MECE336 Microprocessors I
Clock and Delays

Dr. Kurtuluş Erinç Akdoğan

kurtuluserinc@cankaya.edu.tr

Course Webpage: http://MECE336.cankaya.edu.tr

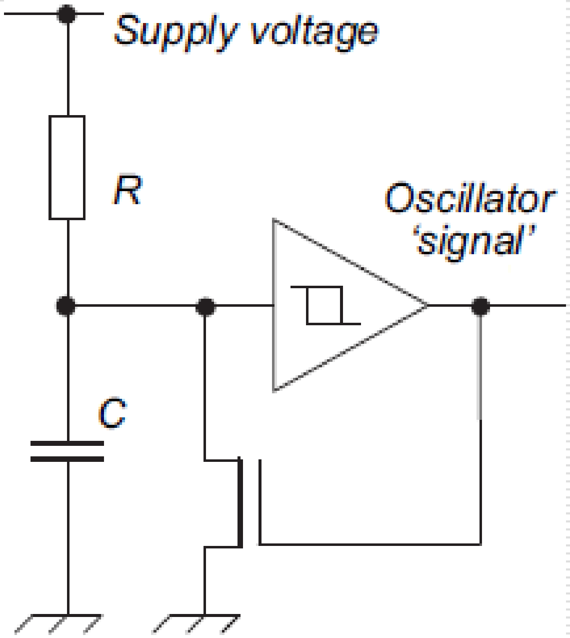
ÇANKAYA ÜNİVERSİTESİ
MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

The Clock Oscillator

- The choice of microcontroller clock source determines some of its fundamental **operating characteristics**.
- While 'faster is better' in terms of operating speed and program execution, faster is definitely worse in terms of **power consumption**, and also possibly in terms of **electromagnetic interference**.
- All timed elements within the microcontroller almost invariably depend on the **clock characteristics**.
- If **stable** and **accurate timing** is required, then the clock oscillator must be stable and accurate.
- With these points in mind, the clock source must be chosen with care and understanding.
- This section starts with a review of the clock technologies available, before moving on to looking at the options offered with the 16F84A.

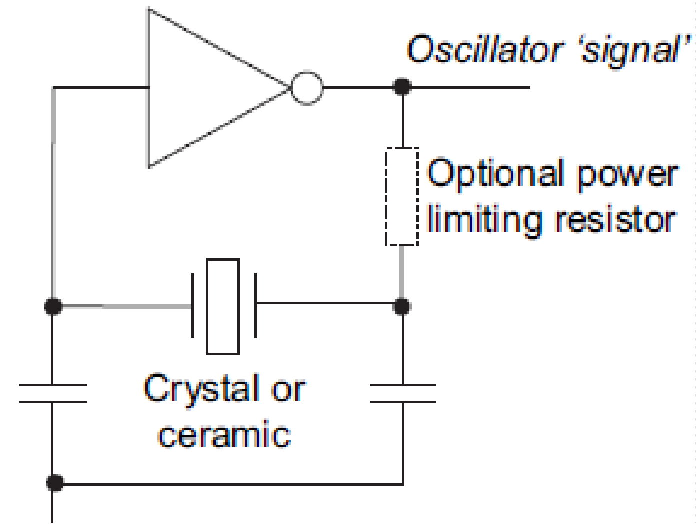
Clock Oscillator Types: Resistor–Capacitor RC Oscillator

- In **RC,** a **capacitor** is charged through a **resistor** from the supply rail. The capacitor voltage drives the input of a **Schmitt trigger buffer**.
- When the Schmitt trigger threshold is exceeded, its output goes high, switching on the MOSFET transistor to which it is connected. The capacitor is quickly discharged, the Schmitt output goes low, the MOSFET is switched off and the charging process starts again.
- This continues for as long as power is maintained. The **clock signal** is taken from the **rectangular waveform** generated at the Schmitt output.
- This simple circuit is integrated onto many larger ICs requiring a clock signal. Users are then usually required to connect resistor and capacitor externally, choosing these to set the desired frequency.
- It is important to note, however, that RC oscillators can be implemented entirely on-chip. They are very **low-cost** and produce a clock signal very **reliably**.
- As resistor, capacitor, power supply and Schmitt trigger threshold values all vary with temperature, their frequency is **not very stable**. They cannot therefore be used where precise timing is required.



Clock Oscillator Types: Crystal Oscillator

- The crystal oscillator depends on the piezo-electric properties of quartz crystal.
- Any mechanical distortion of the material causes a voltage to be produced across opposite sides of it; similarly, if a voltage is applied to the material, a mechanical distortion results.
- Crystals are carefully cut into very thin slices (usually discs), have tiny electrodes attached and are mounted so that they can vibrate.
- When connected in the feedback path across a logic inverter, as the figure shows, the crystal can be forced through piezo-electric action into mechanical vibration. This translates into electrical oscillation, an oscillation that is sustained by the action of the logic gate.



- Small-value capacitors connected from either side of the crystal to ground optimise the electrical conditions needed for this oscillation.
- Crystal vibration occurs at a fixed and remarkably stable frequency – this is the great advantage of the crystal oscillator.
- The crystals themselves tend to be on the expensive side (although cost continues to fall) and mechanically fragile.
- An alternative is the ceramic resonator. This has similar piezo-electric properties to the crystal and is connected in an identical way.
- It is, however, both lower in cost and rather less stable in frequency.
- **Crystals are the only option when precise timing functions, derived from the clock oscillator, are required.**

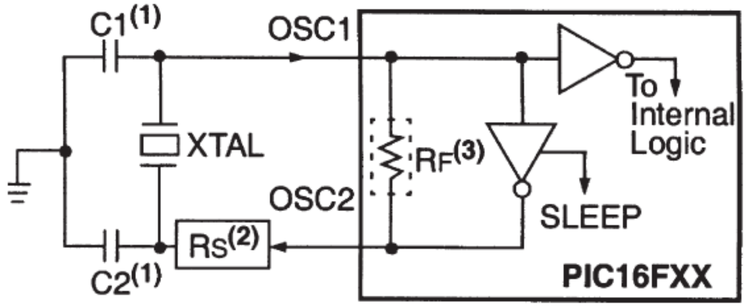
The 16F84A Clock Oscillator

- The 16F84A can be configured to operate in four different oscillator modes selected by setting bits in the **Configuration Word**.
- **XT – crystal**. is the standard crystal configuration. It is intended for crystals or resonators in the range 1–4 MHz.
- **HS – high speed**. is intended for crystal frequencies in the region of 4 MHz or greater, and/or ceramic resonators leading to the highest current consumption of all the oscillator modes.
- **LP – low power**. is intended for low-frequency crystal applications and gives the lowest power consumption possible. Mostly 32.768 kHz is suitable for low-power, time-sensitive applications, for example wristwatches. It will, however, operate at any frequency below around 200 kHz.
- **RC – resistor–capacitor**. For this an external resistor and capacitor must be connected to pin 16. This is the lowest-cost way of getting an oscillator, but should not be used when any timing accuracy is required.

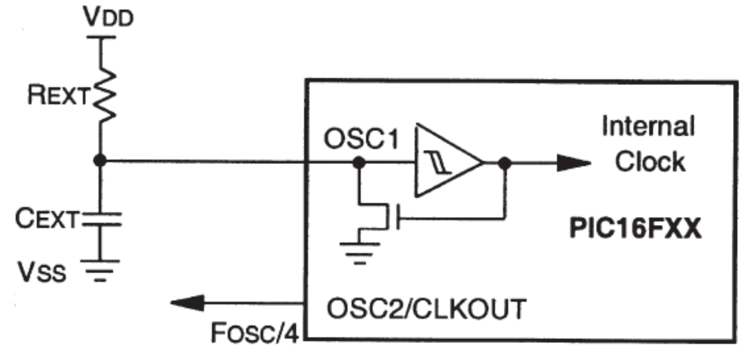
Clock Oscillator: Congurations

- The 16F84A has two oscillator pins, OSC1 (pin 16) and OSC2 (pin 15).
- Between these lies a logic inverter and associated circuitry.
- Either a crystal or a ceramic can be connected to create the oscillator circuit of Figure (a).
- An RC oscillator can also be used, as shown in Figure (b).
- Finally, an external clock source can simply be connected to the OSC1 pin (Figure (c)).

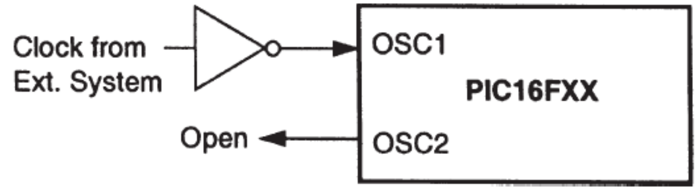
(a) Crystal or ceramic, HS, XT or LP.



(b) Resistor–capacitor.



(c) Externally supplied clock



Clock Oscillator: Component Selection Tables

Crystal

Mode	Freq	OSC1/C1	OSC2/C2
LP	32 kHz 200 kHz	68 - 100 pF 15 - 33 pF	68 - 100 pF 15 - 33 pF
XT	100 kHz 2 MHz 4 MHz	100 - 150 pF 15 - 33 pF 15 - 33 pF	100 - 150 pF 15 - 33 pF 15 - 33 pF
HS	4 MHz 20 MHz	15 - 33 pF 15 - 33 pF	15 - 33 pF 15 - 33 pF

Ceramic

Mode	Freq	OSC1/C1	OSC2/C2
XT	455 kHz 2.0 MHz 4.0 MHz	47 - 100 pF 15 - 33 pF 15 - 33 pF	47 - 100 pF 15 - 33 pF 15 - 33 pF
HS	8.0 MHz 10.0 MHz	15 - 33 pF 15 - 33 pF	15 - 33 pF 15 - 33 pF

Resistor-capacitor

Vdd = 5 Volt

R	C	F	Fosc/4
5 K	100 pF	5.4 MHz	1.3MHz
10 K	100 pF	3.0 MHz	756KHz
100 K	100 pF	328 KHz	82KHz

Power Supply

- A microcontroller is supplied at 5 V traditionally however supply voltages have been pushed down, and 3.3 and 3.0 V supplies are now common.
- Supply current will be dependent on **operating frequency**.
- Not to damage device, ‘absolute maximum ratings’, showing voltage and power dissipation level should not be applied.
- Operating Condition of the PIC 16F84A
  - According to data sheet: supply voltage between 4.0 and 5.5 V (suitable for three AA cells)
  - At least 4.5 V in HS oscillator mode
  - Drop down to 1.5 V without loosing data in RAM in **sleep mode**
- Supply Current
  - About 1.8 mA when running at 4 MHz with supply voltage 5.5 V
  - About 10 mA when running at 20 MHz with supply voltage 5.5 V
  - Low power consumptions for low-power device PIC 16LF84A: 15 μA
- Taking account of only the consumption of the microcontroller, a system powered with three AA cells (4.5 V), each with nominal capacity of 800 mAh.
  - Running at 1.8 mA would give a battery life of 444 hours, or 18.5 days.
  - Running at 10 mA would give 80 hours, or 3.3 days,
  - Running at 15 μA consumption would lead to 53 333 hours, equivalent to 2222 days or **just over six years!**

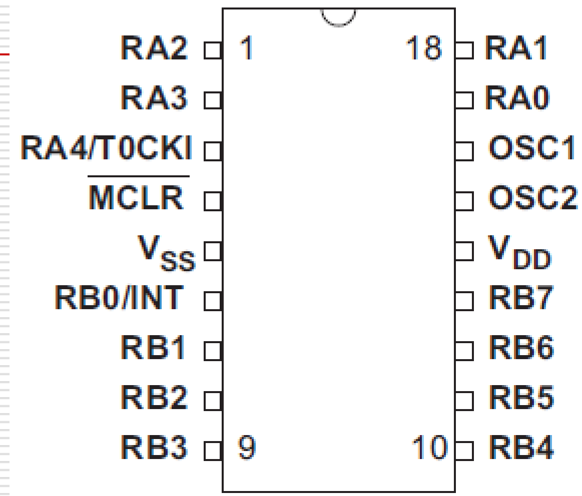
The 16F84A Basic Operating Conditions

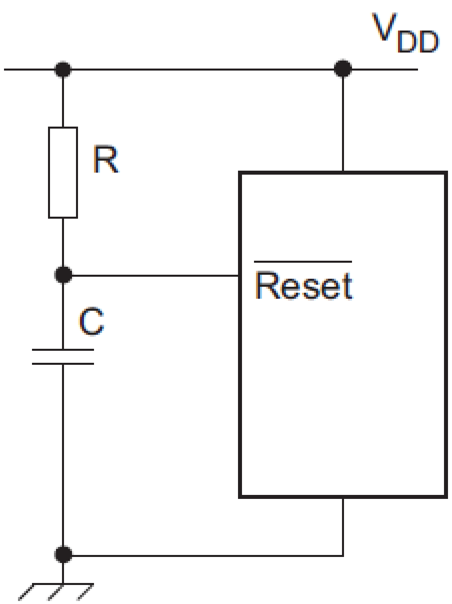
Param No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
	VDD	**Supply Voltage**					
D001		16LF84A	2.0	—	5.5	V	XT, RC, and LP osc configuration
D001 D001A		16F84A	4.0 4.5	— —	5.5 5.5	V V	XT, RC and LP osc configuration HS osc configuration
D002	VDR	**RAM Data Retention Voltage (Note 1)**	1.5	—	—	V	Device in SLEEP mode
D003	VPOR	**VDD Start Voltage** to ensure internal Power-on Reset signal	—	Vss	—	V	See section on Power-on Reset for details
D004	SVDD	**VDD Rise Rate** to ensure internal Power-on Reset signal	0.05	—	—	V/ms	
	IDD	**Supply Current (Note 2)**					
D010		16LF84A	—	1	4	mA	RC and XT osc configuration **(Note 3)** FOSC = 2.0 MHz, VDD = 5.5V
D010		16F84A	—	1.8	4.5	mA	RC and XT osc configuration **(Note 3)** FOSC = 4.0 MHz, VDD = 5.5V
D010A			—	3	10	mA	RC and XT osc configuration **(Note 3)** FOSC = 4.0 MHz, VDD = 5.5V (During FLASH programming)
D013			—	10	20	mA	HS osc configuration (PIC16F84A-20) FOSC = 20 MHz, VDD = 5.5V
D014		16LF84A	—	15	45	µA	LP osc configuration FOSC = 32 kHz, VDD = 2.0V, WDT disabled

Note 1: This is the limit to which V_{DD} can be lowered without losing RAM data.
Note 2: Gives further information on factors that influence supply current.
Note 3: Gives guidance on how to calculate current consumed by the external RC network, when this is used.

Reset: Power-up

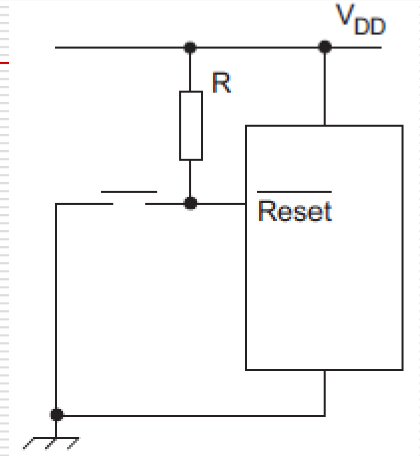
- When the microcontroller powers up, power-up must be detected and force the Program Counter to zero to start running its program from its beginning
- Along with this, it is also very useful to set SFRs so that peripherals are initially in a safe and disabled state.
- This 'ready-to-start' condition is called 'Reset'. The CPU starts running its program when it leaves the Reset condition.
- In the 16F84A there is a Reset input, MCLR ('Master Clear'), on pin 4.
- As long as this is held low, the microcontroller is held in Reset. When it is taken high, program execution starts. If the pin is taken low while the program is running, then program execution stops immediately and the microcontroller is forced back into Reset mode.
- Prior to resetting, to adjust the starting time of program, RC circuit is used to stabilize the embedded system since power supply and the clock oscillator take a finite amount of time to stabilise, and in a complex system power to different parts of the circuit may become stable at different times.





Reset: Power-up

- 16F84A includes some clever on-chip reset circuitry, which in many situations makes the components of Figure (a) or (b) unnecessary.
- A Power-up Timer is included on-chip, which can be enabled by the user with bit 3 of the Configuration Word.
- The 16F84A detects that power has been applied and the Power-up Timer then holds the controller in Reset for a fixed time.
- The circuit of Figure (b) need only be applied if the supply voltage rises very slowly.
- if we don’t want to make use of 16F84A MCLR input then it is essential to recognise that this input must not just be left unconnected.
- The simplest thing to do is to tie it to the supply rail and then forget about it.
- If external reset is desired, add User Reset Button shown in figure.



bit 13-4 **CP:** Code Protection bit
1 = Code protection disabled
0 = All program memory is code protected

bit 3 **$\overline{PWRTE}$**: Power-up Timer Enable bit
1 = Power-up Timer is disabled
0 = Power-up Timer is enabled

bit 2 **WDTE**: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **FOSC1:FOSC0**: Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

Configuration Word

R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	$\overline{PWRTE}$	WDTE	F0SC1	F0SC0
bit13													bit0

MPASM's CONFIG Directive

- Microchip's assembler, MPASM, has a nice feature that allows you to specify, in the source code file, the selected states of the configuration bits for this program.

Using the CONFIG Directive, a Source File Template

```
     LIST   p = p16C77      ; List Directive,
;    Revision History
;
     #INCLUDE    <P16C77.INC>    ; Microchip Device Header File
;
     #INCLUDE    <MY_STD.MAC>    ; File which includes my standard macros
     #INCLUDE    <APP.MAC>       ; File which includes macros specific
                                 ;   to this application
;
; Specify Device Configuration Bits
;
     __CONFIG    _XT_OSC & _PWRTE_ON & _BODEN_OFF & _CP_OFF & _WDT_ON
;
     org   0x00             ; Start of Program Memory
RESET_ADDR   :              ; First instruction to execute after a reset


     end
```

__CONFIG Directive Symbols (From Microchip Header Files)

```
__CONFIG    _XT_OSC & _PWRTE_ON & _BODEN_OFF & _CP_OFF & _WDT_ON
```

- For the symbols available for your device,please refer to that device’s Microchip Include file.
- As long as the correct device is specified (in the LIST and INCLUDE file directives), the correct polarity of all bits is ensured.

Feature	SYMBOLS
Oscillators	_RC_OSC
	_EXTRC_OSC
	_EXTRC_OSC_CLKOUT
	_EXTRC_OSC_NOCLKOUT
	_INTRC_OSC
	_INTRC_OSC_CLKOUT
	_INTRC_OSC_NOCLKOUT
	_LP_OSC
	_XT_OSC
	_HS_OSC
Watch Dog Timer	_WDT_ON
	_WDT_OFF
Power-up Timer	_PWRTE_ON
	_PWRTE_OFF
Brown-out Reset	_BODEN_ON
	_BODEN_OFF
Master Clear Enable	_MCLRE_ON
	_MCLRE_OFF
Code Protect	_CP_ALL
	_CP_ON
	_CP_75
	_CP_50
	_CP_OFF
Code Protect Data EEPROM	_DP_ON
	_DP_OFF
Code Protect Calibration Space	_CPC_ON
	_CPC_OFF

Delay: Basic Information

- The overall speed of the microcontroller operation is entirely dependent on this frequency of clock which is a continuously running fixed-frequency logic square wave.
- Microchip use 'instruction cycle' to define primary unit of time in the action of the processor, for example being used as a measure for how long an instruction takes to execute.
- In PIC midrange microcontrollers the main oscillator signal is divided by four to produce the instruction cycle time.
- For the fastest clock frequency, 20 MHz, the instruction cycle frequency is 5 MHz, with a period of 200 ns.
- The slightly cheaper version of the controller, the 16F84-04, with maximum clock frequency of 4 MHz, has at this frequency an instruction cycle time of 1 μs.
- A popular clock frequency for very low-power applications, including wristwatches, is 32.768 kHz. This has an instruction cycle period of 122.07 μs. The result is very low power, but strictly no high-speed calculations!
- On PIC 16F84A, most instructions take **1 instruction cycle** Exceptions (see instruction set)
  - GOTO, CALL, RETURN, RETFIE RETLW (2 instruction cycles)
  - DECFSZ, INCFSZ, BTFSC, BTFSS (2 instruction cycles if skip, 1 instruction cycle otherwise)

PIC mid-range instruction cycle durations for various clock frequencies

Clock frequency	Instruction cycle	
	Frequency	Period
20 MHz	5 MHz	200 ns
4 MHz	1 MHz	1 μs
1 MHz	250 kHz	4 μs
32.768 kHz	8.192 kHz	122.07 μs

Generating Time Delays And Intervals

- A recurring theme of embedded systems is how we deal with time – how systems respond in a timely way to external events and how they can measure time and generate time delays.
- The initial concept is simple.
  - A memory location is set up to act as a counter, loaded with a certain value and then
  - decremented repeatedly in a loop until it reaches zero.
  - The time taken will depend on the number first placed in the counter and then the time taken for each program loop.
- To implement accurate delays crystal oscillator gives a frequency of excellent accuracy and stability.

```
;Delay of 5ms approx. Instruction cycle time is 5us.
delay5  movlw     D'200'       ;200 cycles called,each taking 5x5=25us
        movwf     delcntr1
del1    nop                    ;1 inst. cycle
        nop                    ;1 inst. cycle
        decfsz    delcntr1,1   ;1 inst. cycle, when no skip
        goto  del1             ;2 inst. cycles
        return
```

moving a number into a memory location (movlw, movwf)

The actual delay loop (del1 … goto del1)

Two nop instructions do nothing at all but take up time

DECFSZ fileReg, d: Decrement the content of f register skip if 0

Destination, d

If d=0, result is placed in WREG

If d=1, result is placed in file register

Delay: Computation

Basic Delay Loop

```
counter   equ      0x0C
          movlw    D'200'
          movwf    counter
loop      nop
          nop
          decfsz   counter,1
          goto     loop

          end
```

Explanation

Decfsz: *if number of register is 0 take 2 instruction cycle, else take 1 instruction cycle.*

Result

- 199 iterations with 5 instruction cycles (including GOTO)
- 1 iteration with 4 instruction cycles (no GOTO in last iteration)
- Total: $199 \cdot 5 + 4 = 999$ instruction cycles from `loop` to `end`
- Assume clock frequency 4 MHz $\rightarrow$ delay is $999 \cdot 1\mu s = 999\ \mu s$

Delay: Different Oscillator Frequencies

Same Delay Loop as Above | **Notes**

```
counter   equ     0x0C
          movlw   D'200'
          movwf   counter
loop      nop
          nop
          decfsz  counter,1
          goto    loop

          end
```

Result

- Total: $199 \cdot 5 + 4 = 999$ instruction cycles from `loop` to `end`
- Assume clock frequency 4 MHz → delay is $999 \cdot 1\mu s = 999\,\mu s$
- Assume clock frequency 20 MHz → delay is $999 \cdot 0.2\,\mu s = 199.8\,\mu s$
- Assume clock frequency 100 kHz → delay is $999 \cdot 40\,\mu s = 39.96\,ms$

→ A different delay loop has to be used for each oscillator frequency

Delay: Example

- Turn on/off PORTB with a frequency of 1 kHz

```
list p=16f84a;
include "p16f84a.inc";
__config _CP_OFF&_WDT_OFF&_XT_OSC;
org 0;
main;
clrf PORTB;
bsf STATUS, RP0;
clrf TRISB;
bcf STATUS, RP0;
bsf PORTB,0;
counter   equ      0x0C
          movlw    D'200'
          movwf    counter
loop      nop
          nop
          decfsz   counter,1
          goto     loop
bcf PORTB,0;
end;
```

Delay: General Formulation of a Single Delay Loop

General Delay Loop **Notes**

```
counter   equ      counterAddress
nIt       equ      N
          movlw    nIt
          movwf    counter
loop      nop
            ⋮
          nop
          decfsz   counter,1
          goto     loop
          nop
          end
```

Result

- Assume loop contains k `nop` instructions and oscillator frequency f
  $\Rightarrow (k+3)\cdot(N-1)+k+2+1=(k+3)\cdot N$ instruction cycles
  $\Rightarrow$ Delay of $(k+3)\cdot N\cdot 4/f$ between `loop` and `end`

Delay: Example Computations

- Realize a delay of 1 ms for an oscillator frequency of f = 20MHz
- Realize a delay of 1 ms for an oscillator frequency of f = 100 kHz
- Realize a delay of 100 ms for an oscillator frequency of f = 100 kHz
- Realize a delay of 100 ms for an oscillator frequency of f = 20MHz

Cascaded Delay Loops

- Assume there are k_1 `nop` instructions in loop 1
- Assume there are k_2 `nop` instructions in loop 2
- Assume the oscillator frequency is f
- Number of instruction cycles of inner loop as before using N_2 and k_2
  $C_2 = (k_2 + 3) \cdot N_2$ instruction cycles in loop 2
- Number of instruction cycles of outer loop
  $C_1 = (k_1 + C_2 + 5) \cdot N_1$ instruction cycles in loop 1
- Overall delay:

$$C_1 \cdot \frac{4}{f} = (k_1 + C_2 + 5) \cdot N_1 \cdot \frac{4}{f} = (k_1 + (k_2 + 3) \cdot N_2 + 5) \cdot N_1 \cdot \frac{4}{f}$$

```
list       p=16f84a
include    "p16f84a.inc"

org        0
counter1   equ       counterAddress1
counter2   equ       counterAddress2
nIt1       equ       N1
nIt2       equ       N2
movlw      nIt1
movwf      counter1
loop1      nop
           :
           nop
           movlw     nIt2
           movwf     counter2
loop2      nop
           :
           nop
           decfsz    counter2,1
           goto      loop2
           nop
           decfsz    counter1,1
           goto      loop1
           nop
           end
```

Cascaded Delay Loops: Example

- Let f = 4MHz. Determine suitable k1; k2;N1;N2 for 100 ms delay
- Let f = 125 kHz. Determine suitable k1; k2;N1;N2 for 1.6 s delay