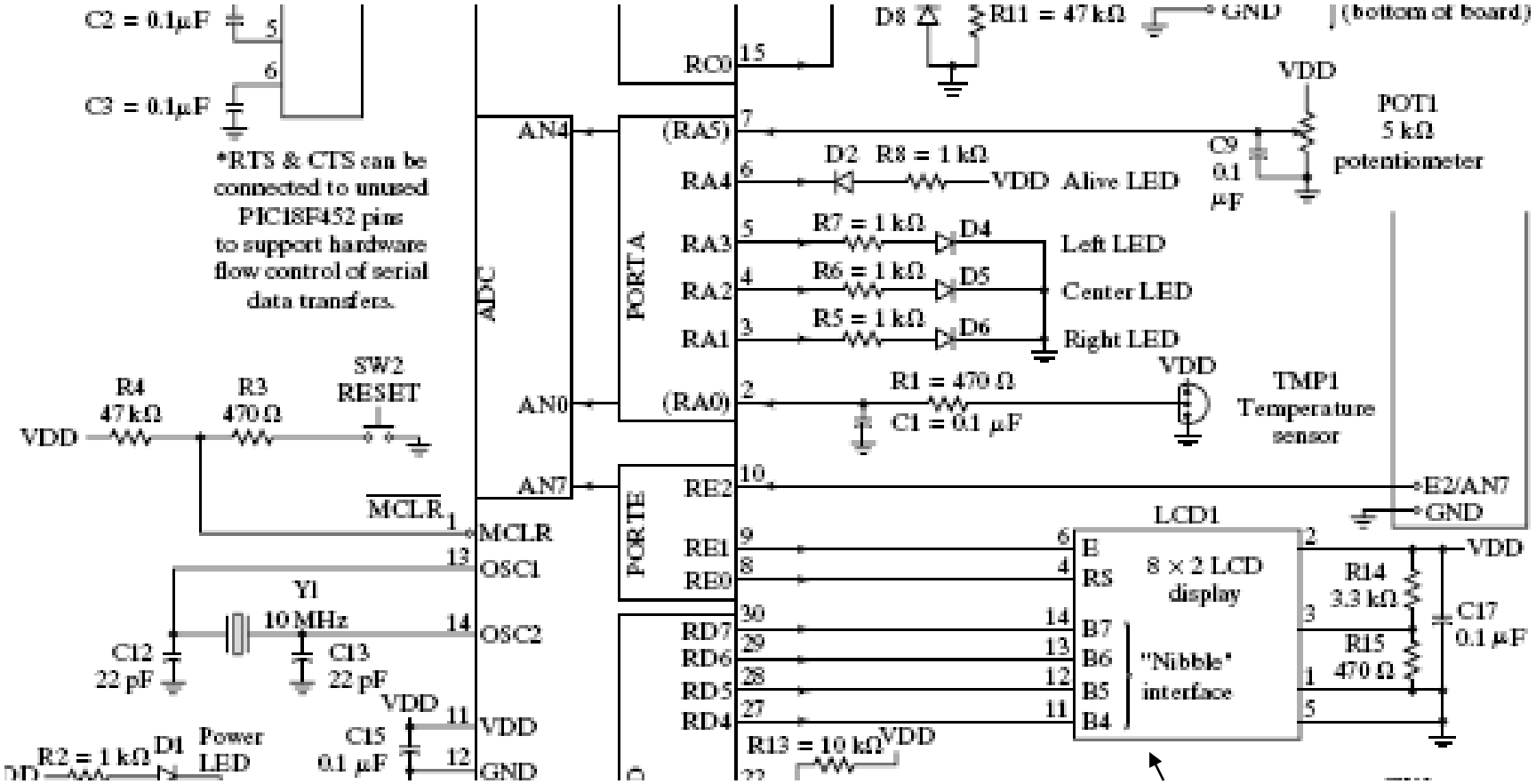


LCD Common Pins

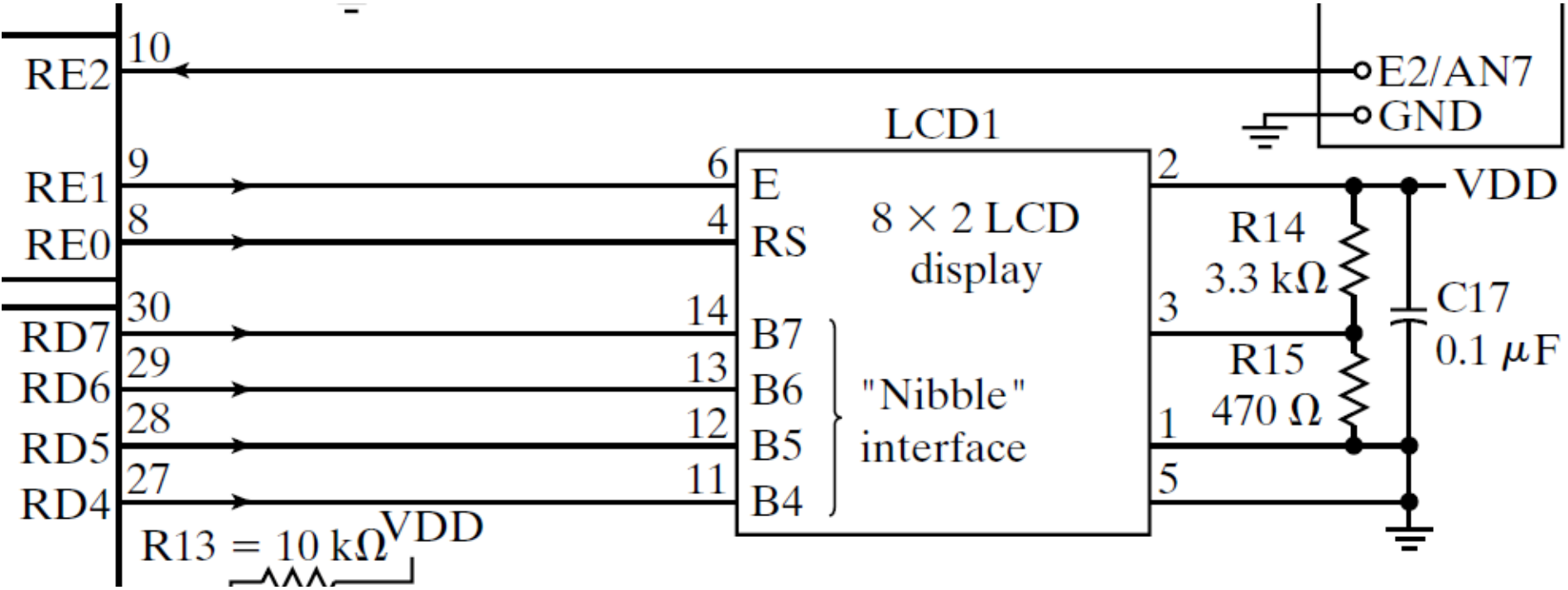
- **Supply, “ground”, and LCD contrast voltage**
- **Register Select (RS)**
  - **RS=0** for sending **instructions** (such as clear screen, or defining characters)
  - **RS=1** for sending **data** to be displayed
- **Enable (E)**
  - Essentially a clock input; a high-low transition will cause the LCD to latch in the data on the data pins
- **Data (D0-D7) or (D0-D3)**
  - The parallel interface pins (can use all 8 or just 4)
- **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

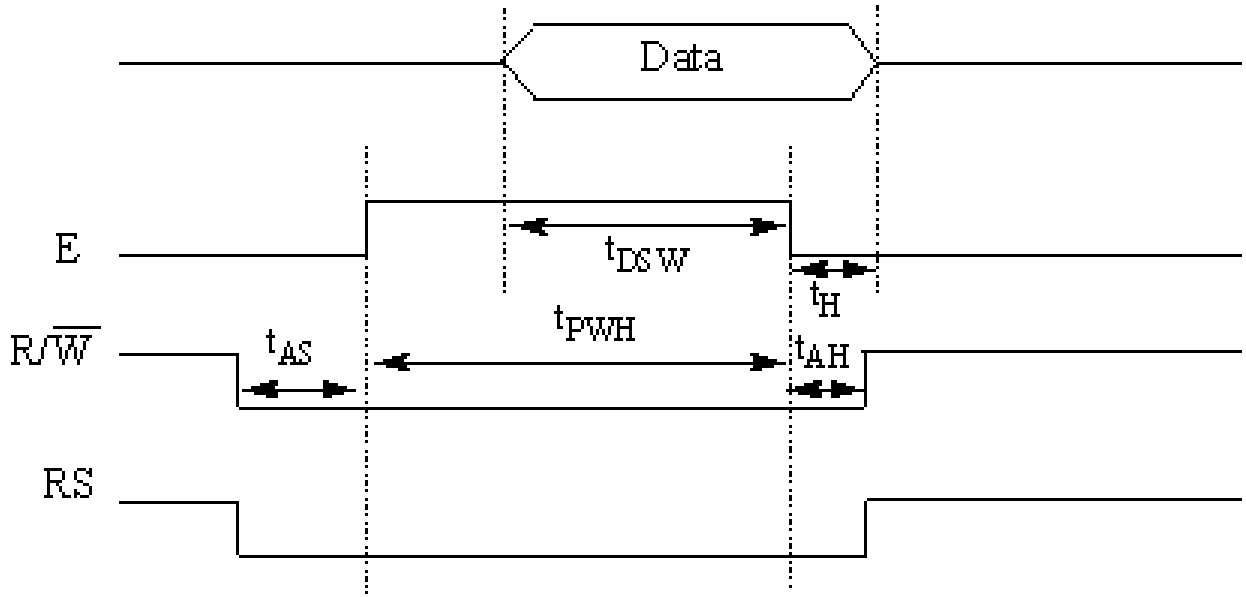
Connections to QwikFlash



LCD Module – Hitachi HD44780 onboard controller


RE2
10
RE1
9
RE0
8
RD7
30
RD6
29
RD5
28
RD4
27
R13 = 10 kΩ
VDD
LCD1
6
E
4
RS
14
B7
13
B6
12
B5
11
B4
8 × 2 LCD
display
"Nibble"
interface
2
3
1
5
E2/AN7
GND
VDD
R14
3.3 kΩ
R15
470 Ω
C17
0.1 μF

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)
- Positions are counted left to right and auto increment can be enabled (no need for cursor positioning for short strings)

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

Questions?

- Textbook Ch. 4.1 and 4.2 for PIC IO examples and more details
- Textbook Ch. 12.1 for LCD details
- LCD Videos
  - https://www.youtube.com/watch?v=mo4_5vG8bbU
  - https://www.youtube.com/watch?v=ZP0KxZl5N2o
  - https://www.youtube.com/watch?v=85LvW1QDLLw

- Start reading Chapter 7
  - PIC Programming in C