

MICROCONTROLLER WORKBOOK SERIES

DIOS LED WORKBOOK



Compatible with
Dios WorkBoard Deluxe
(Not Included)

LEVEL 1 WORKBOOK

Updates

Software and updated manuals can be obtained from the Kronos Robotics web site located at www.kronosrobotics.com.

Forums

A web-based discussions board is located at one of the Kronos Robotics sister sites. These forums cover everything from the Athena product line to motor controller basics and robotics. They are located at: www.kronosrobotics/forums.

Warranty

Kronos Robotics warrants its products against defects in material and workmanship for a period of 90 days. If you discover a defect, Kronos Robotics will, at its option, repair, replace, or refund the item's purchase price. Simply contact Kronos Robotics for an RMA. The customer is responsible for all return shipping expenses.

Copy Policy

Kronos Robotics workbooks are copyrighted material and may not be copied or duplicated in any form.

Disclaimer of Liability

Kronos Robotics cannot be held responsible for any incidental, or consequential damages resulting from the use of any Kronos Robotics products.

15-Day Money-Back Guarantee

It is very important to us here at Kronos Robotics that our customers are completely satisfied. If you are not happy with any product you purchase from Kronos Robotics it may be returned for a full refund or exchange within 15 days of the invoice date.

The returned items must be in unused condition and have original packaging. There will be a restocking fee of 30% only on items that do not meet this condition. Also note that this return policy does not apply to items that you have destroyed or damaged. For example if you hook up a microcontroller backwards you may not return it.

Shipping and Handling fees are non-refundable. The customer is responsible for all return shipping expenses.

Shipping Responsibility

Kronos Robotics ships all packages Priority Air Mail via the United States Postal Service with delivery confirmation. We have found this combination gives the fastest and most reliable delivery at a very reasonable cost.

Once a package has been delivered we are no longer responsible for the package. More specifically, if some one steals the package from your doorstep we will not be held responsible and no refund will be provided.

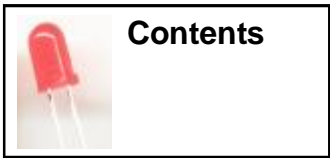
If your package has not been delivered and sufficient time has passed please email us and we will verify delivery. If delivery has been made you will be given the confirmation number and delivery details. Thereafter you must contact your local Post Office for further information.

TradeMarks

PIC is a registered trademark of Microchip Technology, Inc. Windows is a trademark of Microsoft Corporation. 1-wire is a trademark of Dallas Semiconductor.

Contacts

email: sales@kronosrobotics.com
phone: 703 779-9752
fax: 703 779-9753
web: www.kronosrobotics.com



Intro

Intro4

Chapter 1: LED Basics

LED Colors5
 LED Shapes and Sizes5
 Chapter 1 Summery6
 LED Hookup6
 T1 3/4 LED6
 Chapter 1 Questions7

Chapter 2: Single LED Control

Project 1 Connect a Single LED9
 Required Componets for Chapter 29
 Project 2 Control an LED with a Button11
 Project 3: Drive a high Power LED15
 Chapter 2 Summery16
 Chapter 2 Questions17

Chapter 3: Multiple LEDs

Required Componets for Chapter 319
 Project 1 Control 8 LEDs19
 Project 2: Electronic Die21
 Chapter 3 Summery23
 Chapter 3 Questions24


Answers

Answers to Questions26



The Dios LED Workbook is a Level 1 workbook. Kronos Robotics workbooks are graded on a scale of 1-10. Level 1 is the easiest and considered entry level. Level 10 is extremely difficult. Most of our workbooks will fall within the 1 to 5 scale. Workbooks above Level 5 may be considered experimental or highly specialized.

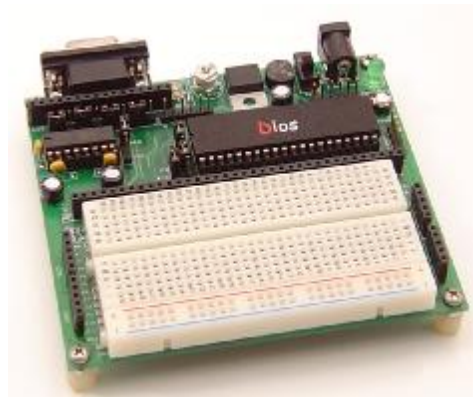
Extra Challenges

We have placed extra challenges in the workbook at various points. A challenge is marked with the  symbol. While the chapter questions at the end of each chapter are used to test and reinforce key points, the challenges are one step beyond and should be considered extra credit. A challenge may exceed the workbook level so if you can't complete a particular challenge don't worry. You can always come back at a later date and try the challenge.

Dios WorkBoard

This workbook is centered around the Dios WorkBoard Deluxe. This board houses a Dios 40 or a DiosPro 40 chip. It has a built-in program port as well as a 1.5 Amp regulator. The board can accept AC at 7 to 14V or DC in any polarity at 7 to 16V.

The breadboard is indexed and will be used to guide the user with exact hookup tables. The user may use the hookup tables, hookup diagram, or schematic diagram. All are included in the workbook.



Dios WorkBoard Deluxe

Workbook Prerequisite

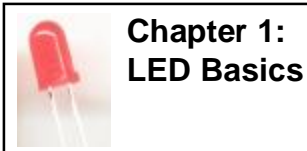
A basic understanding of electronics is required. It is not necessary that you understand electron theory, but you should have a good understanding of what may happen when you reverse the polarity of your power source when connected to a chip or circuit.

You should have assembled and tested your Dios WorkBoard and completed the Dios Tutorial, and be familiar with programming the Dios.

Components Required

You will need the following components to complete this workbook.

- Dios WorkBoard DeluxeKronos Robotics #16452
- Dios 40 or DiosPro 40 MicrocontrollerKronos Robotics #16168 or #16428
- 8, LEDsKronos Robotics #16237
- 8, 390 Ohm ResistorsKronos Robotics #16190
- 1, 10K ResistorKronos Robotics #16193
- 1, ButtonKronos Robotics #16244
- 1, Transistor (2N2222)Kronos Robotics #16143
- High Power LED (**Optional**)RadioShack 276-086
- Hookup WireAny 22 to 26 guage solid wire will do
- Dios Compiler (Free Download)Kronos Robotics Web Site
- Program Code ExamplesIncluded With Compiler
- This WorkbookIncluded With Compiler



Chapter 1: LED Basics

LED : Light Emitting Diode.

An LED is a small semiconductor diode that emits light when charged. This light in most cases is monochromatic, therefore specific colors may be produced. The colors may range from red to violet in the visible range. There are also LED's that can emit higher wavelengths of light in the non-visible infrared range.

LED Shapes and Sizes

LED's come in many sizes, shapes and configurations as shown in Figure 1.1.

Some LED configurations include LED arrays and 7 segment displays.

The most common and least expensive LED is the bullet shaped LED. These LED's are the T1 or the T1 3/4.

The "T" represents the bullet shape. The 1 3/4 is the size and is measured in 1/8" units. So T1 3/4" means bullet shape and a diameter of almost 1/4". (It's actually 5mm)

The T1 is about 1/8" in diameter or 3mm. These are the most popular sizes and can be purchased for less than 10 cents each.

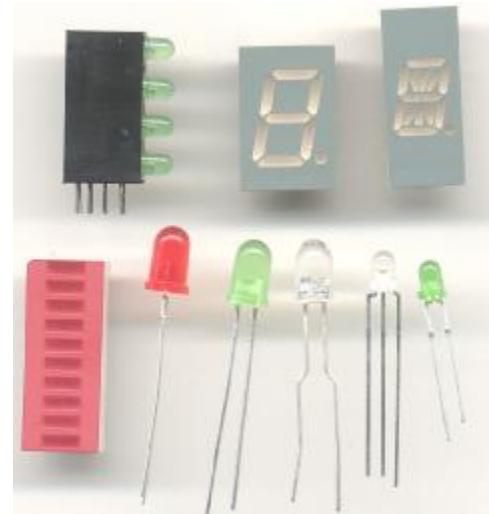


Figure 1.1

LED Colors

Because LED's output a very tight wavelength they can produce specific colors. Figure 1.2 shows a wavelength chart of the visible spectrum of LED's.

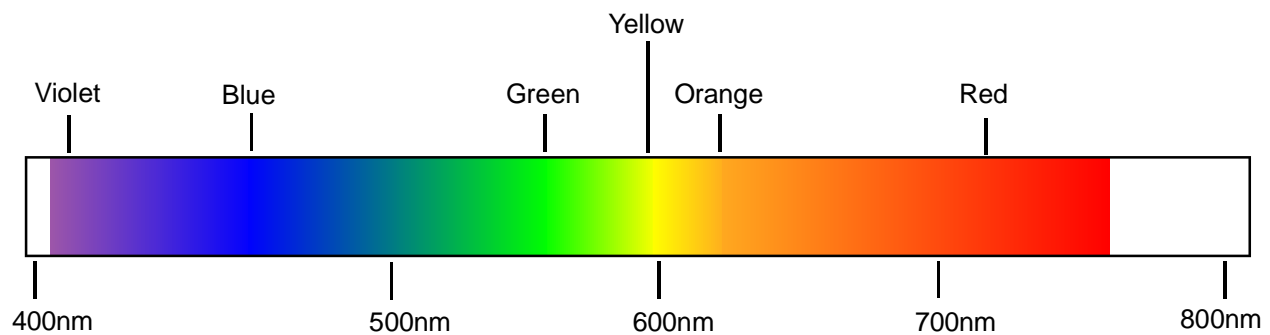


Figure 1.2

Violet LED's are on one end of the spectrum and red LED's are on the other. Beyond the red is the IR wave length and beyond the violet is the UV wave length. White LED's are actually blue LED's covered with some sort of phosphor that causes wavelengths of red through green to be emitted. This wide spectrum creates white light.

How LED's emit the color is determined by the enclosure. First off, they can be clear with the top of the LED acting as a lens. The makeup of this lens can determine the actual viewing angle. These types of LED's are meant to be viewed head-on. They also make nice illuminators because the bulk of the light is sent forward. Even though the LED case is clear, the actual color the LED emits will be determined by its wavelength.

The LED case can also be semi-transparent so that the light is diffused. In this case the whole LED lights up. This type of LED can be viewed from just about any angle. This type of LED makes a better indicator.

T1 3/4 LED

Let's take a closer look at a standard T1 3/4 LED. This type of LED (and many others) have two leads as shown in Figure 1.3. In most cases, one lead will be shorter than the other. The shorter lead is called the Cathode and the longer the Anode. The Anode side of the case will also have a small flat depression on the lower rim of the LED.

LED Hookup

The schematic symbol for the LED is the same as the Diode but with one or more small arrows as shown in Figure 1.4.

To cause an LED to emit light you apply a negative voltage to the Cathode and a positive voltage to the Anode.

LED's have a voltage rating. Most generic LED's fall in the 2-3v range. This means you cannot connect them directly to a 5 volt source. If you connect a LED directly to 5 volts it will glow brightly then die a fast death. Use a resistor in series with the LED to drop the voltage as shown in Figure 1.5.

You can use anywhere from a 100 ohm to 1K resistor to drop the voltage down to acceptable levels. This will give you decent range to adjust the brightness of the LED.

The higher the value of the resistor, the dimmer the LED and the less power the LED will consume.

There are LED's with current limiting resistors built in. They are generally designed for 5v hookup. These LED's make prototyping on breadboards very nice because you can quickly connect one or more LED's to your circuit.

If you connect an LED up backwards it won't light up. Don't worry it won't blow up, it just won't light.

Chapter 1 Summary

That concludes LED basics. We have provided you with a set of questions that target the key points in this chapter.

The answers are on the last page of the workbook. If you answer any of the questions incorrectly simply go back and review this chapter.

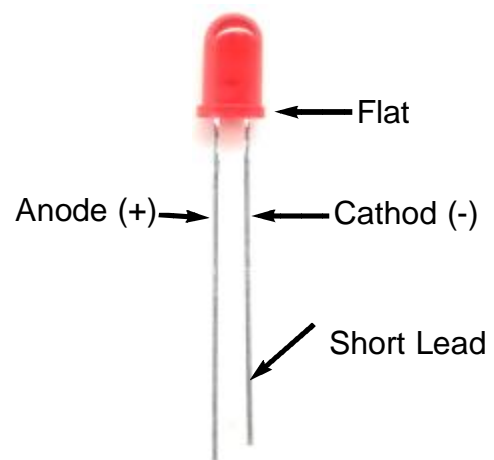


Figure 1.3

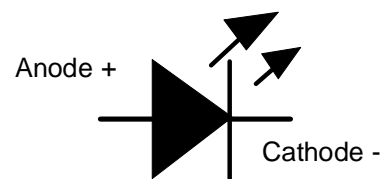


Figure 1.4

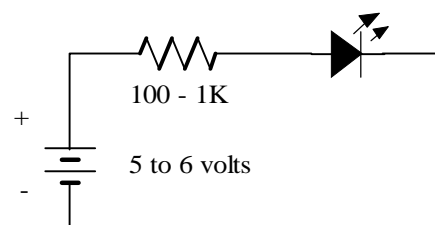


Figure 1.5

Chapter 1 Questions

1. What is the approximate diameter of a T1 3/4 LED?

2. What is the short lead on the LED called?

3. What is the long lead on the LED called?

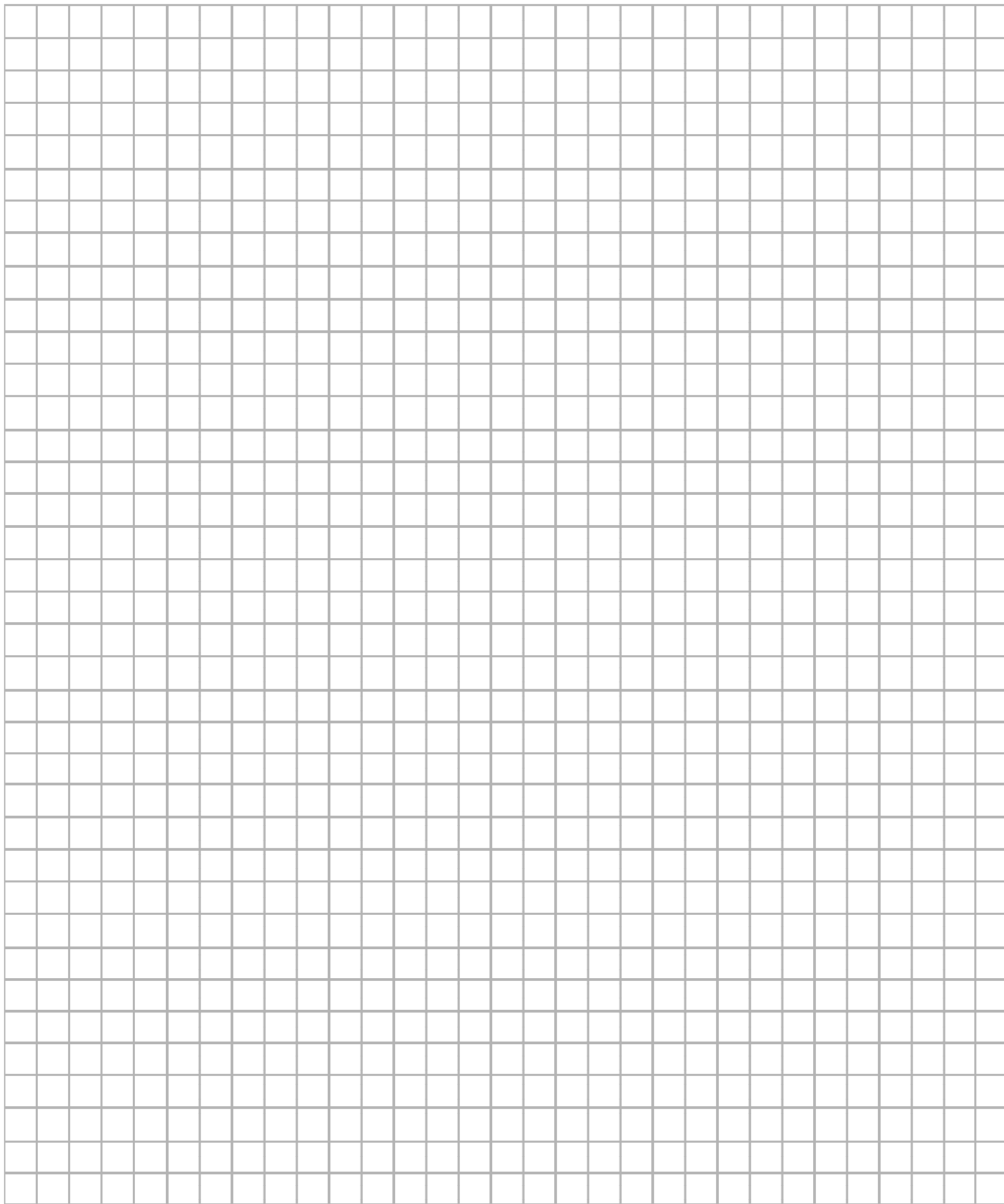
4. Which lead do you connect to the positive side of your power source?

5. What is the difference between a clear and diffused LED?

6. Why do you need to connect a resistor in series with the LED when working with 5 volts?

7. What happens when you connect an LED up backwards?

8. Draw the schematic diagram for an LED.





Chapter 2: Single LED Control

It's time to get our hands dirty. In this chapter we will show you how to connect a single LED to the Dios microcontroller. We will show you how to dim the LED and how to control a larger LED. We will also show you how to use a single LED as an indicator in a single-button menu system.

Required Componets for Chapter 2

- Dios Workboard
- Dios 40 or DiosPro Microcontroller
- Standard LED
- 390 Ohm Resistor
- 10K Resistor
- Button
- Transistor (2N2222)
- High Power LED (**Optional**)

Project 1: Connect a Single LED

Let's Connect an LED to the Dios. These examples will work with both the Dios and DiosPro 40-Pin microcontrollers. Just substitute the appropriate directive to match your microcontroller.

Using Figures 2.1 and 2.2, make the following connections:

- Place a 390 Ohm resistor on the breadboard at positions e30 and f30.
- Place an LED on the breadboard. Anode = i30 Cathode = i29.
- Connect Port 13 to the resistor at a30.
- Connect the Cathode of the LED (f29) to VSS.
- Load your microcontroller with Program 2.1.

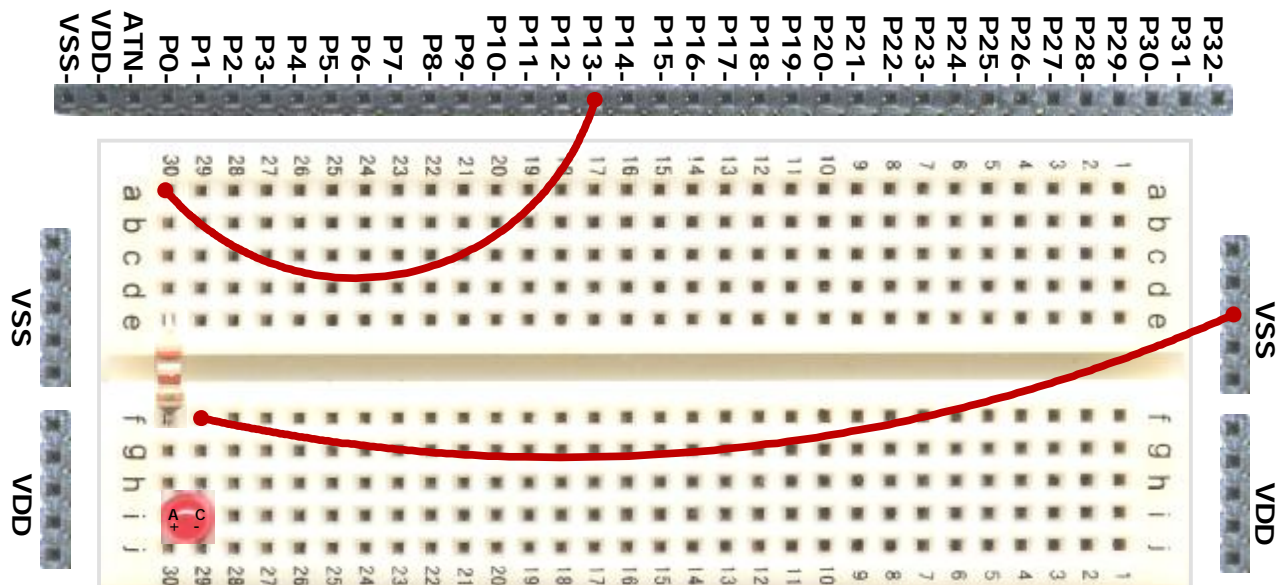


Figure 2.1

The high command will supply 5v to IO port 13 and light the LED. The 390 Ohm resistor drops the voltage to an acceptable level so that the LED is not damaged. Also, without the resistor the LED would most likely pull too much current and could damage the microcontroller.

We use the pause command to slow things down so we can see the changes in the LED state. The pause values are set to 200 which is a delay of 200 milliseconds. This is plenty of time for your eye to register the state changes of the LED. Feel free to experiment with these values to see the different affects. If you change the values below 15 or so you probably won't be able to see the LED turn on and off.

- Using Program 2.1 change the first pause command to **pause 1** and the second pause command to **pause 15**.

Notice that the LED lights but it isn't very bright. This is because the LED is only on for 1ms and off for 15ms. This type of control is called **PWM** or **Pulse Width Modulation**.

Doing PWM this way is easy when you are using simple loops. When your program gets more complicated it can be almost impossible to create a consistent PWM signal using the high and low commands. We need a way to create a PWM signal without affecting the program we are running.

The Dios has a hardware based built-in PWM signal generator. The PWM signal generator is hard wired to Port 13. To activate the PWM generator we use the **PWMinit**, **PWMperiod** and **PWM1duty** functions.

- Load your microcontroller with Program 2.2.

In Program 2.2 you will notice the **include** command was used to load a library. In this case it was the DiosHWPWM library. If you forget to include a library, the Dios software will add it to the end of your program for you.

What is a library?

The Dios compiler has two types of routines. These are called commands and functions. Functions are placed in libraries while commands are more integrated with the language. Libraries are generally used to interface to a chip or particular piece of Dios hardware.

Let's take a look at the PWM functions in detail.

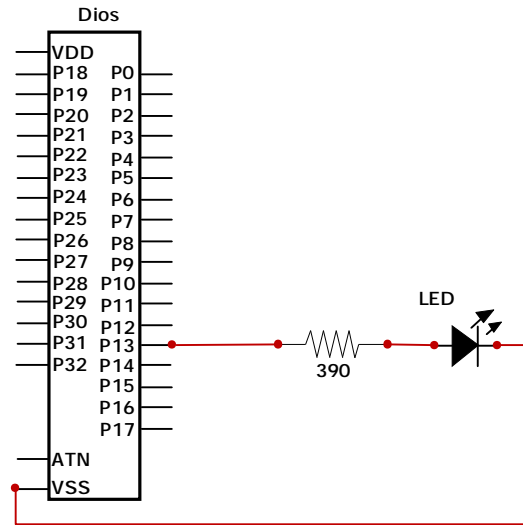


Figure 2.2

```
'Single LED
func main()

    output 13

loop:

    high 13 'Turn on LED
    pause 200 'First Pause
    low 13 'Turn off LED
    pause 200 'Second Pause

    goto loop

endfunc
```

Program 2.1

```
'Dim an LED
func main()

    PWMinit 1
    PWMperiod 100
    PWM1duty 10

loop:

    'Doing Nothing
    goto loop

endfunc
include \lib\DiosHWPWM.lib
```

Program 2.2

PWMinit

This function is used to configure the PWM signal generator.

PWMperiod

This function sets the period of the signal generator. You supply a value between 2 and 255. Note that the period is what determines the frequency of the signal.

PWM1duty

This function sets the duty cycle of the signal. This sets the amount of on time (high) for the period. If you set the period to 100 then you can set the percentage directly. With a period of 100 and a duty of 75 you have a 75% duty cycle; i.e., the signal is high 75% of the time and low 25%.

You can change the duty cycle any time you wish. Think of it as fire and forget. The actual signal for a 50% and 10% duty cycle looks like those shown in Figure 2.3.

- ☞ Here is a challenge for you. Connect a 2nd LED up to port 0. Modify Program 2.2 to make it blink on and off while the LED on port 6 is turned on but dim. Remember you will need to add an output command for port 0 at the beginning of your program.

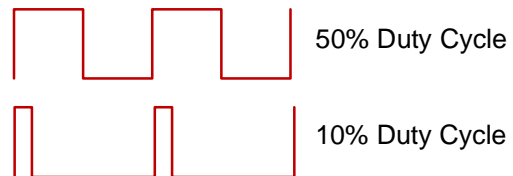


Figure 2.3

Project 2: Control an LED with a Button

An LED can be used with a button to provide feedback. Using Figures 2.4 and 2.5A make the following connections:

- Place a 390 Ohm resistor on the breadboard at positions e30 and f30.
- Place an LED on the breadboard. Anode = i30 Cathode = i29.
- Connect Port 13 to the resistor at a30.
- Connect the Cathode of the LED (f29) to VSS.
- Place a button on the breadboard at positions c16 and c18.
- Place a 10K resistor on the breadboard at positions e13 and e16.
- Connect Port 0 to the button at a16.
- Connect the 10K resistor (e13) to VSS.
- Connect the button (b18) to VDD.
- Load your microcontroller with Program 2.3.

Port 0 is held low with the 10K resistor. This is very important. If the port is left floating it will swing high and low randomly. For this reason we tie it to VSS with a 10K resistor. The button is used to short the port directly to VDD. This overrides the 10K resistor, forcing port 0 high.

The Dinos has 8 built-in resistors which are tied to IO ports 0-7. When activated, these resistors pull the IO ports high. This creates a situation where a button would be used to pull them low. The command to activate the resistors is **pullupon**. The command activates all 8 resistors at once. To turn off the pullup resistors, use the **pullupoff** command.

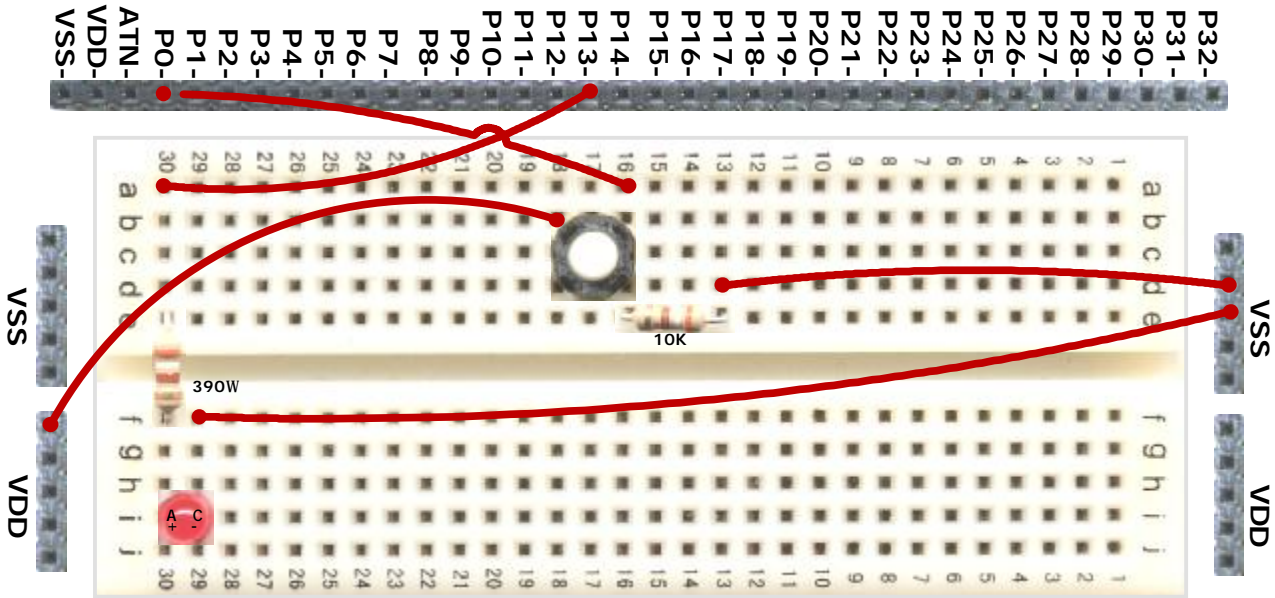


Figure 2.4

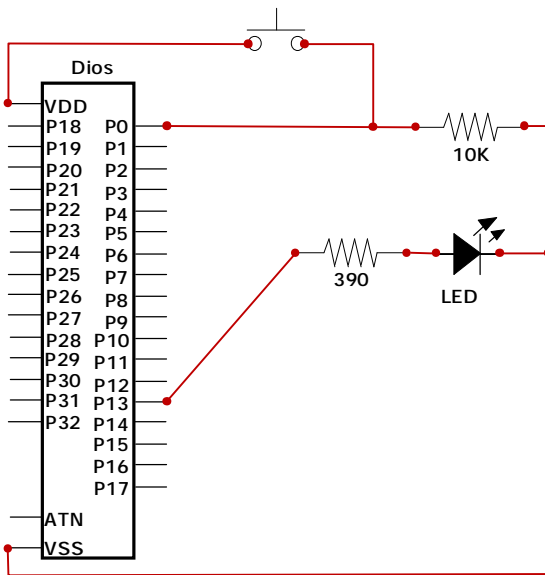


Figure 2.5A

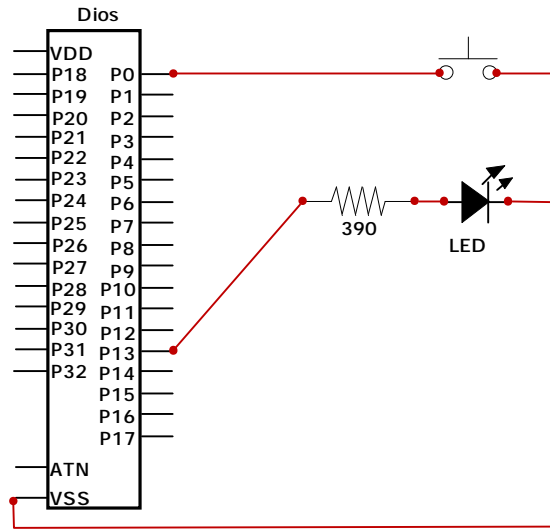


Figure 2.5B

Program 2.3 monitors IO port 0. If IO port 0 is high it issues the high command on port 13. If not it issues the low command on port 13. This is the most simple button indicator.

Challenge

- ☐ Modify Program 2.3 so that it will work with the circuit in Figure 2.5B. Notice that the circuit does not have a pulldown or pullup resistor connected. You must use the built-in resistors.

Hint: Only one command needs to be added to your program and one line needs to be changed.

```
'Use a button to control and LED
func main()

    output 13
    low 13

loop:

    if inp0 = 1 then
        high 13
    else
        low 13
    endif

    goto loop

endfunc
```

Program 2.3

- ❑ Load your microcontroller with Program 2.4. Rebuild the circuit in Figure 2.5A if you need to.



This program does a bit more. Each time you push the button it increments the mode variable then calls a subroutine called blink. The blink subroutine will blink the LED to match the count in the mode variable.

- ❑ Load your microcontroller with Program 2.5.

This program implements a complete menu with the LED and button. You cycle through the menu items the same as you did in Program 2.4. However, when you hold the button down for more than a second or so it will jump to a command handler.

The real work horse portion of this program is the while/wend command just after the loop label. After pausing a few milliseconds it increments a counter. When this counter passes a count of 200 it jumps to the handler.

Challenge

-  ❑ Modify Program 2.4 so that it will work with the circuit in Figure 2.5B. Again, you will have to activate the built-in resistors.
-  ❑ Next Modify Program 2.5 to work with the circuit in Figure 2.5B.

```
'Use a button to control and LED
func main()

    output 13
    low 13

    dim mode,x

    mode = 1

loop:

    if inp0 = 1 then
        mode = mode + 1
        if mode > 5 then
            mode = 1
        endif

        gosub blink
        while inp0 = 1
            wend
        endif

        goto loop

'-----
'Blinks the mode count
blink:

    print "Mode =",mode
    for x = 1 to mode
        high 13
        pause 200
        low 13
        pause 200
    next
    return

endfunc
```

Program 2.4

```

'Use a button to control and LED
func main()

  output 13
  low 13

  dim mode,x,count

  mode = 1

loop:

  if inp0 = 1 then
    while inp0 = 1
      pause 4
      count = count + 1
      if count > 200 then
        count = 0
        goto docommands
      endif
    wend

    count = 0
    mode = mode + 1
    if mode > 5 then
      mode = 1
    endif

    gosub blink
  endif

  goto loop

'-----
docommands:

  branch mode,cont,md1,md2,md3,md4,md5
  goto cont

md1:
  print "Hey Mode 1"
  goto cont

md2:
  print "Hey Mode 2"
  goto cont

md3:
  print "Hey Mode 3"
  goto cont

md4:
  print "Hey Mode 4"
  goto cont

md5:
  print "Hey Mode 5"
  goto cont

cont:
  while inp0=1
  wend
  goto loop

'-----
'Blinks the mode count
blink:
  for x = 1 to mode
    high 13
    pause 200
    low 13
    pause 200
  next
  return
endfunc

```

Program 2.5

Project 3: Drive a High Power LED

There are times when we need to drive very large or high power LED's. The Diodes can drive up to 25ma. To drive more than that you need to use a transistor. Using Figures 2.6 and 2.7, make the following connections:

Note that if you dont have a high power LED you can use a regular LED. Just make sure you use a resistor to drop the voltage.

- Place a 390 Ohm resistor on the breadboard at positions b19 and b22.
- Place the high power LED on the breadboard. Anode = c28 Cathode = c27
- Place a 2N2222 transistor on the breadboard. C = e20 B = e19 E = e18
- Connect Port 6 to the resistor at a22.
- Connect the Anode on the LED (e28) to VDD. Note that if you are using a normal LED, you can make this connection with a 390 Ohm resistor.
- Connect the Cathode on the LED (e27) to the collector on the transistor at d20.
- Connect the emmitter on the transistor (d18) to VSS.
- Load your microcontroller with Program 2.1.

We are using the same program we used with the standared LED just to keep things consistant. You should be able to run all the dim and button code examples as well.

Depending on what you are driving, you can increase the size of the resistor if the transistor overheats.

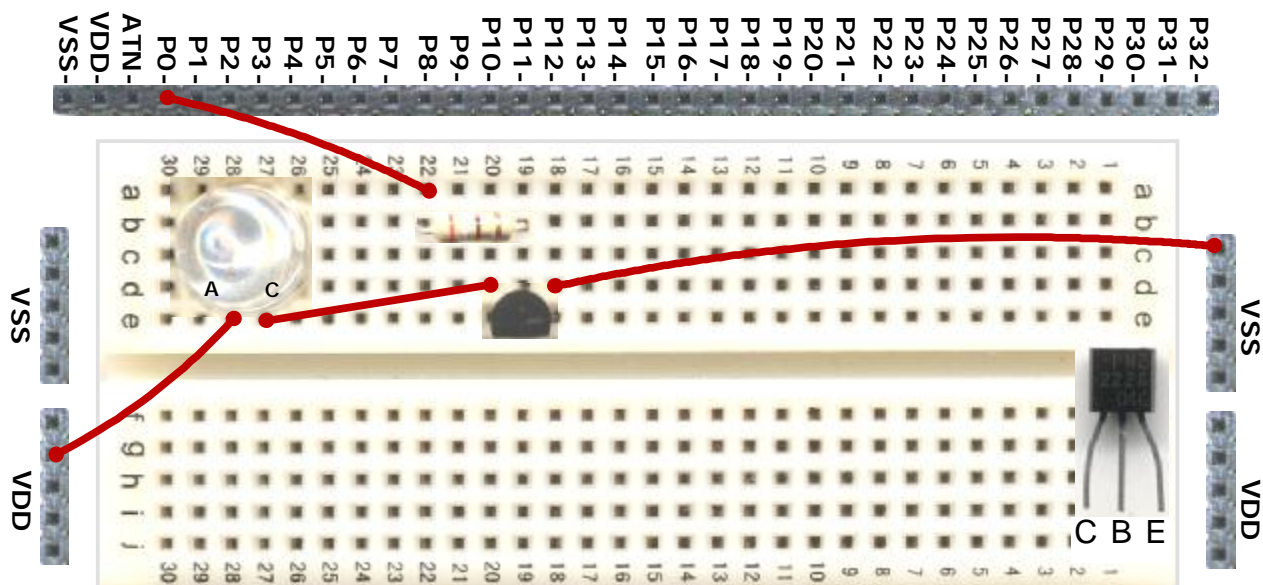


Figure 2.6

Chapter 2 Summary

Chapter 2 was more of a hands-on workout. It is important that you understand the basic concepts here.

First, we showed you how to turn an LED on and off. We then showed you how to dim the LED and introduced you to PWM.

Next, we added a button to an IO port in order to control an LED. We improved upon the design and created a single button/LED menu system.

Finally, in this chapter we used a transistor to control a high power LED.

We have provided you with a set of questions that target the key points in this chapter.

The answers are on the last page of the workbook. Remember if you answer any of the questions incorrectly simply go back and review this chapter.

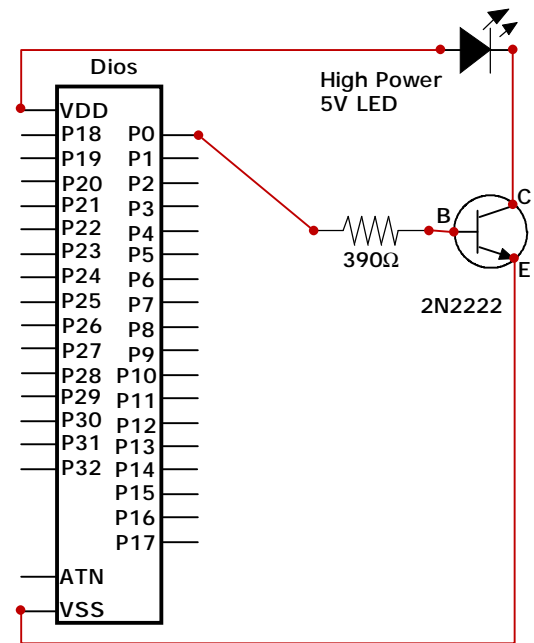


Figure 2.7

Chapter 2 Questions

1. What does PWM stand for?

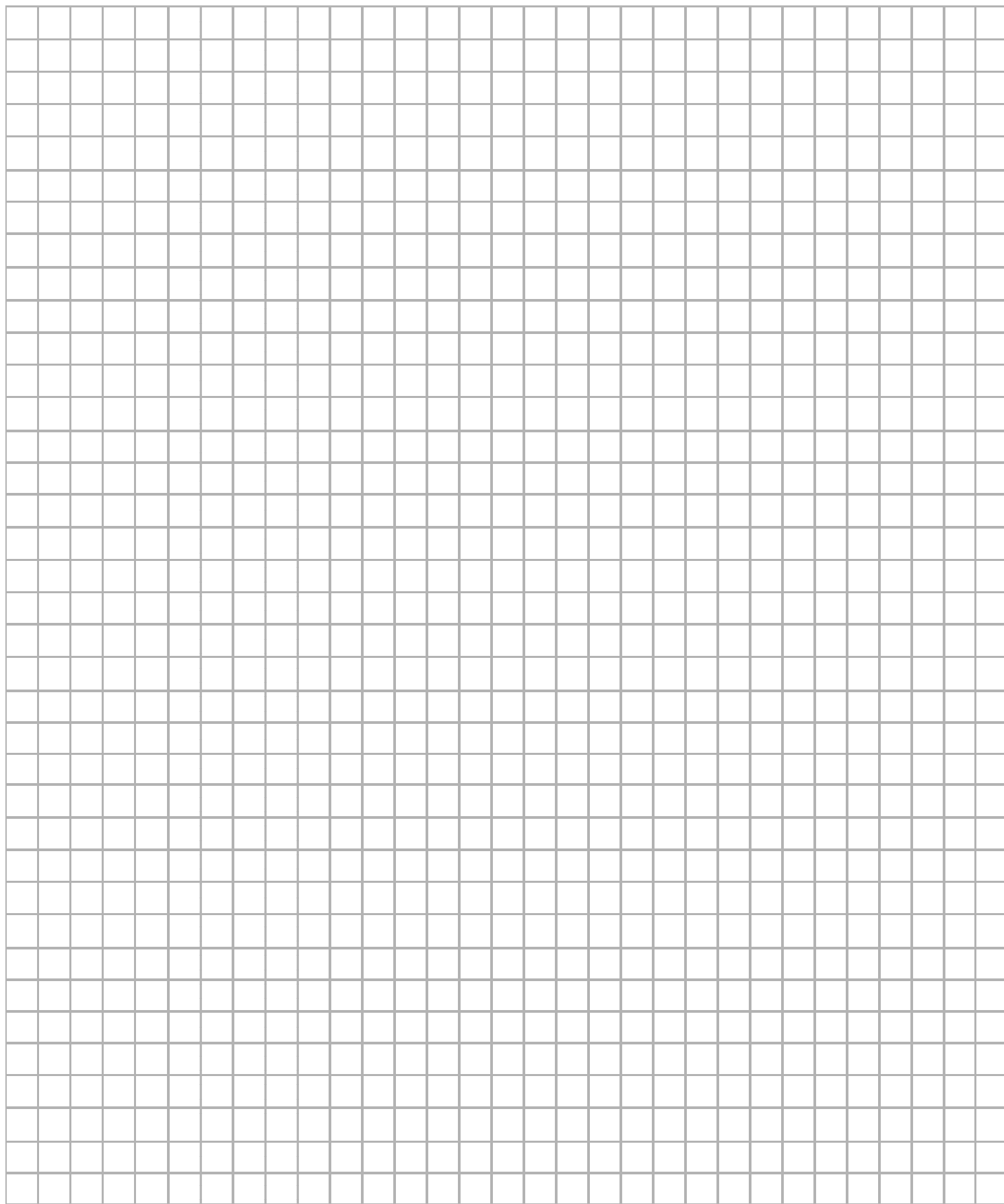
2. What is the duty cycle of the Diod PWM generator with the commands **PWMperiod 100** and **PWM1duty 66** ?

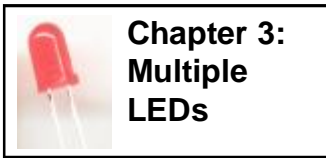
3. What is the duty cycle of the Diod PWM generator with the commands **PWMperiod 200** and **PWM1duty 50** ?

4. Why do we need to hold an IO port low with a resistor when using a button?

5. What is the maximum current the Diod can drive with its IO ports?

6, What command is used to activate the built-in pullup resistors on IO ports 0-7?





Chapter 3: Multiple LEDs

In this chapter we will hookup multiple LEDs. This will allow us to create patterns and display more information then we could with a single LED.

Required Components for Chapter 3

- Dios WorkBoard
- Dios 40 or DiosPro Microcontroller
- 8 Standard LEDs
- 8 390 Ohm Resistors.
- 10K Resistor
- Button

Project 1: Control 8 LEDs

As you might think, you control 8 LEDs much like you control one. Using Figures 3.1 and 3.2, make the following connections:

- Place 8, 390 Ohm resistors on the breadboard at the following positions: e30-f30, e27-f27, e24-f24, e21-f21, e18-f18, e15-f15, e12-f12, e9-f9.
- Place 8, LEDs on the breadboard at the following positions: h30-h29, h27-h26, h24-h23, h21-h20, h18-h17, h15-h14, h12-h11, h9-h8. Note that the first connection on each LED is the Anode. (Long Lead)
- Connect the following IO ports on the breadboard: P0-a30, P1-a27, P2-a24, P3-a21, P4-a18, P5-a15, P6-a12, P7-a9.
- Place the following jumpers on the breadboard: f29-f26, j26-j23, f23-f20, j20-j17, f17-f14, j14-j11, f11-f8, j8-VSS.
- Load your microcontroller with Program 3.1.

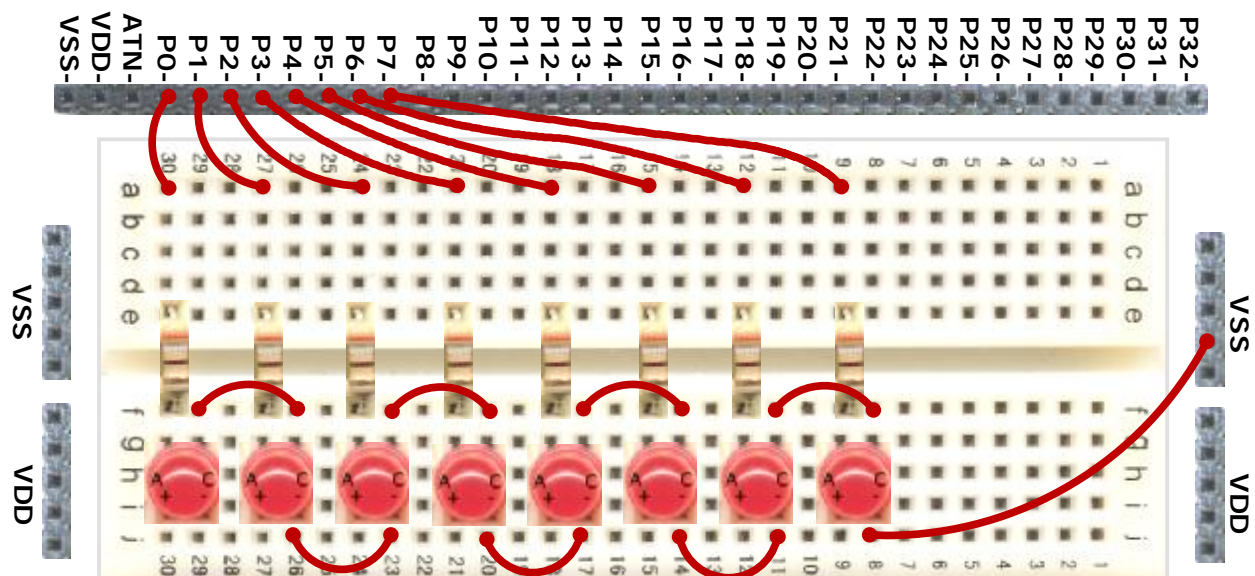


Figure 3.1

This program will turn off all LEDs using the **low** command. It will then set the port pointed to by the **x** variable high. This will in turn light the LED connected to that port.

Notice that the **low** command is setting multiple ports low. Instead of issuing separate **low** commands we can use a single command to set many ports low. You specify a series of numbers separated by a comma indicating the ports you want to set low. Like the **low** command the **high** command may set multiple ports as well.

Like the **low** command the **output** command can set multiple ports as output without issuing separate output commands.

```
Dios
'Multiple LED's
func main()

  dim x

  output 0,1,2,3,4,5,6,7

loop:
  for x = 0 to 7
    low 0,1,2,3,4,5,6,7
    high x
    pause 100
  next
  goto loop

endfunc
```

Program 3.1

□ Load your microcontroller with Program 3.2.

Similar in function to the last example, Program 3.2 causes the LEDs to appear to cycle back and forth. Notice that **for/next** commands are used to cycle the lit LED to the right by incrementing the **x** variable for you. To cycle them back we need to decrement the **x** variable. To decrement with a **for/next** loop we must use a **step -1** option.

The **for/next** command has three arguments that you specify to set the looping parameters. The first is the variable. The variable is used to hold the count. The next two arguments set the starting point and ending point for the count. Using the **step** option you can supply a fourth argument specifying the amount to increment or decrement the counter.

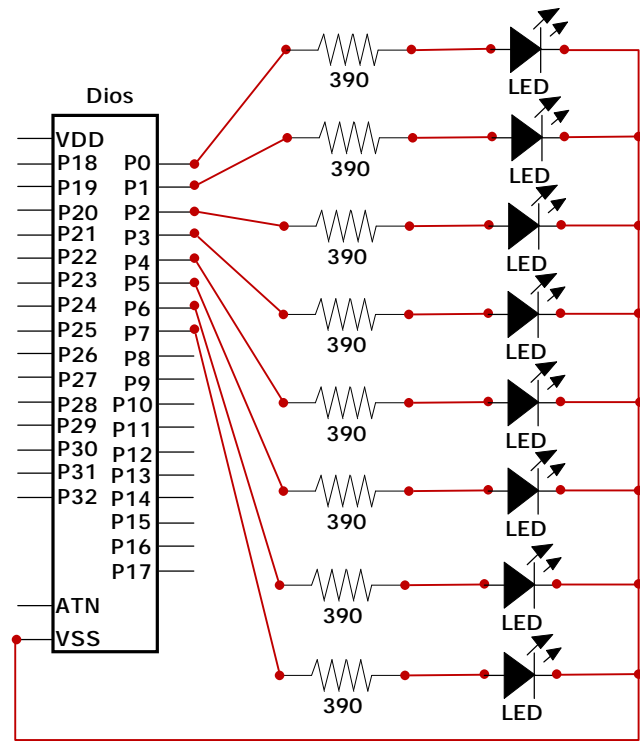


Figure 3.2

```
Dios
'Multiple LED's
func main()

  dim x

  output 0,1,2,3,4,5,6,7

loop:
  for x = 0 to 7
    low 0,1,2,3,4,5,6,7
    high x
    pause 50
  next

  for x = 7 to 0 step -1
    low 0,1,2,3,4,5,6,7
    high x
    pause 50
  next

  goto loop

endfunc
```

Program 3.2

The actual code between the **for** command and the **next** command will be executed.

- ❑ Play with the **pause** values to change the affect. Move the **low** commands around and see what happens.
- ❑ Load your microcontroller with Program 3.3

This program uses the **portset** command to load the individual bits in the x variable into the 8 LEDs. By incrementing the variable we in effect create a binary counter. We have to use the **pause** command to slow things down so that we can see the counter count.

```
Dios
'Multiple LED's
func main()

  dim x

  output 0,1,2,3,4,5,6,7

loop:
  x = x + 1
  portset x
  pause 50

  goto loop

endfunc
```

Program 3.3

Project 2: Electronic Die

Now for something a bit more fun. Let's wire up 7 LEDs to form a die. To roll the die we will use a single button. Using Figures 3.3 and 3.4, make the following connections:

- ❑ Place 7, 390 Ohm resistors on the breadboard at the following positions: a22-a25, a17-a20, a9-a12, j22-j25, j14-j17, j9-j12, f20-f23.
- ❑ Place a 10K Ohm resistor on the breadboard at b4-b7.
- ❑ Place 8, LEDs on the breadboard at the following positions: c22-c21, c17-c16, c12-c11, h22-h21, h17-h16, h12-h11, f19-e14. Note that the first connection on each LED is the Anode. (Long Lead).
- ❑ Place a button on the breadboard at c4-c2.
- ❑ Connect the following ports on the breadboard: P0-b25, P1-b20, P2-b9, P3-i25, P4-i14, P5-i9, P6-g23, P7-a4.
- ❑ Place the following jumpers on the breadboard: e16-e21, b11-b16, d7-d11, e11-f11, g11-g16, h19-h20, i16-j21, d14-d16, VSS-e7, VDD-e2.
- ❑ Load your microcontroller with Program 3.4.

There are two important sections in this program.

Number Selection Section

In this section, as long as the button connected to port 7 is pushed, we increment the x variable. If it is more than 6 we set it back to 1. Once we do this we fall through and display the results. Notice we slow it down by using a **pause** command. This let's us see the LEDs flicker. The LEDs are changing so fast that it looks like they are random.

The **inp7** operator returns the state of IO port 7. This is one way we can test the state of an IO port. You can also use the syntax **inp.X** where X is an IO port number or constant pointing to an IO port. The **ioport(X)** command may also be used.

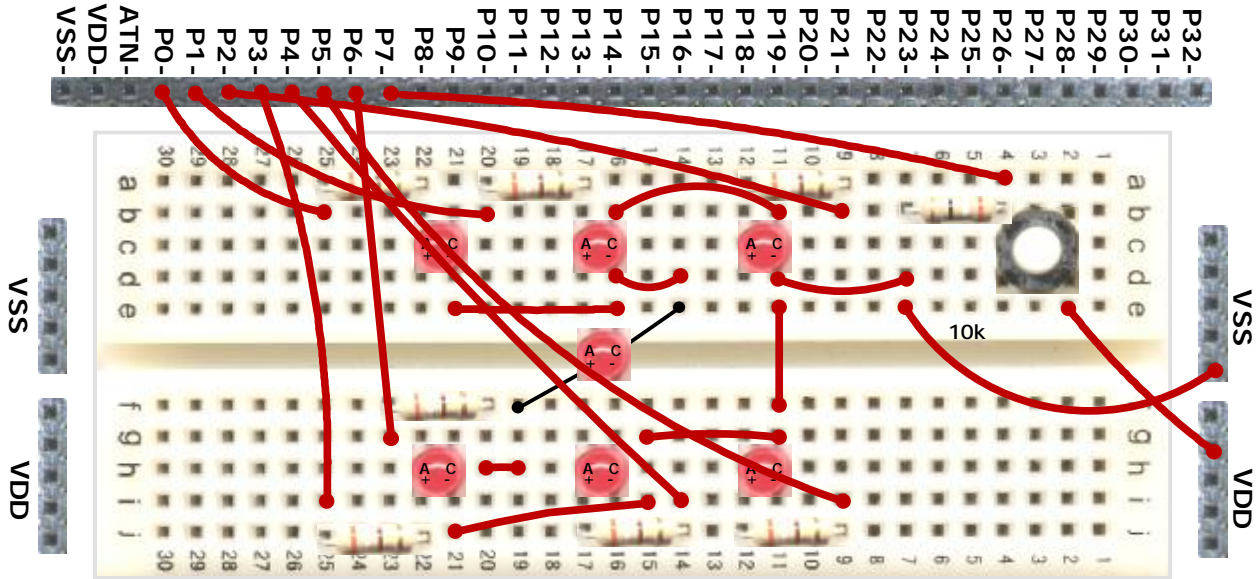


Figure 3.3

Display Section

In this section we use the **branch** command to jump to one of 6 small routines to turn on a particular set of LEDs.

To use the **branch** command you supply a variable. This variable acts as an index to force the program to jump to a location based on that number. You must supply a jump point for each possible value in the variable. If there is no jump point defined for a particular value the command will just fall through to the next command.

Notice that 0 is a valid number and is the first jump point. In our case we will never have a 0 so we just place jump point back to the main loop location at the beginning of the program.

Each routine will use the **low** command to turn off all the LEDs. The **high** command is then used to turn on one or more LEDs to simulate the pips on a die.

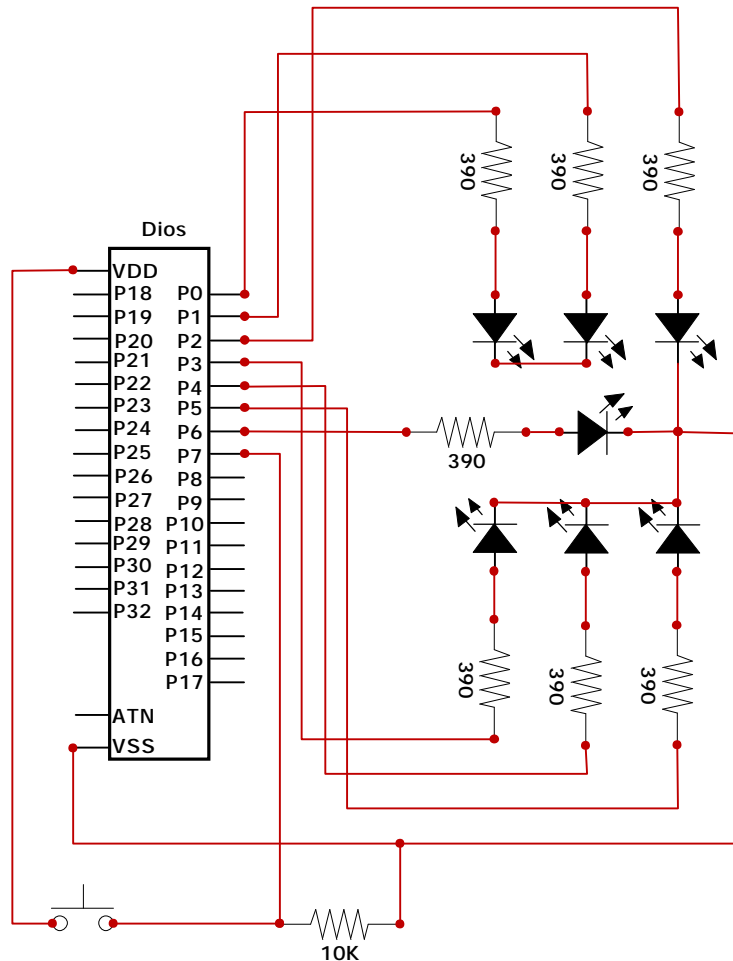


Figure 3.4

Chapter 3 Summery

In this chapter we showed you the basics of controlling more than 1 LED.

First we showed you how to sequence through several LEDs. We then created a binary counter.

Next we created a 6-sided Die with 7 LED's and simulated rolling the Die with a button

We have provided you with a set of questions that target the key points in this chapter.

The answers are on the last page of the workbook. Remember, if you answer any of the questions incorrectly simply go back and review this chapter.

This concludes the Dios LED workbook. Please checkout our other workbooks at www.kronosrobotics.com

```
Dios
'Electronic Die
func main()

  dim x

  output 0,1,2,3,4,5,6

loop:
'-----
'  Number Selection Section
'-----
  if inp7 = 1 then
    x = x + 1
    if x > 6 then
      x = 1
    endif
    pause 10
  else
    goto loop
  endif

'-----
'  Display Section
'-----
  branch x,loop,one,two,three,four,five,six

one:
  low 0,1,2,3,4,5,6
  high 6
  goto loop
two:
  low 0,1,2,3,4,5,6
  high 0,5
  goto loop
three:
  low 0,1,2,3,4,5,6
  high 0,5,6
  goto loop
four:
  low 0,1,2,3,4,5,6
  high 0,2,3,5
  goto loop
five:
  low 0,1,2,3,4,5,6
  high 0,2,3,5,6
  goto loop
six:
  low 0,1,2,3,4,5,6
  high 0,1,2,3,4,5
  goto loop

endfunc
```

Program 3.4

Chapter 3 Questions

1. How do you set multiple ports high or low all at once?

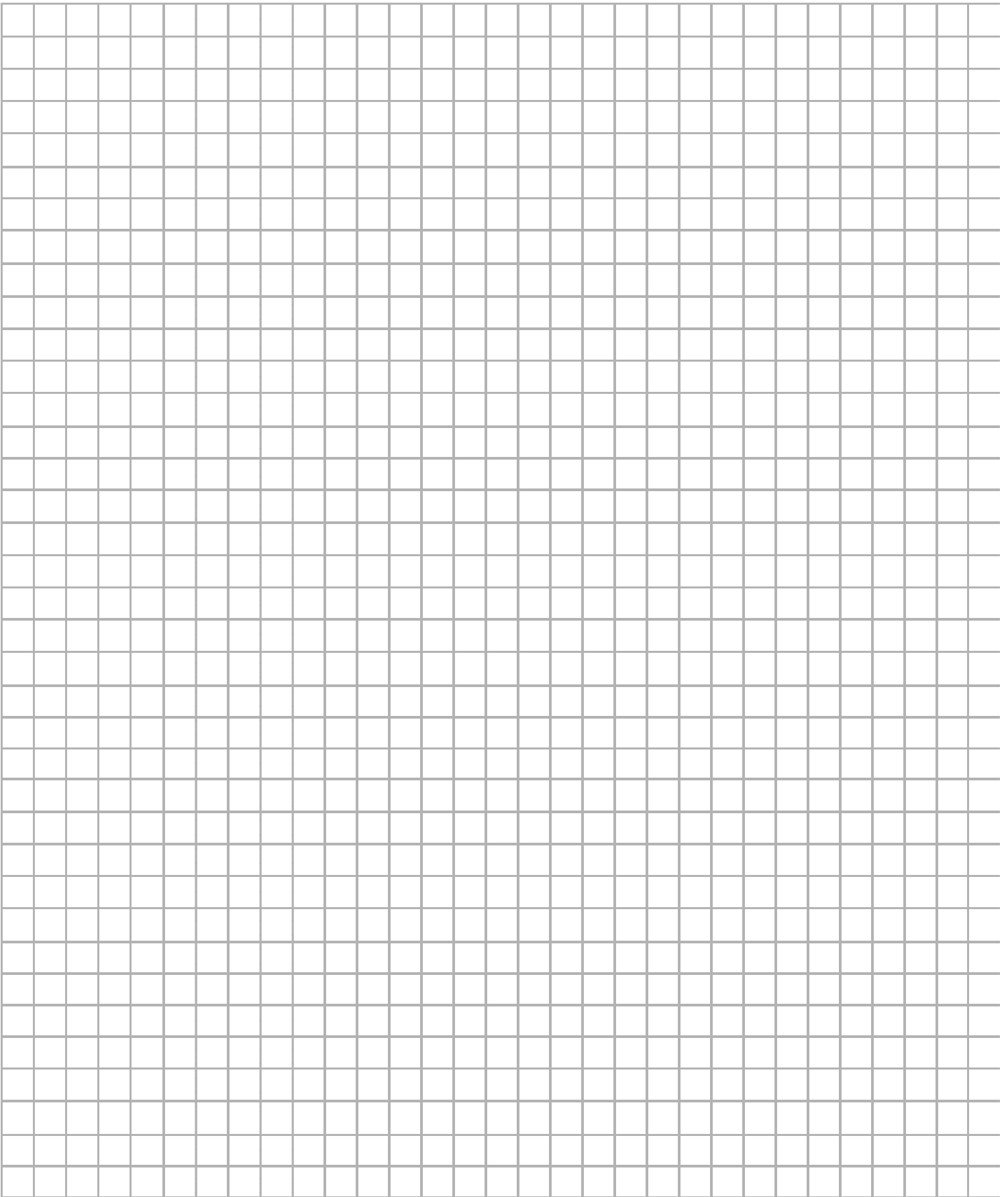
2. What extra option do you need to use to decrement a number with the for/next command?

3. What command allows you to set 8 LEDs all at once using a single variable?

4. Why do we need to use the pause command in most of the program examples?

5. What operator do you use with the **if/then** command to test an IO port?

6. What happens when the branch command does not have a location to jump to for a given value?



6. If falls through to the next command.
5. inp
4. To slow down the state changes of the LED so the human eye can see them.
3. portset
2. step - 1
1. We use the high or low command separating the port numbers with a comma.

Chapter 3 Answers

6. pullupon
5. 25ma
4. So it does not float. This will keep it from randomly changing until the button is pressed.
3. 25%
2. 66%
1. Pulse Width Modulation

Chapter 2 Answers



- 8.
7. It won't light.
6. Most LEDs are rated at 2-3 volts and hooking them up to 5v directly will burn them up.
5. The Clear LED will focus the light toward the front of the LED. A diffused LED will simply glow.
4. Anode
2. Cathode
1. Almost 1/4"

Chapter 1 Answers

