

PIC Programming

by Nicholas Regan

FILE	12125_NICHOLAS_REGAN_PIC_PROGRAMMING_153432_659596483.DO CX (765.19K)		
TIME SUBMITTED	31-JAN-2019 03:27PM (UTC+0000)	WORD COUNT	4007
SUBMISSION ID	100071652	CHARACTER COUNT	20791

Title:
PIC Programming

Module:
Micros, Peripherals and Interfaces
ACDA4004

Electronics and Electrical Engineering HND

University of Wales Trinity St David

By: Nick Regan
Student No: 1803871
Date: 31.1.19

Declaration

The work of this report completed under the supervision of: Dr Stephen Mosey.

I declare that this report is the product of my own work and is not being submitted for any other degree etc.

Signed: N.S. L.

Date: 31.1.19

Contents

Subject	Page Number
Introduction	1
1.1 Explanation of MPLAB IDE	1
Illustration of MPLAB IDE	2
1.2 Components of the MPLAB IDE	2
2. The Assignment Board	4
3. Count Errors	6
4. Count Help	7
5. Count Sound	9
References	11
Figure List	12
Further Reading	12
List of Appendices	13

Introduction

This assignment covers reports of tasks using PIC programming. The tasks involved the use of MPLAB IDE, PICSTART PLUS and a PIC16F84 Assignment Board.

The features and components of the MPLAB IDE are described. The circuit diagram of the Assignment Board is illustrated and described. Reports of the tasks using the Assignment Board are accompanied and illustrated by flow charts.

1. Explanation of MPLAB IDE

In the booklet provided in the module tutorial, Young has described MPLAB as “a programme development suite called an “IDE” which is short for “Integrated Development Environment”. [1]. He explains that It helps people develop programmes for the PIC series of microcontroller integrated circuits. [1] It is produced by Arizona Microchip and can be downloaded free as it is a freeware product.

Bates describes it as “a primary development system toolset for PIC applications.” [2 p62], and Tomar explains further, it is a “large suite of software and hardware development tools integrated within one software package.....development of embedded application on microchip PIC controllers.” [3].

In basic terms, according to Microchip in its review of the product, [4], it is a 32 bit application that can be used on up to date versions of Windows. It requires 16MB memory, ideally 32MB and 45MB of hard disk space. It enables the user to create software development projects, embedded applications, for use with Microchip's PIC. It contains components, several tools which can be used for development and debugging. For example, create source files, group them into a project, assemble file and link source code, debug source code and test it using built in simulators or in-circuit emulators or in-circuit debuggers.

Features (as listed by Tomar in his overview of the product: [3].)

- Create and edit source code using the built-in editor.
- Assemble, compile and link source code.
- Debug the executable logic by watching program flow with the built-in simulator or in real time with in-circuit emulators or in-circuit debuggers.
- Make timing measurements with the simulator or emulator.
- View variables in Watch windows.
- Program firmware into devices with device programmers

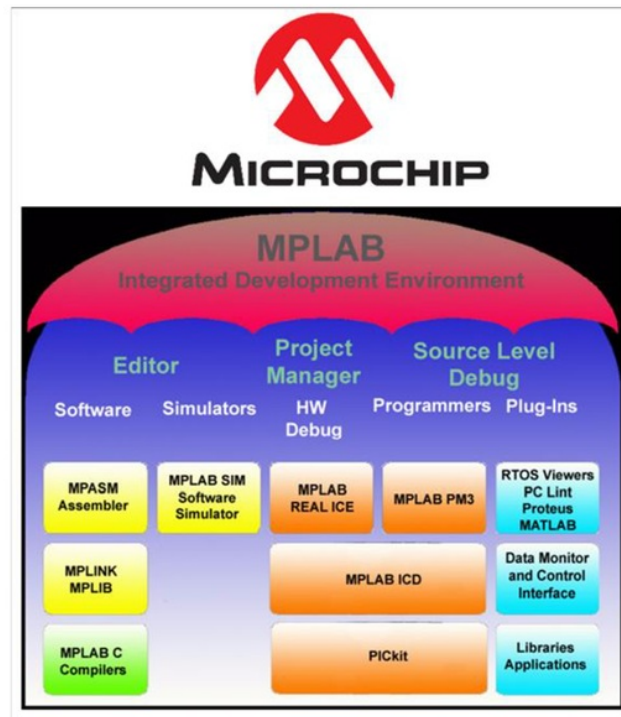


Figure 1: Diagram of MPLAB IDE

1.1 Components of MPLAB IDE

Project Manager

Tomar introduces the project manager as providing “integration and communication between the IDE and language tools”, [3].

The Project Manager organises files, (files to be edited and those connected to them). The files at the beginning of the process are called ‘source files’ and are written according to the rules of the assembler and compiler. They are then converted into “modules of machine code and references to functions and data storage. The linker resolves these placeholders and combines all the modules into a file of executable machine code,” [5]. The linker also produces a file for debugging, and makes it possible for the executing machine codes to be connected back to the source file.

The code is written by a text editor, part of another component known as the Editor.

Editor

Tomar gives a brief description by writing that the editor “programmes the text editor and also acts as a window into the debugger”, [3].

As mentioned above, the code is written by the text editor. The text editor uses colour coding for identification of various parts. The text editor in MPLAB IDE is designed especially for use with Microchip MCU's.

The code is written and the Editor (together with other tools) is able to display code execution in the debugger.

Language Tools

The language tools include the assembler, compiler and linker, to assemble, compile and link source code. The assembler can be used alone for a single file, or with the linker for several files. [5]. Any errors found by the assembler, compiler and linker can be viewed in a results window. [6].

Debugger

According to Microchip in their Quick Start Guide, [5], the debugger allows “breakpoints, single stepping, watch window and all the features of a modern debugger.... Works in conjunction with the Editor to reference information from the target being debugged back to the source code. “

The code that is being produced is tested by using the debugger. MPLAB IDE have software programmes included which simulate the microprocessor that the code is intended for. There are also hardware debuggers that can be used, known as in-circuit emulators (ICE) to act as the target microprocessor. The process is necessary as it allows the user to view variables regularly, at each step, so that changes can be made to correct or improve. [5].

Execution Engines

. Tomar lists the tasks of the execution engines in the overview of MPLAB IDE: [3]

- They use the PC to simulate the instructions and some peripheral function of the PIC Micro devices.
- Create and edit source code using the built in editor Assembler compile and link source code.
- Debug the executable logic by watching programme flow with the built in simulator, or in real time with in-circuit emulation or in-circuit debugger.
- Make timing measurable with the simulator or emulator.
- View variables in Watch Windows
- Programme firmware into device programme.



A printed circuit board (pcb) controls and regulates electrical flow through a circuit. "It serves as a conductor helping currents get where they need to go and do what they need to do." [7]. It consists of several components which are placed in particular positions.

Integrated circuits are miniature pcb's within a microchip.

4

consists of a thin piece of quartz crystal. When a voltage is applied, the crystal changes shape. This is known as a piezo-electric effect. The electrical charge produces a mechanical effect by changing the shape of the crystal and vice versa. A quartz oscillator has been chosen as they have a very high frequency stability". That is "ability to provide a constant frequency output" [7]. One capacitor is placed at each end to ground the quartz oscillator.

Capacitors are cylinder shaped and store electrical energy. They allow a small amount of charge through to maintain a steady flow. [8].

"The seven segment display SSD requires seven outputs from the microcontroller. Active high operation can be provided by a common cathode LED display and the displayed decimal point can be used as the power indicator." [2 p211].

The audio transducer requires one output. A piezo buzzer has sufficient bandwidth and output power and its power consumption is low. [2 p211].

3. Count Errors

When introduced to the count project there were a number of deliberate errors (see appendix A) which needed to be corrected. These errors were:

- Line 30, 31, 33 and 36 – symbol not previously defined (status)
- Line 57 – symbol not previously defined (get count)
- Line 74 – argument out of range. Least significant bit
- Line 75 symbol not previously defined (mcont)
- Line 89 found label after column 1 (en)
- Line 89 illegal opcode (d)
- Line 98 expected end

For lines 30, 31, 33 and 36 status in the memory map has an upper case 'S' whereas the status in the initialisation has a lower case 's'. These have to be the same otherwise the program will not recognise the instruction so to fix this error the upper case 'S' was changed to a lower case 's'.

For line 57 'count' was spelt differently to 'count' on line 41 which was spelt 'countt'. To fix this error, 'countt' was changed to 'count', as this is the correct spelling.

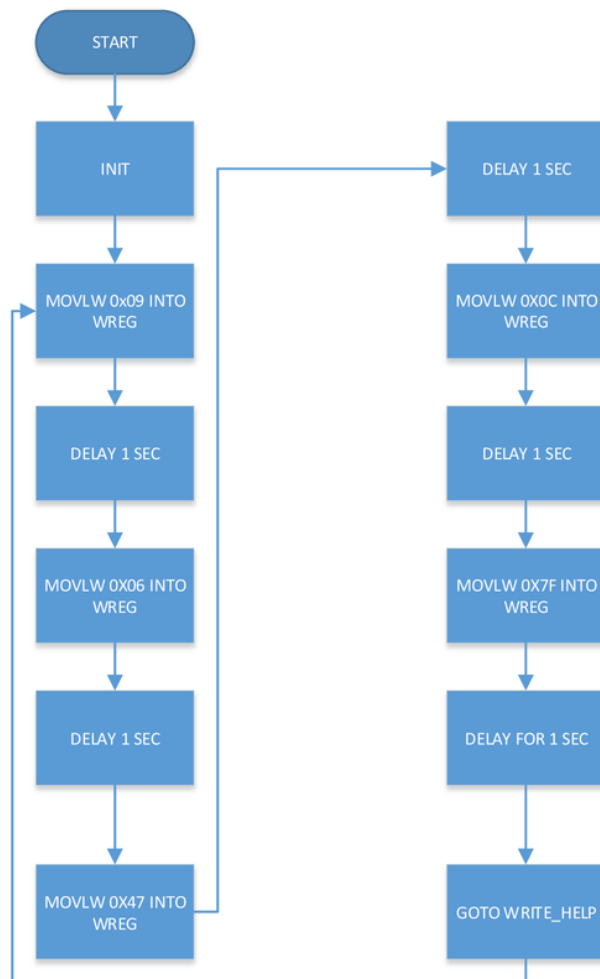
For line 74 the value was 'OXfff' which was incorrect. It should have been 'OXff', so to fix this error 'OXfff' was changed to 'OXff'.

For line 75 was another spelling error occurred as 'mcont' should be spelt 'mcount'. To fix this error, 'mcont' was changed to 'mcount'.

Lastly, to correct the error in lines 89 and 90 the space was deleted between the 'n' and 'd' on line 89 to make the word read 'end'.

These errors were fixed and the program build was successful, (see Appendix B).

4. Count Help



The flowchart shows a program for a microchip to show the word 'help', (see Appendix C) on the assignment board. This is done by programming a microchip PICSTART PLUS programmer and inserting the microchip into the assignment board.

Before the program is made the values for each letter of the word 'help' and the blank letter must be figured out. This is done using the binary table and the seven segments of the display. The values for each letter are:-

- H – 0x09
- E – 0x06
- L – 0x47
- P – 0x0c
- ' ' - 0x7f

As it can be seen in the flowchart the first instruction is the initialisation. This is where the direction of the pins are decided and whether port B is going to be used for inputs or outputs.

The next instruction to be executed is to load the LED status for the letter 'H' (0x09) into the w register also known as the working register, which is where main operations are used.

The next instruction to be executed is to delay the program for 1 second which is done by calling a pause for 200ms five times.

The next instruction to be executed is load the LED status for the letter 'E' (0x06) into the w register and again, delay for 1 second.

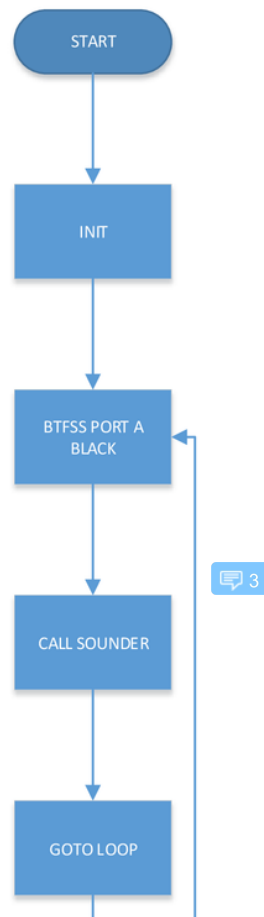
The next instruction is to load the LED status with the letter 'L' (0x47) into the w register and again, delay for 1 second.

The next instruction to be executed is to load the LED status for the letter 'P' (0x0c) into the w register and again, delay for 1 second.

The next instruction to be executed is to load the LED status with the blank letter (0x7f) into the w register, this is to create a space between the last letter and first letter. Again, delay for 1 second.

The last instruction is to go back to 'write_help' which is the label at the start of executions for the word 'help'. This program will then keep repeating until switched off.

5. Count Sound



This flowchart shows the program for a PIC to make sound at 440Hz when the black button is pressed (See Appendix D). Just like the program for 'help' this is done by programming a microchip though PICSTART PLUS and inserting the chip into the assignment board.

The first instruction is initialisation which is where all the variables are set up and the direction of the pins and the inputs and outputs are decided.

The next instruction to be executed is to press the black button. This is port A because port A are all inputs and 'btfss' means bit test file skip on set, so if the black button is pressed then move on to the next instruction but if it's not pressed then jump over the next instruction.

The next instruction is 'call sounder' which makes the sound if the black button is pressed.

And the last instruction is to go to 'loop' which returns to the label at the start of the executions.

References

- [1] J. Young article title: MPLAB v8.92 handout booklet supplied in module Micro's Peripherals and Interface ACDA4004, 2016.
- [2] M. Bates, PIC Microcontrollers An introduction to Microelectronics, 3rd Ed., Oxford, Newnes, 2011.
- [3] A.Tomar, article title: Microchip: MPLAB Integrated Development Environment (IDE) Overview, found online at: www.element14.com/community/docs/DOC-39318/1/microchip-mplab-integrated-development-environment-ide-overview, last updated 25th June 2014,. Accessed on 4th Jan 2019.
- [4] Anon Author, employee of Informer Technologies Inc., article title: MPLAB IDE v8.92 Free, found online at: <https://mplab-ide.software.informer.com/> review of MPLAB updated 21st Dec 2018 Accessed on 8th Jan 2019
- [5] Anon Author, employee of Microchip Technology Inc., article title: MPLAB IDE Quick Start Guide, found online at: http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_Getting_Started_51281g.pdf, 2007, Accessed on 27th January 2019.
- [6] Anon Author, employee of Microchip Technology inc., article title: MPLAB IDE Simulator, Editor User's Guide, 2002, accessed on 28th January 2019
- [7] J. Uhl, article title: What is a Printed Circuit Board?, found online at: <https://www.wisegeek.com/what-is-a-printed-circuit-board.htm>, last updated 25th Dec 2018, accessed on 30th Jan 2019.
- [8] Anon Author, employee of Aspecore, article title: Quartz Crystal Oscillators, found online at: <https://www.electronics-tutorials.ws/oscillator/crystal.html>, updated 2018, accessed on 30th Jan 2019.

Figure List

Figure 1: Diagram of MPLAB IDE taken from A. Tomar, article title: Microchip: MPLAB Integrated Development Environment (IDE) Overview, found online at www.element14/community/docs/DOC-39318//microchip-mplab-integrated-development-environment-ide-overview, last updated 25th June 2014,. Accessed on 4th Jan 2019.

Figure 2: Circuit Diagram taken from Micro's Peripherals and Interfaces Module Tutorial Handbook written by J.Young 2016.

Further Reading

S.R. Rizvi, Microcontroller Programming, An introduction, London, CRC Press, 2012.

J. Stokes, Inside the Machine, San Francisco (USA), No Starch Press Inc. 2007.

N. Senthil Kumar, et al, Microprocessors and Interfacing, New Delhi (India), Oxford University Press, 2012.

R.S. Kaler, Microprocessors An introduction, Second Ed. New Delhi (India), L.K. International Publishing House Pvt Ltd. 2019

List of Appendices

	Page No.
Appendix A	PIC Programme Containing Deliberate Errors, 14
Appendix B	PIC Programme Showing Corrected Errors, 18
Appendix C	PIC Programme for 'HELP' 22
Appendix D	PIC Programme for 'Sound' 27

Appendix A

```
; FILE:          count.asm

; AUTHOR:        JMY

; DATE:         24th March 2005

; DESCRIPTION:    Count to 255 at 1 second intervals and display on a seven
;                  segment display in binary.
;
;                  The pause subroutine takes apx. 0.2 sec.
;
;                  Pause is called 5 times


list    p=16F84          ; select processor type
radix   hex              ; define radix as hexadecimal
;    #include "p16f84.inc"
;-----

;    cpu equates (memory map)

portB   equ    0x06      ;port B address
trisb   equ 0x06          ;trisB address in register bank 1


Status  equ 0x03          ;status register
RP0     equ 5             ;bit 5 of status register
RP1     equ 6             ;bit 6 of status register


count   equ    0x0c      ; counter 0 to 255
nCount  equ    0x0d      ; pause routine inner loop counter
mCount  equ    0x0e      ; pause routine outer loop counter


;-----
```

```

    org    0x000

start

    bcf     status,RP0    ; clear bank selection bits in the status reg
    bcf     status,RP1

    bsf     status,RP0    ; select bank 1
    movlw  0x00           ; load w with 0x00
    movwf  trisb          ; load it into trisb to make portb all outputs
    bcf     status,RP0    ; reselect bank 0

    clrf   portB          ; clear all lines low
    clrf   count          ; initialize counter to 0

get_counnt

    movf   count, w       ; move count to W

    movwf  portB          ; move W to port B to update the display

    addlw  1               ; add +1 to W

    movwf  count          ; save W in count

    call   pause          ; delay 200ms by subroutine
    call   pause
    call   pause

```

```

    call    pause

    call    pause          ; five pause executions equals ~ 1 second

    goto    get_count      ; repeat forever

;-----
;subroutine pause: lets waste some time....
;
;inputs: nothing
;outputs: nothing
;
;does: nothing but waste some time
;corrupts: register W
;
;description: two loops using "mCount" and "nCount" register files
;
;              "mCount" is the outer loop counter (255 down to zero)
;
;              "nCount" is the inner loop counter (255 down to zero)
;
;-----
pause
    movlw  0xffff          ; set w = 255 decimal
    movwf  mCont           ; mCount = w: outer loop counter

loadN
    movlw  0xff            ; set w = 255 decimal
    movwf  nCount          ; nCount = w: inner loop counter

decN
    decfsz nCount, f       ; nCount--

```

```

goto    decN          ; if nCount != 0 then repeat nCount--

decfsz  mCount, f      ; else decrement Count

goto    loadN          ; if mCount != 0 then

                        ; reload nCount to 255 and decrement

return                  ; else exit subroutine

en d

;-----
; Configuration! Important!
;
; at blast time, select:
;     memory unprotected
;     watchdog timer disabled
;     standard crystal (4 MHz)
;     power-up timer on

```

Appendix B

```
; FILE:          count.asm
; AUTHOR:        JMY
; DATE:          24th March 2005
; DESCRIPTION:    Count to 255 at 1 second intervals and display on a seven
;
;                  segment display in binary.
;
;                  The pause subroutine takes apx. 0.2 sec.
;
;                  Pause is called 5 times
```

```
list    p=16F84      ; select processor type
radix    hex          ; define radix as hexadecimal
;    #include "p16f84.inc"
;-----
;    cpu equates (memory map)
portB    equ    0x06      ;port B address
trisb    equ 0x06      ;trisB address in register bank 1

status    equ 0x03      ;status register
RP0      equ 5          ;bit 5 of status register
RP1      equ 6          ;bit 6 of status register

count    equ    0x0c      ; counter 0 to 255
nCount    equ    0x0d      ; pause routine inner loop counter
mCount    equ    0x0e      ; pause routine outer loop counter
```

```
;-----
```

```

org    0x000

start

    bcf      status,RP0    ; clear bank selection bits in the status reg
    bcf      status,RP1

    bsf      status,RP0    ; select bank 1
    movlw 0x00             ; load w with 0x00
    movwf trisb            ; load it into trisb to make portb all outputs
    bsf      status,RP0    ; reselect bank 0

    clrf     portB         ; clear all lines low
    clrf     count        ; initialize counter to 0

get_count
    movf     count, w      ; move count to W

    movwf portB            ; move W to port B to update the display

    addlw 1                ; add +1 to W

    movwf count            ; save W in count

    call     pause         ; delay 200ms by subroutine
    call     pause

```

```

call    pause

call    pause

call    pause        ; five pause executions equals ~ 1 second


goto    get_count        ; repeat forever


;-----
;subroutine pause: lets waste some time....
;
;inputs: nothing
;outputs: nothing
;
;does: nothing but waste some time
;corrupts: register W
;
;description: two loops using "mCount" and "nCount" register files
;
;                "mCount" is the outer loop counter (255 down to zero)
;
;                "nCount" is the inner loop counter (255 down to zero)
;
;-----
pause

    movlw 0xff        ; set w = 255 decimal

    movwf mCount        ; mCount = w: outer loop counter

loadN

    movlw 0xff        ; set w = 255 decimal

    movwf nCount        ; nCount = w: inner loop counter

decN

```

```

    decfsz  nCount, f      ; nCount--
    goto    decN           ; if nCount != 0 then repeat nCount--

    decfsz  mCount, f      ; else decrement Count
    goto    loadN          ; if mCount != 0 then
                                ; reload nCount to 255 and decrement

    return                ; else exit subroutine

end

;-----
; Configuration! Important!
;
; at blast time, select:
;     memory unprotected
;     watchdog timer disabled
;     standard crystal (4 MHz)
;     power-up timer on

```

Appendix C

```
; FILE:          count.asm

; AUTHOR:        JMY

; DATE:          24th March 2005

; DESCRIPTION:    Count to 255 at 1 second intervals and display on a seven
;                  segment display in binary.
;
;                  The pause subroutine takes apx. 0.2 sec.
;
;                  Pause is called 5 times


list    p=16F84          ; select processor type
radix   hex              ; define radix as hexadecimal
;    #include "p16f84.inc"
;-----

;    cpu equates (memory map)

portB   equ    0x06      ;port B address
trisb   equ 0x06          ;trisB address in register bank 1


Status  equ 0x03          ;status register
RP0     equ 5             ;bit 5 of status register
RP1     equ 6             ;bit 6 of status register


count   equ    0x0c      ; counter 0 to 255
nCount  equ    0x0d      ; pause routine inner loop counter
mCount  equ    0x0e      ; pause routine outer loop counter


;-----
```

```

org    0x000

start

    bcf      Status,RP0    ; clear bank selection bits in the status reg
    bcf      Status,RP1

    bsf      Status,RP0    ; select bank 1
    movlw   0x00            ; load w with 0x00
    movwf   trisb          ; load it into trisb to make portb all outputs
    bcf      Status,RP0    ; reselect bank 0

    clrf    portB          ; clear all lines low


write_help

    movlw   0x09            ;load LED status for letter 'h' into w reg
    movwf   portB          ;put W reg onto Port B pin
    call    delay1s

    movlw   0x06            ;load LED status for letter 'e' into w reg
    movwf   portB          ;put W reg onto Port B pin
    call    delay1s

    movlw   0x47            ;load led states for letter 'l' into w reg
    movwf   portB
    call    delay1s

```

```

    movlw 0x0c    ;load led states for letter 'p' into w reg
    movwf portB
    call  delay1s

    movlw 0x7f    ;load led states for letter ' ' into w reg
    movwf portB
    call  delay1s

    goto  write_help    ; repeat forever

;-----
;subroutine pause: lets waste some time....
;
;inputs: nothing
;outputs: nothing
;
;does: nothing but waste some time
;corrupts: register W
;
;description: two loops using "mCount" and "nCount" register files
;              "mCount" is the outer loop counter (255 down to zero)
;              "nCount" is the inner loop counter (255 down to zero)
;
;-----
pause

```

```

        movlw 0xff          ; set w = 255 decimal
        movwf mCount        ; mCount = w: outer loop counter
loadN
        movlw 0xff          ; set w = 255 decimal
        movwf nCount        ; nCount = w: inner loop counter
decN
        decfsz nCount, f     ; nCount--
        goto   decN          ; if nCount != 0 then repeat nCount--

        decfsz mCount, f     ; else decrement Count
        goto   loadN         ; if mCount != 0 then
                                ; reload nCount to 255 and decrement

        return              ; else exit subroutine

```

delay1s

```

        call   pause
        call   pause
        call   pause
        call   pause
        call   pause

```

return

end

;------

; Configuration! Important!

;

; at blast time, select:

; memory unprotected

; watchdog timer disabled

; standard crystal (4 MHz)

; power-up timer on

Appendix D

```
; FILE:          count.as
; AUTHOR:        JMY
; DATE:          24th March 2005
; DESCRIPTION:    Count to 255 at 1 second intervals and display on a seven
;                segment display in binary.
;                The pause subroutine takes apx. 0.2 sec.
;                Pause is called 5 times

list    p=16F84          ; select processor type
radix    hex              ; define radix as hexadecimal
;    #include "p16f84.inc"
;-----
;    cpu equates (memory map)

portB    equ    0x06      ;port B address
trisb    equ 0x06        ;trisB address in register bank 1

portA    equ 0x05
trisa    equ 0x05

Status    equ 0x03        ;status register
RP0        equ 5          ;bit 5 of status register
RP1    equ    6          ;bit 6 of status register
```

```

black equ 0           ;black button connects to bit 0 port A (count UP)
green equ 1           ;green button connects to bit 1 port A (count DO)

nCount equ 0x0d       ; pause routine inner loop counter
mCount equ 0x0e       ; pause routine outer loop counter

```

```

;-----

```

```

org 0x000

start
    bcf      Status,RP0    ; clear bank selection bits in the status reg
    bcf      Status,RP1

    bsf      Status,RP0    ; select bank 1
    movlw 0x00             ; load w with 0x00
    movwf trisb            ; load it into trisb to make portb all outputs

    movlw 0x03
    movwf trisa            ;setting RA0 and RA1 as inputs

    bcf      Status,RP0    ; reselect bank 0

```

```
    clrf    portB    ; clear all lines low
```

Loop

```
    btfss  portA,black    ;is black button pressed
```

```
    call   sounder
```

```
    goto   Loop          ; repeat forever
```

sounder

```
    bsf    portB,7 ; turn sounder on
```

```
    call   pause
```

```
    bcf    portB,7 ; turn sounder off
```

```
    call   pause
```

```
    return
```

```
;------
```

```
;subroutine pause: lets waste some time....
```

```
;
```

```
;inputs: nothing
```

```
;outputs: nothing
```

```
;
```

```
;does: nothing but waste some time
```

```
;corrupts: register W
```

```

;
;description: two loops using "mCount" and "nCount" register files
;
;          "mCount" is the outer loop counter (255 down to zero)
;
;          "nCount" is the inner loop counter (255 down to zero)
;
;-----
pause
    movlw 0x13          ; set w = 255 decimal
    movwf mCount        ; mCount = w: outer loop counter
loadN
    movlw 0x13          ; set w = 255 decimal
    movwf nCount        ; nCount = w: inner loop counter
decN
    decfsz nCount, f     ; nCount--
    goto   decN          ; if nCount != 0 then repeat nCount--

    decfsz mCount, f     ; else decrement Count
    goto   loadN         ; if mCount != 0 then

                        ; reload nCount to 255 and decrement
    return              ; else exit subroutine

delay1s

    call   pause

```

return

end

;-----

; Configuration! Important!

;

; at blast time, select:

; memory unprotected

; watchdog timer disabled

; standard crystal (4 MHz)

; power-up timer on

PIC Programming

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

65/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7



Comment 1

These comments are a little generic. I was looking for the specific function each component has in the circuit.

PAGE 8

PAGE 9

PAGE 10

PAGE 11



Comment 2

misplaced "r"

PAGE 12



Comment 3

You should include captions on your images so you can refer to them in your text more easily.

e.g. This can be seen in figure 3 where....

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

NICHOLAS REGAN

64.60

ENGAGEMENT WITH LITERATURE

including reading, referencing, academic conventions and academic honesty



A comprehensive range of research informed literature embedded in the work. Excellent selection of relevant and credible sources. High-level referencing skills, consistently applied.

14

KNOWLEDGE & UNDERSTANDING

Knowledge of the basic underlying concepts and principles of a subject.



Knowledge is extensive. Exhibits understanding of the breadth and depth of established views.

13

COGNITIVE & INTELLECTUAL

Evaluate underlying concepts and principles of a subject and interpret qualitative and quantitative data in order to develop lines of argument and make sound judgements.



Not in mark scheme

PRACTICAL APPLICATION SKILLS



A very good application of a range of methods, materials, tools and techniques. Evidence of some creativity.

26

TRANSFERABLE SKILLS

Communicate the results of their study/work accurately and reliably, and with structured and coherent arguments; the qualities needed for employment requiring the exercise of some personal responsibility within a structured and managed environment.



Can communicate effectively in a suitable format, but may have minor errors. Mostly coherent work and is in a suitable structure. You should include captions on your figures so you can refer to them more easily in your text.

11.6