Random Items

Timer, Switches, Keypads, Hints, A/D Potentiometer

Return from Subroutine

RETURN — Return from Subroutine

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC

Status Affected: None

Encoding:

00	0000	0000	1000

Description: Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	No operation	Process data	No operation

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
HERE    RETURN
```

After Instruction

PC = TOS

RETLW — Return with Literal in W

Syntax: [*label*] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W;
TOS → PC

Status Affected: None

Encoding:

11	01xx	kkkk	kkkk

Description: The W register is loaded with the eight bit literal 'k'. The program counter is loaded 13-bit address at the Top of Stack (the return address). This is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
HERE    CALL TABLE   ; W contains table
                     ; offset value
        •            ; W now has table value
        •
        •
TABLE   ADDWF PC     ;W = offset
        RETLW k1     ;Begin table
        RETLW k2     ;
        •
        •
        •
        RETLW kn     ; End of table
```

Before Instruction

W = 0x07

After Instruction

W = value of k8

PC = TOS = Address `Here + 1`

Interrupt Handling

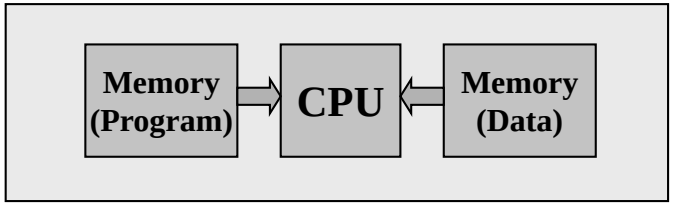
- It is good programming practice to have a “dummy” interrupt handling routine if you are not using interrupts
- If you accidentally enable interrupts, nothing bad happens

```
        org     04h                     ; interrupt vector
        goto    isrService              ; jump to interrupt routine
```

•
•
•

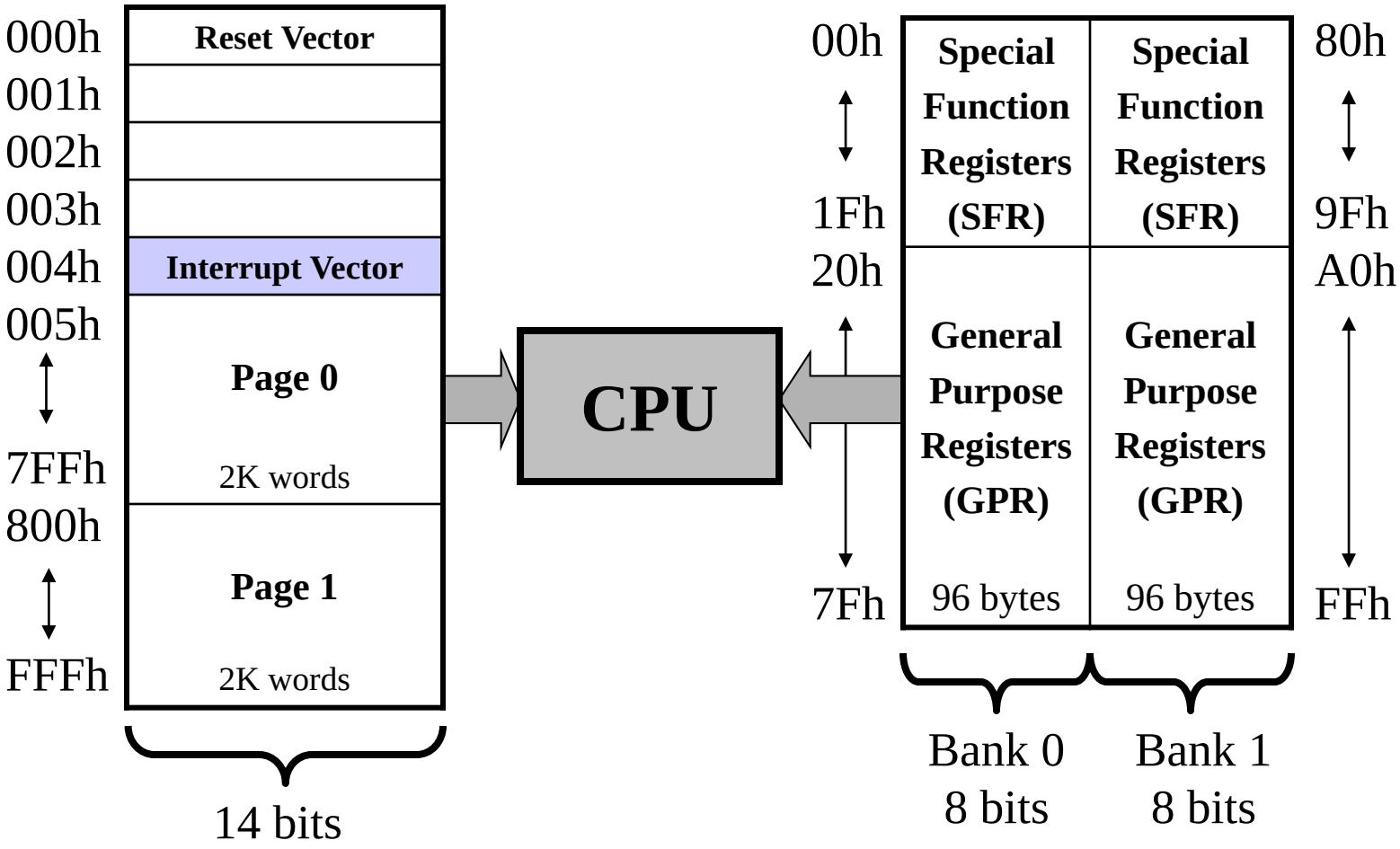
```
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;

isrService

        goto        isrService                  ; error - - stay here

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

MicroChip PIC16F74
Harvard Architecture



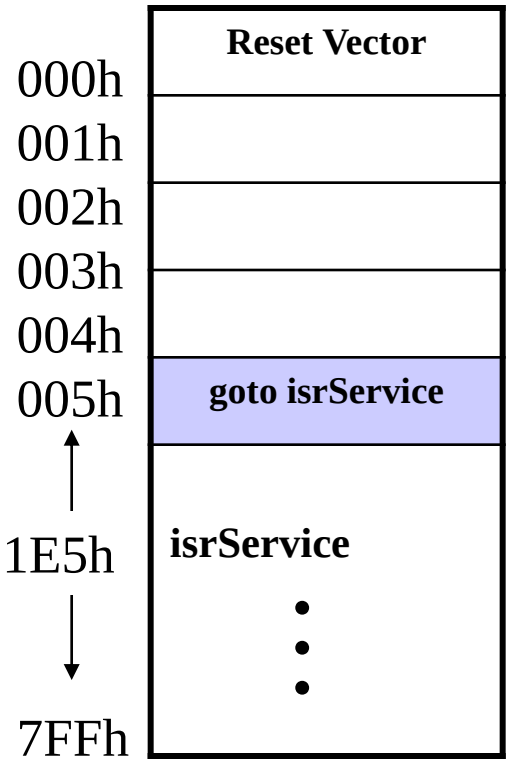
Program Memory

Data Memory



Program Memory

```
org     04h                 ; interrupt vector
goto    isrService          ; jump to interrupt routine
```



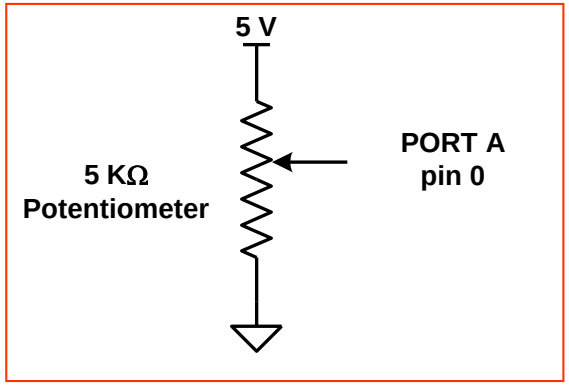
GOTO Unconditional Branch

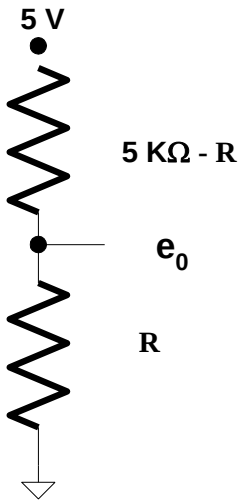
Syntax:	[*label*] GOTO k
Operands:	0 ≤ k ≤ 2047
Operation:	k → PC<10:0> PCLATH<4:3> → PC<12:11>
Status Affected:	None
Encoding:	10 1kkk kkkk kkkk
Description:	GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two cycle instruction.

101 001 1110 0101

14 bits

Operation of the Precision Potentiometer





$$e_0 = \frac{R}{(5\,K - R) + R}(5\,V) = \frac{R}{5\,K}(5\,V)$$

wiper of potentiometer		dial of potentiometer		voltage on Port A pin		register ADRES value
5 KΩ	↕	10.0	↕	5.0 V	↕	255
R	→		→		→	
0 Ω		0.0		0.0 V		0

Potentiometer Examples

When Dial = 10, Output (i.e. input to A/D) is 5.0 V
When Dial = 5, Output (i.e. input to A/D) is 2.5 Volts
When Dial = 0, Output (i.e. input to A/D) is 0.0 Volts

When Dial = 10, ADRESH = 255 Decimal = FF Hex
When Dial = 5, ADRESH = 127 Decimal = 7F Hex
When Dial = 0, ADRESH = 00 Decimal = 00 Hex

Mode 3 (AC Example)

$$\text{ADRES} = 70 \text{ Hex} = 112 \text{ Decimal} = \frac{x}{10.0}(255)$$

solving $\Rightarrow$ x = Dial = 4.4

Timing Examples

$$\text{dial} = 0.392 \quad \Rightarrow \quad \text{ADRES} = \frac{0.392}{10.0}(255) = 10 = 0\text{A h} = \text{B}\,00001010$$

$$= 10 \quad \text{Decimal}$$

2.5 seconds

$$\text{dial} = 5.0 \quad \Rightarrow \quad \text{ADRES} = \frac{5.0}{10.0}(255) = 255 = 80\text{h} = \text{B}\,1000\,0000$$

$$= 128 \quad \text{Decimal}$$

32 seconds

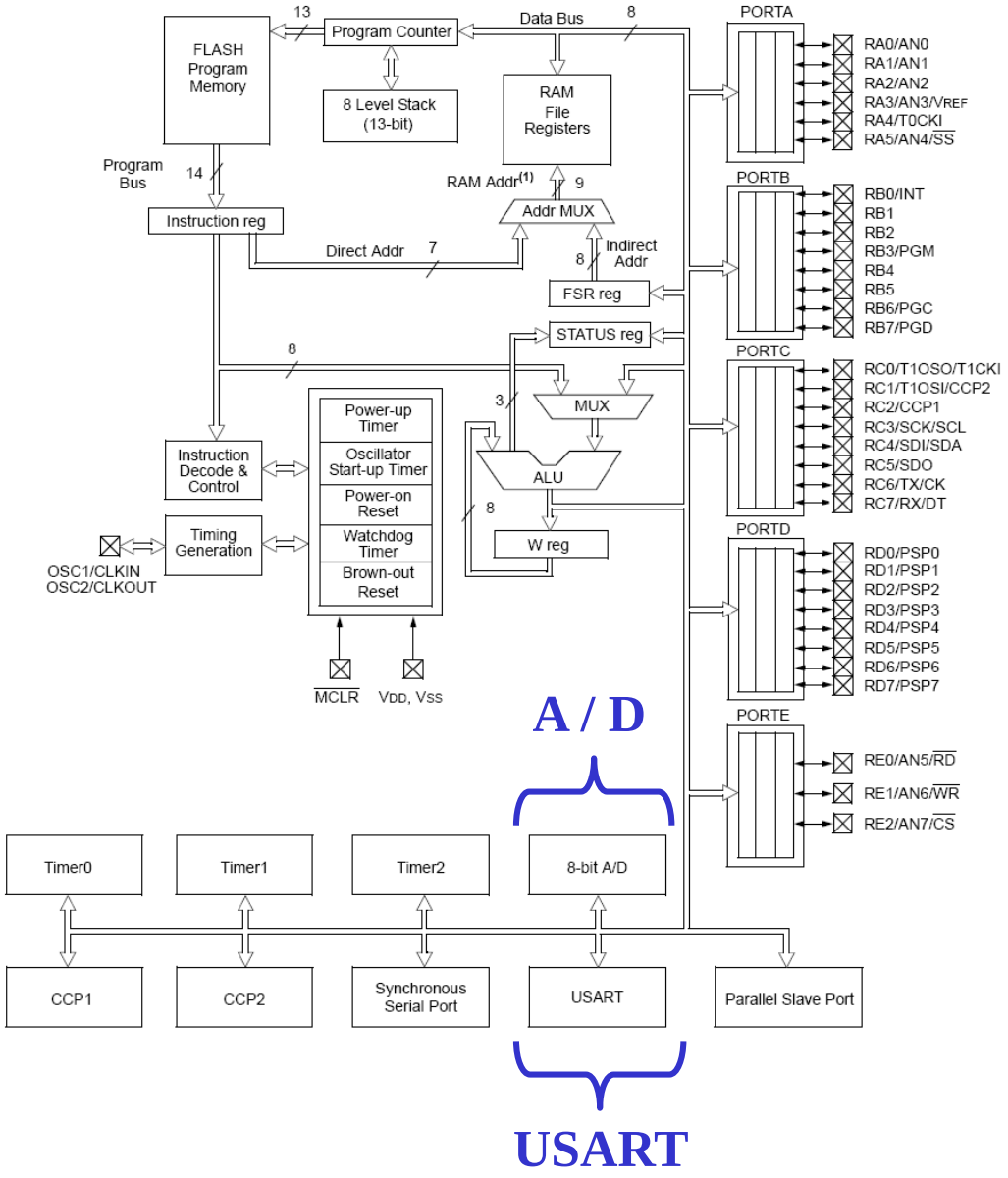
$$\text{dial} = 7.5 \quad \Rightarrow \quad \text{ADRES} = \frac{7.5}{10.0}(255) = 192 = \text{C0h} = \text{B}\,1100\,0000$$

$$= 192 \quad \text{Decimal}$$

48 seconds

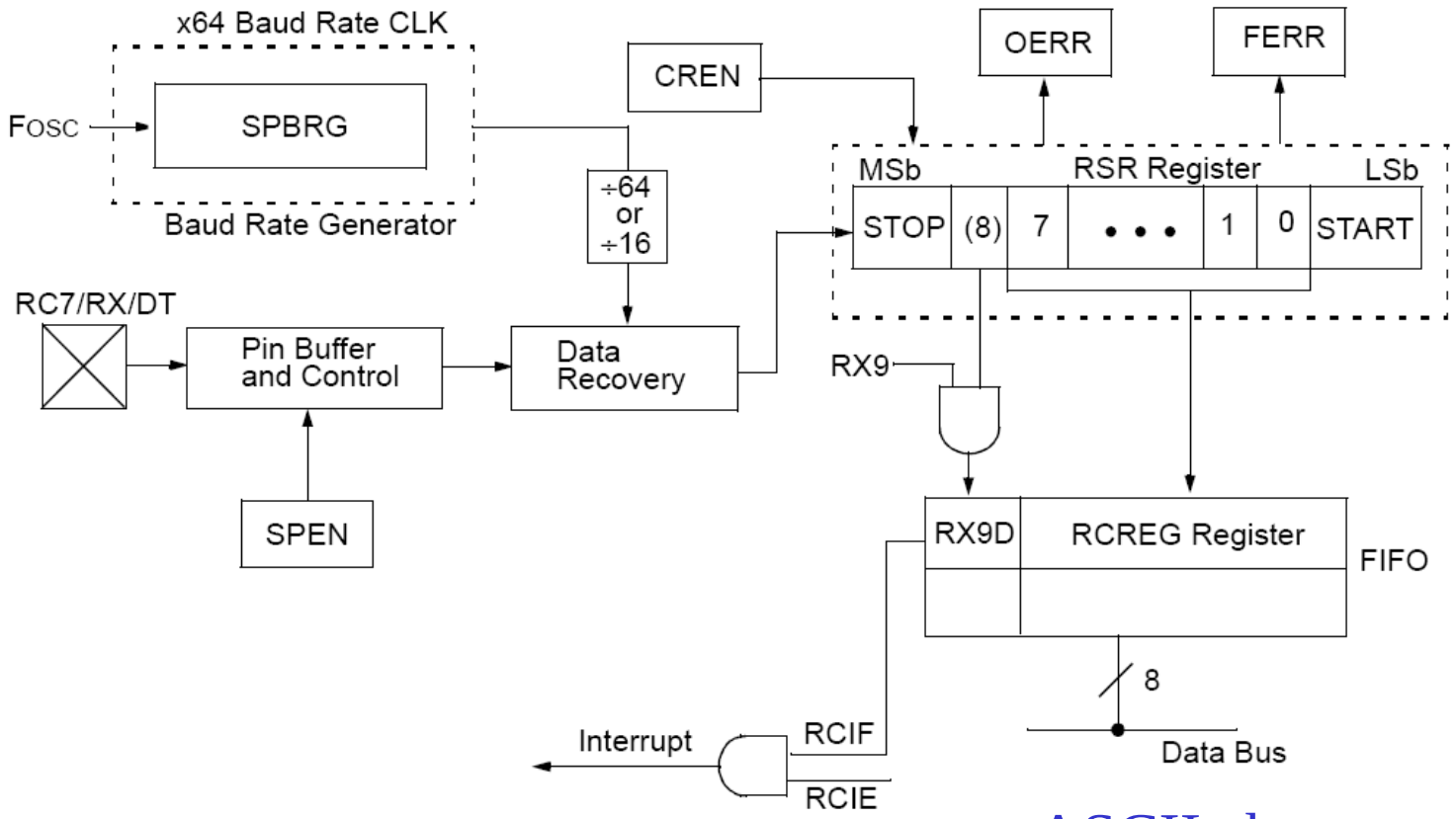
PIC16F747 Schematic

USART can be used to connect to the COM port on a PC



USART Receive Block Diagram

(Universal Synchronous/Asynchronous Receiver-Transmitter)



ASCII character

ASCII
(American Standard Code for Information Interchange)

most significant nibble

least significant nibble

LSB/MSB HEX	HEX BIN	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	(NUL)	(DLE)	Space	0	@	P	'p	
1	0001	(SOH)	(DC1)	!	1	A	Q	a	q
2	0010	(STX)	(DC2)	"	2	B	R	b	r
3	0011	(ETX)	(DC3)	#	2	C	S	c	s
4	0100	(EOT)	(DC4)	$	4	D	T	d	t
5	0101	(ENQ)	(NAK)	%	5	E	U	e	u
6	0110	(ACK)	(SYN)	&	6	F	V	f	v
7	0111	(BEL)	(ETB)	'	7	G	W	g	w
8	1000	(BS)	(CAN)	(	8	H	X	h	x
9	1001	(HT)	(EM)	)	9	I	Y	i	y
A	1010	(LF)	(SUB)	*	:	J	Z	j	z
B	1011	(VT)	(ESC)	+	;	K	[	k	{
C	1100	(FF)	(FS)	,	,	L	\	l	|
D	1101	(CR)	(GS)	-	+	M	]	m	}
E	1110	(SO)	(RS)	.	.	N	^	n	~
F	1111	(SI)	(US)	/	?	O	_	o	DEL

s ⇒ 01110011
h ⇒ 01101000

Using a "Mask"

Suppose you set up your code so that if you receive an "s" from the PC, you start, and if you receive an "h" from the PC, you stop (halt).

ASCII	
s ⇒	01110011
h ⇒	01101000

If zero bit of STATUS = 1
result of XOR was 0

" trapped the s"

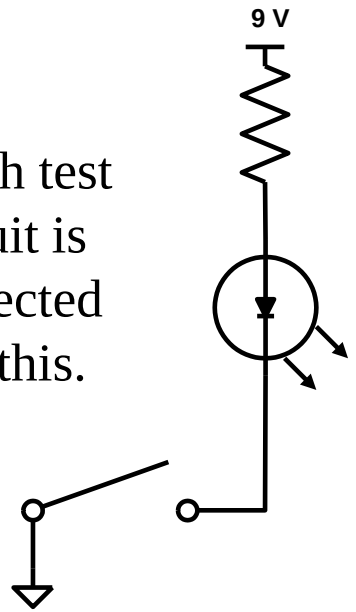
```
movlw   B'01110011'     ; move the mask for "s" to W register
xorwf   RCREG,W         ; exclusive OR with USART receive register
btfsc   STATUS,z        ; check for match
goto    Start           ; goto the start code
movlw   B'01101000'     ; move the mask for "h" to W register
xorwf   RCREG,W         ; exclusive OR with USART receive register
btfsc   STATUS,z        ; check for match
goto    Stop            ; goto the stop code
```

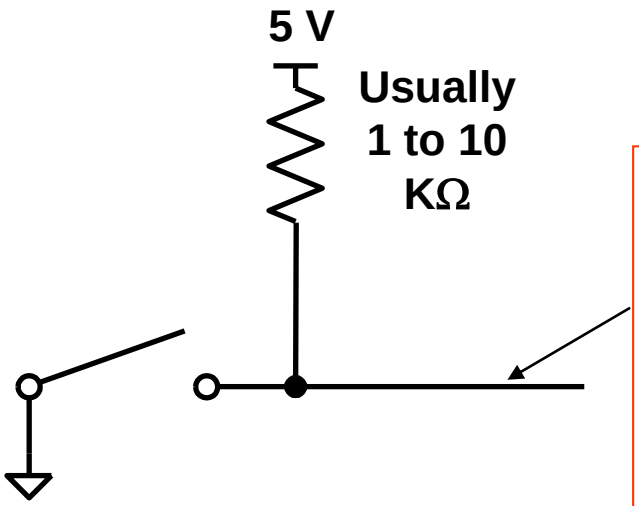
could have also written ⇒ movlw 's'

Switches

- Since a switch can be either on or off, it can be considered to be a logic element.
- Because of the way that the switch is normally wired in an electronic circuit **"on"** corresponds to digital **"low"** & **"off"** corresponds to digital **"high"**

Switch test circuit is connected like this.





This wire is connected to the electronic circuit. When the switch is on (closed) the wire is connected to ground (0 V, logical low). When the switch is off (open) the wire is connected to 5 V (logical high). The resistor limits the current when the switch is closed (so that you do not short your 5 V power supply to ground).

Aside: Throw & Pole

- "Throw" refers to the motion of the switch.
  - Single throw means that the switch terminals are either open (disconnected) or closed (connected).
  - Double throw means that one set of switch terminals are open while another set is closed. When you "throw the switch", you close the first set of terminals and open the second.

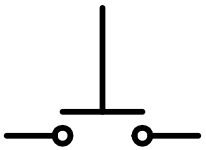
- "Pole" refers to the number of independent connections on the switch.
  - Single pole means that the switch has one set of terminals.
  - Double pole means that the switch has two sets of terminals.

- Switches are typically either Single or Double Throw but can have many poles. You would say something like "single pole double throw" (i.e. you give the number of poles first).

Single Throw Switches

Single Pole

toggle

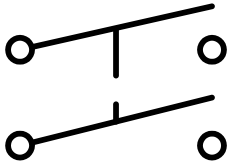


pushbutton

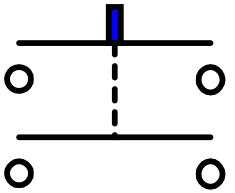
spring return toggle
momentary

Dashed line indicates poles are mechanically connected (i.e. move together) but there is no electrical connection.

Double Pole



toggle

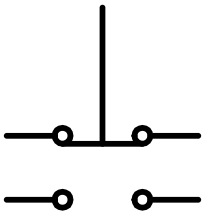


pushbutton

Double Throw Switches

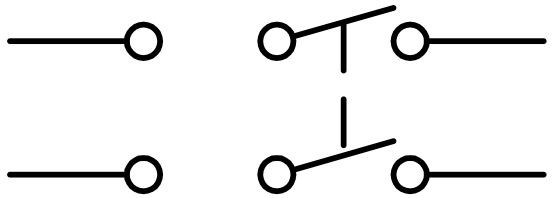
Single Pole

toggle



pushbutton

Double Pole



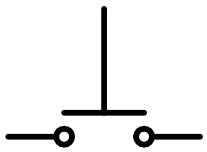
toggle

Some Complexity

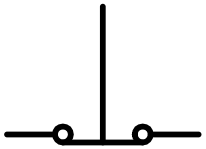
Single pole single throw pusshbutton or momentary switches can be "normally open" meaning that they close when you throw the switch.

Alternatively, they can be "normally closed" meaning that they open when you throw the switch.

These terms would not apply to a toggle switch since they do not have a spring return.



normally open



normally closed

Double pole switches can be "make before break" meaning that they close the second set of poles before they open the first (requires some mechanical complexity).

Alternatively, they can be "break before make" (the simplest) meaning that they open the first set of poles before they close the second.

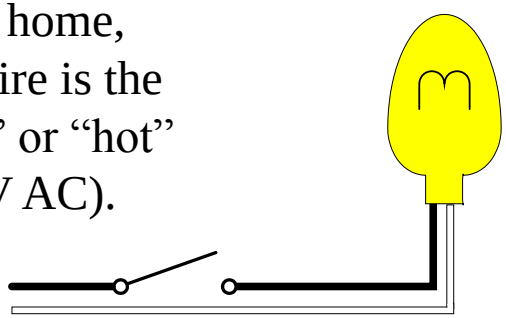
Double throw switches can also have a center position that opens both sets of poles. This is called "on-off-on" as opposed to "on-on".

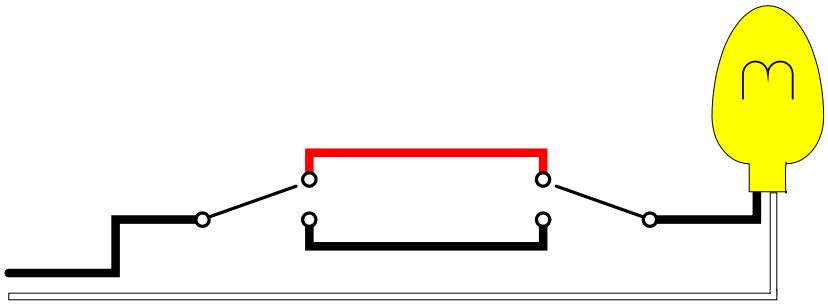
on - off - on

on - on

“ 2 Way” Switch

In the home, black wire is the “supply” or “hot” (115V AC).



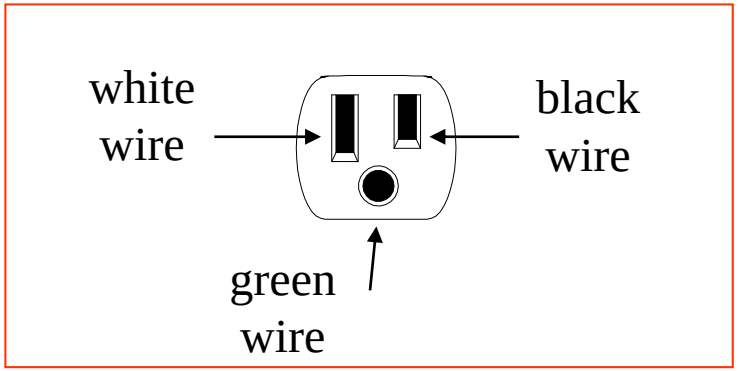


White wire is the “neutral” or “return” (0 V).

Green wire is the “ground” (sometimes connected to water pipes).

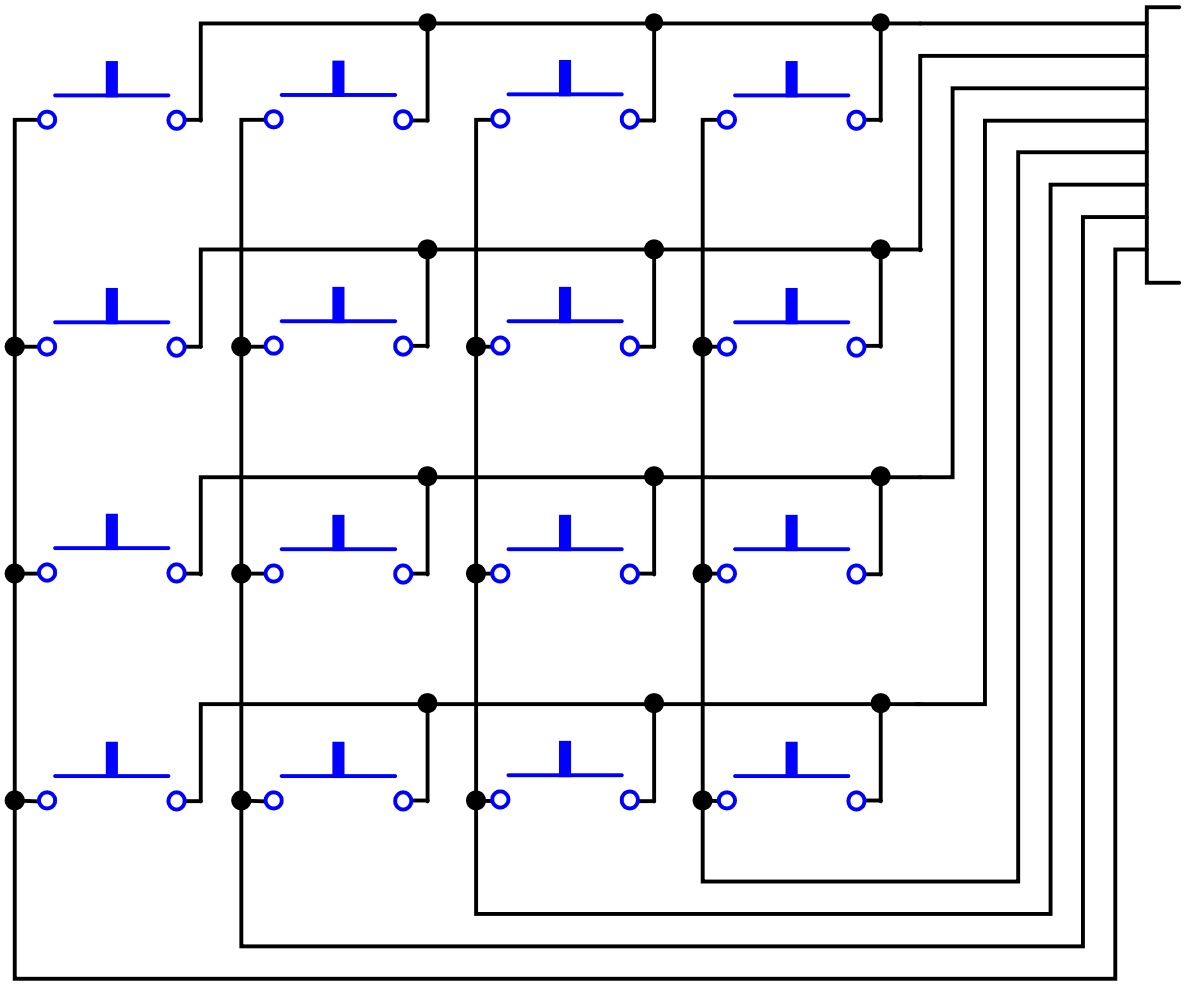
Single pole double throw switch (on – on) turns light on or off from either side of the room.

Single pole single throw switch turns light on or off.



Using a Keypad

Single Pole - Single Throw Pushbutton



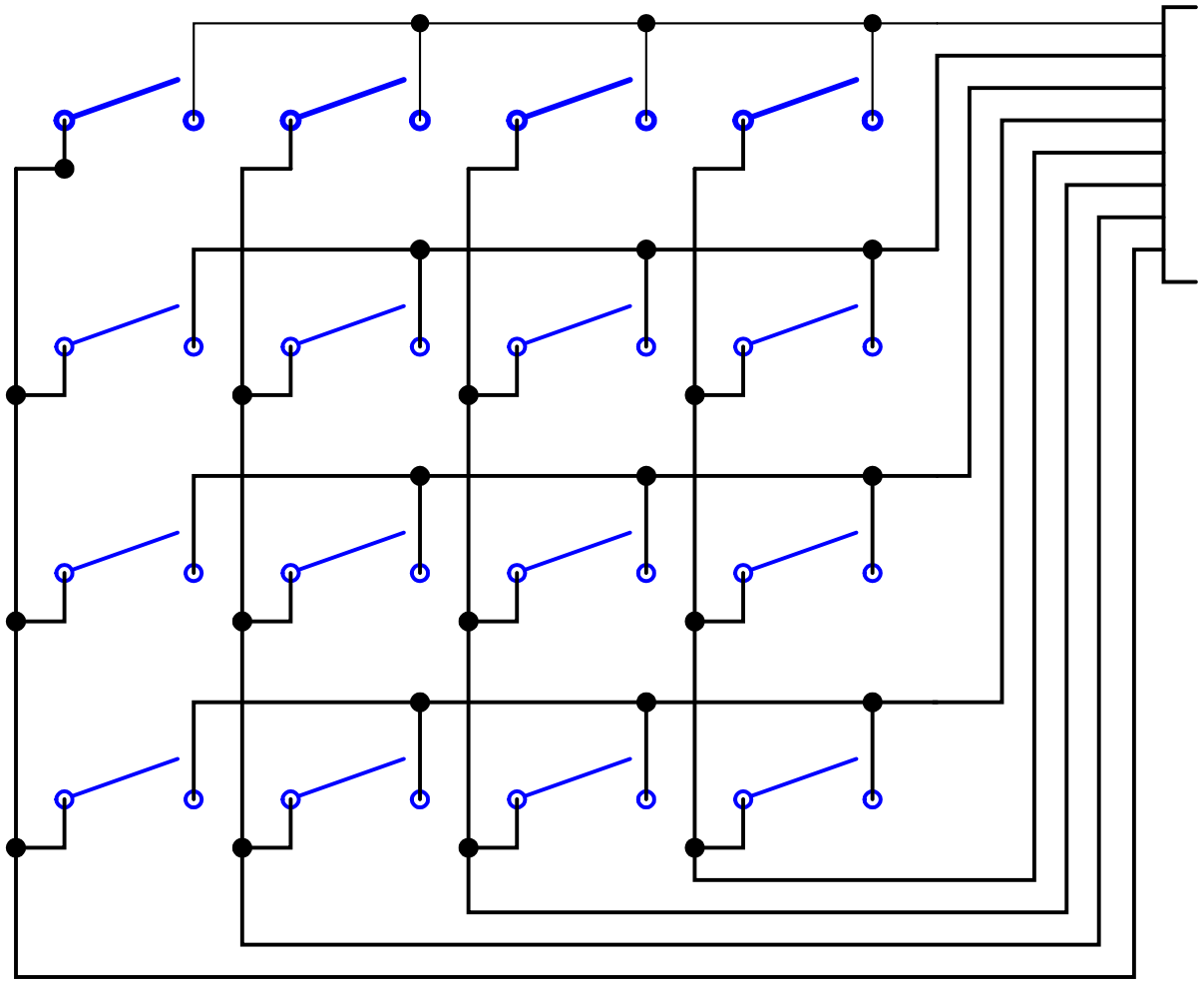
Top 4 Port bits are outputs

Bottom 4 Port bits are inputs

Alternately bring each pin of top 4 bits hi to determine which switch (which row) is pressed on bottom 4 bits.

Toggle Keypad

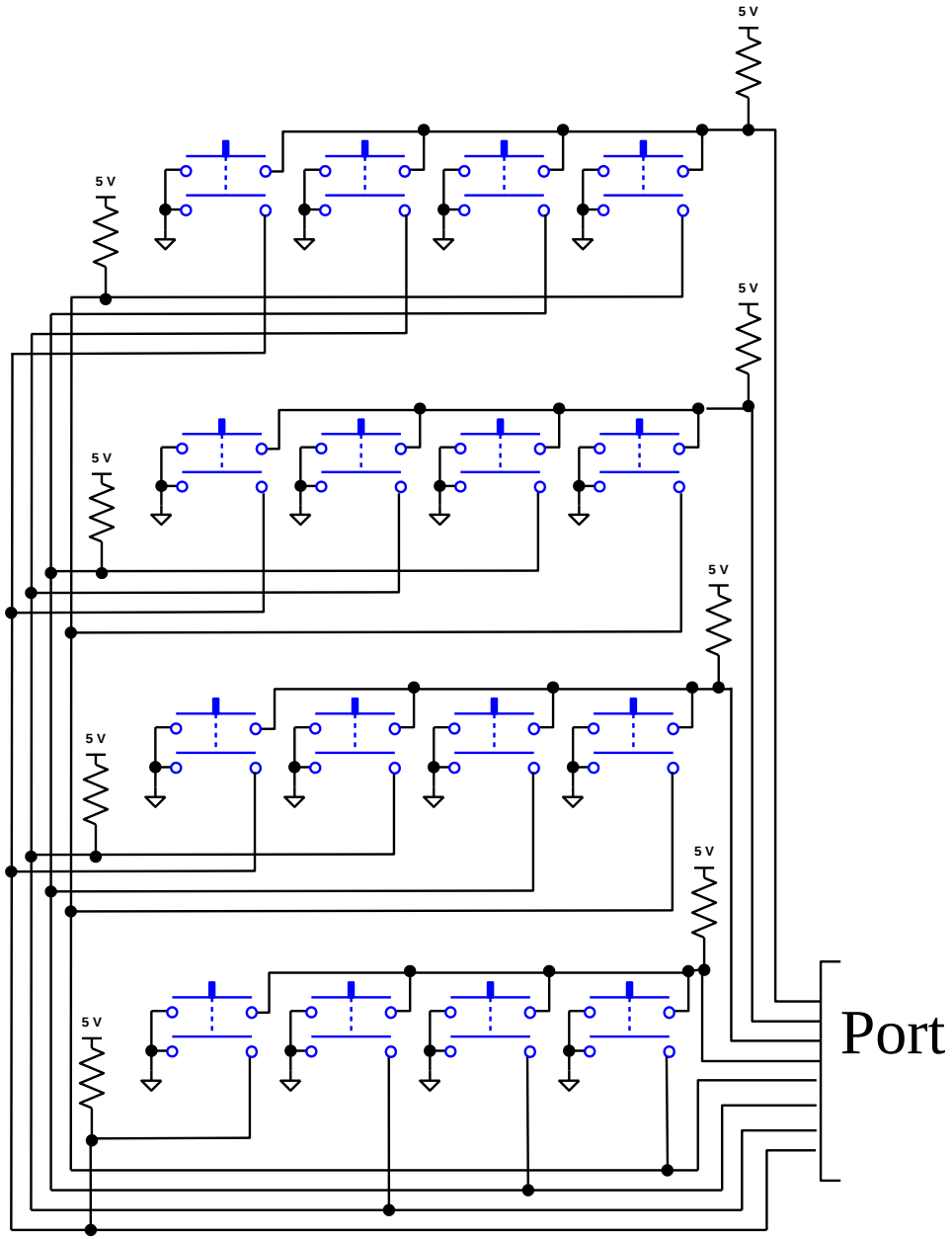
Single Pole - Single Throw Toggle



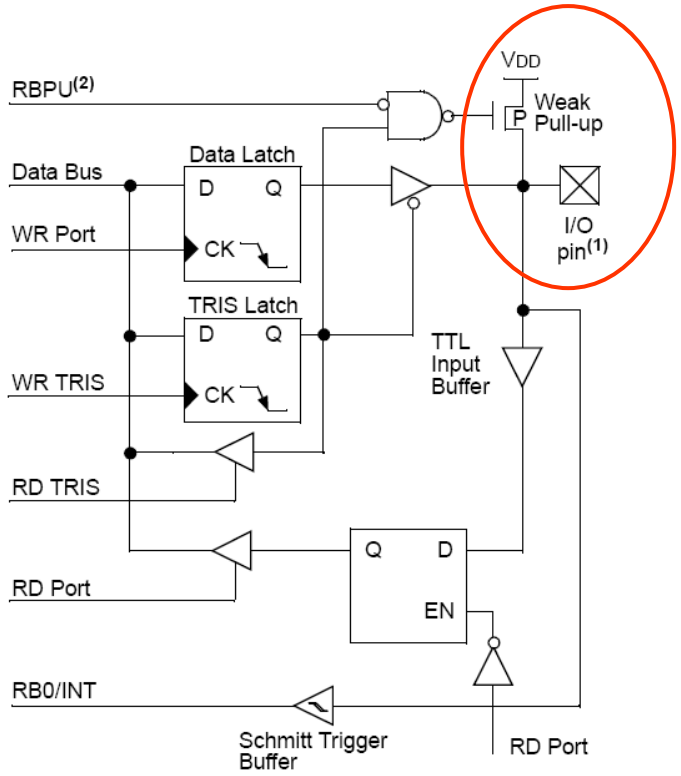
Reading the switches directly requires a double pole single throw keypad.

Note that switches have to be pulled up with a resistor tied to 5V (otherwise they "float" when open).

You read which row is lo on top 4 bits & which column is lo on bottom 4 bits.



Using Port B for a Keypad



Note 1: I/O pins have diode protection to VDD and VSS.

2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (OPTION_REG<7>).

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

"Option" register

You can set the pins on Port B to have "weak" pull-ups so that you do not need the external resistors.

Determine the Function of the Analog / Digital Ports

- For Ports A, B & E you have to determine if the pins are analog or digital and if digital, set the data direction - - input or output (analog is only input)
- Determine Analog or Digital using the register

ADCON1

- If Analog, determine the operation of the A/D using register

ADCON0

- If Digital, set the data direction in TRIS registers

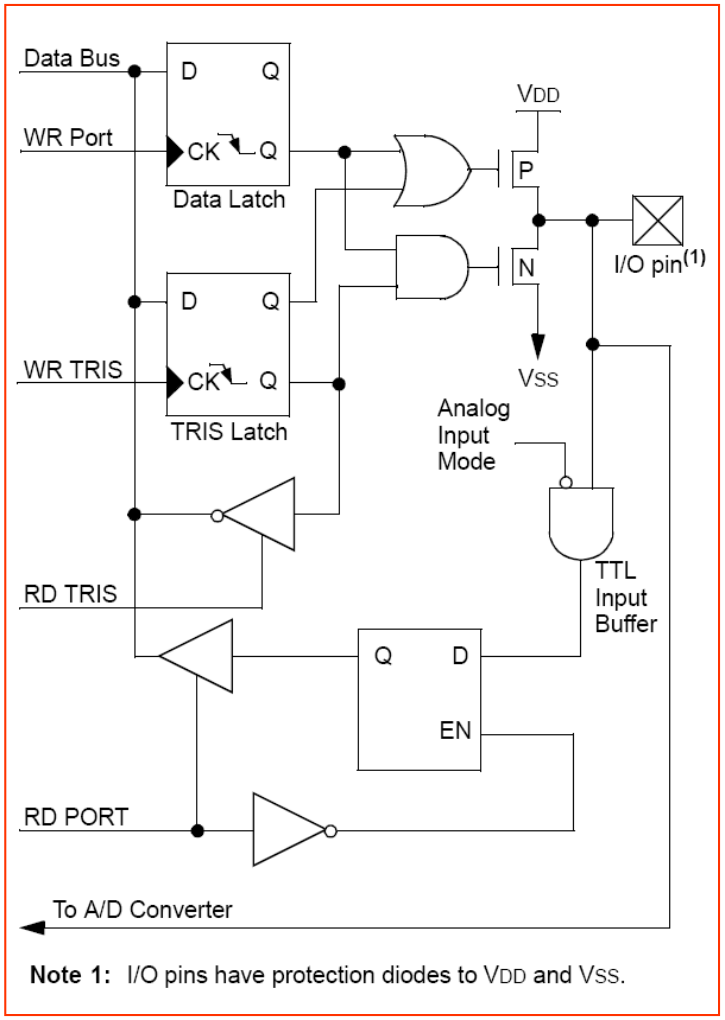
PORTA ⇔ TRISA
PORTB ⇔ TRISB
PORTE ⇔ TRISE

Port A & E

Port A & E Block Diagram

"Analog Input Mode" value determines if port pin is analog or digital

This is set in ADCON1



Rotate Assembler instructions divide or multiply a binary number by 2

rotate right ⇒ divide
rotate left ⇒ multiply

Both rotate instructions are through the carry bit (in STATUS register)

RRF Rotate Right f through Carry

Syntax: [*label*] RRF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

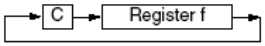
Operation: See description below

Status Affected: C

Encoding:

00	1100	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

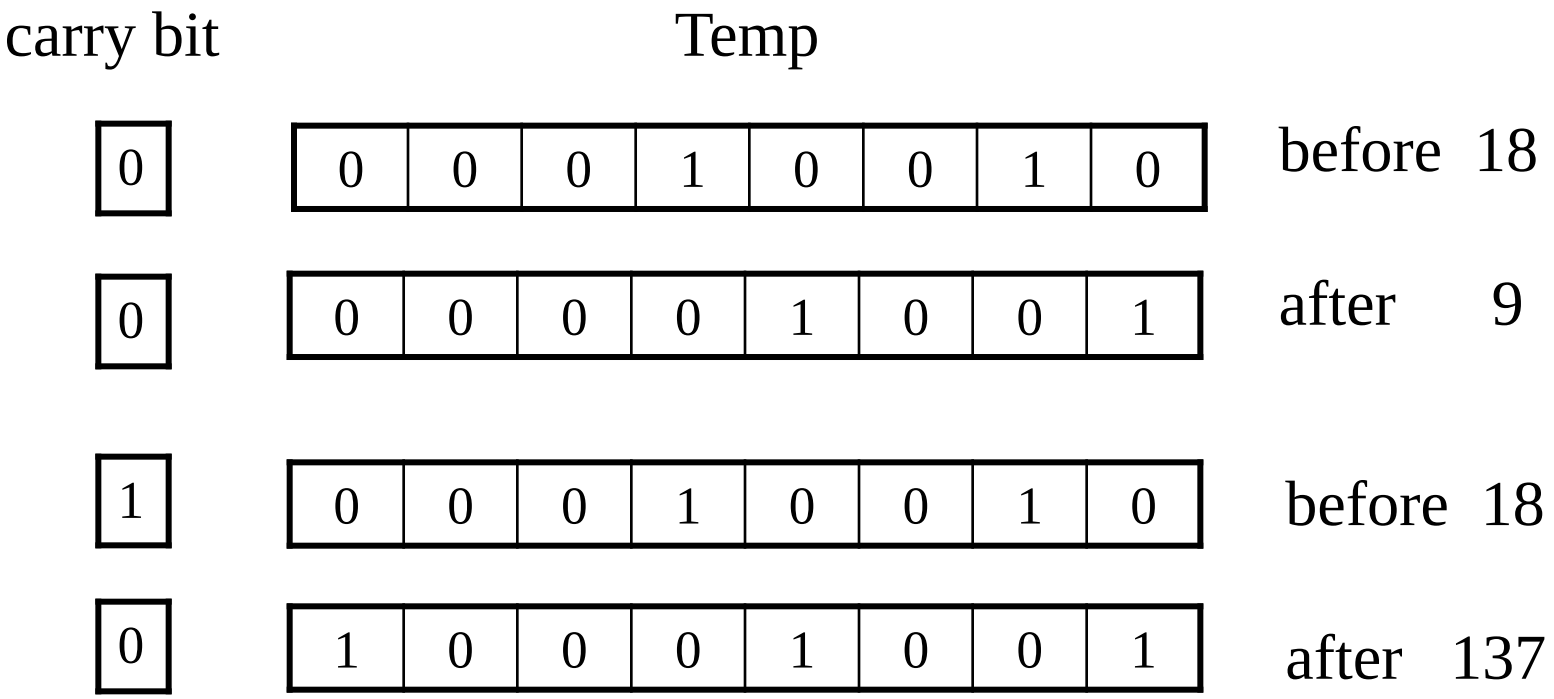
Example 1

```
RRF     REG1,0
```

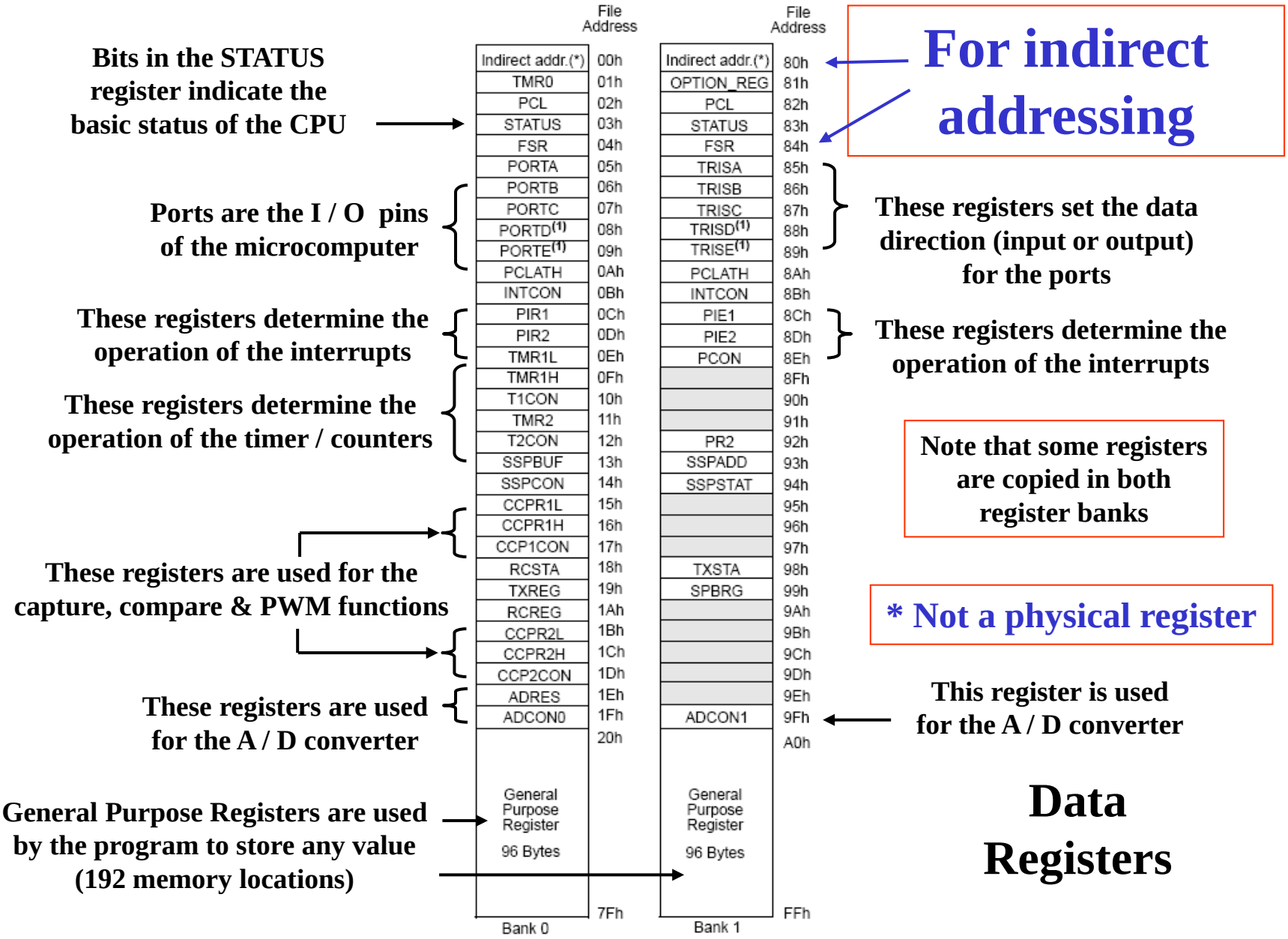
Before Instruction
REG1= 1110 0110
W = xxxx xxxx
C = 0
After Instruction
REG1= 1110 0110
W = 0111 0011
C = 0

Result Depends on the Carry Bit

rrf Temp, F



bcf STATUS, c


Bits in the STATUS register indicate the basic status of the CPU
Ports are the I / O pins of the microcomputer
These registers determine the operation of the interrupts
These registers determine the operation of the timer / counters
These registers are used for the capture, compare & PWM functions
These registers are used for the A / D converter
General Purpose Registers are used by the program to store any value (192 memory locations)
File Address
Indirect addr.(*) 00h
TMR0 01h
PCL 02h
STATUS 03h
FSR 04h
PORTA 05h
PORTB 06h
PORTC 07h
PORTD(1) 08h
PORTE(1) 09h
PCLATH 0Ah
INTCON 0Bh
PIR1 0Ch
PIR2 0Dh
TMR1L 0Eh
TMR1H 0Fh
T1CON 10h
TMR2 11h
T2CON 12h
SSPBUF 13h
SSPCON 14h
CCPR1L 15h
CCPR1H 16h
CCP1CON 17h
RCSTA 18h
TXREG 19h
RCREG 1Ah
CCPR2L 1Bh
CCPR2H 1Ch
CCP2CON 1Dh
ADRES 1Eh
ADCON0 1Fh
20h
General Purpose Register
96 Bytes
7Fh
Bank 0
File Address
Indirect addr.(*) 80h
OPTION_REG 81h
PCL 82h
STATUS 83h
FSR 84h
TRISA 85h
TRISB 86h
TRISC 87h
TRISD(1) 88h
TRISE(1) 89h
PCLATH 8Ah
INTCON 8Bh
PIE1 8Ch
PIE2 8Dh
PCON 8Eh
8Fh
90h
91h
PR2 92h
SSPADD 93h
SSPSTAT 94h
95h
96h
97h
TXSTA 98h
SPBRG 99h
9Ah
9Bh
9Ch
9Dh
9Eh
ADCON1 9Fh
A0h
General Purpose Register
96 Bytes
FFh
Bank 1
For indirect addressing
These registers set the data direction (input or output) for the ports
These registers determine the operation of the interrupts
Note that some registers are copied in both register banks
* Not a physical register
This register is used for the A / D converter
Data Registers

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)

For indirect addressing

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)
Each bank is 128 bytes

bit 4 **$\overline{TO}$**: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 z: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

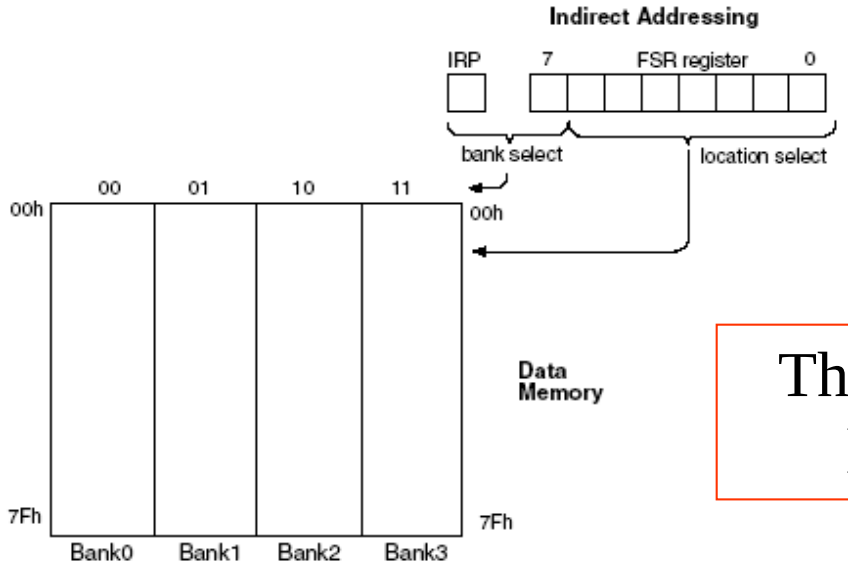
bit 1 **DC**: Digit carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C:** Carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

STATUS Register

Indirect Addressing

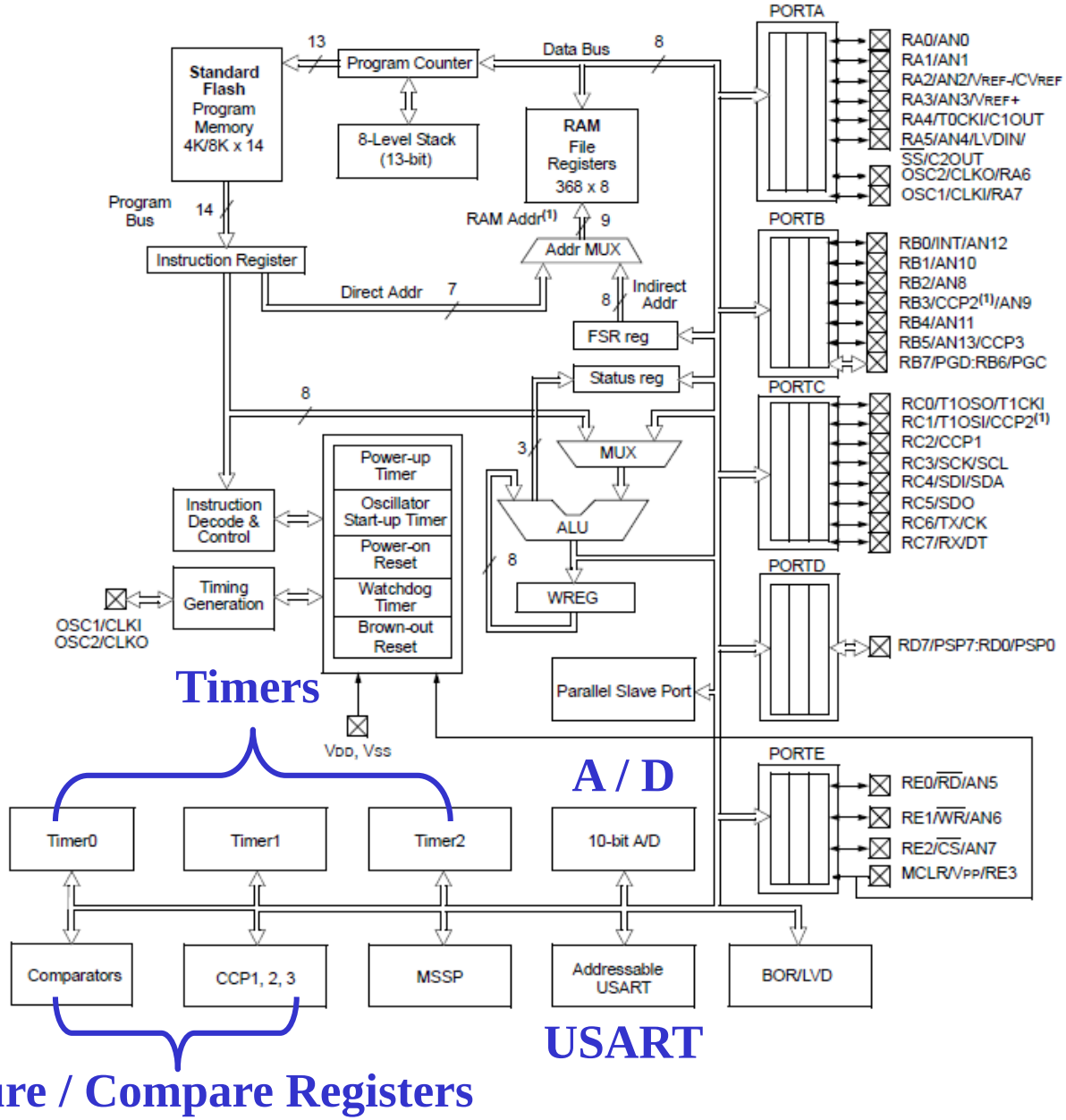


This code clears memory locations 20h to2Fh

```
Temp    EQU     020h        ; set up Temp register
Temp1   EQU     021h        ; set up Temp register
                etc.

        bcf     STATUS, IRP ; set indirect addressing bank 0/1
        movlw   Temp        ; initialize pointer to memory
        movwf   FSR         ; move to File Select Register
Next
        clrf    INDF        ; clear the INDF register
        incf    FSR,F       ; increment the pointer
        btfss   FSR,4       ; all done?
        goto    Next        ; not done, clear next location
Continue                    ; finished, keep going
```

or use a mask →

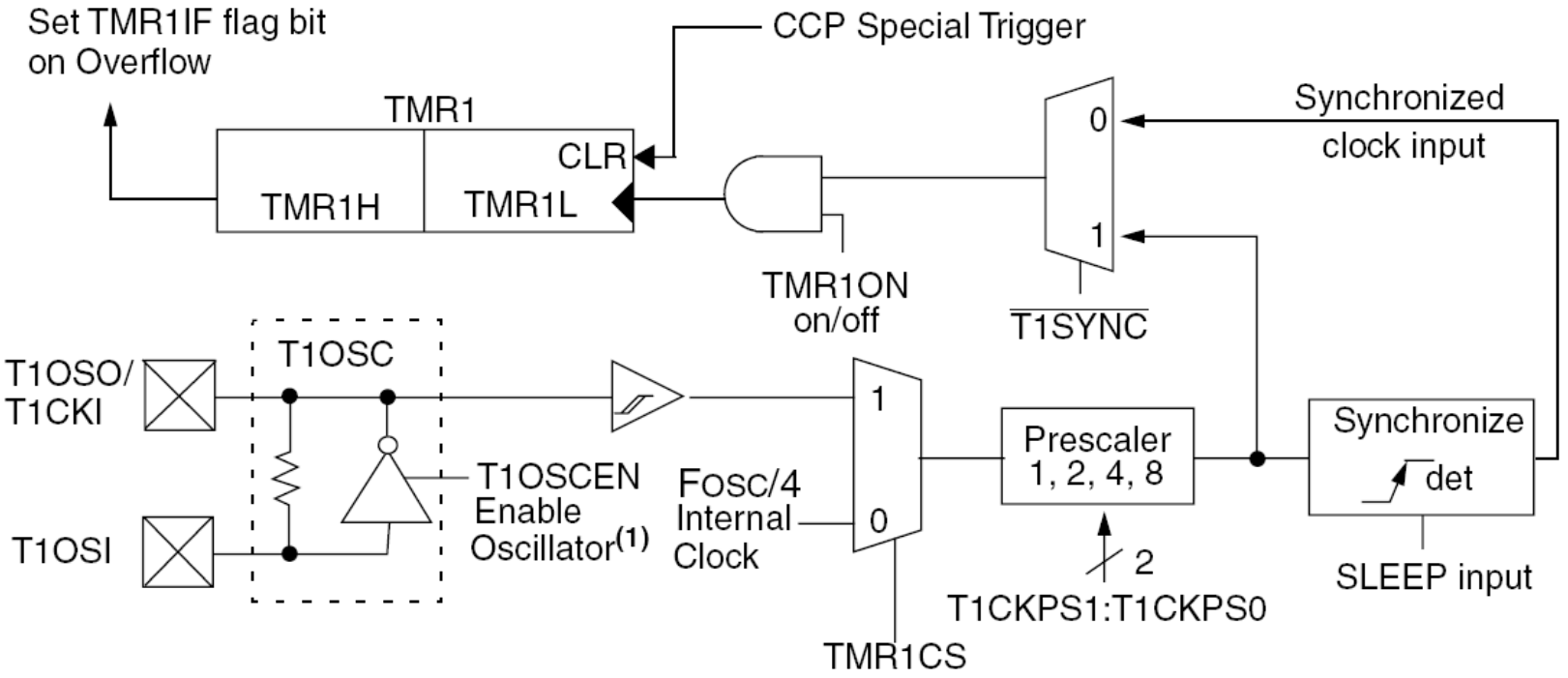
PIC16F747 Schematic



Timer 1

- Similar to the operation of the D flip-flop
- 16 bit timer (has two 8 bit registers)
- Increments from 0000h to FFFFh then rolls over to 0000h
- Source can be a pin or the microcontroller oscillator (incrementing every instruction cycle $F_{osc} / 4$)
  - pin source $\Rightarrow$ acts like a counter (synchronous or asynchronous)
  - oscillator source $\Rightarrow$ acts like a timer
- Can detect when it rolls over (also generates a processor interrupt)
- Can synchronize the pin source with the oscillator
- Can pre-scale the clock signal

Using Timer 1



Note 1: When the T1OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

Setting Up the Timer – T1CON

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN**: Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut-off (the oscillator inverter is turned off to eliminate power drain)

bit 2 **$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Control bit
TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON**: Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Also determines which input pin is used for counter (can also set to use external crystal)

Some Numbers

- Suppose you set Timer 1 to increment every 4 instruction cycles (1:4 prescaler)
  - 4 MHz oscillator $\Rightarrow$ 1 MHz instruction cycle $\Rightarrow$ 250 kHz timer clock cycle
- Suppose you want a ¼ second clock
  - 250 kHz clock $\Rightarrow$ 4 μsec period $\Rightarrow$ 62,500 counts
- A 16 bit counter can hold 2^{16} counts $\Rightarrow$ 65,536 counts
- 65,536 – 62,500 = 3036 counts $\Rightarrow$ 0BDC hex
- If you load the 2 timer registers (TMR1H & TMR1L) with 0B DC hex Timer 1 will overflow (and set flag) ¼ second later

PIR1 Register

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF(1):** Parallel Slave Port Read/Write Interrupt Flag bit
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion is completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag
1 = The SSP interrupt condition has occurred, and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
SPI
A transmission/reception has taken place.
I^2C Slave
A transmission/reception has taken place.
I^2C Master
A transmission/reception has taken place.
The initiated START condition was completed by the SSP module.
The initiated STOP condition was completed by the SSP module.
The initiated Restart condition was completed by the SSP module.
The initiated Acknowledge condition was completed by the SSP module.
A START condition occurred while the SSP module was IDLE (multi-master system).
A STOP condition occurred while the SSP module was IDLE (multi-master system).
0 = No SSP interrupt condition has occurred

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Note 1: PSPIF is reserved on 28-pin devices; always maintain this bit clear.

Timer 1 overflow bit

Capture / Compare / PWM **(CCP)** Module

- Capture – read a timer whenever a pin changes (or 4 or 16 changes)
- Compare – do something when a timer or counter value agrees with the number in a register
  - set or clear a pin
  - set an internal software flag
- PWM – (Pulse Width Modulation) output a quasi-analog signal to control something (DC Motor Case Study)
- Two CCP modules which are identical except for the operation of the special event trigger

CCP Resources

CCP Mode	Timer Resource
Capture	Timer1
Compare	Timer1
PWM	Timer2

Generic Name	CCP1	CCP2	Comment
CCPxCON	CCP1CON	CCP2CON	CCP control register
CCPRxH	CCPR1H	CCPR2H	CCP High byte
CCPRxL	CCPR1L	CCPR2L	CCP Low byte
CCPx	CCP1	CCP2	CCP pin

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	Same TMR1 time-base.
Capture	Compare	The compare should be configured for the special event trigger, which clears TMR1.
Compare	Compare	The compare(s) should be configured for the special event trigger, which clears TMR1.
PWM	PWM	The PWMs will have the same frequency, and update rate (TMR2 interrupt).
PWM	Capture	None
PWM	Compare	None

Setting Up the Compare – CCP1CON

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CCPxX:CCPxY**: PWM Least Significant bits

Capture mode:
Unused

Compare mode:
Unused

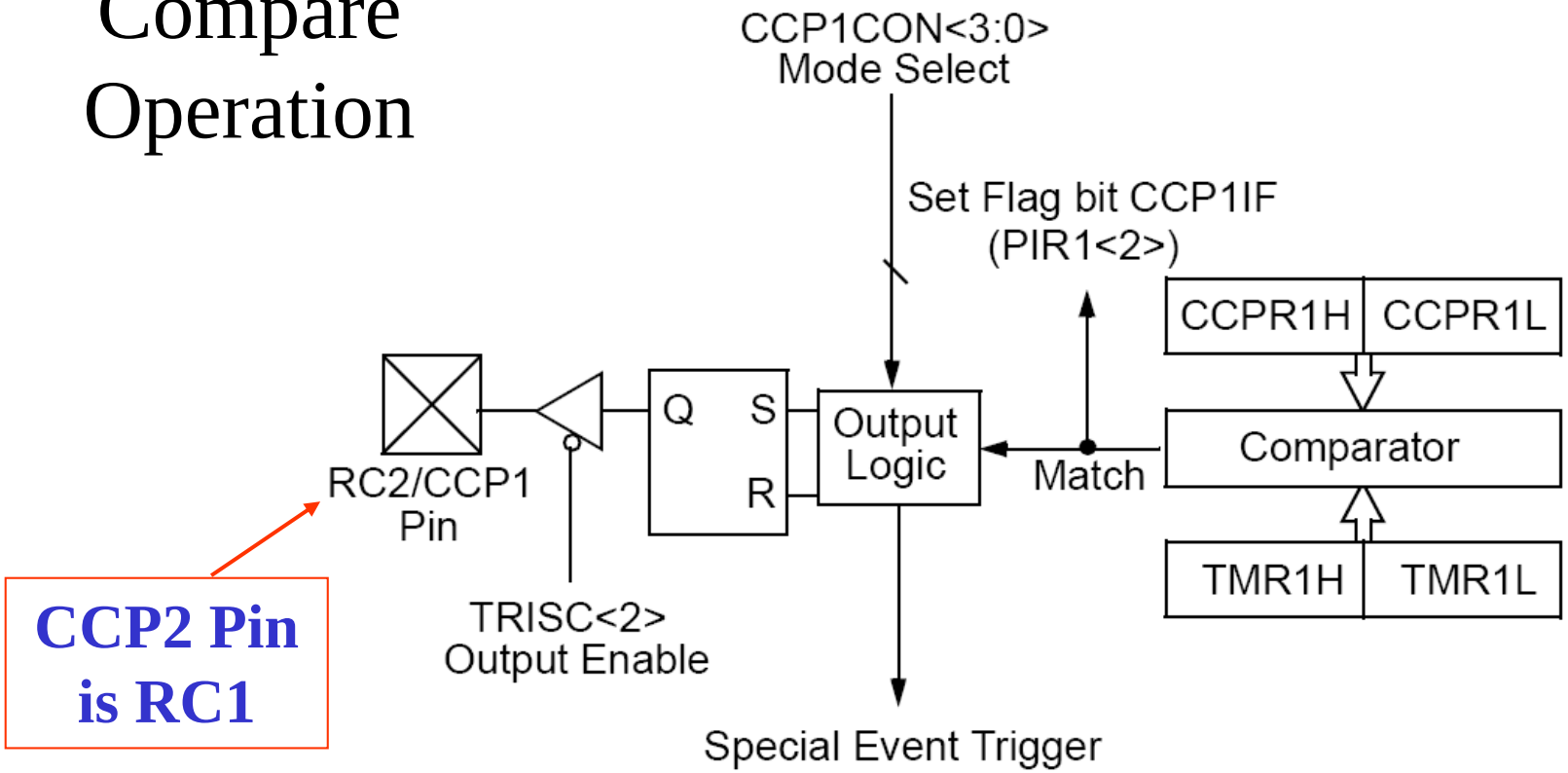
PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCPx module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (CCPxIF bit is set)
1001 = Compare mode, clear output on match (CCPxIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 clears Timer1; CCP2 clears Timer1 and starts an A/D conversion (if A/D module is enabled)
11xx = PWM mode

Unfortunately pin RC1 is affected

Compare Operation



Special Event Trigger will:

- clear TMR1H and TMR1L registers
- NOT set interrupt flag bit TMR1F (PIR1<0>)
- (for CCP2 only) set the GO/$\overline{\text{DONE}}$ bit (ADCON0<2>)

Find the Flag
PIR1 Register

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF(1):** Parallel Slave Port Read/Write Interrupt Flag bit
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion is completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag
1 = The SSP interrupt condition has occurred, and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
SPI
A transmission/reception has taken place.
I^2C Slave
A transmission/reception has taken place.
I^2C Master
A transmission/reception has taken place.
The initiated START condition was completed by the SSP module.
The initiated STOP condition was completed by the SSP module.
The initiated Restart condition was completed by the SSP module.
The initiated Acknowledge condition was completed by the SSP module.
A START condition occurred while the SSP module was IDLE (multi-master system).
A STOP condition occurred while the SSP module was IDLE (multi-master system).
0 = No SSP interrupt condition has occurred

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Note 1: PSPIF is reserved on 28-pin devices; always maintain this bit clear.

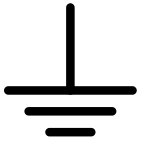
Power Supply

Voltages on the power supply are relative to this terminal. This is connected to the ground (also called “common”) of your circuits.

This terminal is connected to the ground (green wire) of the electrical outlet. For safety with high voltage & power, you often connect a wire between this ⏚ terminal and **COM**. In Mudd, the electrical outlet ground is poor and very noisy, so it is recommended that you **do not connect** this terminal to your circuit.

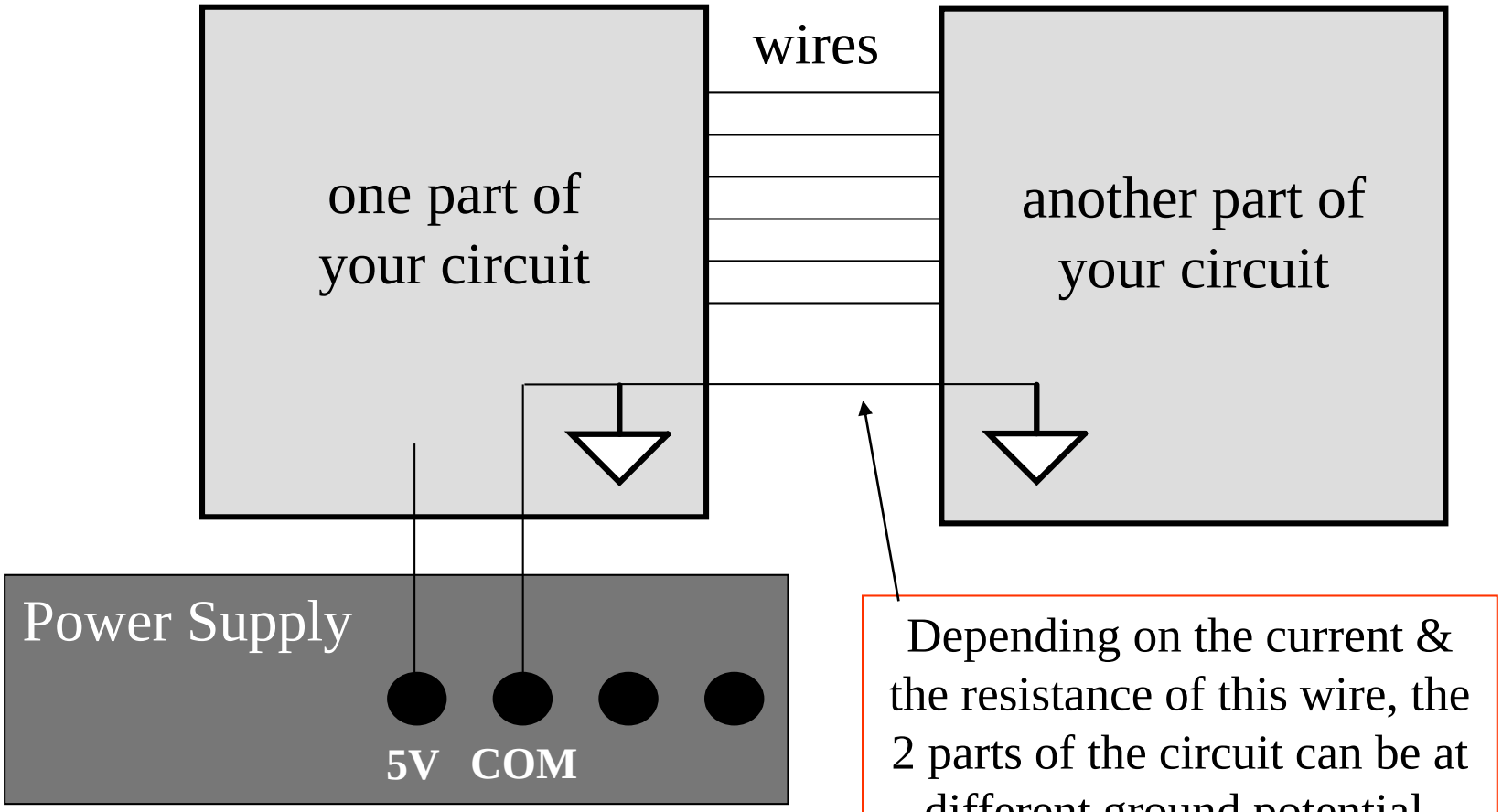
Ground

On schematics, I will draw ground like this. This typically indicates “signal ground”. It means that you should connect that point on the circuit to the common (**COM**) terminal on the power supply.



This symbol typically indicates “power ground”. On the power supply, it is physically connected to the round terminal on the outlet. The grounds in Mudd are terrible and you should not connect this power supply terminal to your circuit since it will only produce noise.

Ground Loops



Ground Loop
Serious Problem